# Package 'swamp'

July 2, 2014

**Type** Package

**Title** Visualization, analysis and adjustment of high-dimensional data in respect to sample annotations.

**Version** 1.2.3

**Depends** impute, amap, gplots, MASS

**Author** Martin Lauss

**Maintainer** Martin Lauss <martin.lauss@med.lu.se>

**Description** The package contains functions to connect the structure of the data with the information on the samples. Three types of associations are covered: 1. linear model of principal components. 2. hierarchical clustering analysis. 3. distribution of features-sample annotation associations. Additionally, the inter-relation between sample annotations can be analyzed. Simple methods are provided for the correction of batch effects and removal of principal components.

**License** GPL (>= 2)

**LazyLoad** yes

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2013-03-13 14:11:11

## R topics documented:

---

`swamp-package` *Visualization, analysis and adjustment of high-dimensional data in respect to sample annotations.*

---

## Description

The package contains functions to connect the structure of the data with sample information. Three types of analyses are covered: 1. linear models of principal components. 2. hierarchical clustering analysis. 3. distribution of associations between individual features and sample annotations. Additionally, the inter-relation between sample annotations can be analyzed. We include methods for batch adjustment by a. median-centering, b. linear models and c. empirical bayes (combat). A method to remove principal components from the data is also included.

## Details

| | |
|---|---|
| Package: | swamp |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2011-01-14 |
| License: | GPLv2 |
| LazyLoad: | yes |

This package aims to find the associations between a high-dimensional data matrix, typically obtained form high-throughput analysis, to the sample annotations, be they technical (e.g. batch surrogates) or biological information. High-dimensional data usually has a much larger number of features than samples. This requires specific analysis to see "how the data looks like" and in which way the technical and biological information on the samples is reflected in the dataset. Sample annotations can be analyzed for their association to principal componenents (prince(), prince.plot(), prince.var.plot()) as well as clusters from HCA (hca.plot(), hca.test()). The distribution of association of sample annotations to the features can be plotted and analysed (feature.assoc(), dense.plot(), corrected.p()). Batch surrogate variables might be associated with biological annotations, hence making batch adjustment of the data problematic. The associations between the sample annotations can be calculated and plotted (confounding()). If unwanted batch effects have been identified in the previous steps, two simple methods are provided to adjust the data (quickadjust.zero(),

quickadjust.ref()). Another batch adjustment uses linear models, with the advantage to remove numerical variables and several variables at once (adjust.linearmodel()). In addition, the popular ComBat batch adjustment has been adopted for the package to combine batches of small sample size (combat()). Principal components confounded by batch variables can be reomved from the data (kill.pc()). To use the functions of this package, the data matrix has to be a numeric matrix and the sample annotations have to be a data.frame with numeric or factors with 2 or more levels. NAs are allowed in both the data matrix and the annotation dataframe, except for the functions adjust.linearmodel() and combat().

## Author(s)

Martin Lauss martin.lauss@med.lu.se

## Examples

```
##### first you need a dataset (matrix)
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1  ## lets put in some larger values
set.seed(200)
##### second you need sample annotations (data.frame)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))
              # Level "B" of Factor 1 marks the samples with the larger values



##### perform the functions

## principal components
res1<-prince(g,o,top=10,permute=TRUE)
prince.plot(prince=res1)
  # to plot p values of linear models: lm(principal components ~ sapmle annotations).
  # to see if the variation in the data is associated with sample annotations.
res2<-prince.var.plot(g,show.top=50,npermute=10)
  # to see how many principal components carry informative variation

## hierarchical clustering analysis
hca.plot(g,o)
  # to show a dendrogram with sample annotations below
res3<-hca.test(g,o,dendcut=2,test="fisher")
  # to test if the major clusters show differences in sample annotations

## feature associations
res4a<-feature.assoc(g,o$Factor1,method="correlation")
  # to calculate correlation between one sample annotation and each feature
res4b<-feature.assoc(g,o$Factor1,method="t.test",g1=res4a$permuted.data)
res4c<-feature.assoc(g,o$Factor1,method="AUC",g1=res4a$permuted.data)
dense.plot(res4a)
  # to plot the distribution of correlations in the observed data
```

```
  # in comparison to permuted data
dense.plot(res4b)
dense.plot(res4c)
res5<-corrected.p(res4a)
  # to correct for multiple testing and find out how many features are
  # significantly associated to the sample annotation
names(which(res5$padjust<0.05))
names(which(res5$adjust.permute<0.05))
names(which(res5$adjust.rank<0.05))

## associations between sample annotations
res4<-confounding(o,method="fisher")
  # to see how biological and technical annotations are inter-related

## adjustment for batch effects
gadj1<-quickadjust.zero(g,o$Factor1)
  # to adjust for batches (o$Factor1)
  # using median centering of the features for each batch
prince.plot(prince(gadj1,o,top=10))

gadj2<-quickadjust.ref(g,o$Factor1,"B")
  # to adjust for batches (o$Factor)
  # by adjusting the median of the features to the median of a reference batch.
prince.plot(prince(gadj2$adjusted.data,o,top=10))

gadj3<-kill.pc(g,pc=1)
  # to remove one or more principal components (here pc1) from the data
prince.plot(prince(gadj3,o,top=10))

lin1<-adjust.linearmodel(g,o$Numeric1)
  # to adjust for numerical variables using a linear model
prince.plot(prince(lin1,o,top=10))

com1<-combat(g,o$Factor1,batchcolumn=1)
  # to use the empirical Bayes framework of ComBat, popular for small-sample sizes
prince.plot(prince(com1,o,top=10))
```

---

`adjust.linearmodel` *Batch adjustment using a linear model*

---

## Description

The function uses a linear model for each feature: with the feature as dependent variable and technical variables (batches) as regressors.

