

## Description

This function is essentially a wrapper function that allows all the normalizations needed or desired to be done using one command. It can carry out spatial, background, dye, and within and across condition normalization. Any subset of these can be done on the data. Note that, although there are many arguments to this function, only the first 11 need to be considered unless the user wishes to adjust the methods used to normalize the data.

## Usage

```
all.norm(dat, x=NULL, y=NULL, dat.log.scale=TRUE,
        empty=NULL, invariantset=NULL, conditions=NULL,
        spat=TRUE, bkg=FALSE, dye=TRUE, cond=TRUE,
        subset=NULL, ignore=NULL, scale.norm=TRUE,
        span.loc=0.5, degree.loc=1, span.scale=0.75, degree.scale=1,
        zero.centred=FALSE, n.sample = 1*10^4
        bkg.method=c("deterministic","probabilistic"), mu.emp=NULL, sd.emp=NULL,
        qnt1=0.5, add.one=TRUE,
        span=0.2, cond.method=c("quantile","global"))
```

## Arguments

<code>dat</code>	Matrix of unnormalized microarray expression with rows being arrays and columns being spots on the arrays. Note that when dye normalization is carried out, each pair of channels corresponds to a cy3 and cy5 channel of an array.
<code>x</code>	Vector of length <code>nrow(dat)</code> containing x-coordinates of all the spots on the arrays. This needs to be supplied for spatial normalization.
<code>y</code>	Vector of length <code>nrow(dat)</code> containing y-coordinates of all the spots on the arrays. This needs to be supplied for spatial normalization.
<code>dat.log.scale</code>	Logical value indicating whether or not the data in <code>dat</code> are on the log scale. The default value ( <code>TRUE</code> ) indicates that the data are on the log scale.
<code>empty</code>	Used in background normalization to define the empty control spots. Normally a vector of indices or names. However, it can also be a single number, $n$ , in which case the $n$ -th smallest values in <code>xpr</code> will be assumed to be the empty cells. If it is not specified ( <code>NULL</code> ) when <code>method = "probabilistic"</code> then <code>mu.emp</code> and <code>sd.emp</code> must be specified. See <code>bkg.norm</code> for more details.
<code>invariantset</code>	A vector or an integer defining the invariant spots that are to be used in the dye normalization and/or across condition normalization. If a vector its elements should index the columns of <code>dat</code> that represent invariant spots. If an integer, then it indicates the largest change in rank that

spots to be included in the invariant set can take. If unspecified, then all spots will be assumed to be invariant across conditions (and dyes). See `dye.norm` and `cond.norm` for more details on how `invariantset` is used in dye and across condition normalization.

<code>conditions</code>	A vector the length of <code>(nrow(dat))</code> containing labels for the conditions on the different arrays or array/channels. This is used in within and across condition normalization.
<code>spat</code>	Logical indicator of whether spatial normalization should be conducted; it is conducted by default.
<code>bkg</code>	Logical indicator of whether background normalization should be conducted; the default is not to conduct it. See Details.
<code>dye</code>	Logical indicator of whether dye normalization should be conducted; it is conducted by default.
<code>cond</code>	Logical indicator of whether within and across condition normalization should be conducted; it is conducted by default.
<code>subset, ignore, scale.norm, span.loc, degree.loc, cr span.scale, degree.scale, zero.centred,</code>	Arguments in <code>spat.norm</code> that can be altered if desired or necessary. See <code>bkg.norm</code> .
<code>bkg.method</code>	The <code>method</code> ("deterministic" by default) for global background normalization; see <code>bkg.norm</code> .
<code>mu.emp</code>	Optional argument that can be used in background normalization. See <code>bkg.norm</code> .
<code>sd.emp</code>	Optional argument that can be used in background normalization. See <code>bkg.norm</code> .
<code>qnt1</code>	Argument in <code>bkg.norm</code> that can be altered if desired. See <code>bkg.norm</code> .
<code>add.one</code>	Argument in <code>bkg.norm</code> that can be altered if desired. See <code>bkg.norm</code> .
<code>span</code>	Span of the loess used in the dye normalization.
<code>cond.method</code>	The <code>method</code> ("quantile" by default) for across and within condition normalizations; see <code>cond.norm</code> .

## Details

All normalizations (`spat.norm`, `dye.norm`, `cond.norm` are conducted by default with the exception of background normalization (`bkg.norm`).

The unnormalized expression values (`dat`) are assumed (by default) to be on the log scale. If this is not the case, then set `dat.log.scale = FALSE`.

## Value

A matrix of the normalized expression values.

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

## See Also

spat.norm, bkg.norm, dye.norm, cond.norm

## Examples

---

bkg.norm *Background normalization*

---

## Description

This performs a global background normalization using either a probabilistic or deterministic method. If used, it should be done after spatial normalization and before any other normalization is carried out.

## Usage

```
bkg.norm(xpr, empty=NULL, method=c("deterministic","probabilistic"),  
         dat.log.scale=TRUE, mu.emp=NULL, sd.emp=NULL, qntl=0.5, add.one=TRUE)
```

## Arguments

xpr	Microarray data to be background normalized. The data are assumed to be in the form of a vector. Although they can be supplied in matrix format, they are converted to a vector internally and the output is always given in vector format.
empty	Use this to define the empty control spots. Normally a vector of indices or names. However, it can also be a single number, $n$ , in which case the $n$ -th smallest values in xpr will be assumed to be the empty cells. If it is not specified (NULL) when method = "probabilistic" then mu.emp and sd.emp must be specified. See Details.
method	Can be "probabilistic" or "deterministic" (the default) methods described in the Details below.
dat.log.scale	Logical indicating whether the data are on the log scale (TRUE) or on the original skewed scale (FALSE). The background normalization in bkg.norm works on the original scale and so if dat.log.scale=TRUE the data in xpr are exponentiated.
mu.emp	If available, an estimate of the empirical mean of the background spots can be given in this argument. This is only needed (and used) for method = "probabilistic" when empty is unspecified (NULL), in which case sd.emp must also be given.

<code>sd.emp</code>	If available, an estimate of the empirical standard deviation of the background spots can be given in this argument. This is only needed (and used) for <code>method = "probabilistic"</code> when <code>empty</code> is unspecified (NULL), in which case <code>mu.emp</code> must also be given.
<code>qnt1</code>	The quantile of the <code>empty</code> spots used in the <code>"deterministic"</code> background subtraction <code>method</code> . It is 0.5 (the median) by default. See Details below.
<code>add.one</code>	Logical indicating whether one is added to each of the background values. Adding one stops zero normalized values occurring which are problematic if the data is to be logged at a later stage. By default this is added. If the data are given on the log scale ( <code>dat.log.scale=TRUE</code> ), then <code>add.one</code> will be forced to be TRUE as the function returns the normalized data on the log scale.

## Details

`bkg.norm` carries out global background normalization since local background normalization can suffer from problems of contamination and increased variance. However, note that Wit and McClure (2004) suggest not doing any background normalization and that `all.norm` does not do background normalization by default.

The data can be specified on either the log or original scale and are returned in the scale given. Empty spots should be defined using `empty` by either specifically or as a certain number of the smallest values on the array. If they are not specified, then `mu.emp` and `sd.emp` must be given appropriate values when `method="probabilistic"`. Two methods for global background normalization can be chosen, which though different tend to give very similar results.

The (default) `"deterministic"` global background normalization `method` uses the following equation:

$$\hat{y}_i = \max(y_i - \mu_e, 0),$$

where the original intensities  $y_i$  have some representative value  $\mu_e$  of the `empty` spots subtracted from it. Any negative values are replaced by zeros. The value for  $\mu_e$  used here is a suitable quantile, `qnt1`, with the median (`qnt1=0.5`) being the default. If no empty spots are provided, then the smallest value in `xpr` is subtracted from all the values in `xpr`.

The `"probabilistic"` `method` for global background normalization is more sophisticated. This approach finds the conditional expectation of the true signal given the observed signal,

$$E(s_i | s_i + b_i)$$

where the observed signal is thought of as the sum of the true signal  $s_i$  and the background  $b_i$ , which are assumed to follow exponential and normal distributions respectively. Estimates of the background distribution's mean and standard deviation can be calculated from the `empty` spots or else need to be specified in `mu.emp` and `sd.emp`. The method is described further in Wit and McClure (2004) and was introduced in Irizarry et al. (2003). However, even when the assumptions underlying this approach are true, the simpler `"deterministic"` approach appears to do at least as well.

Finally note that if `add.one=TRUE` then 1 will be added to all the normalized values. This ensures that there are no zero values if the data are subsequently logged. If the data are given on the log scale then 1 is added regardless, since after normalizing on the (exponentiated) original scale the data are logged before being returned.

## Value

The function returns a vector of global background normalized expression values on the same scale as that of the data given in `xpr`. Note that the object specified as `xpr` is converted (if necessary) into vector format using `as.vector`, so matrices (and data frames) will lose their positional information.

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

R. Irizarry et al. (2003) Exploration, normalization, and summaries of high density oligonucleotide array probe level data, "Biostatistics" 4(2), 249–64.

## See Also

`all.norm`, `bkg.norm`, `dye.norm`, `cond.norm`

## Examples

---

<code>breast</code>	<i>DATA: Breast Cancer microarray experiment</i>
---------------------	--

---

## Description

This data set details microarray experiment for 62 breast cancer patients. The microarrays used are small custom arrays with 57 genes. For two genes both the 5' and 3' versions are included. Rather than measuring gene expression, this experiment aims to measure gene amplification or deletion, which refers to the number of copies of a particular DNA sequence within the genome. The aim of the experiment is to find out the key genomic factors involved in aggressive and non-aggressive forms of breast cancer. The experiment was conducted by the Dr. John Bartlett and Dr. Caroline Witton in the Division of Cancer Sciences and Molecular Pathology of the University of Glasgow at the city's Royal Infirmary. Many thanks to them for allowing us to include this data.

## Usage

```
data(breast)
```

## Format

A list containing:

**dat** A 62 by 59 matrix containing the amplification values of 59 clones. The clones represent 57 genes; for two genes both the 5' and 3' version were included. Each row represents the gene amplification profile of a breast cancer tumour from a single patient. The amplification values are calculated by taking the ratio of tumour samples versus a reference sample.

**arrays** A vector of length 62 identifying the array (1, 2, ... 62), from which each row of **dat** came.

**age.at.diag** A vector of length 62, indicating the age at which the patients were diagnosed with breast cancer and a biopsy was taken.

**surv.time** A vector of length 62, indicating the follow-up time for each patient. For patients that have died (**any.death=1**) this indicates the additional survival time beyond diagnosis. For patients that have not died (**any.death=0**) this indicates the time from diagnosis to the end of the study.

**cancer.death** A binary vector of length 62, indicating whether or not the patient has died of breast cancer (0 = did not die of breast cancer, 1 = died of breast cancer).

**any.death** A binary vector of length 62, indicating whether or not the patient has died before the end of the study due to any cause (0 = did not die, 1 = died). This is can be regarded as a censoring indicator.

**cancer.grade** A vector of length 62, indicating the severity of the breast cancer tumour. This grade uses the Scarff-Bloom-Richardson system to score the tumour on the frequency of cell mitosis, tubule formation and nuclear pleomorphism. Grade 1 represents well-differentiated breast cells, i.e. the cells generally appear normal and are not growing rapidly and the cancer is arranged in small tubules. Grade 2 represents moderately-differentiated breast cells. Grade 3 represents poorly differentiated breast cells, i.e. the cells do not appear normal and tend to grow and spread more aggressively. Survival rates are typically lower for higher cancer grades.

**size** A vector of length 62, indicating the size of the tumour in mm.

**npi** A vector of length 62, indicating the Nottingham Prognostic Index. This index is used to help classifying the tumours into different groups, while controlling for non-genomic influences. The index is calculated by combining three prognostic factors: tumour size (cm x 0.2); lymph node stage (1 = node negative, 2 = 1-3 metastatic nodes 3 =  $\geq 4$  metastatic nodes;) and histological grade (1-3, good, moderate, poor). A prognostic index  $< 3.4$  implies a good prognosis, between 3.4 and 5.4 implies a moderately good prognosis and over 5.4 implies a poor prognosis.

## Source

Dr. John Bartlett and Dr. Caroline Witton, Division of Cancer Sciences and Molecular Pathology, University of Glasgow, Glasgow Royal Infirmary.

