

# Package ‘ChemometricsWithR’

January 27, 2015

**Type** Package

**Title** Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences

**Version** 0.1.8

**Author** Ron Wehrens

**Maintainer** Ron Wehrens <ron.wehrens@gmail.com>

## Description

The package provides functions and scripts used in the book “Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences” by Ron Wehrens, Springer (2011).

**Depends** ChemometricsWithRData (>= 0.1.3), MASS, pls

**Suggests** nnet, randomForest, ada, rrcov, sfsmisc, ipred, fastICA, rda, TIMP, class, e1071, rpart, cluster, ALS, ptw, dtw, boot, leaps, lars, elasticnet, subselect, kohonen, signal, mclust

**License** GPL (>= 2)

**LazyLoad** yes

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-08-04 12:15:21

## R topics documented:

ChemometricsWithR-package . . . . .	2
AdjRkl . . . . .	2
classvec2classmat . . . . .	3
Error . . . . .	4
Evaluation . . . . .	5
GA . . . . .	6
gini . . . . .	8
MCR . . . . .	8
PCA . . . . .	10
PCA.plot . . . . .	12
pick.peaks . . . . .	13

SA . . . . .	14
unsigned.range . . . . .	16

<b>Index</b>	<b>17</b>
--------------	-----------

ChemometricsWithR-package

*Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences*

## Description

Package containing scripts and functions from the book "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences", by Ron Wehrens. All R code in the book is available in demos: typing `demo(chapter2)` will run the examples from the second chapter. A list of errata can be found in the `inst` directory of the installed package.

## Author(s)

Ron Wehrens <ron.wehrens@iasma.it>

## References

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

## See Also

[chemometrics](#)

AdjRkl

*Adjusted Rand Index*

## Description

The Adjusted Rand Index is a measure of similarity for two groupings or clusterings. A value of 1 indicates total agreement.

## Usage

`AdjRkl(part1, part2)`

## Arguments

<code>part1</code>	First partitioning.
<code>part2</code>	Second partitioning.

**Value**

Number.

**Author(s)**

Ron Wehrens

**References**

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

**Examples**

```
if (require("kohonen")) {
  data(wines, package = "kohonen")
  wines.dist <- dist(scale(wines))
  wines.sl <- hclust(wines.dist, method = "single")
  wines.cl <- hclust(wines.dist, method = "complete")

  AdjRk1(cutree(wines.sl, 4), cutree(wines.cl, 4))
} else {
  cat("Package kohonen not available.\nInstall it by typing 'install.packages(\"kohonen\")'")
}
```

---

classvec2classmat	<i>Convert a classification vector into a matrix or the other way around.</i>
-------------------	---

---

**Description**

Functions toggle between a matrix representation, where class membership is indicated with one '1' and for the rest zeros at each row, and an class vector (maybe integers or class names). The classification matrix contains one column per class. Conversion from a class matrix to a class vector assigns each row to the column with the highest value. An optional argument can be used to assign only those objects that have a probability higher than a certain threshold (default is 0).

**Usage**

```
classvec2classmat(yvec)
classmat2classvec(yamat, threshold=0)
```

**Arguments**

yvec	class vector. Usually integer values, but other types are also allowed.
yamat	class matrix: every column corresponds to a class.
threshold	only classify into a class if the probability is larger than this threshold.

**Value**

`classvec2classmat` returns the classification matrix, where each column consists of zeros and ones; `classmat2classvec` returns a class vector (integers).

**Author(s)**

Ron Wehrens

**Examples**

```
classes <- c(rep(1, 5), rep(2, 7), rep(3, 9))
classmat <- classvec2classmat(classes)
classmat
classmat2classvec(classmat)
```

---

Error

*Often-used error functions*

---

**Description**

Error functions for classification and regression

**Usage**

```
rms(x, y)
err.rate(x, y)
```

**Arguments**

`x`, `y` True or predicted values, either numbers or factors.

**Value**

Function `rms` returns the root-mean-square error for real-valued `x` and `y` vectors. Function `err.rate` returns the fraction of non-matching cases in `x` and `y` (real numbers or factors).

**Author(s)**

Ron Wehrens

**References**

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

---

Evaluation	<i>Evaluation function examples for SA- or GA-based variable selection in classification applications.</i>
------------	--

---

### Description

Two examples of functions that can be used in variable selection for classification. The outcome of these functions should be maximized by the optimization.