## Usage

```
adjust.linearmodel(g, o.batches, robust.lm = F, small.memory = F)
```

## Arguments

| | |
|---|---|
| g | the input data in form of a matrix with features as rows and samples as columns. NAs are allowed. |
| o.batches | contains the batch variable(s). a numeric or factor vector, or a dataframe. |
| robust.lm | default=F, if set to true robust linear models are performed by rlm of package MASS. The calculations take longer than using lm because a loop is used. |
| small.memory | default=F, if set to true robust a loop through the rows is used in the lm function. This reduces the risk of running out of memory, however computation time is longer. |

## Details

For each feature a lm(feature~., batches) is performed. The residuals of the fitted model are returned. (The means of the features of g are added to the residuals, as the residuals are centered, which may not be desired.) Note the following possibilties of using a linear model for batch adjustment: 1. Technical variables (Batches) can be numeric. 2. Numerical technical variables can be used in transformed forms (log, exp,...). 3. Several batch variables, be it numeric or factors, can be corrected at once, by just adding more regressors to the model. By default the function performs lm. If robust.lm = T, robust linear models are performed using the rlm funcion of MASS. NAs are not allowed in g. Samples that contain NAs in o.batches are returned unadjusted.

## Value

A numeric matrix which is the adjusted dataset.

## Note

robust linear models require the package MASS

## Author(s)

Martin Lauss

## Examples

```
# data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
   paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
# patient annotations as a data.frame, annotations should be numbers and factors but not characters.
# rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50), Numeric2=colMeans(g),row.names=colnames(g))

##unadjusted.data
res1<-prince(g,o,top=10)
```

```
prince.plot(res1)

##batch adjustment
lin1<-adjust.linearmodel(g,o$Numeric2)
lin2<-adjust.linearmodel(g,o[,c("Numeric2","Factor2")]) # also correct for Factor2

##prince.plot
prince.plot(prince(lin1,o,top=10))
prince.plot(prince(lin2,o,top=10))
```

---

`combat` *ComBat algorithm to combine batches.*

---

## Description

Performs ComBat as described by Johnson et al.

## Usage

```
combat(g, o.withbatch, batchcolumn = NULL, par.prior = T, prior.plots = T)
```

## Arguments

| | |
|---|---|
| `g` | the input data in form of a matrix with features as rows and samples as columns. |
| `o.withbatch` | the batch annotation as a factor vector or within a dataframe that contains additional biological co-variates. make sure that the order of annotation is the same as in g. rownames (o) must be identical to colnames (g). when submitting a data.frame o.withbatch it can contain only factors. |
| `batchcolumn` | Required. Specify the batch column number of a dataframe ; set to 1 for a vector. All columns have to be factors, no NAs allowed. |
| `par.prior` | if 'T' uses the parametric adjustments, if 'F' uses the nonparametric adjustments. if you are unsure what to use, try the parametric adjustments (they run faster) and check the plots to see if these priors are reasonable. |
| `prior.plots` | if 'T' will give prior plots with black as a kernal estimate of the empirical batch effect density and red as the parametric estimate. |

## Details

The R-code of the ComBat algorithm has been taken from the webpage jlab.byu.edu/ComBat and input and output were adopted to the swamp package. ComBat uses parametric and non-parametric empirical Bayes frameworks for adjusting data for batch effects. The method is robust to outliers and performs particularly well with small sample sizes. ComBat can handle only categorical batch variables in its current development stage. Biological covariates can be added to the model (also categorical).

## Value

A numeric matrix which is the adjusted dataset.

## Note

R coded algorithm directly from Johnson WE

## Author(s)

Martin Lauss

## References

Johnson WE, Li C, Rabinovic A. Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics. 2007 Jan;8(1):118-27.

## Examples

```
# data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
# patient annotations as a data.frame, annotations should be numbers and factors but not characters.
# rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Factor3=factor(c(rep("X",15),rep("Y",20),rep("Z",15))),
              Numeric1=rnorm(50),
              row.names=colnames(g))

##unadjusted.data
res1<-prince(g,o,top=10)
prince.plot(res1)

##batch adjustment for Factor 3
com1<-combat(g,o$Factor3,batchcolumn=1)
##batch adjustment for Factor 3; with covariate
com2<-combat(g,o[,c("Factor2","Factor3")],batchcolumn=2)

##prince.plot
prince.plot(prince(com1,o,top=10))
prince.plot(prince(com2,o,top=10))
```

confounding *Heatmap of interrelation of sample annotations*

## Description

The function tests the relationships of the sample annotations and plots the heatmap of the p-values.