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## Description

A wrapper function for visualising samples. This can be done using a Sammon map, a principal component analysis (PCA) plot or via hierarchical clustering (either ordinary or using `hipam`).

## Usage

```
cluster.samples(dat, method="sammon", labels=NULL, metric="euclidean",
               title="Clustering Samples", linkage="average",
               cex.main=2, cex.sub=1.5, cex.lab=1.2,...)
```

## Arguments

<code>dat</code>	Expression data of the samples to be clustered. Should be given as a matrix, whose rows should represent the samples.
<code>method</code>	Character string specifying the method to be used on <code>dat</code> . Possible methods include "sammon" (the default), "pca", "hipam" and "hierarchical". See Details for more information.
<code>labels</code>	Labels can be given to the samples using this argument. If given, <code>labels</code> should be a vector of length <code>nrow(dat)</code> . If not given, the samples rows positions are used instead.
<code>metric</code>	Character string specifying the distance measure to be used. Can be a power distance ("manhattan", "euclidean", "cubic", "quartic"), 1 minus the correlation ("correlation",) or 1 minus the absolute correlation "abscorr".
<code>title</code>	Character string to be added as the title of the plot.
<code>linkage</code>	The type of linkage to be used in hierarchical clustering. See <code>hclust</code> for more details.
<code>cex.main</code>	Character expansion scalar for the title.
<code>cex.sub</code>	Character expansion scalar for the information given in the subtitle.
<code>cex.lab</code>	Character expansion scalar for the sample labels.
<code>...</code>	Other arguments can be supplied that depend on the clustering methods used. See <code>text</code> (for Sammon and PCA), <code>plot.tree</code> (for HIPAM) or <code>plclust</code> for possibilities.

## Details

This function can visualize the similarity between samples using Sammon mapping, a principal component analysis (PCA) plot or a dendrogram.

If a Sammon map is chosen (`method="sammon"`), then a plot of all of the samples is produced with each sample labelled using `label` and positioned using `sammon`. A similar plot is shown when `method="pca"` using the first two principal components from a PCA on the samples.

If a dendrogram is desired, then there are two choices: an ordinary dendrogram created using `hclust` and `plclust` is shown if `method="hierarchical"`; if `method="hipam"` is chosen, then a HIPAM clustering is done on the data using `hipam` and the resulting dendrogram is plotted using `plot.tree`.

### Author(s)

E. Wit and J. McClure

### References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

### See Also

`sammon`, `par`, `hipam`, `plot.tree`, `hclust`, `plclust`,

### Examples

```
data(skin)
labels <- paste(skin$array,substr(skin$condition,1,1),skin$dye,sep="")
# Sammon plot of similarity of samples:
cluster.samples(skin$dat,labels=labels)
# PCA plot of similarity of samples:
cluster.samples(skin$dat,method="pca",labels=labels)
# dendrogram of similarity of samples:
cluster.samples(skin$dat,method="hierarchical",labels=labels)
```

---

`cond.norm`

*Normalization within and across conditions*

---

### Description

It is essential when comparing data from different arrays (or channels) that they are being measured on as similar scale as is possible. `cond.norm` allows this to be done using either a quantile normalization method or a global location and scale normalization method. By supplying the condition each array or array/channel combination represents, replicates can be normalized before across condition normalization takes place.

### Usage

```
cond.norm(x, conditions=NULL, invariantset=NULL, dat.log.scale=TRUE,
          method=c("quantile","global"))
```

## Arguments

<code>x</code>	A matrix whose rows represent arrays (or array/channel combinations with 2-channel arrays) and whose columns are genes/spots.
<code>conditions</code>	A vector the length of <code>(nrow(x))</code> containing labels for the conditions on the different arrays or array/channels.
<code>invariantset</code>	A vector or an integer. If a vector it should define the invariant elements (genes/spots) in the columns of <code>x</code> that are to be used in aligning the distributions. If a scalar it should be an integer indicating the largest change in rank that spots to be included in the invariant set can take. If unspecified (NULL), then all the data will be used in the quantile normalization. When the "global" method is used <code>invariantset</code> is ignored.
<code>dat.log.scale</code>	Logical value indicating whether or not the data are on the log scale. If FALSE, then logs will be taken of <code>x</code> , as the original scale of microarray data is highly skewed. The default is TRUE.
<code>method</code>	Character string defining the method to be used. This can either be "global", a simple location and scale normalization, or the recommend (and default) "quantile" approach. See Details.

## Details

Two methods for normalization are available in `cond.norm`. The default method uses quantiles. A simpler and potentially less effective approach simply normalizes globally the location and scale of different arrays.

The "global" method simply standardizes values in `x` within each row by removing their row median and dividing by their row median absolute deviation (MAD). These standardized values are then converted back to their approximate original scale by multiplying by the overall MAD of `x` and then adding the overall median of `x`.

The "quantile" method attempts a more sophisticated normalization by making the distributions of the results within each row of `x` the same for replicates and for invariant genes. The normalization takes place in two stages, assuming that there are replicates within each condition.

It is within-condition normalization that is conducted first. For a set of replicate rows, the data in each row are ranked. Each value is then replaced by the median of the values in the other rows with the same rank. This ensures that the distributions across each replicate with any condition will be the same.

Once this has been done, across-condition normalization is carried out using a similar quantile normalization. However, it is often unwise to use all the genes, since gene expressions will be expected to differ between conditions. Generally, however, it would be expected that some genes or control spots will not differ across any of the conditions. These *invariant* set of genes/controls should be used to calculate to calibrate the normalization, using linear interpolation. Note that the maximum and minimum values of each array are automatically added to the invariant set in order to ensure that the linear interpolation does not become *extrapolation*. The larger the invariant set, the better the calibration; however, remember that this should be tempered by the fact that adding genes that do vary to the invariant set has the opposite effect.

See Wit and McClure (2004) for more on these two methods.

**Value**

A matrix of within and across condition normalized expression values.

**Author(s)**

E. Wit and J. McClure

**References**

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

**See Also**

all.norm, spat.norm, bkg.norm, dye.norm

**Examples**

---

<code>cor.crit.val</code>	<i>Calculation of critical values for correlation test</i>
---------------------------	--

---

**Description**

This function calculates the correlation critical values for the test that the population correlation is zero.

**Usage**

```
cor.crit.val(alpha, n)
```

**Arguments**

<code>alpha</code>	A vector of the probabilities for which the correlation critical values are needed
<code>n</code>	The sample size

**Details**

For each given alpha and for the given sample size this calculates the appropriate correlation critical value. It assumes the test is two-sided and returns only the positive critical value. It is used in `cor.visual`.

**Value**

The positive critical value for the test that the population correlation is zero for each given alpha and for the sample size.

## Author(s)

J. McClure and E. Wit

## Examples

```
cor.crit.val(c(0.05,0.01),n=20)
```

---

cor.visual

*Plotting correlations of a subset of variables*

---

## Description

This function plots correlations between each member of a subset of variables and either the variables in the complement to this set, or else all the other variables. Only correlations above or equal to a specified cut-off are plotted.

## Usage

```
cor.visual(cor.mat, selected.variables, cor.cutoff=0, show.plot=TRUE,  
          cf.unchosen.only=FALSE, col=NULL, max.per.plot=10, max.plots=3,  
          signif.levels=NULL, n.obs=NULL, absv=TRUE)
```

## Arguments

cor.mat	A matrix of correlations between the variables.
selected.variables	A vector defining a subset of the variables. It can be: a logical vector the length of <code>nrow(cor.mat)</code> ; a numeric vector specifying the variables from the column/row numbers in <code>cor.mat</code> ; a character vector of the names of the variables in <code>dimnames(cor.mat)</code> . If unspecified, then all variables are selected.
cor.cutoff	Correlation cut-off. Only correlations whose value is above or equal to this cut-off will be displayed. The default value 0. Note that absolute correlations will be considered if <code>absv</code> is TRUE.
show.plot	Logical value indicating whether or not plot should be displayed. By default it is TRUE.
cf.unchosen.only	Logical value. If TRUE correlations between selected variables are not considered. If FALSE (the default), all possible correlations are considered.
col	Vector specifying the colouring for the plot. If <code>signif.levels</code> and <code>n.obs</code> are specified, then this must be a vector of length 3, specifying the colours for the three different levels of significance the correlations can take. Otherwise it specifies the colour for of each predictor plotted; in this case it can be the length of number of predictors or a single colour.
max.per.plot	Maximum number of displays per plot. See details.

<code>max.plots</code>	Maximum number of plots that can be placed into the Graphics Device's figure region. See details.
<code>signif.levels</code>	If specified this needs to be a numeric vector of length 2 that defines three different levels of significance for the correlations. It's two values should be p-values. If specified, then <code>n.obs</code> must also be given. See details.
<code>n.obs</code>	A scaler giving the number of observations per variable that the correlations are based on. This should only be specified if <code>signif.levels</code> is given. See details.
<code>absv</code>	Logical value, specifying whether or not the absolute values of correlations are considered. By default, absolute values are considered.

## Details

For every selected variable, the correlations between that variable and every eligible variables that are greater than or equal to the cut-off are shown. Eligible variables can be either any of the other variables or only those not in `selected.variables`. Absolute correlations will be considered if `absv=TRUE`.

Note that `max.per.plot` and `max.plots` may need to be adjusted from their default values if the number of variables chosen in `selected.variables` is greater than the default `max.per.plot*max.plots`. However, having more predictors than this value may lead to the plot becoming overly crowded.

p-value cut-offs from tests of whether or not the population correlations are zero can be specified by `signif.levels` (two p-values must be given) and by indicating the number of observations that `cor.mat` is calculated from using `n.obs`. Correlations whose p-values are lower than the smaller p-value in `signif.levels` will be coloured black. Correlations with p-values between the values given in `signif.levels` will be shaded dark grey and those whose p-values are larger than both values in `signif.levels` will be shaded light grey. These default colours can be altered using `col`. No attempt at correcting for multiple testing is made.

## Value

A list containing one element for of the selected variables. Each of these elements is vector of correlations equal to or above the cut-off.

## Author(s)

J. McClure and E. Wit

## See Also

`lasso.cor`

## Examples

```
x <- matrix(rnorm(1000), ncol=50)
cor.x <- cor(x)
cor.visual(cor.x, cor.cutoff=0.3, selected.variables=sample(50,10))
# Try different values for cor.cutoff
```

---

<code>discover.internal</code>	<i>Subfunctions called from the microarray discovery functions or their subfunctions</i>
--------------------------------	--

---

### Description

These are subfunctions needed for the microarray discovery functions, but are not directly used.

### Author(s)

E. Wit and J. McClure

### See Also

`pamsam`, `hipam`, `twoway.pam`, `cluster.samples`, `sammon.plot`

---

<code>dye.norm</code>	<i>Dye normalization</i>
-----------------------	--------------------------

---

### Description

This function `dye` normalizes two-channel microarray data using an invariant set of spots/genes if they are available. It uses a loess smoother on the transformed MA plot of the data. Spatial normalization (see `spat.norm`) and, if it is to be used, background normalization (see `bkg.norm`) should be done before `dye` normalization.

### Usage

```
dye.norm(cy3, cy5, invariantset=NULL, dat.log.scale=TRUE, span=0.2)
```

### Arguments

<code>cy3</code>	A vector containing data from the Cy3 channel. If Cy3/Cy5 dyes aren't being used, then this should just be the data from one of the dye channels and <code>cy5</code> be the data from the other. Note that it does not make a difference to the normalization which channel actually goes in <code>cy3</code> .
<code>cy5</code>	A vector containing data from the Cy5 channel. If Cy3/Cy5 dyes aren't being used, then this should just be the data from one of the dye channels and <code>cy3</code> be the data from the other. Note that it does not make a difference to the normalization which channel actually goes in <code>cy5</code> .
<code>invariantset</code>	A vector or an integer. If a vector it should define the invariant elements (genes/spots) in <code>cy3</code> and <code>cy5</code> that are to be used in the loess smoothing. If a scalar it should be an integer indicating the largest change in rank that spots to be included in the invariant set can take. If unspecified ( <code>NULL</code> ), then all the data will be used in the loess.

`dat.log.scale` Logical value indicating whether or not the data are on the log scale. If `FALSE`, then logs will be taken of `cy3` and `cy5`, as the original scale of microarray data is highly skewed. The default is `TRUE`.

`span` The span of the loess carried out on the MA transformed data. See Details.

