

# Package ‘mvdalab’

October 13, 2022

**Type** Package

**Title** Multivariate Data Analysis Laboratory

**Version** 1.7

**Date** 2022-10-05

**Author** Nelson Lee Afanador, Thanh Tran, Lionel Blanchet, and Richard Baumgartner

**Maintainer** Nelson Lee Afanador <nelson.afanador@gmail.com>

**Description** An open-source implementation of latent variable methods and multivariate modeling tools. The focus is on exploratory analyses using dimensionality reduction methods including low dimensional embedding, classical multivariate statistical tools, and tools for enhanced interpretation of machine learning methods (i.e. intelligible models to provide important information for end-users). Target domains include extension to dedicated applications e.g. for manufacturing process modeling, spectroscopic analyses, and data mining.

**License** GPL-3

**LazyData** true

**Imports** car, ggplot2, MASS, moments, parallel, penalized, plyr,  
reshape2, sn

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-10-05 23:00:14 UTC

## R topics documented:

mvdalab-package . . . . .	3
acfplot . . . . .	4
ap.plot . . . . .	5
bca.cis . . . . .	6
bidiaplots.fit . . . . .	7
BiPlot . . . . .	9
boot.plots . . . . .	11
coef.mvdareg . . . . .	12

coefficients.boots . . . . .	13
coefficients.mvdareg . . . . .	14
coefficientsplot2D . . . . .	16
coefsplot . . . . .	17
College . . . . .	18
contr.niets . . . . .	18
ellipse.mvdalab . . . . .	19
imputeBasic . . . . .	20
imputeEM . . . . .	21
imputeQs . . . . .	22
imputeRough . . . . .	23
introNAs . . . . .	24
jk.after.boot . . . . .	25
loadings . . . . .	26
loadings.boots . . . . .	28
loadingsplot . . . . .	29
loadingsplot2D . . . . .	30
mewma . . . . .	31
model.matrix . . . . .	32
MultCapability . . . . .	33
MVcis . . . . .	35
MVComp . . . . .	36
mvdaboot . . . . .	37
mvdaloo . . . . .	39
mvrnorm.svd . . . . .	41
my.dummy.df . . . . .	42
no.intercept . . . . .	43
pca.nipals . . . . .	43
pcaFit . . . . .	45
PE . . . . .	46
Penta . . . . .	47
perc.cis . . . . .	48
plot.cp . . . . .	49
plot.mvcomp . . . . .	50
plot.mvdareg . . . . .	52
plot.plusminus . . . . .	53
plot.R2s . . . . .	54
plot.smc . . . . .	55
plot.sr . . . . .	56
plot.wrtpls . . . . .	57
plsFit . . . . .	58
plusminus.fit . . . . .	61
plusminus.loo . . . . .	62
plusMinusDat . . . . .	63
plusminusFit . . . . .	64
predict.mvdareg . . . . .	66
print.mvdalab . . . . .	67
print.plusminus . . . . .	68

proCrustes . . . . .	68
R2s . . . . .	70
ScoreContrib . . . . .	71
scoresplot . . . . .	72
SeqimputeEM . . . . .	73
smc . . . . .	74
smc.acfTest . . . . .	75
sr . . . . .	76
T2 . . . . .	78
Wang_Chen . . . . .	79
Wang_Chen_Sim . . . . .	79
weight.boots . . . . .	80
weights . . . . .	81
weightsplot . . . . .	83
weightsplot2D . . . . .	84
wrtpls.fit . . . . .	84
Xresids . . . . .	86
XresidualContrib . . . . .	87
y.loadings . . . . .	88
y.loadings.boots . . . . .	89

**Index** **91**

---

mvdalab-package	<i>Multivariate Data Analysis Laboratory (mvdalab)</i>
-----------------	--

---

### Description

Implementation of latent variables methods. The focus is on explorative analysis using dimensionality reduction methods, such as Principal Component Analysis (PCA), and on multivariate regression based on Partial Least Squares regression (PLS). PLS analyses are supported by embedded bootstrapping and variable selection procedures.

### Details

Package: mvdalab  
 Type: Package  
 Version: 1.0  
 Date: 2015-08-10  
 License: GPL-3

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@gmail.com>), Thanh Tran (<thanh.tran@mvdalab.com>),  
Lionel Blanchet (<lionel.blanchet@mvdalab.com>), Richard Baumgartner (<richard\_baumgartner@merck.com>)  
Maintainer: Nelson Lee Afanador (<nelson.afanador@gmail.com>)

---

acfplot

*Plot of Auto-correlation Funcion*

---

**Description**

This function computes the autocorrelation function estimates for a selected parameter.

**Usage**

```
acfplot(object, parm = NULL)
```

**Arguments**

object	an object of class <code>mvdareg</code> , i.e., <code>plsFit</code> .
parm	a chosen predictor variable; if <code>NULL</code> a random predictor variable is chosen

**Details**

This function computes the autocorrelation function estimates for a selected parameter, via `acf`, and generates a graph that allows the analyst to assess the need for an autocorrelation adjustment in the [smc](#).