## Usage

```
confounding(o, method = "chisq", workspace = 2e+07, smallest = -20,
            diagonal.zero = F, label = colnames(o), note = T, notecol = "black",
           notecex = 1, breaks = 50, col = c(heat.colors(48), "white"), key = T,
            cexRow = 1, cexCol = 1, margins = c(7,7), colsep = NULL,
            rowsep = NULL, sepcolor = "black", sepwidth = c(0.05,0.05))
```

## Arguments

| | |
|---|---|
| o | the sample annotations in the form of a data.frame, with the sample names as rownames(o). o can contain factors with 2 or more levels and numeric variables; no character variables are allowed. NAs are allowed and cases are removed at calculations. |
| method | statistical test to be used when two factors are tested, this can be either "fisher" or "chisq" to use fisher.test() or chisq.test(), respectively. default = "chisq". fisher.test is however preferable as it is an exact test. Note that fisher.test() is computationally expensive and can cause R to crash. |
| workspace | workspace to use if test="fisher". |
| smallest | a numeric value. log10(p-values) less than smallest are set to smallest for plotting. default = -20. e.g. a log10 p-value of -37 will be set to -20. Smallest has to be less than 0. |
| diagonal.zero | set to TRUE to force diagonal p-values to be 0. |
| label | vector containing names of the sample annotation. default=colnames(o) |
| note | set to TRUE to print the p-values in the cells of the plot. |
| notecol | to determine the color of the notes. |
| notecex | to determine the font size of the notes. |
| breaks | either a number (default=50) or a numeric vector (default would be seq(-20,0,length.out=50)) of breaks for the colors. |
| col | a vector of colors with a length of breaks-1. default=c(heat.colors(48), "white")). |
| key | whether the color key should be printed, default=TRUE. |
| cexRow | font size of row label. default=1. |
| cexCol | font size of column label. default=1. |
| margins | a vector with the margins for columns and rows. default=c(7,7). |
| colsep | same as in heatmap.2 function. |

| | |
|---|---|
| rowsep | same as in heatmap.2 function. |
| sepcolor | same as in heatmap.2 function. |
| sepwidth | same as in heatmap.2 function. |

## Details

Technical and biological annotations are often interrelated, leading to confounding. This function tests the interelation of all sample annotations, be they technical batch surrogates or biological measures. Two sample annotations are compared at a time. If both are factors, fisher.test() or chisq() test can be used. Note that fisher.test() is computationally expensive and might cause R to crash at large sample numbers. If one sample annotation is numeric a linear modeal is used in the form of lm(numeric sample annotation~other sample annotation). The p-value is derived from the F-statistic of the linear model. The p-value from lm() is equivalent to the cor.test() p-value in the case of two numeric variables. NAs in the sample annotations are allowed and result in deletion of the NA case. It should be noted however, that different number of NAs in various sample annotations lead to different power of the comparisons. Matrices that specify for each comparison the test and sample number used are returned. With NAs in the data it is possible that a pair of sample annotations does not provide two different values each. In such a pair that does not show variance for both annotations the output is set to NA. The function uses heatmap.2() from the package gplots to plot the p-values.

## Value

a list with components

| | |
|---|---|
| p.values | a numeric square matrix that contains the p-values for associations between sample annotations. |
| n | a numeric square matrix that contains the number of samples at each test. |
| test.function | a character square matrix that contains the test function used at each test. |
| classes | a character vector that contains the classes of the variables in o. |

## Note

requires the package gplots

## Author(s)

Martin Lauss

## Examples

```
# patient annotations as a data.frame, annotations should be numbers and factors but not characters.
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Factor3=factor(c(rep("X",15),rep("Y",20),rep("Z",15))),
              Numeric1=rnorm(50))

## calculate and plot interrelations
```

```
res4<-confounding(o,method="fisher")
```

---

`corrected.p` *Correction of p-values for associations between features and sample annotation*

---

## Description

The function corrects for multiple testing of associations of features to sample annotation. Adjustment is done by padjust() or by the p-values from permuted data.

## Usage

```
corrected.p(feature.assoc, correction = "fdr", adjust.permute = T,
            adjust.rank = T, ties.method = "first")
```

## Arguments

| | |
|---|---|
| `feature.assoc` | a list with the p-values of feature associations, typically created by the function feature.assoc(). (If not created by feature.assoc() the list has to contain the elements observed.p and permuted.p.) |
| `correction` | adjustment method to use for padjust(). default="fdr". must be one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" or "none". |
| `adjust.permute` | if set to TRUE (default), the p-values will be adjusted by observed.p divided by permuted.p for each rank in observed.p and permuted.p. |
| `adjust.rank` | if set to TRUE (default), the p-values will be adjusted by calculating for every observed p-value the proportion of smaller permuted.p values to smaller observed.p values. |
| `ties.method` | if adjust.permute=TRUE or adjust.rank=TRUE the method for handling ties can be either "first" or "random". Tied p-values are likely when using "AUC" as method to measure feature associations. default="first". |