## Details

A dye normalization is carried out on the data from one microarray experiment given as two vectors, `cy3` and `cy5`. The data need to be on the log scale, so they logs are taken if `dat.log.scale = FALSE`. An MA transformation of the plot is then carried out on the data. For each spot, this finds the average value (on the log scale) of its Cy3 and Cy5 values (*A*) and the difference between them,  $\log(\text{Cy3}) - \log(\text{Cy5})$  (*M*). This conducts an approximate 45 degree transformation of the data.

A problem with using a regression or smoothing technique on the non-MA transformed data is that there is no sense in which one channel should predict the other, rather than *vice-versa*. The results of the normalization would change if the vector assigned to the `cy3` were instead assigned to `cy5`. However, the average (*A*) values can be considered a predictor variable for the differences (*M*) and it is on this transformed version of the data that the function conducts the smoothing.

The smoothing used is loess with a default `span` of 0.2, which Yang et al. (2002) reported as a suitable value for dye normalization. Once the smoothing has been done, this ‘trend’ is removed from the differences (*M*) and then the *A* and trend-removed *M* values are back-transformed to give the dye normalized Cy3 and Cy5 values.

## Value

The functions returns a list containing the two vectors of the dye normalized values: `cy3.norm` and `cy5.norm`.

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

Yang YH, et al. (2002) "Normalization for cDNA microarray data", Technical Report 589, Department of Statistics, University of California at Berkeley, CA.

## See Also

`all.norm`, `spat.norm`, `bkg.norm`, `cond.norm`

## Examples

**Description**

This is a hierarchical clustering method that is based on partitioning around medoids (PAM). It is divisive in that it starts with all objects assumed to be in one single cluster and divides then into smaller and smaller clusters. It shares similarities with both DIANA (Kaufman and Rousseeuw 1990) and HOPACH (Pollard and van der Laan 2002).

HIPAM uses either a local or global method to determine when to stop branching. Both methods use the average silhouette width (asw) and at any stage in building the tree attempt to subdivide each already accepted cluster.

Once created the tree can then be visualized using Sammon mapping.

**Usage**

```
hipam(x, method="local", metric="euclidean", show.plot=TRUE,
      asw.tol=0, maxsplit=4, local.const=NULL, ...)
```

**Arguments**

<code>x</code>	Data to be clustered. Should be given as a matrix or data.frame, with its rows representing the objects to be clustered and the columns the variables.
<code>method</code>	Character string indicating the method of cluster validation. This can either be "local" (the default) or "global". See Details.
<code>metric</code>	Character string specifying the distance measure to be used. Can be a power distance ("manhattan", "euclidean", "cubic", "quartic"), 1 minus the correlation ("correlation"), or 1 minus the absolute correlation "abscorr".
<code>show.plot</code>	Logical indicating whether a Sammon plot should be given of the chosen clustering's medoids. The default is TRUE.
<code>asw.tol</code>	This allows a tolerance (or penalty, if asw.tol is negative) to be included when branch splitting. In global HIPAM this tolerance is added to the overall asw of any proposed clustering. In local HIPAM this tolerance is added to the asw of the proposed partition of a cluster. The default value is 0.
<code>maxsplit</code>	The maximum number of clusters that any existing (or proposed) cluster can be split into when searching for the best tree.
<code>local.const</code>	Only used in local HIPAM. If specified when <code>metric="local"</code> , this is a value (between $-1$ and $1$ ) that the asw of any proposed partition of an existing cluster must be greater than. In this case, <code>asw.tol</code> will be ignored. By default, this is not specified.
<code>...</code>	Can be used to give arguments to the function <code>pamsam</code> that <code>hipam</code> makes use of when clustering.

## Details

This clustering starts by splitting the data into between 2 and `maxsplit` clusters, using average silhouette width (`asw`). Each cluster is then investigated to see if it can be split further. If a split is accepted, then the clusters within that split can then also be investigated. These searches for finer branching of the tree continue until all accepted clusters have been considered for partition.

The method for deciding whether further splits are accepted depends on whether the global or local `method` is used. In the global method it is the overall `asw` of the all the data that determines whether a new split is accepted. As long as the `asw` of the putative clustering is greater than the present `asw` minus `asw.tol`, the proposed new clustering is accepted. The number of sub-clusters that any cluster is potentially divided into is between 2 and `maxsplit` and is selected using the `asw` only within the objects in that cluster.

The global method tends to find only a few clusters in large data sets. It is similar to looking at the earth from space, with only land, sea and ice being distinguishable. Looking locally allows urban, agricultural and mountainous areas to be distinguished, for example. This is the idea behind local HIPAM. Here the sub-dividing of an existing cluster is decided by considering how well the sub-clusters distinguish amongst only themselves rather than other branches of the tree.

There are two methods for determining when a branch should not be split further. The default method decides whether a split should be accepted by comparing the `asw` of the putative (mother) split with the average `asw` from the best further (child) splits of each cluster in the putative split. If the mother `asw` is greater than the child `asw`, then the mother split is accepted, since there is less homogeneity in the mother clustering than in the child clusterings. Note that a tolerance (`asw.tol`) can be subtracted from the average of the child average silhouette widths when deciding whether the mother split is to be accepted.

The other method for determining branch splitting used in local HIPAM compares the putative split's `asw` to a set level determined by `local.const`.

As with `pamsam`, HIPAM uses PAM or CLARA (when there are too many objects). In fact, the PAM and CLARA clustering is done using `pamsam`.

Once the algorithm has found the HIPAM tree clustering it then visualises the clustering, unless `show.plot=FALSE`. If there are only two clusters in the tree, then all the data are plotted in a two-dimensional Sammon plot using their cluster labels as identifiers. If the tree contains more than two clusters, then a Sammon plot of the medoids is plotted with lines drawn indicating the hierarchy, unless any of the medoids of an ancestor cluster is also the medoid of one or more of its descendant clusters in the tree. In this case a dendrogram is drawn, where a one dimensional Sammon map is used at each level of the tree to put the most similar clusters near each other. See `plot.tree` for more details on plotting HIPAM objects.

More information on HIPAM can be found in Wit and McClure (2004).

## Value

A list containing: a vector of the objects' cluster labels (`clustering`), where the clusters are from the final clustering (i.e. the clusters without descendants); the `asw` of the final clustering; the number of levels (`n.levels`) in the tree; the cluster medoids of all of the clusters in the tree; the activity status (`active`) of each cluster (these should all be `FALSE`);

the `development` matrix of the tree (this contains one row for each of the final clusters and indicates the ancestors of the final clusters at each levels of the tree in its columns); the number of clusters (`num.of.clusters`) in the final clustering; a character string indicating the dissimilarity measure used (`metric`).

## Note

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

L. Kaufman and P. J. Rousseeuw (1990) "Finding Groups in Data" Chichester: Wiley.

K. S. Pollard and M. L. van der Laan (2002) "Resampling based methods" Working Paper 121, Berkeley Division of Biostatistics, UC Berkeley.

## See Also

`pam`, `clara`, `sammon`, `twoway.pam`, `plot.tree`

## Examples

```
data(skin)
# A HIPAM clustering of the first 400 genes in skin dataset:
skin.hp400 <- hipam(t(skin$dat[,1:400]))
skin.hp400$clustering
tabulate(skin.hp400$clustering)
# (compare these results with PAMSAM (see pamsam examples))
```

---

`impute.missing`            *Impute missing data*

---

## Description

This function imputes missing values using  $k$ -nearest neighbours (k-NN).

## Usage

```
impute.missing(dat, k = 10, dist.obj=NA, metric="manhattan",method="knn.wt")
```

## Arguments

<code>dat</code>	The data matrix whose missing values are to be imputed. Its rows should be objects and its columns variables. For microarray data, samples should be in the rows and genes in the columns. Missing values should be coded as <code>NA</code> .
<code>k</code>	Number of nearest neighbours to be used in the k-NN; this should be an integer greater than zero. The default value is 10.
<code>dist.obj</code>	The distance object between rows in <code>dat</code> can be supplied, if it is available, using this argument.
<code>metric</code>	Character string specifying the distance measure to be used. Has to be one of the ones accepted by the <code>dist</code> function. See <code>dist</code> for more details.
<code>method</code>	Character string specifying the method used for imputation. Options include <code>"knn"</code> (ordinary k-NN imputation) and <code>"knn.wt"</code> (weighted k-NN imputation; the default). See Details.

## Details

This function imputes missing data using k-nearest neighbours. Having specified `k`, the number of nearest neighbours to use, a missing value in a row is replaced by the average of the values for that column in the `k` most similar rows. The (dis)similarity between rows is defined using the `metric`; the R function `dist` is used to calculate the dissimilarities. Manhattan distance is a good choice for `metric` as it is robust to outliers.

If `method="knn.wt"`, then the average taken of the `k` nearest neighbours is a weighted one. The weights used are the inverse of the dissimilarities.

The default value for the number of nearest neighbours to use in the imputation is 10. If there are only a few rows in `dat`, then a lower value for `k` should be considered. Generally, the imputation will work better, the more rows there and particularly if there are replicates amongst the rows.

## Value

The function returns a matrix or data frame (depending on what `dat` is) that is the same as `dat`, except that the missing values have been replaced by the values imputed in the function.

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

`dist`

## Examples

---

kcv	<i>[SUBFUNCTION] A k-fold cross validation for step.lda and step.knn</i>
-----	--

---

## Description

This is a sub-function for step.lda and step.knn. It conducts a k-fold cross-validation error for the stepwise procedure (search.step.lda or search.step.knn) used to create the model found in step.lda or step.knn.

## Usage

```
kcv(x, cl, fold, type=c("step.lda","step.knn"), method, k.vec=c(1,15,2), l=0)
```

## Arguments

x	A matrix of explanatory variables used in creating the model.
cl	A factor specifying the class of each observation in x.
fold	The number of folds to be used in the k-fold cross validation.
type	Character string specifying whether an LDA ("step.lda") or KNN ("step.knn") stepwise search model is being used.
method	Type of LDA to be conducted. See lda in MASS library for details.
k.vec	Vector of the number of nearest neighbours (k) to search through.
l	Minimum vote for definite decision, otherwise 'doubt' (see knn in class package for details)

## Details

See step.lda or step.knn

## Value

An list containing: the k-fold cross-validation estimate of the class of the objects in x (est.cl); details of the k-fold cross validation error of the procedure (cross.validation).

## Author(s)

J. McClure and E. Wit

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

step.lda, step.knn

---

<code>lasso.cor</code>	<i>Plotting correlations between predictors in a LASSO fit and those excluded</i>
------------------------	---

---

## Description

Plots all correlations above or equal to a cut-off between predictors in a LASSO fit and those excluded from the fit. These correlations are also printed out. This is an exploratory tool that can be useful for seeing which predictors have been excluded because of their correlation with one of the included predictors.

## Usage

```
lasso.cor(plot.lasso.obj, cor.cutoff = 0.5, predictors = NULL,  
          cor.mat = NULL, col = NULL, max.per.plot = 10,  
          max.plots = 3, absv = TRUE)
```

## Arguments

<code>plot.lasso.obj</code>	A <code>plot.lasso</code> object, where the tuning parameter (or <code>chosen.var</code> ) has been set to the desired value. See details.
<code>cor.cutoff</code>	Correlation cut-off. Only correlations whose value is above or equal to this cut-off will be displayed. If unspecified, this is taken to be 0.5. Note that absolute correlations will be considered if <code>absv</code> is <code>TRUE</code> .
<code>predictors</code>	The original data matrix used for the LASSO, containing all the predictors (both those included in the LASSO fit and the excluded ones). Predictors should be the columns and observations the rows. This or <code>cor.mat</code> must be specified.
<code>cor.mat</code>	Matrix containing the correlations between all of the predictors in the original matrix. This or <code>predictors</code> must be specified.
<code>col</code>	Vector specifying the colour of each predictor. Can be the length of number of predictors or a single colour.
<code>max.per.plot</code>	Maximum number of displays per plot. See details.
<code>max.plots</code>	Maximum number of plots that can be placed into the Graphics Device's figure region. See details.
<code>absv</code>	Logical value, specifying whether or not the absolute values of correlations are considered. By default, absolute values are considered.

## Details

A `plot.lasso` object needs to be supplied for this to work. An the appropriate value for the tuning parameter in the `plot.lasso` command is needed in order that the particular LASSO fit of interest is the one that `lasso.cor` considers. Only correlations between predictors in the LASSO fit and excluded predictors are considered—correlations between included predictor or between excluded predictors are not considered.