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

This function is built using the `acf` function in the **stats** R package.

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer-Verlag.

**See Also**

[smc](#), [smc.acfTest](#)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
acfplot(mod1, parm = NULL)
```

---

`ap.plot`*Actual versus Predicted Plot and Residuals versus Predicted*

---

### Description

This function provides the actual versus predicted and actual versus residuals plot as part of a model assessment

### Usage

```
ap.plot(object, ncomp = object$ncomp, verbose = FALSE)
```

### Arguments

<code>object</code>	an object of class <code>mvdareg</code> , i.e., <code>plsFit</code> .
<code>ncomp</code>	number of components used in the model assessment
<code>verbose</code>	output results as a data frame

### Details

This function provides the actual versus predicted and residuals versus predicted plot as part of model a assessment across the desired number of latent variables. A smooth fit (dashed line) is added in order to detect curvature in the fit.

### Value

The output of `ap.plot` is a two facet graph for actual versus predicted and residuals versus predicted plots.

### Author(s)

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

### See Also

[plsFit](#)

### Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
ap.plot(mod1, ncomp = 2)
```

---

 bca.cis

*Bias-corrected and Accelerated Confidence Intervals*


---

### Description

Computes bootstrap BCa confidence intervals for chosen parameters for PLS models fitted with `validation = "oob"`.

### Usage

```
bca.cis(object, conf = .95, type = c("coefficients",
  "loadings", "weights"))
```

### Arguments

<code>object</code>	an object of class "mvdareg", i.e. plsFit.
<code>conf</code>	desired confidence level
<code>type</code>	input parameter vector

### Details

The function computes the bootstrap BCa confidence intervals for any fitted `mvdareg` model. Should be used in instances in which there is reason to suspect the percentile intervals. Results provided across all latent variables (LVs). As such, it may be slow for models with a large number of LVs.

### Value

A `bca.cis` object contains component results for the following:

<code>ncomp</code>	number of components in the model
<code>variables</code>	variable names
<code>boot.mean</code>	mean of the bootstrap
BCa percentiles	confidence intervals
proportional bias	calculated bias
<code>skewness</code>	skewness of the bootstrap distribution
<code>a</code>	acceleration constant

### Author(s)

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

## References

There are many references explaining the bootstrap and its implementation for confidence interval estimation. Among them are:

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman & Hall.

Hinkley, D.V. (1988) Bootstrap methods (with Discussion). *Journal of the Royal Statistical Society, B*, 50, 312:337, 355:370.

## See Also

[plsFit](#), [mvdaboot](#), [boot.plots](#)

## Examples

```
data(Penta)
## Number of bootstraps set to 250 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "oob", boots = 250)
bca.cis(mod1, conf = .95, type = "coefficients")
## Not run:
bca.cis(mod1, conf = .95, type = "loadings")
bca.cis(mod1, conf = .95, type = "weights")

## End(Not run)
```

---

bidiagpls.fit

*Bidiag2 PLS*

---

## Description

Bidiagonalization algorithm for PLS1

## Usage

```
bidiagpls.fit(X, Y, ncomp, ...)
```

## Arguments

X	a matrix of observations. NAs and Infs are not allowed.
Y	a vector. NAs and Infs are not allowed.
ncomp	the number of components to include in the model (see below).
...	additional arguments. Currently ignored.

**Details**

This function should not be called directly, but through `plsFit` with the argument `method="bidiagpls"`. It implements the Bidiag2 scores algorithm.

**Value**

An object of class `mvdareg` is returned. The object contains all components returned by the underlying fit function. In addition, it contains the following:

<code>loadings</code>	X loadings
<code>weights</code>	weights
<code>D2</code>	bidiag2 matrix
<code>iD2</code>	inverse of bidiag2 matrix
<code>Ymean</code>	mean of reponse variable
<code>Xmeans</code>	mean of predictor variables
<code>coefficients</code>	regression coefficients
<code>y.loadings</code>	y-loadings
<code>scores</code>	X scores
<code>R</code>	orthogonal weights
<code>Y</code>	scaled response values
<code>Yactual</code>	actual response values
<code>fitted</code>	fitted values
<code>residuals</code>	residuals
<code>Xdata</code>	X matrix
<code>iPreds</code>	predicted values
<code>y.loadings2</code>	scaled y-loadings
<code>fit.time</code>	model fitting time
<code>val.method</code>	validation method
<code>ncomp</code>	number of latent variables
<code>contrasts</code>	contrast matrix used
<code>method</code>	PLS algorithm used
<code>scale</code>	scaling used
<code>validation</code>	validation method
<code>call</code>	model call
<code>terms</code>	model terms
<code>model</code>	fitted model

**Author(s)**

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>), Thanh Tran (<[thanh.tran@mvdalab.com](mailto:thanh.tran@mvdalab.com)>)



## References

Indahl, Ulf G., (2014) The geometry of PLS1 explained properly: 10 key notes on mathematical properties of and some alternative algorithmic approaches to PLS1 modeling. Journal of Chemometrics, 28, 168:180.