### Usage

```
lda.loofun(x, grouping, subset, ...)  
pls.cvfun(x, response, subset, ...)
```

### Arguments

x	Data matrix: independent variables used by <code>eval.fun</code>
grouping	Class vector, possibly a factor
response	Dependent variable, typically a real number
subset	A vector containing the indices of the variables to be included
...	Further arguments, such as the number of latent variables to use in <code>plscvfun</code>

### Details

The evaluation function should give high values for good subsets, and low values for bad subsets. The `lda.loofun` function simply counts the number of correct predictions in LOO crossvalidation, and subtracts the number of variables in the subset. Function `pls.cvfun` returns the mean squared error of cross-validation.

### Value

One value indicating the quality of the subset

### Author(s)

Ron Wehrens

### References

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

### See Also

[GA](#), [SA](#)

**Description**

A set of functions implementing simple variable selection in classification applications using genetic algorithms.

**Usage**

```
GAfun(X, C, eval.fun, kmin, kmax, popsize = 20, niter = 50,
      mut.prob = 0.05, ...)
GAfun2(X, C, eval.fun, kmin, kmax, popsize = 20, niter = 50,
      mut.prob = 0.05, ...)

GA.init.pop(popsiz, nvar, kmin, kmax)
GA.select(pop, number, qlts, min.qlt = 0.4, qlt.exp = 1)
GA.mut(subset, maxvar, mut.prob = 0.01)
GA.X0(subset1, subset2)
```

**Arguments**

X	Data matrix: independent variables used by eval.fun
C	Class vector, used by eval.fun
eval.fun	evaluation function. Should take a data matrix, a class vector (or factor), and a subset argument
kmin	Minimal number of variables to retain
kmax	Maximal number of variables to retain
popsiz	Size of the GA population
niter	Number of iterations
mut.prob	Mutation probability
...	Further arguments to the evaluation function
nvar	The total number of variables to choose from
pop, subset, subset1, subset2	A (part of a) population of trial solutions
number	The number of trial solutions that may produce offspring
qlts	Vector of quality measures for members in a population
min.qlt	Minimal quality of a trial solution to be considered as a future parent
qlt.exp	Quality scaling parameter: the larger this number, the more discrimination between good and bad solutions, and the more greedy the search characteristics
maxvar	Number of variables to choose from

## Details

The function generates a population of trial solutions, each containing a number of variables to be retained. For every member of the population, the evaluation function calculates a quality measure, which determines the chance of that member to create offspring. In a process of "survival of the fittest", this leads to subsets for which the evaluation function has a maximal value.

The initialization is done randomly. Selection is simple threshold selection. Mutation swaps variables in or out of the subset; the cross-over type is uniform. Functions `GA.init.pop`, `GA.select`, `GA.mut` and `GA.XO` are auxiliary functions, not meant to be called directly by the user.

## Value

Functions `GAFun` and `GAFun2` both return a list containing the following fields:

<code>best</code>	The best subset
<code>best.q</code>	The quality of the best subset
<code>n.iter</code>	The number of iterations

In addition, the outcome of `GAFun2` also contains

<code>qualities</code>	A matrix containing the best, median and worst quality value throughout the optimization
------------------------	--

## Author(s)

Ron Wehrens

## References

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

## See Also

[Evaluation, SA](#)

## Examples

```
if (require("pls")) {
  data(gasoline, package = "pls")
  ## Usually more iterations are needed
  GAobj <- GAFun(gasoline$NIR, gasoline$octane,
               eval.fun = pls.cvfun, niter = 20,
               kmin = 3, kmax = 25, ncomp = 2)
  GAobj
} else {
  cat("Package pls not available.\nInstall it by typing 'install.packages(\"pls\")'")
}
```

---

`gini`

*Gini impurity index for cart objects*

---

### Description

A simple implementation of the Gini impurity index for classification and regression trees. Not meant to be called directly - included for demonstration purposes.

### Usage

```
gini(x, class, splitpoint)
```

### Arguments

<code>x</code>	Numeric vector of length <code>n</code> .
<code>class</code>	Class labels, length <code>n</code> .
<code>splitpoint</code>	Tentative split point.

### Value

The Gini impurity index, given a certain split point, a vector of possible splits, and a vector of class labels. Lower values indicate more pure leaves.