If none of the correlations between a predictor in the LASSO fit and the excluded predictors are greater than the cut-off, then that predictor will not be included in the figure.

Note that `max.per.plot` and `max.plots` may need to be adjusted from their default values if the numbers of predictors in the LASSO is greater than the default `max.per.plot*max.plots`. However, having more predictors than this value may lead to the plot becoming overly crowded.

Specifying `chosen.var` in `plot.lasso` can allow any subset of predictors to be chosen as the subset to be considered in `lasso.cor`. However, non-LASSO subsets for visualising correlations are better investigated using the `cor.visual` function that `lasso.cor` calls.

## Value

A list containing one element for of the predictors included in the LASSO. Each of these elements is vector of correlations equal to or above the cut-off.

## Author(s)

J. McClure and E. Wit

## See Also

`plot.lasso`, `cor.visual`

## Examples

```
library(lars)
x <- matrix(rnorm(600), ncol=30)
y <- x[,1] + x[,2] + rnorm(20)
xy.las <- lars(x,y,"lasso")
xy.plot.obj <- plot.lasso(xy.las, tuning=0.4, title="Example of LASSO plot")
lasso.cor(xy.plot.obj, cor.cutoff=0.3, predictors=x)
# Try different values for cor.cutoff
```

---

`ma.plots`

*Microarray Plots*

---

## Description

These functions create an image of microarray expression (`ma.image`, `ma.bw`), a perspective plot (`ma.persp`) or a contour plot (`ma.contour`). In the case of `ma.image` and `ma.bw`, the former creates a colour image and the latter a greyscale one.

## Usage

```
ma.image(mat,xlab="",ylab="",col=terrain.colors(10),
         title="Spatial distribution of expression values across microarray",...)
ma.bw(mat,xlab="",ylab="",col=grey(0:25/25),
      title="Spatial distribution of expression values across microarray",...)
ma.persp(mat,theta=-45,phi=40,xlab="",ylab="",zlab="expression",shade=0.2,
         title="Spatial distribution of expression values across microarray",...)
ma.contour(mat,xlab="",ylab="",
          title="Spatial distribution of expression values across microarray",...)
```

## Arguments

<code>mat</code>	Matrix containing the data to be plotted. The data can be given in two ways. In the first, an entry's row and column defines its x and y positions on the microarray. In the second, there are three columns in the data; the first two give the x and y positions (respectively) of each spot on the array, whose normalized value is given in the third column.
<code>xlab</code>	Character string giving the name for the x-axis.
<code>ylab</code>	Character string giving the name for the y-axis.
<code>col</code>	Vector of colour range to be used for image plots.
<code>title</code>	String containing the title for the plot.
<code>theta, phi</code>	Angles (in degrees) defining the viewing direction in perspective plots; 'theta' gives the azimuthal direction and 'phi' the colatitude. See <code>persp</code> for more details.
<code>zlab</code>	Character string giving the name for the z-axis in perspective plots.
<code>shade</code>	Shading strength for "shadows" that would be created by a light source in perspective plots. See <code>persp</code> for more details.
<code>...</code>	Additional graphical parameters (see <code>image</code> , <code>persp</code> , <code>contour</code> , and <code>par</code> ).

## Details

Note that these functions are very similar to the `image`, `persp` and `contour` functions in R. The main thing these functions do is convert typical microarray storage formats into objects of the form that the `image`, `persp` and `contour` require.

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

`image`, `persp`, `contour`

## Examples

```
data(skin)
# The fourth row of skin$dat represents Cy5-dyed cancerous tissue
# from the second array:
list(array=skin$arrays[4],condition=skin$conditions[4],dye=skin$dyes[4])
# Different microarray plots for this data:
ma.image(cbind(skin$x,skin$y,skin$dat[4,]))
ma.bw(cbind(skin$x,skin$y,skin$dat[4,]))
ma.persp(cbind(skin$x,skin$y,skin$dat[4,]))
ma.contour(cbind(skin$x,skin$y,skin$dat[4,]))
```

---

mammary

*DATA: Mouse mammary tissue microarray experiment*

---

## Description

This data set details a large time course Affymetrix microarray experiment carried out using mammary tissue from female mice across different developmental stages. Eighteen different time points were used covering 4 different developmental stages (virgin, pregnant, lactation and involution), with three biological replicates used at each of these times. In total mammary tissue from 54 mice was used.

It was conducted by Prof. Barry Gusterson's group at the University of Glasgow's Division of Cancer Science and Molecular Pathology. Many thanks to Prof. Gusterson and Dr. Torsten Stein for allowing us to include this data.

## Usage

```
data(mammary)
```

## Format

A list containing:

**dat** A 54 by 12488 matrix containing the normalized expression values of 12488 probe sets, representing around 8,600 genes over the 54 Affymetrix arrays, with each row representing the expression values of one of the arrays.

**arrays** A vector of length 54 identifying the array that each row of **dat** came from. Each identifier indicates first the stage (**v** for virgin, **p** for pregnant, **l** for lactation and **i** for involution) and the time point within that, followed after the full stop by the replicate number. Thus "i20.1" indicates that the array is the first replicate for day 20 of the involution stage. Within the virgin stage, the time points after its stage letter (**v**) refer to age in weeks from birth; for the other stages time points refer to the time in days since the beginning of that stage.

**stages** A vector of length 54 identifying the stages ("**virgin**", "**pregnant**", "**lactation**" or "**involution**") that the tissue used for each array comes from that the data in each row of **dat** represent.

**age** The age in days of the mice. Note that this is only an indicator since it assumes that the start of pregnancy, lactation and involution all happen exactly two days after the oldest age in their preceding stage. It also assumes that the 6 week old mice are exactly 42 days old, etc.

## Details

Four main stages of development were investigated: first the virgin stage, where mice at 6, 10 and 12 weeks of age were used; then pregnancy, where tissue from mice at 1, 2, 3, 8.5, 12.5, 14.5 and 17.5 days after conception was used; next lactation, where tissue from mice at 1, 3, and 7 days after giving birth was used; finally, involution, where tissue from mice at 1, 2, 3, 4 and 20 days after lactation stops was used. Three mice are used at each of these 18 time points, leading to 54 being used in total. Each animal's mammary tissue is used for the one array.

Note that the half days for the pregnant mice over 8 days old is due to the exact time of conception not being known, but it being somewhere between 8 and 9 days for 8.5 etc.

Note also that involution is the period of time after lactation stops where the milk secreting cells die.

For more details see Wit and McClure (2004) and Stein et al. (2004).

## Source

Prof. Barry Gusterson's and Dr. Torsten Stein, University of Glasgow's Division of Cancer Science and Molecular Pathology.

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

T. Stein et al. (2004) "Involution of the mouse mammary gland is associated with ..." Breast Cancer Research, 6, R75–R91

---

normalization subfunctions

*Subfunctions called from the normalization functions or their subfunctions*

---

## Description

These are subfunctions needed for the normalization functions, but are not directly used.

## Author(s)

E. Wit and J. McClure

## See Also

ma.image, ma.bw, ma.persp, ma.contour, all.norm, spat.norm, bkg.norm, dye.norm, cond.norm

---

od subfunctions

*Subfunctions called from od or its subfunctions*

---

### Description

These are subfunctions needed for od, but are not directly used.

### Author(s)

E. Wit and J. McClure

### See Also

od

---

od

*Optimal design for 2-channel microarray experiments*

---

### Description

Function to search for and display A- and D-optimal designs for 2-channel microarray experiments. Simulated annealing is used to find the best design, unless a loop design is required, in which case the optimal (interwoven) loop design is found.

### Usage

```
od(nt, ns, optimality="A", method="SA", contrasts=TRUE,
   dye=TRUE, extend.design=NULL, weight=1, n.iter=1000,
   ok.final.temp=0.0001, initial.temp=10, max.temp.step=0.99,
   initial=NULL, plot.it=TRUE, show.off=FALSE)
```

### Arguments

nt	number of treatment levels (e.g. number of time points or conditions)
ns	number of slides available
optimality	type of optimality, i.e. "A" (A-optimality) or "D" (D-optimality). A-optimality minimizes $\text{Trace}((X^t X)^{-1})$ , which corresponds to minimum average variance of the parameter estimates. D-optimality minimizes $\det(X^t X)^{-1}$ , which corresponds to minimum generalized variance of the parameter estimates.
method	Method for searching for an optimal design. "SA" uses simulated annealing. "loop" finds the best interwoven loop design. Note that the option "loop" will ignore the dye argument, because interwoven loop designs automatically balance dye assignments. Specifying "loop" will also ignore the extend.design argument because, given a particular design, it is typically impossible to extend that design to an interwoven loop design, barring some special situations.

<code>contrasts</code>	only used for A-optimality, where the default is <code>TRUE</code> . It defines whether the objective function, $\text{Trace}((X^t X)^{-1})$ , is for the contrasts, or for the arbitrary canonical parameterization ( <code>set-first-to-zero</code> ) chosen by us. This is not an issue for D-optimality, which is independent from the particularly chosen parameterization.
<code>dye</code>	Whether or not a term for the dye effect term should be included in the design matrix as an extra parameter. The default value is <code>TRUE</code> .
<code>extend.design</code>	An optional argument, requiring a design matrix or od-object, from which the design matrix is extracted. If this is specified, then the od function is interpreted as extending the <code>extend.design</code> matrix as optimally as possible to include another <code>nt</code> new conditions using another <code>ns</code> new arrays.
<code>weight</code>	The weight parameter, <code>w</code> , in weighted A-optimal design for time course experiments. The default value of 1 is equivalent to the unweighted approach. The parameter defines the weight between timepoints $i$ and $j$ as $w^{(abs(i-j))}$ . A value less than 1 will favour nearby comparisons, whereas a value greater than 1 will favour long distance comparisons. Ignored when D-optimality is used.
<code>n.iter</code>	Number of iterations of the simulated annealing method.
<code>ok.final.temp</code>	An important optimization parameter. Setting this parameter closer to zero, the final result will be more sensitive to the true optimum. For example, if the final temperature is 0.001, then a solution that varies 0.1 be accepted with probability of 0.37 (since $0.999^{(1/0.001)} = 0.37$ ). This is particularly crucial for large designs. The programme tries to reach " <code>ok.final.temp</code> ", within the constraints set by the <code>n.iter</code> , <code>initial.temp</code> and <code>max.temp.step</code> arguments.
<code>initial.temp</code>	The starting temperature for the simulated annealing. See Wit and McClure (2004) for details.
<code>max.temp.step</code>	In order for Simulated Annealing to converge to the true maximum, Kirkpatrick et al. (1983) showed that the temperature should cool via a logarithm. This can be very slow. We instead follow common practice to cool the temperature geometrically. The parameter <code>max.temp.step</code> ensures that the multiplicative cooling factor is not smaller than that. If <code>n.iter</code> is too small, the preferred final temperature ( <code>ok.final.temp</code> ) may not be reached.
<code>initial</code>	The starting design matrix for the algorithm. If specified this should be a matrix or an od object. However, the algorithm does not require that a starting matrix is specified. Only used when <code>method="SA"</code> .
<code>plot.it</code>	If <code>TRUE</code> (the default) it produces a plot of the design found by od using the function <code>plot.od</code> .
<code>show.off</code>	If <code>TRUE</code> it shows plots of all the intermediate designs found as the algorithm iterates. In this case the function will take longer to run in order that each iteration's plot is on screen for long enough to be seen. The default value for <code>show.off</code> is <code>FALSE</code> .

## Details

Given a number of treatments/timepoints (`nt`) and slides (`ns`), the algorithm searches for an A-optimal or D-optimal design (see `optimality`). The search is done using simulated annealing or by restricting the search to interwoven loop designs (see `method`). A plot of the design can also be given (see `plot.it`) using the `plot.od` function. A dye effect term can be included (see `dye`).

For A-optimal designs, the objective can be defined by contrasts or a set-first-to-zero parameterization (see `contrasts`). Weighted A-optimal design can also be done (see `weight`).

A number of different parameters control the simulated annealing and these can be altered to improve the search if necessary (see `n.iter`, `ok.final.temp`, `initial.temp` and `max.temp.step`). Also in simulated annealing, a starting design can be specified (see `initial`). If extra slides are available to extend an existing design, then the `od` function can search for the optimal way of doing this (see `extend.design`) when simulated annealing is used.