Manne R., Analysis of two partial-least-squares algorithms for multi-variate calibration. Chemom. Intell. Lab. Syst. 1987; 2: 187:197.

## See Also

[plsFit](#)

---

BiPlot	<i>Generates a biplot from the output of an 'mvdareg' and 'mvdapca' object</i>
--------	--

---

## Description

Generates a 2D Graph of both the scores and loadings for both "mvdareg" and "mvdapca" objects.

## Usage

```
BiPlot(object, diag.adj = c(0, 0), axis.scaling = 2,
       cov.scale = FALSE, comps = c(1, 2),
       col = "red", verbose = FALSE)
```

## Arguments

object	an object of class "mvdareg" or "mvdapca".
diag.adj	adjustment to singular values. see details.
axis.scaling	a graphing parameter for extending the axis.
cov.scale	implement covariance scaling
comps	the components to illustrate on the graph
col	the color applied to the scores
verbose	output results as a data frame

## Details

"BiPlot" is used to extract a 2D graphical summary of the scores and loadings of PLS and PCA models.

The singular values are scaled so that the approximation becomes  $X = GH'$ :

$X = ULV' = (UL^{\alpha_1})(L^{\alpha_2}V') = GH'$ , and where  $\alpha_2$  is to  $(1 = \alpha)$

The rows of the G matrix are plotted as points, corresponding to observations. The rows of the H matrix are plotted as vectors, corresponding to variables. The choice of alpha determines the following:

$c(0, 0)$ : variables are scaled to unit length and treats observations and variables symmetrically.

$c(0, 1)$ : This biplot attempts to preserve relationships between variables wherein the distance between any two rows of  $G$  is proportional to the Mahalanobis distance between the same observations in the data set.

$c(1, 0)$ : This biplot attempts to preserve the distance between observations where in the positions of the points in the biplot are identical to the score plot of first two principal components, but the distance between any two rows of  $G$  is equal to the Euclidean distance between the corresponding observations in the data set.

`cov.scale = FALSE` sets `diag.adj` to  $c(0, 0)$  and multiplies  $G$  by  $\sqrt{n - 1}$  and divides  $H$  by  $\sqrt{n - 1}$ . In this biplot the rows of  $H$  approximate the variance of the corresponding variable, and the distance between any two points of  $G$  approximates the Mahalanobis distance between any two rows.

Additional scalings may be implemented.

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

### References

SAS Stat Studio 3.11 (2009), User's Guide.

Additional information pertaining to biplots can be obtained from the following:

Friendly, M. (1991), SAS System for Statistical Graphics, SAS Series in Statistical Applications, Cary, NC: SAS Institute

Gabriel, K. R. (1971), "The Biplot Graphical Display of Matrices with Applications to Principal Component Analysis," *Biometrika*, 58(3), 453–467.

Golub, G. H. and Van Loan, C. F. (1989), *Matrix Computations*, Second Edition, Baltimore: Johns Hopkins University Press.

Gower, J. C. and Hand, D. J. (1996), *Biplots*, London: Chapman & Hall.

Jackson, J. E. (1991), *A User's Guide to Principal Components*, New York: John Wiley & Sons.

### Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
BiPlot(mod1, diag.adj = c(0, 0), axis.scaling = 2, cov.scale = FALSE)

## Not run:
data(Penta)
mod2 <- pcaFit(Penta[, -1], ncomp = 4)
BiPlot(mod2, diag.adj = c(0, 0), axis.scaling = 2.25, cov.scale = FALSE)

## End(Not run)
```

**Description**

This takes an mvdaReg object fitted with validation = "oob" and produces a graph of the bootstrap distribution and its corresponding normal quantile plot for a variable of interest.

**Usage**

```
boot.plots(object, comp = object$ncomp, parm = NULL,  
           type = c("coefs", "weights", "loadings"))
```

**Arguments**

object	an object of class "mvdaReg", i.e., a plsFit.
comp	latent variable from which to generate the bootstrap distribution for a specific parameter
parm	a parameter for which to generate the bootstrap distribution
type	input parameter vector

**Details**

The function generates the bootstrap distribution and normal quantile plot for a bootstrapped mvdaReg model given validation = "oob" for type = c("coefs", "weights", "loadings"). If parm = NULL a parameter is chosen at random.

**Value**

The output of boot.plots is a histogram of the bootstrap distribution and the corresponding normal quantile plot.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**See Also**

[bca.cis](#)

**Examples**

```
data(Penta)  
## Number of bootstraps set to 300 to demonstrate flexibility  
## Use a minimum of 1000 (default) for results that support bootstrapping  
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],  
              ncomp = 2, validation = "oob", boots = 300)  
boot.plots(mod1, type = "coefs", parm = NULL)
```

**Description**

Functions to extract information from mvdaLab objects.

**Usage**