### Author(s)

Ron Wehrens

### References

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

---

MCR

*Functions for Multivariate Curve Resolution*

---

### Description

Multivariate Curve Resolution, or MCR, decomposes a bilinear matrix into its pure components. A classical example is a matrix consisting of a series of spectral measurements on a mixture of chemicals for following the reaction. At every time point, a spectrum is measured that is a linear combination of the pure spectra. The goal of MCR is to resolve the pure spectra and concentration profiles over time.



**Usage**

```
mcr(x, init, what = c("row", "col"), convergence = 1e-08,
    maxit = 50)
opa(x, ncomp)
efa(x, ncomp)
```

**Arguments**

x	Data matrix
init	Initial guess for pure compounds
what	Whether the pure compounds are rows or columns of the data matrix
convergence	Convergence criterion
maxit	Maximal number of iterations
ncomp	Number of pure compounds

**Details**

MCR uses repeated application of least-squares regression to find pure profiles and spectra. The method is iterative; both EFA and OPA are methods to provide initial guesses.

**Value**

Function `mcr` returns a list containing

C	An estimate of the pure "concentration profiles"
S	An estimate of the pure "spectra"
resids	The residuals of the final decomposition
rms	Root-mean-square values of the individual iterations

Function `opa` returns a list containing

pure.compounds:	A matrix containing <code>ncomp</code> pure compounds, usually spectra at specific time points
selected:	The wavelengths leading to the estimates of the pure concentration profiles

Function `efa` returns a list containing

pure.compounds:	A matrix containing <code>ncomp</code> pure compounds, usually concentration profiles at specific wavelengths
forward:	The development of the singular values of the reduced data matrix when increasing the number of columns in the forward direction
backward:	The development of the singular values of the reduced data matrix when increasing the number of columns in the backward direction

Usually, `opa` and `efa` are employed in opposite ways: if `opa` is used to find the "purest" row of a data matrix, one would typically employ `efa` to find the "purest" column, and vice versa.

**Author(s)**

Ron Wehrens

**References**

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

**Examples**

```
data(bdata)
D1.efa <- efa(bdata$d1, 3)
matplot(D1.efa$forward, type = "l")
matplot(D1.efa$backward, type = "l")
matplot(D1.efa$pure.comp, type = "l")

D1.opa <- opa(bdata$d1, 3)
matplot(D1.opa$pure.comp, type = "l")

if (require(MASS)) {
  D1.mcr.efa <- mcr(bdata$d1, D1.efa$pure.comp, what = "col")
  matplot(D1.mcr.efa$C, type = "l", main = "Concentration profiles")
  matplot(t(D1.mcr.efa$S), type = "l", main = "Pure spectra")
}
```

---

PCA

*Principal Component Analysis*

---

**Description**

Functions for PCA: creating a PCA object, extracting variances, scores and loadings for individual PCs, projecting new data in the PC space, and reconstruction using a limited number of PCs.

**Usage**

```
PCA(X, warn = TRUE)
## S3 method for class 'PCA'
summary(object, varperc = 90, pc.select = c(1:5,10), ...)
variances(object, npc = maxpc)
## S3 method for class 'PCA'
scores(object, npc = maxpc, ...)
## S3 method for class 'PCA'
loadings(object, npc = maxpc, ...)
reconstruct(object, npc = maxpc)
project(object, npc = maxpc, newdata, ldngs)
```

**Arguments**

<code>X</code>	a matrix, with each row representing an object.
<code>warn</code>	logical, whether or not to give a warning when the data are not mean-centered.
<code>object</code>	an object of class "PCA" (see below).
<code>varperc</code>	variance threshold in the summary function.
<code>...</code>	extra arguments, e.g., for printing the variance table ( <code>digits = ...</code> ).
<code>pc.select</code>	PCs to be included in the summary function.
<code>npc</code>	the number of PCs to be returned.
<code>newdata</code>	data (with the same number of variables as the original data) that are to be projected into the space of the first <code>npc</code> PCs.
<code>ldngs</code>	loadings to be used; by default the PCA loadings.

**Value**

Function `PCA` returns an object of class "PCA" with components

<code>scores</code>	object weights per PC.
<code>loadings</code>	variable weights per PC.
<code>var</code>	variance explained per PC.
<code>totalvar</code>	The total variance in the data set.

Function `summary.PCA` gives a short summary of the PCA model, stating how many PCs are needed to cover a certain percentage of the total variance, and for selected PCs gives the (cumulative) variance explained.

Function `variances` returns the variances associated with each PC.