## Value

An `od` object and plot of the final chosen design (see `plot.it`). Intermediate simulated annealing designs can also be plotted as the function runs (see `show.off`).

The `od` object is a list containing: the score (lower=better) of the final design (`score`); the design matrix (`design`); . It also contains a number of the arguments of `od`: the type of optimality (`optimality`); whether or not the design is for contrasts (`for.contrasts`); whether a dye effect term is included (`estimating.dye.effect`); whether it was a reference design (`reference.design` (always `FALSE`) at present); the weight parameter (`weight`).

When the simulated annealing method is used, the object also includes: The fraction of proposed designs accepted at their iteration (`f.accepted`); the initial temperature parameter (`initial.temp`); the number of iterations (`n.iterations`); the factor by which the temperature cools (`temp.step`); the final temperature (`final.temp`); the pre-extension design, if slides are being added to an existing design (`extending.the.design`).

When an (interwoven) loop design is searched for, the object also includes a vector containing the different jump sizes of each loop (`jumps`).

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi (1983) Optimization by simulated annealing, "Science" 220(4598), 671–80.

## See Also

`plot.od`

## Examples

```
# Default version (uses simulated annealing and A-optimality):
od1 <- od(nt=5,ns=15)
# Same as od1, but this time searches for interwoven loop design:
od2 <- od(nt=5,ns=15, method="loop")
# Same as od2, but this time uses D-optimality:
od3 <- od(nt=5,ns=15, optimality="D", method="loop")
# Extension to od1 with 2 more treatments and 6 slides:
od5 <- od(nt=2,ns=6,extend.design=od1)
# Same as od1, but using D-optimality:
od6 <- od(nt=5,ns=15, optimality="D")
```

---

op

*Optimal Pooling of RNA samples for Microarray Experiments*

---

## Description

Function to calculate the optimal pool size as well as the number of microarrays for a particular experiment to attain a preset level of confidence in detecting a certain preset fold-change level.

## Usage

```
op(bio.sd=0.6, measure.sd=0.6, array.cost = 700,
   sample.cost= 50, fold=1.5, alpha=0.1)
```

## Arguments

<code>bio.sd</code>	biological variation, i.e. the standard deviation in gene expression of a sample in the population of interest.
<code>measure.sd</code>	technical variation, i.e. the standard deviation that would be observed if the same sample would be applied to another array.
<code>array.cost</code>	Cost of a single array (including dyes).
<code>sample.cost</code>	Cost of extracting a single RNA sample (excluding labelling).
<code>fold</code>	Fold change that one would like to be able to detect.
<code>alpha</code>	$1-\alpha$ represents the confidence level at which you would like to be able to detect the fold change.

## Details

We consider a biological system with standard deviation of `bio.sd` and a microarray with technical standard deviation of `measure.sd`, in which an array including dyes costs `array.cost` units and preparing a hybridization sample costs `sample.cost` units. In order to be able to detect a `fold` change with  $1-\alpha$  confidence, the function will calculate the optimal number of arrays and the optimal number of samples per pool in each hybridization sample for each array.

It is important to note that the dyes and other labelling costs should be included in the array costs, rather than in the sample costs.

The default values for the biological and technical variation were derived from a mouse study described in Churchill (2002).

### Value

A list containing the optimal number of arrays (`num.arrays`) and the optimal number of RNA samples per hybridization sample on each array (`num.samples.per.pool`).

### Author(s)

E. Wit and J. McClure

### References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

G.A. Churchill (2002) Fundamentals of experimental design for cDNA microarrays, "Nature Genetics", 32, pp.490-5.

### See Also

od

### Examples

```
# Results with the default values of the arguments:
op()
# Same as before, but with smaller biological variation.
# Note that pooling RNA is no longer worthwhile:
op(bio.sd=0.15)
# Same as before, but with larger biological variation,
# which would be relevant for more complex biological
# systems, such as humans. Note that pooling becomes even
# more valuable:
op(bio.sd=1)
```

---

pamsam

*PAMSAM partition clustering and visualization*

---

### Description

This is a clustering method that is based on partitioning around medoids (PAM), and which uses average silhouette width (asw) to determine the number of clusters. A number of further partitions and collapses of clusters are attempted after the initial clustering to try to further improve the clustering. The chosen partition can then be visualized using Sammon mapping.

## Usage

```
pamsam(x, k=NA, metric="manhattan", show.plot=TRUE, maxclust=50,
       further.maxclust=20, exhaustive=FALSE, cluster.rows=TRUE,
       maxpam=800, clara.samples=8, clara.sampsiz=150, x.inflation=1.2,
       response.name="Response", var.name="Index",
       axes.col="darkgray", line.col="darkgray", number.col="black",
       asw.sampsiz=200, asw.samp.type="ceiling",
       max.check=20, total.check=100)
```

## Arguments

<code>x</code>	Data to be clustered. Should be given as a matrix or data.frame, with its rows representing the objects to be clustered and the columns the variables. If given the other way round, then <code>cluster.rows</code> must be changed to <code>FALSE</code> .
<code>k</code>	Leave this unspecified ( <code>NA</code> ) if you wish <code>pamsam</code> to search for the best number of clusters. Give <code>k</code> a number if you wish that specific number of clusters to be created.
<code>metric</code>	Character string specifying the distance measure to be used. Can be a power distance (" <code>manhattan</code> ", " <code>euclidean</code> ", " <code>cubic</code> ", " <code>quartic</code> "), 1 minus the correlation (" <code>correlation</code> "), or 1 minus the absolute correlation " <code>abscorr</code> ".
<code>show.plot</code>	Logical indicating whether a Sammon plot should be given of the chosen clustering's medoids. The default is <code>TRUE</code> .
<code>maxclust</code>	The maximum number of clusters considered in the initial search. The default is 50.
<code>further.maxclust</code>	The maximum number of clusters considered when further splitting of the clusters is conducted. The default is 20.
<code>exhaustive</code>	If <code>TRUE</code> , then then all values for <code>k</code> between 2 and <code>maxclust</code> will be considered. If <code>FALSE</code> (the default), then a grid search for <code>k</code> is done.
<code>cluster.rows</code>	Logical indicating whether or not the objects to be clustered are given in the rows (the default) or not.
<code>maxpam</code>	The maximum number of objects for which the clustering algorithm will use the PAM method. Above this value, the sampling based CLARA method is used. The value <code>maxpam</code> should take depends on the speed of the computer used. Experiment with different values if clustering many objects.
<code>clara.samples</code>	The number of samples used in the CLARA method (see <code>samples</code> in <code>clara</code> for more details).
<code>clara.sampsiz</code>	The size of the samples used in the CLARA method (see <code>sampsiz</code> in <code>clara</code> for more details).
<code>x.inflation</code>	The proportion by which the x-axis of the Sammon plot is increased in order to display the graph at each medoid.

<code>response.name</code>	Character string to be added as the y-axis label of the medoids' graphs, indicating the units of all of the variables.
<code>var.name</code>	Character string to be added as the x-axis label of the medoids' graphs.
<code>axes.col</code>	The colour of the axes of the graph for each medoid. The default is dark grey.
<code>line.col</code>	The colour of the line on the graph for each medoid. The default is dark grey.
<code>number.col</code>	The colour of the cluster labels for each medoid. The default is black.
<code>asw.sampsize</code>	Sample size used in the estimation of asw when the number of objects is larger than <code>asw.sampsize</code> .
<code>asw.samp.type</code>	String indicating the type of sampling method to be used when sampling is needed to calculate average silhouette width. Stratified sampling is generally recommended, to ensure that each cluster is well represented in the calculation. Stratified sampling methods can be altered according to how the sample size for each cluster is determined from the (probably) non-integer value that proportionate frequencies would indicate. This can be <code>"round"</code> , where values are rounded to the nearest whole number; <code>"ceiling"</code> , where nearest larger integer is used (the default method). Simple random sampling ( <code>"srs"</code> ) can also be used, as can equal sampling ( <code>"equal"</code> ), where each cluster is sampled the same number times (and sampling is done by replacement).
<code>max.check</code>	The maximum number of partitions and collapses that should be tried without any improvement in asw. The default value is 20.
<code>total.check</code>	The maximum number of partitions and collapses that should be tried. The default value is 100.

## Details

The clustering algorithm searches for the clustering that has the highest average silhouette width (asw), unless the number of clusters,  $k$ , has been specified, in which case, a PAM (or CLARA) clustering is done for the given number of clusters.

The algorithm searches in two stages, using. In the first a grid or exhaustive (see `exhaustive`) search is conducted using PAM (or CLARA) with values of  $k$  between 2 and `maxclust`. Once this initial clustering has been done, a number (with a maximum of `max.check`) of further partitioning or collapsing operations are attempted to improve the asw. Each time a cluster is chosen at random and further partitioning or collapsing of that cluster is attempted. If the operation tried does not succeed for the chosen cluster, then the other operation is also tried.

In a further partitioning step, the chosen cluster is itself clustered using the same method that the initial clustering did in the first stage (this time the maximum number of clusters is decided set using `further.maxclust`). If splitting the chosen cluster in this way increases asw, then the step is accepted and a new clustering is created with the objects in the chosen cluster now split into the clusters found in the partitioning step.

In a collapsing step, the chosen cluster is joined to another cluster. Each of the other clusters are checked as a potential collapse partner for the chosen cluster. If the best of these possible collapses improves the overall asw, then then the collapse is accepted.

CLARA is used instead of PAM when the number of objects to be clustered is greater than `maxpam`. CLARA is simply a sampling version of PAM that finds the medoids by sampling the objects a `clara.samples` number of times. Once medoids are found, all the objects are assigned to their nearest medoid's group. See Kaufman and Rousseeuw (1990), or the `pam` and `clara` functions in the `cluster` library for more details of PAM and CLARA.

Once the final clustering has been found, PAMSAM then visualizes the medoids of the clusters. The medoids are plotted in a two-dimensional Sammon mapping, indicating their approximate similarity to each other. Each medoid is represented by a small graph that shows the medoids' value across the different variables. This can be particularly useful where the variables represent different time points, or otherwise ordered conditions. For more details on Sammon mapping see `sammon` in the `MASS` library.

Information on PAMSAM can be found in Wit and McClure (2004).

### Value

A list containing: the cluster `medoids`; a vector of the objects' cluster labels (`clustering`); the `asw` of the clustering; the number of clusters (`num.of.clusters`); a list containing information on the progress of the clustering algorithm (`info`); another list containing information on the `asw` and `sesw` (standard error of the silhouette widths) profiles at each stage of the search (`profiles`); a character string indicating the dissimilarity measure used (`metric`).

### Note

### Author(s)

E. Wit and J. McClure

### References

- E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.
- L. Kaufman and P. J. Rousseeuw (1990) "Finding Groups in Data" Chichester: Wiley.

### See Also

`pam`, `clara`, `sammon`, `hipam`, `twoway.pam`

### Examples

```
data(skin)
# A PAMSAM clustering of the first 400 genes in skin dataset:
skin.ps400 <- pamsam(t(skin$dat[,1:400]))
skin.ps400$clustering
tabulate(skin.ps400$clustering)
# (compare these results with HIPAM (see hipam examples))
```

**Description**

Produces a plot of the LASSO model fitted. The standardized coefficients of the predictors are given at each stage of the LASSO. It is based on Hastie's more general plot.lars function in the lars library. However, it allows a subset of the predictors to be highlighted, in particular predictors included in LASSO at any value of the norm fraction tuning parameter.

**Usage**

```
plot.lasso(x, tuning, norm.prop.plot = 1, end.tuning, chosen.var,
           breaks = FALSE, omit.zeros = TRUE, eps = 1e-10, title,
           ...)
```

**Arguments**

x	LASSO object produced by the lars function.
tuning	Scalar specifying proportion of the norm fraction. Variables whose coefficients are non-zero in LASSO for the tuning value will be highlighted in the plot unless chosen.var is specified.
norm.prop.plot	Specifies the limits of the horizontal axis on the plot. A scalar defining the largest proportion of the norm fraction to be displayed.
end.tuning	Logical value. If 'TRUE', then the horizontal axis only extends as far as tuning parameter chosen. The default value is 'TRUE' when a tuning value is given and is 'FALSE' otherwise.
chosen.var	Vector defining variables of interest to be highlighted in the plot. Can be: a logical vector; a numeric vector specifying the column numbers of the chosen variables; a character vector with names of the variables of interest if the data given to the lars function includes this in the column names. If specified it overrides the tuning argument.
breaks	If 'TRUE', then vertical lines are drawn at each break point in the piecewise linear coefficient paths.
omit.zeros	If 'TRUE', then the variables never included at any stage of the LASSO are not plotted.
eps	The cut-off for deciding whether a variable should be included when omit.zero=TRUE
title	Character string defining title for the coefficients' plot.
...	Other plotting arguments.