```
## S3 method for class 'mvdareg'  
coef(object, ncomp = object$ncomp, type = c("coefficients",  
      "loadings", "weights", "y.loadings"), conf = .95, ...)
```

**Arguments**

object	an mvdareg object, i.e. a plsFit.
ncomp	the number of components to include in the model (see below).
type	specify model parameters to return.
conf	for a bootstrapped model, the confidence level to use.
...	additional arguments. Currently ignored.

**Details**

These are usually called through their generic functions `coef` and `residuals`, respectively. `coef.mvdareg` is used to extract the regression coefficients, loadings, or weights of a PLS model.

If `comps` is missing (or is `NULL`), all parameter estimates are returned.

**Value**

coefficients	a named vector, or matrix, of coefficients.
loadings	a named vector, or matrix, of loadings.
weights	a named vector, or matrix, of weights.
y.loadings	a named vector, or matrix, of y.loadings.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**See Also**

[coef](#), [coefficients.boots](#), [coefficients](#), [loadings](#), [loadings.boots](#), [weights](#), [weight.boots](#)

**Examples**

```

data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "loo")
coef(mod1, type = "coefficients")

```

---

coefficients.boots      *BCa Summaries for the coefficient of an mvdareg object*

---

**Description**

Computes bootstrap BCa confidence intervals for regression coefficients, along with expanded bootstrap summaries.

**Usage**

```
coefficients.boots(object, ncomp = object$ncomp, conf = 0.95)
```

**Arguments**

object	an object of class mvdareg, i.e., a plsFit.
ncomp	number of components in the model
conf	desired confidence level

**Details**

The function computes the bootstrap BCa confidence intervals for fitted mvdareg models where validation = "oob". Should be used in instances in which there is reason to suspect the percentile intervals. Results provided across all latent variables or for specific latent variables via ncomp.

**Value**

A coefficients.boots object contains component results for the following:

variable	variable names
actual	Actual loading estimate using all the data
BCa percentiles	confidence intervals
boot.mean	mean of the bootstrap
skewness	skewness of the bootstrap distribution
bias	estimate of bias w.r.t. the loading estimate
Bootstrap Error	estimate of bootstrap standard error
t value	approximate 't-value' based on the Bootstrap Error
bias t value	approximate 'bias t-value' based on the Bootstrap Error

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

There are many references explaining the bootstrap. Among them are:

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman & Hall.

Hinkley, D.V. (1988) Bootstrap methods (with Discussion). *Journal of the Royal Statistical Society, B*, 50, 312:337, 355:370.

**See Also**

[coef](#), [coefficients](#), [coefsplot](#), [coefficients](#)

**Examples**

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "oob", boots = 300)
coefficients.boots(mod1, ncomp = 2, conf = .95)
```

---

`coefficients.mvdareg` *Extract Summary Information Pertaining to the Coefficients resulting from a PLS model*

---

**Description**

Functions to extract regression coefficient bootstrap information from mvdalab objects.

**Usage**

```
## S3 method for class 'mvdareg'
coefficients(object, ncomp = object$ncomp, conf = .95, ...)
```

**Arguments**

<code>object</code>	an mvdareg object. A fitted model.
<code>ncomp</code>	the number of components to include in the model (see below).
<code>conf</code>	for a bootstrapped model, the confidence level to use.
<code>...</code>	additional arguments. Currently ignored.

**Details**

`coefficients` is used to extract a bootstrap summary of the regression of a PLS model.

If `comps` is missing (or is `NULL`), summaries for all regression estimates are returned. Otherwise, if `comps` is given parameters for a model with only the requested component `comps` is returned.

Bootstrap summaries provided are for actual regression coefficients, bootstrap percentiles, bootstrap mean, skewness, and bias. These summaries can also be extracted using `coefficients.boots`

**Value**

A `coefficients` object contains a data frame with columns:

<code>variable</code>	variable names
<code>Actual</code>	Actual loading estimate using all the data
<code>BCa percentiles</code>	confidence intervals
<code>boot.mean</code>	mean of the bootstrap
<code>skewness</code>	skewness of the bootstrap distribution
<code>bias</code>	estimate of bias w.r.t. the loading estimate
<code>Bootstrap Error</code>	estimate of bootstrap standard error
<code>t value</code>	approximate 't-value' based on the Bootstrap Error
<code>bias t value</code>	approximate 'bias t-value' based on the Bootstrap Error

**Author(s)**

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

**See Also**

[coef](#), [coefficients.boots](#), [coefficients](#)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
coefficients(mod1)
```

---

coefficientsplot2D	<i>2-Dimensional Graphical Summary Information Pertaining to the Coefficients of a PLS</i>
--------------------	--

---

### Description

Functions to extract 2D graphical coefficients information from mvdaLab objects.

### Usage

```
coefficientsplot2D(object, comps = c(1, 2), verbose = FALSE)
```

### Arguments

object	an mvdaLab object.
comps	a vector of length 2 corresponding to the number of components to include.
verbose	output results as a data frame

### Details

coefficientsplot2D is used to extract a graphical summary of the coefficients of a PLS model. If comp is missing (or is NULL), a graphical summary for the 1st and 2nd components is returned.

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdaLab.com>)

### See Also

[loadingsplot2D](#), [weightsplot2D](#)

### Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
coefficientsplot2D(mod1, comp = c(1, 2))
```



---

coefsplo	<i>Graphical Summary Information Pertaining to the Regression Coefficients</i>
----------	--

---

## Description

Functions to extract regression coefficient bootstrap information from mvdalab objects.