## Details

As high-dimensional data contains many features, the p-values of feature associations have to be corrected for multiple testing. The number of features that are significantly associated with sample annotation can show how strog the data is connected to the respective sample annotation. The p-values can be adjusted using the standard correction methods of padjust(). Additionally two methods that use the p-values from permuted data are proposed. First, p-values are adjusted by observed.p divided by permuted.p for each rank in observed.p and permuted.p. For instance if the third lowest p-value in the observed associations is 1e-9 and the third lowest p value in the permuted data is 1e-4, this p-value is correced by 1e-9 divided by 1e-4 which is 1e-5. Second, p-values are adjusted by calculating for every observed p-value the proportion of smaller permuted.p values to smaller observed.p values. For instance, we have a p-value of 1e-3 which is ranked as the 300-lowest p-value in the observed data. In the permuted data there are 3 p-values that are lower than

1e-3. In the 300 p-values we suspect 3 of them to occur by chance, hence the adjusted p-value is 3 divided by 300 which is 0.01. This correction method can yield p-values of 0 and is less robust when only a few permuted.p are smaller than the observed.p. Both proposed correction methods may acutally show similar results to padjust(observed.p,method="fdr")

## Value

a list with components

| | |
|---|---|
| `padjust` | a numeric vector containing the corrected p-values using padjust(). |
| `adjust.permute` | a numeric vector containing the corrected p-values using observed.p divided by permuted.p at each rank |
| `adjust.rank` | a numeric vector containing the corrected p-values by calculating for every observed p-value the proportion of smaller permuted.p values to smaller observed.p values |

## Author(s)

Martin Lauss

## Examples

```
# data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
# patient annotations as a data.frame, annotations should be numbers and factor but not characters.
# rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))


#calculate feature associations
res4a<-feature.assoc(g,o$Factor1,method="correlation")
#correct the p-values
res5<-corrected.p(res4a)
  names(which(res5$padjust<0.05))
  names(which(res5$adjust.permute<0.05))
  names(which(res5$adjust.rank<0.05))
```

---

`dense.plot` *Density plots of feature associations in observed and permuted data*

---

## Description

The function plots the distribution of feature associations for a specified sample annotation for both observed and reshuffled data.

## Usage

```
dense.plot(feature.assoc, lty = 1:2, col = 1:2, lwd = c(2, 2), ylab = "",
           main = "", cex.main = 1, cex.lab = 1, cex.axis = 1)
```

## Arguments

| | |
|---|---|
| feature.assoc | A list of feature associations, typically created by the function feature.assoc(). (If not created by feature.assoc() the list has to contain the elements observed, permuted and method.) |
| lty | a numeric vector containing the line types for the observed and permuted density lines. default=1:2. |
| col | the colors for the observed and permuted density lines. default=1:2. |
| lwd | the line widths. default=c(2,2). |
| ylab | optional labeling of y-axis. |
| main | optional titel. |
| cex.main | optional titel font size. |
| cex.lab | optional axis label font size. |
| cex.axis | optional axis font size. |

## Details

The function plots the distribution of associations of features with a sample annotation calculated by feature.assoc(). The function uses plot.density() for the observed data and adds the permuted data using lines(density()). The x-axis is dependent on the method used to measure association, e.g. if the method was "correlation", then xlim is c(-1,1) and xlab="Corrlation".

## Author(s)

Martin Lauss

## Examples

```
## data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
   paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
## patient annotations as a data.frame, annotations should be numbers and factor but not characters.
## rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

# calculate the associations to Factor 1
res4a<-feature.assoc(g,o$Factor1,method="correlation")
res4b<-feature.assoc(g,o$Factor1,method="t.test",g1=res4a$permuted.data)
   # uses t.test instead, reuses the permuted data generated in res4a
```

```
res4c<-feature.assoc(g,o$Factor1,method="AUC",g1=res4a$permuted.data)
   # uses AUC instead, reuses the permuted data generated in res4a

# plot distribution of associations in observed and permuted data
dense.plot(res4a)
dense.plot(res4b)
dense.plot(res4c)
```

---

`feature.assoc` *Associations of the features to a sample annotation in observed and reshuffled data.*

---

## Description

This function calculates the associations of each feature of the data matrix to a specified sample annotation. Either Pearson correlation, t-test statistic, Area Under Curve or R squared is used as measure of association. In parallel, the features in permuted data are tested for comparison.

## Usage

```
feature.assoc(g, y, method = "correlation", g1 = NULL, exact = 1)
```

## Arguments

| | |
|---|---|
| `g` | the input data in form of a matrix with features as rows and samples as columns. Missing values are allowed. |
| `y` | a factor or numeric vector which contains the sample information. Typically a variable of the data.frame o used in the remaining functions of this package. y can be a factor with 2 or more levels or a numeric vector. y cannot be a character vecor. y has to be of the same length as ncol (g). Missing values are allowed and those cases are removed from the calculations. |
| `method` | if y is a factor with two levels, this method is used for calculation of the association. The method can be one of "correlation", "t.test", or "AUC". If y is a factor with >2 levels lm() is used automatically, if y is numeric cor() is used automatically to determine the associations. |
| `g1` | As there are different ways to generate a randomized dataset, a pre-calculated permutation set can be specified here. Else the permutation data is generated within the function by reshuffling the values for each feature. g1 has to be a matrix with the same dimensions as g. |
| `exact` | if method="AUC", exact determines how wilcox.test() treats ties. |