**Details**

The coefficient values plotted have been normalized. If a subset of variables is selected using chosen.var or tuning, then these variables will be highlighted in the plot.

## Value

If a subset of variables is selected using `chosen.var` or `tuning`, then these variables will be printed, along with their un-normalized coefficient values at the right-hand end of the plot (or at the value for the tuning proportion, if that is given) and their column numbers.

## Author(s)

J. McClure, E. Wit and T. Hastie

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

## See Also

`lars`, `cv.lars` and `plot.lars` (all in `lars` library)

## Examples

```
x <- matrix(rnorm(1000), ncol=50)
y <- x[,1] + x[,2] + rnorm(20)
xy.las <- lars(x,y,"lasso")
cv.lars(x,y,type="lasso") # Allows appropriate tuning parameter fraction to be selected
plot.lasso(xy.las, tuning=0.4, title="Example of LASSO plot")
# Alter tuning according to cv.lars.
# Note that plot.lasso must be specified; plot will create a plot.lars figure.
```

---

`plot.od`

*Plotting function for od*

---

## Description

This function plots the 2-channel microarray experiments designs found by the `od` function. It plots the treatments/timepoints in either a circular formation or a in a Sammon or multidimensional scaling (mds) method. The treatments/timpoints are represented as circular nodes; each array being shown as arrows joining the two treatments/timpoints to be placed on that array.

## Usage

```
plot.od(x, method="circle", magic=1, search.details=TRUE,
        main=NULL, sub=NA, circle.col="darkgrey",
        optimality="A", contrasts=TRUE,weight=1,dye=TRUE, ...)
```

## Arguments

<code>x</code>	an 'od' object; the result of 'od'. It can also be a design matrix (a matrix object); if so, then <code>optimality</code> , <code>contrasts</code> , <code>weight</code> and <code>dye</code> should be specified; otherwise these are ignored and the information is taken from <code>x</code> .
<code>method</code>	The plotting method to be used. The default is "circle" which plots the treatments in a circle. The two other methods, "sammon" and "mds", using dimension reduction techniques to place treatments connected to each other near one another in the plot; see details for more information.
<code>magic</code>	A constant ( $>0$ ) that can be added to $X^t X$ to alter the appearance of Sammon or mds plots; see method and details for more information. It's default value is 1.
<code>search.details</code>	If TRUE (the default), then simulated annealing search details (number of iterations and acceptance rate) will be added on the plot.
<code>main</code>	Add an appropriate main title (as a character string) to the plot.
<code>sub</code>	Add an appropriate subtitle (as a character string) to the plot.
<code>circle.col</code>	Colour(s) for the nodes representing the treatments/timepoints. Specific colours for each treatment/timepoint can be specified using a vector of the length of the number of treatments/timepoints. The default is for all to be dark grey.
<code>optimality</code>	type of optimality, i.e. "A" (A-optimality) or "D" (D-optimality).
<code>contrasts</code>	only used for A-optimality, where the default is TRUE. It defines whether the objective function, $\text{Trace}((X^t X)^{-1})$ , is for the contrasts, or for the arbitrary canonical parameterization (set-first-to-zero) chosen by us. This is not an issue for D-optimality, which is independent from the particularly chosen parameterization.
<code>weight</code>	The weight parameter, $w$ , in weighted A-optimal design for time course experiments. The default value of 1 is equivalent to the unweighted approach. The parameter defines the weight between timepoints $i$ and $j$ as $w^{(abs(i-j))}$ . A value less than 1 will favour nearby comparisons, whereas a value greater than 1 will favour long distance comparisons. Ignored when D-optimality is used.
<code>dye</code>	Whether or not a term for the dye effect term should be included in the design matrix as an extra parameter. The default value is TRUE.
<code>...</code>	Further graphical parameters from 'par'.

## Details

The default 'method', "circle", plots treatments/timepoints as circular nodes that form a circle and which are connected to other nodes via arrows. Each arrow represents an array. The node at the start of the arrow is the treatment/timepoint to be hybridized with one particular dye (e.g. Cy 3); the node at the pointed end of the arrow is the treatment/timepoint to be hybridized with the other dye (e.g. Cy 5) on this array.

The thickness of the arrows indicates how many arrays connect the two relevant treatments/timepoints in that dye "direction".

When the "sammon" or "mds" 'method' is chosen, the nodes are arranged, so that treatments/timepoints that are most closely 'connected' to each other tend to appear nearest each other in the plot. A Sammon map ("sammon") or multidimensional scaling ("mds") plot of  $1/X'X$  are used to arrange the nodes, where  $X$  is the design matrix and  $1/X'X$  indicates that each element is inverted and not that the whole matrix is inverted. A 'magic' constant ( $<0$ ) is added to each element in  $X'X$  to alter the appearance of the plot. The default value is 1, but lower values can sometimes be more graphically pleasing.

### Author(s)

E. Wit and J. McClure

### References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

### See Also

od

### Examples

```
od1 <- od(nt=5,ns=15, plot.it=FALSE)
# Default "circular" method:
plot.od(od1)
# Sammon and multidimensional scaling versions:
plot.od(od1, method="sammon")
plot.od(od1, method="mds")
# Try different values for 'magic' alter the appearance of the plot.
# e.g.:
plot.od(od1, method="sammon", magic=0.6)
plot.od(od1, method="mds", magic=0.3)
# Give different colours for the treatments/timepoints:
plot.od(od1, circle.col=1:5)
```

---

plot.step.knn

*Sammon plot of stepwise KNN*

---

### Description

This function plots a step.knn object. It creates a Sammon plot of the observations labelled according to whether they are training or test cases and which group or predicted group they are in. For training cases where k-fold cross validation of the method is done, the k-fold classification accuracy of each prediction can be shown.

## Usage

```
plot.step.knn(x, train.col="darkgrey", test.col="black",  
             cv.col=c("springgreen2","red"), bw=FALSE, legend=2, ...)
```

## Arguments

<code>x</code>	The output of <code>step.knn</code> .
<code>train.col</code>	The colour for the training cases, unless cross-validation has been conducted (see <code>cv.col</code> ). It is "darkgrey" by default.
<code>test.col</code>	The colour for the test cases; "black" by default.
<code>cv.col</code>	Vector containing colours for the cross-validation prediction of the training cases, By default, these are a green colour for correct prediction and a red for incorrect prediction, unless <code>bw=TRUE</code> is chosen. If <code>bw=TRUE</code> , then both correct and incorrect predictions will be coloured "darkgrey".
<code>bw</code>	If TRUE then the plot produced will appear in black and white only. If FALSE, a colour plot will be produced. See <code>cv.col</code> .
<code>legend</code>	The corner of the plot in which the legend should appear. 1 means the bottom left, 2 the top left, 3 the top right and 4 the bottom right corner.
<code>...</code>	Other graphical parameters can be supplied.

## Details

A 2-dimensional Sammon plot of the training data in `x` will be shown (together the test data if any has been supplied). The objects in this plot will be identified by their classification group (or predicted classification in the case of test cases) and will be coloured according to whether they are training or test cases. In the case of k-fold cross validation, a cross will appear over incorrectly predicted training cases and a circle over each correctly predicted cases.

## Author(s)

J. McClure and E. Wit

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

## See Also

`step.knn`

## Examples

```
x <- matrix(rnorm(1000), ncol=50)  
x[,2] <- x[,2] + rep(c(-1,1),each=10)  
x[,12] <- x[,12] + rep(c(-1,1,1,-1),each=5)  
y <- rep(c("A","B","C","D"), each=5)
```

```

z <- matrix(rnorm(400),ncol=50)
z[,2] <- z[,2] + rep(c(-1,1),each=4)
z[,12] <- z[,12] + rep(c(-1,1,1,-1),each=2)
step.knn.obj <- step.knn(train=x, class=y, test=z, kcv=TRUE,
                        k.vec=seq(1,9,2), fold=5, plot.it=FALSE)
plot.step.knn(x=step.knn.obj, legend=2)
# Try different values for legend if it covers some of the data.

```

---

plot.step.lda                      *Sammon plot of stepwise LDA*

---

## Description

This function plots a step.lda object. It creates a Sammon plot of the observations labelled according to whether they are training or test cases and which group or predicted group they are in. For training cases where k-fold cross validation of the method is done, the k-fold classification accuracy of each prediction can be shown.

## Usage

```

plot.step.lda(x, train.col="darkgrey", test.col="black",
             cv.col=c("springgreen2","red"), bw=FALSE, legend=2, ...)

```

## Arguments

x	The output of step.lda.
train.col	The colour for the training cases, unless cross-validation has been conducted (see cv.col). It is "darkgrey" by default.
test.col	The colour for the test cases; "black" by default.
cv.col	Vector containing colours for the cross-validation prediction of the training cases, By default, these are a green colour for correct prediction and a red for incorrect prediction, unless bw=TRUE is chosen. If bw=TRUE, then both correct and incorrect predictions will be coloured "darkgrey".
bw	If TRUE then the plot produced will appear in black and white only. If FALSE, a colour plot will be produced. See cv.col.
legend	The corner of the plot in which the legend should appear. 1 means the bottom left, 2 the top left, 3 the top right and 4 the bottom right corner.
...	Other graphical parameters can be supplied.

## Details

A 2-dimensional Sammon plot of the training data in x will be shown (together the test data if any has been supplied). The objects in this plot will be identified by their classification group (or predicted classification in the case of test cases) and will be coloured according to whether they are training or test cases. In the case of k-fold cross validation, a cross will appear over incorrectly predicted training cases and a circle over each correctly predicted cases.

## Author(s)

J. McClure and E. Wit

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

## See Also

step.lda

## Examples

```
x <- matrix(rnorm(1000), ncol=50)
x[,2] <- x[,2] + rep(c(-1,1),each=10)
x[,12] <- x[,12] + rep(c(-1,1,1,-1),each=5)
y <- rep(c("A","B","C","D"), each=5)
z <- matrix(rnorm(400),ncol=50)
z[,2] <- z[,2] + rep(c(-1,1),each=4)
z[,12] <- z[,12] + rep(c(-1,1,1,-1),each=2)
step.lda.obj <- step.lda(train=x, class=y, test=z, kcv=TRUE, method="mle",
                        fold=5, plot.it=FALSE)
plot.step.lda(x=step.lda.obj, legend=2)
# Try different values for legend if it covers some of the data.
```

---

plot.tree

*HIPAM tree plotting function*

---

## Description

This function plots a given HIPAM tree object as a traditional dendrogram, or as a two-dimensional Sammon map. In the latter form it can be either of all the objects or of all the cluster medoids, with branches linking each cluster to its parent.

## Usage

```
plot.tree(x, method="2d", reorder=TRUE,
          title="Proposed Hierarchical PAM Clustering", data=NULL)
```

## Arguments

x	The HIPAM object to be plotted.
method	Character string indicating the type of plot to be shown. If "2d" (the default), then a two-dimensional Sammon plot of the clusters is drawn, with branches connecting each of the clusters to the root cluster (which is labelled 'R'). If "tree", then a typical hierarchical tree. If "all", then a two-dimensional Sammon plot of all the data is shown with each object labelled by its cluster; in this case, the original <code>data</code> must be given as well.

<code>reorder</code>	When <code>code="tree"</code> , this indicates whether the order in which the clusters appear in each level of the tree is decided using a one-dimensional Sammon mapping of that level's cluster medoids. The default ( <code>TRUE</code> ) does this, but this can cause the branches to appear 'tangled' sometimes; if this occurs change it to <code>reorder=FALSE</code> .
<code>title</code>	Character string giving the title for the plot.
<code>data</code>	If <code>method="all"</code> , then the original data need to be given in this argument.

## Details

The function plots the result of the `hipam` function as: a traditional dendrogram (`method="tree"`), a two-dimensional Sammon plot of all the cluster medoids linked by branches to show the underlying tree structure (`method="2d"`, the default); or a a two-dimensional Sammon plot of all the data labelled their (terminal) cluster.