## Usage

```
coefsplo(object, ncomp = object$ncomp, conf = 0.95, verbose = FALSE)
```

## Arguments

object	an mvdareg object. A fitted model.
ncomp	the number of components to include.
conf	for a bootstrapped model, the confidence level to use.
verbose	output results as a data frame

## Details

coefficients is used to extract a graphical summary of the regression coefficients of a PLS model.

If comps is missing (or is NULL), a graphical summary for the nth component regression estimates are returned. Otherwise, if comps is given parameters for a model with only the requested component comps is returned.

Bootstrap graphical summaries provided are when method = oob.

## Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

## Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
coefsplo(mod1, ncomp = 1:2)
```

---

College	<i>Data for College Level Examination Program and the College Qualification Test</i>
---------	--

---

**Description**

Scores obtained from 87 college students on the College Level Examination Program and the College Qualification Test.

**Usage**

College

**Format**

A data frame with 87 observations and the following 3 variables.

Science Science (CQT) - numerical vector

Social Social science and history (CLEP) - numerical vector

Verbal Verbal (CQT) - numerical vector

**Source**

Johnson, R.A., Wichern, D.W. (2002) Applied Multivariate Statistical Analysis. Prentice Hall.

---

contr.niets	<i>Cell Means Contrast Matrix</i>
-------------	-----------------------------------

---

**Description**

This function generates a cell means contrast matrix to support PLS models.

**Usage**

```
contr.niets(n, contrasts)
```

**Arguments**

n A vector of levels for a factor, or the number of levels.

contrasts a logical indicating whether contrasts should be computed; set to FALSE in order to generate required contrast matrix.

**Details**

This function uses `contr.treatment` to generate a cell means contrast matrix in support of PLS models.

**Value**

For datasets with categorical variables it produces the needed design matrix.

**Author(s)**

Nelson Lee Afanador

**Examples**

```
# Three levels
levels <- LETTERS[1:3]
contr.niets(levels)

# Two levels
levels <- LETTERS[1:2]
contr.niets(levels)
```

---

 ellipse.mvdalab

*Ellipses, Data Ellipses, and Confidence Ellipses*


---

**Description**

This function draws confidence ellipses for covariance and correlation matrices derived from either a matrix or dataframe.

**Usage**

```
ellipse.mvdalab(data, center = c(0, 0), radius = "chi", scale = TRUE,
  segments = 51, level = c(0.95, 0.99), plot.points = FALSE, pch = 1, size = 1,
  alpha = 0.5, verbose = FALSE, ...)
```

**Arguments**

data	A dataframe
center	2-element vector with coordinates of center of ellipse.
radius	Use of the Chi or F Distributions for setting the radius of the confidence ellipse
scale	use correlation or covariance matrix
segments	number of line-segments used to draw ellipse.
level	draw elliptical contours at these (normal) probability or confidence levels.
pch	symbols to use for scores
size	size to use for scores
alpha	transparency of scores
plot.points	Should the points be added to the graph.
verbose	output results as a data frame
...	additional arguments. Currently ignored.

**Details**

ellipse uses the singular value decomposition in order to generate the desired confidence regions. The default confidence ellipse is based on the chisquare statistic.

**Value**

Returns a graph with the ellipses at the stated as levels, as well as the ellipse coordinates.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Fox, J. (2008) Applied Regression Analysis and Generalized Linear Models, Second Edition. Sage.  
Fox, J. and Weisberg, S. (2011) An R Companion to Applied Regression, Second Edition, Sage.

**Examples**

```
data(iris)
ellipse.mvdalab(iris[, 1:2], plot.points = FALSE)
ellipse.mvdalab(iris[, 1:2], center = colMeans(iris[, 1:2]), plot.points = TRUE)
```

---

imputeBasic

*Naive imputation of missing values.*

---

**Description**

Imputes the mean or median for continous variables; highest frequency for categorical variables.

**Usage**

```
imputeBasic(data, Init = "mean")
```

**Arguments**

data	a dataset with missing values
Init	For continous variables impute either the mean or median

**Details**

A completed data frame is returned. For numeric variables, NAs are replaced with column means or medians. For categorical variables, NAs are replaced with the most frequent levels. If object contains no NAs, it is returned unaltered.

**Value**

imputeBasic returns a list containing the following components:

```
Imputed.DataFrame           Final imputed data frame
Imputed.Missing.Continuous  Imputed continous values
Imputed.Missing.Factors     Imputed categorical values
```

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
dat <- introNAs(iris, percent = 25)
imputeBasic(dat)
```

---

imputeEM

*Expectation Maximization (EM) for imputation of missing values.*


---

**Description**

Missing values are iteratively updated via an EM algorithm.