## Details

For each feature the association to the sample annotation is calculated. If the sample annotaion is a factor with 2 levels, it can be chosen whether Pearson correlation, t.test statistic or Area Under Curve (AUC) is used as measure of association. The uncorrected p-values for the strength of associations are calculated by cor.test(), t.test() and wilcox.test() respectively. The distribution of

these associations can be seen using dense.plot() function. For instance this can reveal a group of positively associated features. The order of the levels in levels(y) is decisive, e.g. for correlation the factors are transformed by as.numeric(), whereby the first level becomes 1 and the second level becomes 2. Hence, a positive association means higher values in samples with level 2 and a negative assocation means higher values in level 1. This should also be true for t.test and AUC, but please re-check. If the annotation is a factor with more than 2 levels, lm() is automatically used with R squared as the measure of association and the p-value as obtained from the F statistic. If the annotation is a numeric vector, correlation is used (with cor.test() for p-value). NAs are allowed in both the data matrix and the annotation vector and is treated by case-wise deletion for the calculations. To see the relelvance of the associations, the calculations are repeated with permuted data, which can be either pre-entered as g1 or otherwise is calculated within the function by reshuffling the values for each feature.

## Value

a list with components

| | |
|---|---|
| `observed` | a numeric vector containing the association of features to sample annotation in the observed data. |
| `permuted` | a numeric vector containing the association of features to sample annotation in the permuted data. |
| `observed.p` | a numeric vector containing the p-values for association of features to sample annotation in the observed data. |
| `permuted.p` | a numeric vector containing the p-values for association of features to sample annotation in the permuted data. |
| `method` | the method used as measure of association, which can be one of "correlation", "t.test", "AUC" or "linear.model". |
| `class.of.y` | a character that states the class of y. |
| `permuted.data` | the matrix of the permuted data. |

## Author(s)

Martin Lauss

## Examples

```
## data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
   paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
## patient annotations as a data.frame, annotations should be numbers and factor but not characters.
## rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

# calculate the associations to Factor 1
```

```
res4a<-feature.assoc(g,o$Factor1,method="correlation")
res4b<-feature.assoc(g,o$Factor1,method="t.test",g1=res4a$permuted.data)
    # uses t.test instead, reuses the permuted data generated in res4a
res4c<-feature.assoc(g,o$Factor1,method="AUC",g1=res4a$permuted.data)
    # uses AUC instead, reuses the permuted data generated in res4a
str(res4a)
```

---

hca.plot *Dendrogram with according sample annotations*

---

## Description

The function plots the dendrogram from hierarchical cluster analysis with colorcoded sample annotations below.

## Usage

```
hca.plot(g, o, method = "correlation", link = "ward", colored = palette(),
         border = NA, code = colnames(o), cex.code = 1,
         breaks = round(nrow(oreihe)/4),
       cutcolors = colorpanel(breaks, low = "green", mid = "black", high = "red"))
```

## Arguments

| | |
|---|---|
| g | the input data in form of a matrix with features as rows and samples as columns. |
| o | the corresponding sample annotations in the form of a data.frame. A single sample annotation variable as a vector is allowed and will be transformed to a data.frame. rownames (o) must be identical to colnames (g). o can contain factors and numeric variables. No character variables are allowed. NAs are allowed and blank spaces are plotted. |
| method | the distance method for the clustering. default="correlation". hcluster from the package amap is used and method must be one of "euclidean", "maximum", "manhattan", "canberra" "binary" "pearson", "correlation", "spearman" or "kendall". |
| link | the agglomeration principle for the clustering. default="ward". hcluster from the package amap is used and link must be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". |
| colored | a vector of colors in which factor variables of o will be colorcoded. default are the 8 colors of palette(). the first level is plotted in the first color, the second in the second color and so on. for annotation with more than 8 levels colors should be added here. |
| border | a color for the borders in the annotation rectangels rect(). default=NA. |
| code | vector containing names of the sample annotations. default=colnames(o). |
| cex.code | font size of code. |
| breaks | a number that determines in how many bins a numeric annotation is cut using the cut() function. |

`cutcolors` a vector of color in which numeric variables will be colored. length(cutcolors) has to be the number of breaks. a colorpanel is default to plot the numeric values as a color gradient, with low values in green and high values in red.

## Details

The data is clustered using the amap package. The plot works for sample annotations as a data.frame or as a single vector. NAs are allowed in both data matrix and sample annotation data.frame. If the annotation is a factor, the annotations come in the colororder specified by colored. If the annotation is numeric, breaks and cutcolors is used which is currently set to be a colorpanel().

## Note

requires the packages amap and gplots

## Author(s)

Martin Lauss

## Examples

```
# data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
   paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
# patient annotations as a data.frame, annotations should be numbers and factor but not characters.
# rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

## hca plot
hca.plot(g,o)
```

---

# hca.test *Tests for annotation differences among sample clusters*

---

## Description

The main clusters of a dendrogram are tested for different patient annotations.