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

`hipam`, `sammon`

## Examples

```
data(skin)
# A HIPAM clustering of the first 400 genes in skin dataset:
skin.dat400 <- t(skin$dat[,1:400])
skin.hp400 <- hipam(skin.dat400, show.plot=FALSE)
# The default 2d method:
plot.tree(skin.hp400)
# As a dendrogram, with ordered clusters:
plot.tree(skin.hp400, method="tree")
# As a traditional dendrogram:
plot.tree(skin.hp400, method="tree", reorder=FALSE)
# Showing the cluster labels of all the data:
plot.tree(skin.hp400, method="all", data=skin.dat400)
```

## Description

This function plots covariate information about each of the clusters in a PAMSAM object. It is a visual tool to see how distinct the PAMSAM clusters are in terms of the chosen covariate.

## Usage

```
sammon.plot(pamsam.obj, covar, method="", radians=FALSE,  
            alpha=0.95, cv=NA, ...)
```

## Arguments

<code>pamsam.obj</code>	The output from a PAMSAM clustering on a particular data set. See <code>pamsam</code> for more details.
<code>covar</code>	Vector of covariate data. This should be of the same length of the number of objects clustered by the <code>pamsam.obj</code> . It can be a continuous variable (numerical object), a binary or multinomial response variable (factor or character). Note that if it is numeric but takes only two values, it is treated as binary.
<code>method</code>	If this is "circular" and <code>covar</code> is numeric, the covariate information will be treated as representing degrees (or radians if <code>radians=TRUE</code> ) and the angular means and standard deviations are calculated and plotted in circles.
<code>radians</code>	When <code>method="circular"</code> , the angles in <code>covar</code> will be treated as radians, if <code>radians=TRUE</code> . Otherwise they will be treated as degrees.
<code>alpha</code>	The size of the confidence interval to be shown when <code>covar</code> is numeric. For example, if 0.95 (the default), then a 95% confidence interval is drawn (assuming a normal distribution).
<code>cv</code>	This sets the factor by which the standard deviation of each cluster's covariates (when <code>covar</code> is numeric) should be multiplied. Use this instead of <code>alpha</code> if desired.
<code>...</code>	Other parts of <code>sammon.plot</code> can be altered if desired.

## Details

This function plots covariate information `covar` about the clusters of a PAMSAM object (`pamsam.obj`). Binary, multinomial, or continuous data can be supplied.

For binary and multinomial data bar charts will be drawn for each medoid in the PAMSAM clustering. The position of the charts in the plot is determined from a Sammon mapping of the medoids. The numbers in each sub-category in are shown in the plot as are the labels of the clusters.

For continuous data the mean for each cluster is plotted, together with its confidence interval (assuming a normal distribution) or error bar, in a rectangle. Each rectangle is placed at a point in the plot determined by a Sammon mapping of the cluster medoids.

If the covariate is angular data (`method="circular"`) then, rather than showing the means and confidence intervals in rectangles, they are shown in circles (having found the appropriate means). In this case the angles can be given in degrees (the default) or radians (`radians=TRUE`).

Note that for continuous data, the confidence interval or error bars size can be set by `alpha` or `cv`.

## Note

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

`pamsam`

## Examples

```
# A contrived example:
data(skin)
# A PAM(SAM) clustering of the first 200 genes in skin dataset into 4 clusters:
skin.ps200 <- pamsam(t(skin$dat[,1:200]),k=4)
# Invented covariate data:
bin.covar <- sample(c("A","B"),200,replace=TRUE)
mult.covar <- sample(c("A","B","C","D"),200,replace=TRUE)
cont.covar <- rnorm(200)
sammon.plot(skin.ps200,bin.covar)
sammon.plot(skin.ps200,mult.covar)
sammon.plot(skin.ps200,cont.covar)
```

---

`search.step.knn`      *[SUBFUNCTION] Stepwise search for KNN used by step.knn*

---

## Description

This is a sub-function for `step.knn`. It conducts the stepwise search. A leave-one-out cross validation approach is used as an optimizing function.

## Usage

```
search.step.knn(train, class, k.vec=seq(1,15,2), l=1)
```

## Arguments

<code>train</code>	A matrix of explanatory variables used in creating the model.
<code>class</code>	A factor specifying the class of each observation in train.
<code>k.vec</code>	Vector of the number of nearest neighbours (k) to search through.
<code>l</code>	Minimum vote for definite decision, otherwise 'doubt' (see <code>knn</code> in <code>class</code> package for details)

## Details

See `step.knn`

## Value

List containing: logical vector indicating variables to be included in the KNN (which); the value of the optimizing function for the chosen model (rate); the estimated leave-one-out cross validation class (`cv.class`) and the posterior probabilities associated with these estimates (`cv.posterior`) the KNN method used (`method`).

## Author(s)

J. McClure and E. Wit

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

`step.knn`

---

`search.step.lda`      *[SUBFUNCTION] Stepwise search for LDA used by `step.lda`*

---

## Description

This is a sub-function for `step.lda`. It conducts the stepwise search. A leave-one-out cross validation approach is used as an optimizing function.

## Usage

```
search.step.lda(x, class, method="mle")
```

## Arguments

<code>x</code>	A matrix of explanatory variables used in creating the model.
<code>class</code>	A factor specifying the class of each observation in train.
<code>method</code>	Type of LDA to be conducted. See <code>lda</code> in MASS library for details.

## Details

See `step.lda`

## Value

List containing: logical vector indicating variables to be included in the LDA (`which`); the value of the optimizing function for the chosen model (`rate`); the estimated leave-one-out cross validation class (`cv.class`) and the posterior probabilities associated with these estimates (`cv.posterior`) the LDA method used (`method`).

## Author(s)

J. McClure and E. Wit

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

## See Also

`step.lda`

---

`skin`

*DATA: Skin cancer microarray experiment*

---

## Description

This data set details a two-channel microarray experiment carried out on cancerous and normal (control) skin tissue samples using 4 arrays. It was conducted by Dr. Nighean Barr at the Cancer Research UK Beatson Laboratories in Glasgow. Many thanks to her for allowing us to include this data.

## Usage

`data(skin)`

## Format

A list containing:

**dat** A 8 by 9216 matrix containing the unnormalized expression values of the 9216 genes and control spots over the 4 two-channel arrays, with each row representing the expression values of one of the arrays in one of the channels.

**arrays** A vector of length 8 identifying the array (1, 2, 3 or 4) that each row of **dat** came from.

**conditions** A vector of length 8 identifying the condition ("cancer" or "normal") that the data in each row of **dat** represent.

**dyes** A vector of length 8 identifying the dye ("Cy3" or "Cy5") channel of the data in each row of **dat**.

**x** A vector of length 9216 containing the x coordinate of each gene or control spot (i.e. of each column of **dat**).

**y** A vector of length 9216 containing the y coordinate of each gene or control spot (i.e. of each column of **dat**)

## Source

Dr. Nighean Barr, Cancer Research UK Beatson Laboratories, Glasgow.

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

---

spat.norm

*Spatial microarray normalization*

---

## Description

This function carries out a spatial normalization on the data using loess. A location and scale spatial normalization are carried out. This type of normalization should be done before any other types of normalization (e.g. within or across condition normalization) are carried out.

## Usage

```
spat.norm(xpr, x=NULL, y=NULL, dat.log.scale=TRUE,  
          subset=NULL, ignore=NULL, scale.norm=TRUE,  
          span.loc=0.5, degree.loc=1, span.scale=0.75, degree.scale=1,  
          zero.centred=FALSE, n.sample = 1*10^4)
```

## Arguments

<code>xpr</code>	Microarray data to be spatially normalized. This can either be a vector of expression values or a matrix (or equally a data frame). In matrix format, <code>xpr</code> can be given in two ways. In the first, an entry's row and column defines its x and y positions on the microarray. In the second, there are three columns in the data; the first two give the x and y positions (respectively) of each spot on the array, whose expression value is given in the third column.
<code>x</code>	If <code>xpr</code> is given as a vector then the x-coordinates for the values in <code>xpr</code> should be given in <code>x</code> .
<code>y</code>	If <code>xpr</code> is given as a vector then the y-coordinates for the values in <code>xpr</code> should be given in <code>y</code> .
<code>dat.log.scale</code>	Logical value indicating whether or not the data are on the log scale. If <code>FALSE</code> , then logs will be taken of <code>xpr</code> , as the original scale of microarray data is highly skewed. The default is <code>TRUE</code> .
<code>subset</code>	A optional vector indicating the subset of values to be used in the loess spatial normalization. It is recommended that control spots not be included in the normalization and <code>subset</code> can be used to define non-control spots on the array. See <code>loess</code> and Wit and McClure (2004) for more details.
<code>ignore</code>	A vector indicating which values in <code>xpr</code> should be ignored before applying the loess (the complement set to <code>subset</code> ). Its contents are ignored if <code>subset</code> is specified. See <code>subset</code> .
<code>scale.norm</code>	A logical value indicating whether scale normalization should be included in the spatial normalization. This removes both second order local spatial effects as well as first order ones. The default value is <code>TRUE</code> .
<code>span.loc</code>	The span of the loess used in the location (first order) smoothing. The default value of 0.5 recommended in Wit and McClure (2004) is a robust value. See <code>loess</code> for more details about span.
<code>degree.loc</code>	The degree of the polynomial used in the location (first order) smoothing loess. The default value of 1 is that recommended in Wit and McClure (2004). Using a larger order polynomial can cause the smoother to be unreliable at the edges of the microarray.
<code>span.scale</code>	The span of the loess used in the scale (second order) smoothing. The default value of 0.75 recommended in Wit and McClure (2004) is a robust value. See <code>loess</code> for more details about span.
<code>degree.scale</code>	The degree of the polynomial used in the scale (second order) smoothing loess. The default value of 1 is that recommended in Wit and McClure (2004). Using a larger order polynomial can cause the smoother to be unreliable at the edges of the microarray.
<code>zero.centred</code>	If <code>TRUE</code> , <code>spat.norm</code> will return values centred on zero. If <code>FALSE</code> (the default), <code>spat.norm</code> returns the values on their original scale, although some may be negative.
<code>n.sample</code>	The maximum number of spots to be included in the loess. If there are more spots to be included in the loess than are in <code>n.sample</code> , then a sample of size <code>n.sample</code> is chosen from the spots to be included.

## Details

A local loess on the data is carried out to find the any spatial trend affecting local average expression values. This trend is then removed from the original data. Optionally (and by default) second order trends (trends in the variability of expression) across the array can be removed (see `scale.norm`).

A subset of values to be included (or ignored) in the loess can be specified (see `subset` and `ignore`).

The degree and span of the loesses for both location and scale trend can be specified (see `span.loc`, `degree.loc`, `span.scale`, `degree.scale`), although the default values for these arguments will conduct the spatial normalization robustly. Note that using a degree greater than 1 for either location or scale loess can lead to poor estimates of trend at the edges of a microarray.

## Value

The function returns a list containing a vector of spatially normalized expression values (`spat.norm.xpr`) and a matrix containing the x and y coordinates of these values (`coords`).

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

`all.norm`, `bkg.norm`, `dye.norm`, `cond.norm`

## Examples

---

`step.knn`

*Stepwise KNN*

---

## Description

This function searches for the "best" combination of variables and k to be included in a k-nearest neighbours (KNN) model using a forward stepwise method. A leave-one-out cross validation approach is used as an optimizing function. A true k-fold cross-validation error for the procedure can be found.

## Usage

```
step.knn(train, class, test=NULL, kcv=TRUE, fold=5, k.vec=seq(1,15,2),
         l=0, plot.it=TRUE,bw=FALSE,...)
```

## Arguments

<code>train</code>	A matrix of explanatory variables used in creating the model.
<code>class</code>	A factor specifying the class of each observation in train.
<code>test</code>	An optional matrix of cases that need to be classified.
<code>kcv</code>	A logical value indicating whether k-fold cross validation of the procedure should be carried out. This is done by default.
<code>fold</code>	The number of folds to be used in the k-fold cross validation.
<code>k.vec</code>	Vector of the number of nearest neighbours (k) to search through.
<code>l</code>	Minimum vote for definite decision, otherwise 'doubt' (see <code>knn</code> in class package for details)
<code>plot.it</code>	A logical indicator of whether a Sammon plot of the <code>step.knn</code> 's output should be given. This is done by default.
<code>bw</code>	If TRUE then the Sammon plot produced (when <code>plot.it=TRUE</code> ) will appear in black and white only. If FALSE, a colour plot will be produced.
<code>...</code>	Other <code>plot.step.knn</code> arguments used can be supplied.