**Usage**

```
imputeEM(data, impute.ncomps = 2, pca.ncomps = 2, CV = TRUE, Init = "mean",
         scale = TRUE, iters = 25, tol = .Machine$double.eps^0.25)
```

**Arguments**

data	a dataset with missing values.
impute.ncomps	integer corresponding to the minimum number of components to test.
pca.ncomps	minimum number of components to use in the imputation.
CV	Use cross-validation in determining the optimal number of components to retain for the final imputation.
Init	For continous variables impute either the mean or median.
scale	Scale variables to unit variance.
iters	For continous variables impute either the mean or median.
tol	the threshold for assessing convergence.

**Details**

A completed data frame is returned that mirrors a `model.matrix`. NAs are replaced with convergence values as obtained via EM. If object contains no NAs, it is returned unaltered.

**Value**

`imputeEM` returns a list containing the following components:

<code>Imputed.DataFrames</code>	A list of imputed data frames across <code>impute.comps</code>
<code>Imputed.Continuous</code>	A list of imputed values, at each EM iteration, across <code>impute.comps</code>
<code>CV.Results</code>	Cross-validation results across <code>impute.comps</code>
<code>ncomps</code>	<code>impute.comps</code>

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>), Thanh Tran (<thanh.tran@mvdalab.com>)

**References**

B. Walczak, D.L. Massart. Dealing with missing data, Part I. *Chemom. Intell. Lab. Syst.* 58 (2001); 15:27

**Examples**

```
dat <- introNAs(iris, percent = 25)
imputeEM(dat)
```

---

imputeQs

*Quartile Naive Imputation of Missing Values*

---

**Description**

Missing value imputed as 'Missing'.

**Usage**

```
imputeQs(data)
```

**Arguments**

`data` a dataset with missing values

**Details**

A completed data frame is returned. For continuous variables with missing values, missing values are replaced with 'Missing', while the non-missing values are replaced with their corresponding quartile assignment. For categorical variable with missing values, missing values are replaced with 'Missing'. This procedure can greatly increases the dimensionality of the data.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
dat <- introNAs(iris, percent = 25)
imputeQs(dat)
```

---

imputeRough	<i>Naive Imputation of Missing Values for Dummy Variable Model Matrix</i>
-------------	---

---

**Description**

After generating a cell means model matrix, impute expected values (mean or median for continuous; highest frequency for categorical).

**Usage**

```
imputeRough(data, Init = "mean")
```

**Arguments**

data	a dataset with missing values
Init	For continuous variables impute either the mean or median

**Details**

A completed data frame is returned that mirrors a `model.matrix`. NAs are replaced with column means or medians. If object contains no NAs, it is returned unaltered. This is the starting point for `imputeEM`.

**Value**

`imputeRough` returns a list containing the following components:

Initials	Imputed values
Pre.Imputed	Pre-imputed data frame
Imputed.Dataframe	Imputed data frame

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
dat <- introNAs(iris, percent = 25)
imputeRough(dat)
```

---

introNAs

*Introduce NA's into a Dataframe*

---

**Description**

Function for testing missing value imputation algorithms

**Usage**

```
introNAs(data, percent = 25)
```

**Arguments**

data	a dataset without missing values.
percent	the percent data that should be randomly assigned as missing

**Details**

A completed data frame is returned with the desired percentage of missing data. NAs are assigned at random.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
dat <- introNAs(iris)
dat
```



---

jk.after.boot	<i>Jackknife After Bootstrap</i>
---------------	----------------------------------

---

### Description

This function calculates the jackknife influence values from a bootstrap output `mvdareg` object and plots the corresponding jackknife-after-bootstrap plot.

### Usage

```
jk.after.boot(object, ncomp = object$ncomp,  
              type = c("coefficients", "loadings", "weights"),  
              parm = NULL)
```

### Arguments

<code>object</code>	an <code>mvdareg</code> object. A fitted model.
<code>ncomp</code>	the component number to include in the jackknife-after-bootstrap plot assessment.
<code>type</code>	input parameter vector.
<code>parm</code>	predictor variable for which to perform the assessment. if <code>NULL</code> one will be chosen at random.

### Details

The centred jackknife quantiles for each observation are estimated from those bootstrap samples in which a particular observation did not appear. These are then plotted against the influence values.

The resulting plots are useful diagnostic tools for looking at the way individual observations affect the bootstrap output.

The plot will consist of a number of horizontal dotted lines which correspond to the quantiles of the centred bootstrap distribution. For each data point the quantiles of the bootstrap distribution calculated by omitting that point are plotted against the jackknife values. The observation number is printed below the plots. To make it easier to see the effect of omitting points on quantiles, the plotted quantiles are joined by line segments. These plots provide a useful diagnostic tool in establishing the effect of individual observations on the bootstrap distribution. See the references below for some guidelines on the interpretation of the plots.

### Value

There is no returned value but a graph is generated on the current graphics display.

### Author(s)

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

## References

- Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.
- Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society, B*, 54, 83:127.

## Examples

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "oob", boots = 300)
jk.after.boot(mod1, type = "coefficients")
## Not run:
jk.after.boot(mod1, type = "loadings")
jk.after.boot(mod1, type = "weights")

## End(Not run)
```

---

loadings

*Summary Information Pertaining to the Bootstrapped Loadings*

---

## Description

Functions to extract loadings bootstrap information from mvdalab objects.