## Usage

```
hca.test(g, o, dendcut = 2, method = "correlation", link = "ward",
         test = "chisq", workspace = 2e+07)
```

## Arguments

| | |
|---|---|
| g | the input data in form of a matrix with features as rows and samples as columns. |
| o | the corresponding sample annotations in the form of a data.frame. Sample annotation as a single vector is allowed and will be transformed to a data.frame. rownames (o) must be identical to colnames (g). o can contain factors and numeric variables. No character variables are allowed. NAs are allowed. |
| dendcut | the number of clusters to cut the dendrogram tree (using cutree()). default=2. |
| method | the distance method for the clustering. default="correlation". hcluster from the package amap is used and method must be one of "euclidean", "maximum", "manhattan", "canberra" "binary" "pearson", "correlation", "spearman" or "kendall". |
| link | the agglomeration principle for the clustering. default="ward". hcluster from the package amap is used and link must be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". |
| test | the test to use for the annotations that are factors. this can be either "fisher" or "chisq" to use fisher.test() or chisq.test(), respectively. default = "chisq". However fisher.test is preferable as it is an exact test. Note that fisher.test() is computationally expensive and can cause R to crash. |
| workspace | workspace to use if test="fisher" |

## Details

The function clusters the samples using amap and then cuts the dendrogram into a specified number of clusters. The obtained sample clusters are tested for differences in sample annotations. fisher.test() or chisq.test() is used if the annotation is a factor, lm(annotation~clusters) is used for numeric annotations. The p-values for the dependence of sample annotation and sample clusters are returned.

## Value

a list with components

| | |
|---|---|
| p.values | a numeric vector containing the p.values for the annotation variable. |
| classes | the classes of the annotation variables in o. |

## Note

requires the package amap

## Author(s)

Martin Lauss

## Examples

```
## data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
```

```
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
## patient annotations as a data.frame, annotations should be numbers and factor but not characters.
## rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

# perform the test for the main 2 clusters
res3<-hca.test(g,o,dendcut=2,test="fisher")
        # use test="chisq" for large ncol(g) to avoid crash of R
res3$p.values
```

---

`kill.pc` *Removes principal components from a data matrix*

---

## Description

Does not destroy your personal computer. Really. (No warranty).

## Usage

```
kill.pc(g, pc, imputeknn = F, center = T)
```

## Arguments

| | |
|---|---|
| `g` | the input data in form of a matrix with features as rows and samples as columns. |
| `pc` | the principal components to be removed in form of a numeric vector of length 1 or more. e.g. to remove pc1 and pc3 use pc=c(1,3), to remove only pc3 use pc=3. |
| `imputeknn` | default=FALSE. missing values in the data matrix can be imputed by imputeknn=TRUE. The function knn.impute from the package impute is used with default settings. |
| `center` | default=TRUE. the features are mean-centered before singular value decompositon. this is a pre-requisite for principal component analysis, change only if you are really convinced that centering is not necessary. |

## Details

A specific principal component might be associated with several interelated batch surrogate variables but free from biological associations. In such a case it may be useful to take out such a principal component from the data. The svd() function resolves the data matrix X into X = U*D*V. D is then set to zero for the unwanted principal components and the data X is recalculated. If you use the default center=TRUE make sure you also use prince() with the default center=TRUE. Using different settings for center for the two functions is not fully compatible.

## Value

a matrix which is the new data with the specified principal components removed.

### Note

requires the package impute.

### Author(s)

Martin Lauss

### Examples

```
# data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
   paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
# patient annotations as a data.frame, annotations should be numbers and factors but not characters.
# rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

## pca on unadjusted data
res1<-prince(g,o,top=10)
prince.plot(res1)

## take out pc1
gadj3<-kill.pc(g,pc=1)
prince.plot(prince(gadj3,o,top=10))
```

---

## prince — *Linear models of prinicipal conponents dependent on sample annotations*

---

### Description

The function calculates the principal components of the data and finds associations between the prinicpal components and the sample annotations using linear regressions.

### Usage

```
prince(g, o, top = 25, imputeknn = F, center = T, permute = F)
```

### Arguments

| | |
|---|---|
| g | the input data in form of a matrix with features as rows and samples as columns. |
| o | the corresponding sample annotations in the form of a data.frame. rownames (o) must be identical to colnames (g). o can contain factors with 2 or more levels and numeric variables; no character variables are allowed. NAs are allowed (these samples are ommited in lm()). |

| | |
|---|---|
| top | the number of top principal components to be analyzed, default is set to 25. |
| imputeknn | default=FALSE. missing values in the data matrix can be imputed by imputeknn=TRUE. The function knn.impute from the package impute is used with default settings. |
| center | default=TRUE. the features are mean-centered before singular value decomposiiton. this is a pre-requisite for principal component analysis, change only if you are really convinced that centering is not necessary. |
| permute | default=FALSE. if set to TRUE a permuted data matrix is generated with the values for each feature shuffled. The linear models are also calculated for this permuted dataset. |

## Details

To calculate the principal components of a data matrix, the function prcomp is used. The function prcomp uses singular value decomposition instead of eigen decomposition of the covariance matrix, the actual principal component analysis. As prcomp cannot handle missing values they have to be imputed beforehands, imputeknn=TRUE can be used for k-nearest neighbor imputation from package impute, k is 10. A linear model is perfomed to find associations of the principal components with a dataframe of sample annotations. The f-statistics of lm(principal component i ~ sample annotation j) is used to derive the p-value for these associations. The results can be plotted using the prince.plot() function. If permute=TRUE the analysis will be repeated on permuted data, in which for each feature the values have been randomly shuffled.