Function `scores` returns the scores associated with each PC.

Function `loadings` returns the loadings associated with each PC.

Function `reconstruct` returns the reconstruction of the original data matrix, based on `npc` PCs.

Function `project` projects the new data into the subspace spanned by the given loadings. If argument `ldngs` is given, arguments `pcamod` and `npc` are not needed.

**Author(s)**

Ron Wehrens

**References**

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

**See Also**

[plot.PCA](#)

**Examples**

```
data(wines, package = "ChemometricsWithRData")
wines.PC <- PCA(scale(wines))
```

PCA.plot

*Principal Component Analysis plotting functions***Description**

Plotting functions for PCA: for scores, loadings, scores and loadings simultaneously (a biplot), and variances (a screeplot, where the log of the explained variance is plotted for each PC).

**Usage**

```
## S3 method for class 'PCA'
scoreplot(object, pc = c(1, 2), pcscores = scores(object),
          show.names = FALSE, xlab, ylab, xlim, ylim, ...)
## S3 method for class 'PCA'
loadingplot(object, pc = c(1, 2), ploadings = loadings(object),
            scalefactor = 1, add = FALSE, show.names = FALSE,
            xlab, ylab, xlim, ylim, col = "blue", min.length =
            0.01, varnames = NULL, ...)
## S3 method for class 'PCA'
biplot(x, pc = c(1,2),
       show.names = c("none", "scores", "loadings", "both"),
       score.col = 1, loading.col = "blue",
       min.length = .01, varnames = NULL, ...)
screeplot(object, type = c("scree", "percentage"), npc, ...)
```

**Arguments**

object, x	an object of class "PCA" (see below).
pc	which PCs to show.
pcscores	matrix of scores, by default the scores of the PCA model object.
show.names	show names rather than plotting symbols. For loadingplot and scoreplot a logical (default: FALSE), for biplot one of 'scores', 'loadings', 'both' or 'none' (default).
xlab, ylab, xlim, ylim, col	graphical parameters of the plot.
ploadings	matrix of loadings, by default the loadings of the PCA model object.
scalefactor	scaling factor for the loadings; used internally, when the loadingplot function is called from within biplot.PCA.
add	logical, whether to add to the existing plot (again, useful when loadingplot is called from within biplot.PCA).

npc	how many PCs to show in the scree plot (starting from 1).
type	show a real screeplot (scree) or show the percentage of variance explained (percentage).
score.col, loading.col	colours of the scores and loadings in a biplot.
min.length	minimal length of loading vectors to be plotted by arrows. Vectors that are too short lead to warning messages, are not interesting, and only clutter the graphic.
varnames	alternative vector of variable names.
...	Graphical arguments passed on to lower-level plotting functions.

### Details

Score plots and loading plots show the amount of explained variance at the axis labels only when PCA has been performed at mean-centered data.

### Author(s)

Ron Wehrens

### References

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

### See Also

[PCA](#)

### Examples

```
data(wines, package = "ChemometricsWithRData")
wines.PC <- PCA(scale(wines))
scoreplot(wines.PC, col = wine.classes, pch = wine.classes)
loadingplot(wines.PC, show.names = TRUE)
biplot(wines.PC, score.col = wine.classes)
screeplot(wines.PC)
```

---

pick.peaks	<i>Peak-picking function.</i>
------------	-------------------------------

---

### Description

Function to identify local maxima in a vector, typically a spectrum or a chromatogram.

### Usage

```
pick.peaks(x, span)
```

**Arguments**

x	Numerical vector.
span	Neighbourhood, used to define local maxima.

**Value**

A vector containing positions of local maxima in the input data.

**Author(s)**

Ron Wehrens

**Examples**

```
if (require("ptw")) {
  data(lcms, package = "ptw")
  plot(lcms[1,,1], type = "l", xlim = c(1000, 1500))
  abline(v = pick.peaks(lcms[1,,1], 20), col = "blue")
} else {
  cat("Package ptw not available.\nInstall it by typing 'install.packages(\"ptw\")'")
}
```

---

SA

---

*Simulated Annealing for variable selection in classification*


---

**Description**

A set of functions implementing simple variable selection in classification applications using simulated annealing

**Usage**

```
SAfun(x, response, eval.fun, Tinit, niter = 100,
      cooling = 0.05, fraction = 0.3, ...)
SAfun2(x, response, eval.fun, Tinit, niter = 100,
       cooling = 0.05, fraction = 0.3, ...)

SAstep(curr.set, maxvar, fraction = .3, size.dev = 1)
```