## Usage

```
## S3 method for class 'mvdareg'
loadings(object, ncomp = object$ncomp, conf = .95, ...)
```

## Arguments

object	an mvdareg or mvdapaca object. A fitted model.
ncomp	the number of components to include in the model (see below).
conf	for a bootstrapped model, the confidence level to use.
...	additional arguments. Currently ignored.

## Details

loadings is used to extract a summary of the loadings of a PLS or PCA model. If ncomps is missing (or is NULL), summaries for all loadings estimates are returned. Otherwise, if comps is given parameters for a model with only the requested component comps is returned.

Bootstrap summaries are provided for mvdareg objects where validation = "oob". These summaries can also be extracted using loadings.boots

**Value**

A loadings object contains a data frame with columns:

variable	variable names
Actual	Actual loading estimate using all the data
BCa percentiles	confidence intervals
boot.mean	mean of the bootstrap
skewness	skewness of the bootstrap distribution
bias	estimate of bias w.r.t. the loading estimate
Bootstrap Error	estimate of bootstrap standard error
t value	approximate 't-value' based on the Bootstrap Error
bias t value	approximate 'bias t-value' based on the Bootstrap Error

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

There are many references explaining the bootstrap. Among them are:

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society, B*, 54, 83:127.

**See Also**

[loadingsplot](#), [loadings.boots](#), [loadingsplot2D](#)

**Examples**

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "oob", boots = 300)
loadings(mod1, ncomp = 2, conf = .95)

data(iris)
pc1 <- pcaFit(iris)
loadings(pc1)
```

loadings.boots

*BCa Summaries for the loadings of an mvdareg object***Description**

Computes bootstrap BCa confidence intervals for the loadings, along with expanded bootstrap summaries.

**Usage**

```
loadings.boots(object, ncomp = object$ncomp, conf = .95)
```

**Arguments**

object	an object of class "mvdareg", i.e., a plsFit.
ncomp	number of components in the model.
conf	desired confidence level.

**Details**

The function computes the bootstrap BCa confidence intervals for fitted mvdareg models where `validation = "oob"`. Should be used in instances in which there is reason to suspect the percentile intervals. Results provided across all latent variables or for specific latent variables via `ncomp`.

**Value**

A `loadings.boots` object contains component results for the following:

variable	variable names
actual	Actual loading estimate using all the data
BCa percentiles	confidence intervals
boot.mean	mean of the bootstrap
skewness	skewness of the bootstrap distribution
bias	estimate of bias w.r.t. the loading estimate
Bootstrap Error	estimate of bootstrap standard error
t value	approximate 't-value' based on the Bootstrap Error
bias t value	approximate 'bias t-value' based on the Bootstrap Error

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

## References

There are many references explaining the bootstrap. Among them are:

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society, B*, 54, 83:127.

## Examples

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "oob", boots = 300)
loadings.boots(mod1, ncomp = 2, conf = .95)
```

---

loadingsplot

*Graphical Summary Information Pertaining to the Loadings*

---

## Description

Functions to extract graphical loadings information from `mvdareg` and `mvdapca` object.

## Usage

```
loadingsplot(object, ncomp = object$ncomp, conf = 0.95, verbose = FALSE)
```

## Arguments

<code>object</code>	an <code>mvdareg</code> or <code>mvdapca</code> object.
<code>ncomp</code>	the number of components to include.
<code>conf</code>	for a bootstrapped model, the confidence level to use.
<code>verbose</code>	output results as a data frame

## Details

"loadingsplot" is used to extract a graphical summary of the loadings of a PLS model. If "comps" is missing (or is NULL), a graphical summary for the nth component estimates are returned. Otherwise, if comps is given parameters for a model with only the requested component comps is returned.

Bootstrap graphical summaries provided are when "method = oob"

## Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**See Also**

[loadings](#), [loadings.boots](#), [loadingsplot2D](#)

**Examples**

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "oob", boots = 300)
loadingsplot(mod1, ncomp = 1:2)
```

---

loadingsplot2D	<i>2-Dimensional Graphical Summary Information Pertaining to the Loadings of a PLS or PCA Analysis</i>
----------------	--

---

**Description**

Functions to extract 2D graphical loadings information from mvdalab objects.

**Usage**

```
loadingsplot2D(object, comps = c(1, 2), verbose = FALSE)
```

**Arguments**

object	an mvdareg or mvdapca object.
comps	a vector or length 2 corresponding to the number of components to include.
verbose	output results as a data frame

**Details**

loadingsplot2D is used to extract a graphical summary of the loadings of a PLS model. If comp is missing (or is NULL), a graphical summary for the 1st and 2nd components are returned.

**Author(s)**

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

**See Also**

[coefficientsplot2D](#), [weightsplot2D](#)

**Examples**

```

data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "loo")
loadingsplot2D(mod1, comp = c(1, 2))