## Details

The stepwise search is conducted using the `knn` function in `class`. This conducts a leave-one-out cross validation on each subset of variables it is given over each of the possible number of nearest neighbours given in `k.vec`. The cross validation error produced is used as an optimizing function.

To get an objective cross validation of the method, the matrix `train` must be split into `k` groups, or folds (`k` being defined by the argument `fold` in `step.knn`; note that `k` here has nothing to do with the `k` in `k-nearest neighbours`). The whole search procedure must be applied to each of the complements of the `k` groups and tested on that group. The proportion of incorrectly predicted observations found by this approach gives the cross validation error for the method.

A Sammon plot of the model is produced if `plot.it=TRUE` (see `plot.step.knn`).

## Value

An list containing the training data (`train.set`) and their associated true classes (`train.cl`); if calculated, the classes as estimated by the k-fold cross validation (`cv.est.cl`) and a logical vector indicating whether observations were correctly predicted by `cv.est.cl` (`correct.cl`); if supplied, the matrix of test cases (`test.set`) and the classes as predicted by the chosen KNN model (`est.test.cl`); a logical vector indicating which variables are selected by the chosen KNN model (`selected.var`); a vector of the different class types (`levels`); the number of nearest neighbours to be used in the chosen KNN model (`num.nearest.neighbours`); details of the lowest ("best") value of the function used to build the model (`build.error`); details of the k-fold cross validation error of the procedure (`cross.validation`).

## Author(s)

J. McClure and E. Wit

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

## See Also

knn , knn.cv (in class library) step.lda, plot.step.knn, kcv, search.step.knn

## Examples

```
x <- matrix(rnorm(1000), ncol=50)
x[,2] <- x[,2] + rep(c(-1,1),each=10)
x[,12] <- x[,12] + rep(c(-1,1,1,-1),each=5)
y <- rep(c("A","B","C","D"), each=5)
z <- matrix(rnorm(400),ncol=50)
z[,2] <- z[,2] + rep(c(-1,1),each=4)
z[,12] <- z[,12] + rep(c(-1,1,1,-1),each=2)
step.knn(train=x, class=y, test=z, kcv=TRUE, k.vec=seq(1,9,2),
         fold=5, plot.it=TRUE, bw=FALSE)
```

---

step.lda

*Stepwise LDA*

---

## Description

This function searches for the "best" combination of variables to be included in a linear discriminant analysis (LDA) using a forward stepwise method. A leave-one-out cross validation approach is used as an optimizing function. A true k-fold cross-validation error for the procedure can be found.

## Usage

```
step.lda(train, class, test=NULL, kcv=TRUE, method="mle",
        fold=5, plot.it=TRUE, bw=FALSE, ...)
```

## Arguments

train	A matrix of explanatory variables used in creating the model.
class	A factor specifying the class of each observation in train.
test	An optional matrix of cases that need to be classified.
kcv	A logical value indicating whether k-fold cross validation of the procedure should be carried out. This is done by default.

<code>method</code>	Type of LDA to be conducted. See <code>lda</code> in MASS library for details.
<code>fold</code>	The number of folds to be used in the k-fold cross validation.
<code>plot.it</code>	A logical indicator of whether a Sammon plot of the <code>step.lda</code> 's output should be given. This is done by default.
<code>bw</code>	If TRUE then the Sammon plot produced (when <code>plot.it=TRUE</code> ) will appear in black and white only. If FALSE, a colour plot will be produced.
<code>...</code>	Other <code>plot.step.lda</code> arguments used can be supplied.

## Details

The stepwise search is conducted using the `lda` function in MASS. This conducts a leave-one-out cross validation on each subset of variables it is given. The cross validation error produced is used as an optimizing function.

To get an objective cross validation of the method, the matrix `train` must be split into `k` groups (`k` being defined by the argument `fold` in `step.lda`). The whole search procedure must be applied to each of the complements of the `k` groups and tested on that group. The proportion of incorrectly predicted observations found by this approach gives the cross validation error for the method.

A Sammon plot of the model is produced if `plot.it=TRUE` (see `plot.step.lda`).

## Value

An list containing the training data (`train.set`) and their associated true classes (`train.cl`); if calculated, the classes as estimated by the k-fold cross validation (`cv.est.cl`) and a logical vector indicating whether observations were correctly predicted by `cv.est.cl` (`correct.cl`); if supplied, the matrix of test cases (`test.set`) and the classes as predicted by the chosen LDA model (`est.test.cl`); a logical vector indicating which variables are selected by the chosen LDA model (`selected.var`); a vector of the different class types (`levels`); details of the lowest ("best") value of the function used to build the model (`build.error`); details of the k-fold cross validation error of the procedure (`cross.validation`); the method used for the LDA (`method`).

## Author(s)

J. McClure and E. Wit

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

## See Also

`lda` (in MASS library), `step.knn`, `plot.step.lda`, `kcv`, `search.step.lda`

## Examples

```
x <- matrix(rnorm(1000), ncol=50)
x[,2] <- x[,2] + rep(c(-1,1),each=10)
x[,12] <- x[,12] + rep(c(-1,1,1,-1),each=5)
y <- rep(c("A","B","C","D"), each=5)
z <- matrix(rnorm(400),ncol=50)
z[,2] <- z[,2] + rep(c(-1,1),each=4)
z[,12] <- z[,12] + rep(c(-1,1,1,-1),each=2)
step.lda(train=x, class=y, test=z, kcv=TRUE, method="mle",
         fold=5, plot.it=TRUE, bw=FALSE)
```

---

tuberculosis

*DATA: Tuberculosis time-course microarray experiment*

---

## Description

This data set details a two-channel time-course microarray experiment on Mycobacterium Tuberculosis. In the experiment, the bacterium was grown in vitro over the course of 30 days under controlled circumstances with a fixed amount of growth medium. mRNA was harvested after 6, 14, 20 and 30 days from separate flasks. For day 6, mRNA was extracted from two flasks, divided in four and applied to four microarrays. For each of the other days, mRNA was extracted from a single flask, divided in four hybridization samples and applied to four microarrays. On each microarray also a reference sample of genomic DNA was hybridized. The experiment was conducted by the bacterial microarray group at St. George's Hospital in London under supervision of Dr. Jason Hinds and Prof. Philip Butcher. Many thanks to them for allowing us to include this data.

## Usage

```
data(tuberculosis)
```

## Format

A list containing:

**dat** A 32 by 4624 matrix containing normalized expression values of the 3924 genes and 700 control spots over the 16 two-channel arrays, with each row representing the expression values of one of the arrays in one of the channels. The normalization was carried out with normalization routines provided by Dr. Lorenz Wernisch.

**arrays** A vector of length 32 identifying the array (1, 2, ... 16) that each row of **dat** came from.

**conditions** A vector of length 32 identifying the condition ("reference", "day 6", "day 14", "day 20", "day 30") that the data in each row of **dat** represent.

**dyes** A vector of length 32 identifying the dye ("Cy3" or "Cy5") used to label the sample in each row of **dat**.

**x** A vector of length 4624 containing the x coordinate of each spot on the array.

- y** A vector of length 4624 containing the y coordinate of each spot on the array.
- days** A numeric vector of length 32, containing the time points at which each of the samples were harvested. The genomic DNA reference samples are coded -1. This vector is the numeric equivalent of **conditions**.
- array.labels** A vector of length 32, indicating the time point, the (technical) replicate index and the sample type (**r** = genomic DNA reference sample, **s** = mRNA signal). For example, **06.1.s** stands for the expression profile of the first replicate of the mRNA harvested at day 6.
- genes** A vector of length 4624 of all the spot names. These include the 3924 M. Tuberculosis genes (indicated by Rv) and 700 control spots.

### Source

The Bacterial Microarray Group at St. George's Hospital in London (Bug@s).

### References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference" Chichester: Wiley.

---

`twoway.pam`

*Two-way PAMSAM clustering*

---

### Description

This function conducts a two-way clustering of the data matrix it is given, so that both its rows and columns are clustered. For microarray data, the columns should refer to genes and the rows to arrays.

### Usage

```
twoway.pam(dat, k.row=NA, k.col=NA, row.metric="euclidean", col.metric="abscor",
  reduce.cols=FALSE, alpha=0.05, col.lab="Column clusters",
  row.lab="Row clusters", col=heat.colors(100), breaks=NULL,
  line.col="black", order.metric="manhattan")
```

### Arguments

- dat** Matrix of data to be clustered.
- k.row** The number of clusters that the rows are to be partitioned into. If no value is given for **k.row** (the default), **twoway.pam** searches for the appropriate value.
- k.col** The number of clusters that the columns are to be partitioned into. If no value is given for **k.col** (the default), **twoway.pam** searches for the appropriate value.

<code>row.metric</code>	The measure used to define dissimilarities between rows. By default this is "euclidean"; other options include other power distances ("manhattan", "cubic", "quartic"), 1 minus the correlation ("correlation"), or 1 minus the absolute correlation "abscorr".
<code>col.metric</code>	The measure used to define dissimilarities between columns. By default this is 1 minus the absolute correlation ("abscor"); other options include 1 minus the correlation ("correlation") and power distances ("manhattan", "euclidean", "cubic", "quartic").
<code>reduce.cols</code>	If TRUE, the number of columns can be reduced by eliminating those that do not predict the row clustering. A p-value of <code>alpha</code> is used to determine whether each column is included.
<code>alpha</code>	Applies when <code>reduce.cols=TRUE</code> . It give the p-value cut-off for the each of the ANOVAs that decide which columns are to be selected.
<code>col.lab</code>	Character string giving the label for the columns in the plot.
<code>row.lab</code>	Character string giving the label for the rows in the plot.
<code>col</code>	Vector indicating the intensity in the created image. A greyscale can be created using <code>grey(0:25/25)</code> .
<code>breaks</code>	A set of breakpoints for the colours: must give one more breakpoint than colour. See <code>image</code> for more details.
<code>line.col</code>	Character string indicating the colour of the lines that separate the clusters in the plot.
<code>order.metric</code>	The dissimilarity measure used in the 1-dimensional Sammon mapping of the row and column clusters. The default value uses "manhattan" distance, but other power distances ("euclidean", "cubic", "quartic"), 1 minus the correlation ("correlation"), or 1 minus the absolute correlation "abscorr" can be specified.

## Details

The function first clusters the rows using `pamsam`. It then clusters the columns in the same way, having optionally removed those columns that do not differ across the row clusters.

An image plot is created of the re-ordered data matrix, where the rows and columns are rearranged according to the two clusterings. The order of the row clusters is determined using a one dimensional Sammon mapping on their medians; the same approach is used to order the column clusters.

## Value

The function returns a list containing: a vector of the `dat` row numbers that each row in the image refers to (`row.order`); a vector of the `dat` column numbers that each column in the image refers to (`col.order`); the clustering label of each row in the image (`row.clustering`); the clustering label of each column in the image (`col.clustering`).

## Author(s)

E. Wit and J. McClure

## References

E. Wit and J. McClure (2004) "Statistics for Microarrays: Design, Analysis and Inference"  
Chichester: Wiley.

## See Also

pamsam

## Examples

```
data(skin)
# A PAMSAM clustering of the first 400 genes in skin dataset:
skin.twp400 <- twoway.pam(skin$dat[,1:400])
# Removing "uninformative" genes:
skin.twp400 <- twoway.pam(skin$dat[,1:400],reduce.cols=TRUE)
```

---

xwithouty

*x without y*

---

## Description

This function simply returns the vector `x` excluding the entries that are in `y`.

## Usage

```
x %w/o% y
```

## Arguments

`x` Vector whose elements not in `y` are wanted.  
`y` Vector whose elements are to be removed from `x`.

## Details

The function is similar to (and uses) the internal `%in%` function. `%w/o%` returns the vector given before it, having excluded all elements that are in the vector after it.

It is used in `search.step.lda` and `search.step.knn`.

## Value

A vector containing all elements of the vector before the function's name apart from the elements after the functions name

## Note

**Author(s)**

J. McClure and E. Wit

**References****See Also**

search.step.lda, search.step.knn

**Examples**

```
x <- 1:10  
y <- 5:7  
x %w/o% y
```