**Arguments**

x	Data matrix: independent variables used by eval.fun
response	Class vector, used by eval.fun
eval.fun	evaluation function. Should take a data matrix, a class vector (or factor), and a subset argument
Tinit	Initial temperature

niter	Maximal number of iterations
cooling	Cooling speed
fraction	Size of the desired subset, as a fraction of the total number of variables
...	Further arguments to the evaluation function
curr.set	Current trial solution
maxvar	The total number of variables to choose from
size.dev	Parameter governing the variability in size of subsequent subsets

### Details

Simulated Annealing (SA) starts with a random subset, and proceeds by random moves in the solution space. In this implementation, a new solution may deviate in length at most `size.dev` variables: at most two variables may be swapped in or out at each step. If a step is an improvement, it is unconditionally accepted. If not, acceptance is a stochastic process depending on the current temperature - with high temperatures, "bad" moves are more likely to be accepted than with low temperatures. The process stops after a predefined number of iterations.

### Value

Functions `SAfun` and `SAfun2` both return a list containing the following fields:

best	The best subset
best.q	The quality of the best subset

In addition, the outcome of `SAfun2` also contains

qualities	A vector containing quality values of solutions seen throughout the optimization
accepts	A vector containing logicals indicating which solutions were accepted and which were rejected

### Author(s)

Ron Wehrens

### References

R. Wehrens. "Chemometrics with R - Multivariate Data Analysis in the Natural Sciences and Life Sciences". Springer, Heidelberg, 2011.

### See Also

[Evaluation, GA](#)

**Examples**

```
if (require("pls")) {
  data(gasoline, package = "pls")
  ## usually more than 50 iterations are needed
  SAobj <- SAFun(gasoline$NIR, gasoline$octane,
                eval.fun = pls.cvfun, Tinit = 3,
                fraction = .02, niter = 50, ncomp = 2)
  SAobj
} else {
  cat("Package pls not available.\nInstall it by typing 'install.packages(\"pls\")'")
}
```

---

unsigned.range

*Unsigned range of the data vector including zero.*

---

**Description**

Function returning the range of the data where, if necessary, the range is extended to include zero. Not meant to be called directly by the user.

**Usage**

```
unsigned.range(x)
```

**Arguments**

x                    Numeric vector.

**Value**

A vector of two numbers.

**Note**

From the R stats package (see biplot.default).



# Index

- \*Topic **classif**
  - classvec2classmat, 3
- \*Topic **cluster**
  - AdjRkl, 2
- \*Topic **hplot**
  - PCA.plot, 12
- \*Topic **manip**
  - Error, 4
  - gini, 8
  - MCR, 8
  - pick.peaks, 13
- \*Topic **misc**
  - unsigned.range, 16
- \*Topic **multivariate**
  - PCA, 10
- \*Topic **optimize**
  - Evaluation, 5
  - GA, 6
  - SA, 14
- \*Topic **package**
  - ChemometricsWithR-package, 2
- AdjRkl, 2
- biplot.PCA (PCA.plot), 12
- chemometrics, 2
- ChemometricsWithR
  - (ChemometricsWithR-package), 2
- ChemometricsWithR-package, 2
- classmat2classvec (classvec2classmat), 3
- classvec2classmat, 3
- efa (MCR), 8
- err.rate (Error), 4
- Error, 4
- Evaluation, 5, 7, 15
- GA, 5, 6, 15
- GAFun (GA), 6
- GAFun2 (GA), 6
- gini, 8
- lda.loofun (Evaluation), 5
- loadingplot (PCA.plot), 12
- loadings.PCA (PCA), 10
- MCR, 8
- mcr (MCR), 8
- opa (MCR), 8
- PCA, 10, 13
- PCA.plot, 12
- pick.peaks, 13
- plot.PCA, 11
- plot.PCA (PCA.plot), 12
- pls.cvfun (Evaluation), 5
- project (PCA), 10
- reconstruct (PCA), 10
- rms (Error), 4
- SA, 5, 7, 14
- SAfun (SA), 14
- SAfun2 (SA), 14
- SAstep (SA), 14
- scoreplot (PCA.plot), 12
- scores.PCA (PCA), 10
- screepplot (PCA.plot), 12
- summary.PCA (PCA), 10
- unsigned.range, 16
- variances (PCA), 10