## Value

a list as a prince object with components

| | |
|---|---|
| pr | the output list of the prcomp function with the principal components contained in pr$x. |
| linp | matrix containing the F-test p-values from lm(principal component~sample annotation). |
| rsquared | matrix containing the R-squared values from lm(principal component~sample annotation). |
| prop | numeric vector containing the percentage of the overall variation for each principal component. |
| o | the input data.frame containing the patient annotations. |
| pr.perm | if permute=T it contains the outcome list from prcomp for the permuted data. |
| linpperm | if permute=T it contains the p-values for the permuted data. |
| rsquaredperm | if permute=T it contains the R-squared values for the permuted data. |
| propperm | if permute=T it contains the percentages of variation for the permuted data. |
| imputed | if imputeknn=T it contains the data matrix with the imputed values. |

## Note

requires the package impute

### Author(s)

Martin Lauss

### Examples

```
## data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
## patient annotations as a data.frame, annotations should be numbers and factor but not characters.
## rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

## calculate principal components and use linear models to calculate
## their dependence on sample annotations
res1<-prince(g,o,top=10,permute=TRUE)
str(res1)
res1$linp # to see the p values
```

---

## prince.plot *Heatmap of the associations between principal components and sample annotations*

---

### Description

This function uses the calculations performed by prince() and plots the log10 of p-values obtained by the linear models.

### Usage

```
prince.plot(prince, label = colnames(prince$o), smallest = -20, note = F,
            notecol = "black", notecex = 1,
            breaks = seq(-20,0,length.out=100), col = heat.colors(99),
            margins = c(5, 7), key = T, cexRow = 1, cexCol = 1,
            xlab = "Principal Components (Variation)", colsep = NULL,
            rowsep = NULL, sepcolor = "black", sepwidth = c(0.05,0.05),
            Rsquared = F, breaksRsquared = seq(0,1,length.out=100) )
```

### Arguments

| | |
|---|---|
| prince | an object generated by the function prince() |
| label | vector containing names of the sample annotation. |

| | |
|---|---|
| smallest | a numeric value. log10(p-values) less than smallest are set to smallest for plotting. default = -20. e.g. a log10 p-value of -37 will be set to -20. Smallest has to be less than 0. |
| note | set to TRUE to print the p-values in the cells of the plot. |
| notecol | to determine the color of the notes |
| notecex | to determine the font size of the notes |
| breaks | either a number (default=100) or a numeric vector (default would be seq(-20,0,length.out=100)) of breaks for the colors |
| col | a vector of colors with a length of breaks-1. default=heat.colors(99). |
| margins | a vector with the margins for columns and rows. default=c(5,7). |
| key | whether the color key should be printed, default=TRUE. |
| cexRow | font size of label. default=1. |
| cexCol | font size of column labeling. default=1. |
| xlab | an additional character vector to print at the bottom. |
| colsep | same as in heatmap.2 function. |
| rowsep | same as in heatmap.2 function. |
| sepcolor | same as in heatmap.2 function. |
| sepwidth | same as in heatmap.2 function. |
| Rsquared | set to TRUE to print Rsquared values instead of log10 p-values. Missing values in object o will flaw the comparability of p-values. |
| breaksRsquared | same format as argument breaks. will be used when Rsquared=TRUE |

## Details

This plot indicates batch effects, and shows the influence of the biological annotations on the overall variation. The input has to be an object generated by the prince() function. The plot is based on the heatmap.2 function from the gplots package. Colors, breaks, font size and margins can be changed and cell notes can be added. The log10 of p-values from linear model are plotted. If Rsquared is TRUE the R-squared values from linear model are plotted instead.

## Note

requires the package gplots

## Author(s)

Martin Lauss

## Examples

```
## data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
```

```
## patient annotations as a data.frame, annotations should be numbers and factors but not characters.
## rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

## calculate principal components and use linear models to calculate their
## dependence on sample annotations
res1<-prince(g,o,top=10)
## plot p-values as heatmap
prince.plot(prince=res1)
```

---

`prince.var.plot` *ScreePlot of the data variation covered by the principal components*

---

## Description

To identify the number of top principal components with relevant variation, this function plots the variation contained in the pc for both observed data and reshuffled data.

## Usage

```
prince.var.plot(g, show.top = dim(g)[2], imputeknn = F,
                center = T, npermute = 10)
```

## Arguments

| | |
|---|---|
| `g` | the input data in form of a matrix with features as rows and samples as columns. |
| `show.top` | the number of top principal components to be shown in the plot (cannot exceed ncol(g) or nrow(g)). |
| `imputeknn` | default=FALSE. missing values in the data matrix can be imputed by imputeknn=TRUE. The function knn.impute from the package impute is used with default settings. |
| `center` | default=TRUE. the features are mean-centered before singular value decomposiiton. this is a pre-requisite for principal component analysis, change only if you are really convinced that centering is not necessary. |
| `npermute` | the number of reshuffled datasets. default=10. A permuted data matrix is generated with the values for each feature shuffled. From the permutation sets the median percentage of variation for each principal component is taken. |

## Details

The function prcomp() is used to calculate the variation of the data contained in the principal components. As prcomp cannot handle missing values they have to be imputed beforehands, using imputeknn=TRUE.

## Value

a list with components

`real.variation` a vector containing the percentage of variation for each principal component in the observed data.

`permuted.variation` a matrix containing the percentages of variation for each principal component in the reshuffled data sets.

## Note

requires the package impute

## Author(s)

Martin Lauss

## Examples

```
## data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1
     # the first 100 features show higher values in the samples 26:50

## to plot the variations
res2<-prince.var.plot(g,show.top=50,npermute=10)
str(res2)
```

---

# quickadjust.ref *Batch adjustment by median-scaling to a reference batch*

---

## Description

The function adjusts for batches by adjusting the median of the features to the median of a reference batch.

## Usage

```
quickadjust.ref(g, batches, refbatch)
```

## Arguments

| | |
|---|---|
| `g` | the input data in form of a matrix with features as rows and samples as columns. NAs are allowed. |
| `batches` | a factor with two or more levels and with same length as ncol(g), each level has to contain at least 2 samples. |
| `refbatch` | a character that determines the reference batch. this character has to be a level of batches. |

## Details

The batches are adjusted to a reference batch. The values of the reference batch remain unchanged. For each feature the median of the batches is determined. NAs are removed for median() calculations. The median of the reference batch is divided by the median of the non-reference batch, which is the scaling factor. All values in the non-reference batch are multiplied by the scaling factor. This way all batches will have the same median as the reference batch for each feature. Scaling factors get inflated when data was already feature-centered before. Hence, this method is only advisable for uncentered data. This is a quick and simple method of batch adjustment, that probably does not work for every batch effect, especially when sample numbers per batch are low. The efficiency of batch adjustment can be checked by prince.plot(prince(g,o)) or prince.plot(prince(g, data.frame(batch,batch))).

## Value

a list with components

`adjusted.data` A numeric matrix which is the adjusted dataset.

`scaling.factors`
A numeric matrix containing the scaling factors for each feature in each batch.

## Author(s)

Martin Lauss

## Examples

```
## The function is currently defined as
# data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
# patient annotations as a data.frame, annotations should be numbers and factors but not characters.
# rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
              Numeric1=rnorm(50),row.names=colnames(g))

##unadjusted.data
res1<-prince(g,o,top=10)
prince.plot(res1)

##batch adjustment
gadj2<-quickadjust.ref(g,o$Factor1,"B")
str(gadj2)
##prince.plot
prince.plot(prince(gadj2$adjusted.data,o,top=10))
    # note the high number of variation covered by the first principal component.
    # This is caused by infalted scaling factor as the features of the
    # input matrix g are already centered around zero.
```

```
# this adjustment method should be used only on uncentered data.
```

---

`quickadjust.zero` *Batch adjustment by median-centering*

---

## Description

The function centers the median of each feature in each batch to zero.

## Usage

```
quickadjust.zero(g, batches)
```

## Arguments

| | |
|---|---|
| `g` | the input data in form of a matrix with features as rows and samples as columns. NAs are allowed. |
| `batches` | a factor with two or more levels and with same length as ncol(g), each level has to contain at least 2 samples. |

## Details

The values of a feature are split up into batches and each part is median-centered to zero, i.e. the batch median is subtracted from every value in the batch. As a result the feature will have a median of zero in all batches. NAs are removed for median() calculations. This is a quick and simple method of batch adjustment, that probably does not work for every batch effect, especially when sample numbers per batch are low. The efficiency of batch adjustment can be checked by prince.plot(prince(g,o)) or prince.plot(prince(g, data.frame(batch,batch))).

## Value

A numeric matrix which is the adjusted dataset.

## Author(s)

Martin Lauss

## Examples

```
# data as a matrix
set.seed(100)
g<-matrix(nrow=1000,ncol=50,rnorm(1000*50),dimnames=list(paste("Feature",1:1000),
          paste("Sample",1:50)))
g[1:100,26:50]<-g[1:100,26:50]+1 # the first 100 features show higher values in the samples 26:50
# patient annotations as a data.frame, annotations should be numbers and factors but not characters.
# rownames have to be the same as colnames of the data matrix
set.seed(200)
o<-data.frame(Factor1=factor(c(rep("A",25),rep("B",25))),
              Factor2=factor(rep(c("A","B"),25)),
```

```
                Numeric1=rnorm(50),row.names=colnames(g))

    ##unadjusted.data
    res1<-prince(g,o,top=10)
    prince.plot(res1)

    ##batch adjustment
    gadj1<-quickadjust.zero(g,o$Factor1)
    ##prince.plot
    prince.plot(prince(gadj1,o,top=10))
```

# Index