## Not run:
data(Penta)
mod2 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "loo")
loadingsplot2D(mod2, comp = c(1, 2))

## End(Not run)

data(iris)
pc1 <- pcaFit(iris)
loadingsplot2D(pc1, comp = c(1, 2))

```

---

mewma	<i>Generates a Hotelling's T2 Graph of the Multivariate Exponentially Weighted Average</i>
-------	--

---

**Description**

Generates a Hotelling's T2 Graph for mewma objects.

**Usage**

```

mewma(X, phase = 1, lambda = 0.2, conf = c(0.95, 0.99),
      asymptotic.form = FALSE)

```

**Arguments**

X	a dataframe.
phase	designates whether the confidence limits should reflect the current data frame, phase = 1 or future observations, phase = 2.
lambda	EWMA smoothing parameter
conf	the confidence level(s) to use for upper control limit.
asymptotic.form	use asymptotic convergence parameter for scaling the covariance matrix.

**Details**

mewma is used to generates a Hotelling's T2 graph for the multivariate EWMA.

**Value**

The output of `mewma` is a graph of Hotelling's T2 for the Multivariate EWMS, and a list containing a data frame of univariate EWMA's and the multivariate EWMA T2 values.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Lowry, Cynthia A., et al. "A multivariate exponentially weighted moving average control chart." *Technometrics* 34.1 (1992): 46:53.

**Examples**

```
mewma(iris[, -5], phase = 1, lambda = 0.2, conf = c(0.95, 0.99),
      asymptotic.form = FALSE)
```

---

`model.matrix`

`model.matrix` creates a design (or model) matrix.

---

**Description**

This function returns the `model.matrix` of an `mvdareg` object.

**Usage**

```
## S3 method for class 'mvdareg'
model.matrix(object, ...)
```

**Arguments**

`object` an `mvdareg` object  
`...` additional arguments. Currently ignored.

**Details**

"`model.matrix.mvdareg`" is used to returns the `model.matrix` of an `mvdareg` object.

**Value**

The design matrix for a PLS model with the specified formula and data.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)



**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
model.matrix(mod1)
```

---

 MultCapability

*Principal Component Based Multivariate Process Capability Indices*


---

**Description**

Provides three multivariate capability indices for correlated multivariate processes based on Principal Component Analysis.

**Usage**

```
MultCapability(data, lsIs, usIs, targets, ncomps = NULL, Target = FALSE)
```

**Arguments**

data	a multivariable dataset
lsIs	is the vector of the lower specification limits
usIs	is the vector of the upper specification limits
targets	is the vector of the target of the process
ncomps	is the number of principal component to use
Target	Use targets for calculation of univariate PpKs; otherwise the average is used

**Details**

ncomps has to be set prior to running the analysis. The user is strongly encouraged to use `pcaFit` in order to determine the optimal number of principal components using cross-validation.

When the parameter targets is not specified, then is estimated of centered way as  $targets = lsIs + (usIs - lsIs)/2$ .

Ppk values are provided to allow the user to compare the multivariate results to the univariate results.

**Value**

A list with the following elements:

For `mpca_wang`, the following is returned:

ncomps	number of components used
mcp_wang	index greater than 1, the process is capable
mcpk_wang	index greater than 1, the process is capable

mcpm\_wang        index greater than 1, the process is capable  
 mcpmk\_wang       index greater than 1, the process is capable

For mcp\_xe, the following is returned:

ncomps            number of components used  
 mcp\_wang\_2       index greater than 1, the process is capable  
 mcpk\_wang\_2      index greater than 1, the process is capable  
 mcpm\_wang\_2      index greater than 1, the process is capable  
 mcpmk\_wang\_2     index greater than 1, the process is capable

For mpca\_wang\_2, the following is returned:

ncomps            number of components used  
 mcp\_xe            index greater than 1, the process is capable  
 mcpk\_xe           index greater than 1, the process is capable  
 mcpm\_xe           index greater than 1, the process is capable  
 mcpmk\_xe          index greater than 1, the process is capable

For Ppk, the following is returned:

Individual.Ppks  
                   univariate Ppks; index greater than 1, the process is capable

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

### References

- Wang F, Chen J (1998). Capability index using principal components analysis. *Quality Engineering*, 11, 21-27.
- Xekalaki E, Perakis M (2002). The Use of principal component analysis in the assessment of process capability indices. *Proceedings of the Joint Statistical Meetings of the American Statistical Association, The Institute of Mathematical Statistics, The Canadian Statistical Society*. New York.
- Wang, C (2005). Constructing multivariate process capability indices for short-run production. *The International Journal of Advanced Manufacturing Technology*, 26, 1306-1311.
- Scagliarini, M (2011). Multivariate process capability using principal component analysis in the presence of measurement errors. *AStA Adv Stat Anal*, 95, 113-128.
- Santos-Fernandez E, Scagliarini M (2012). "MPCI: An R Package for Computing Multivariate Process Capability Indices". *Journal of Statistical Software*, 47(7), 1-15, URL <http://www.jstatsoft.org/v47/i07/>.

**Examples**

```

data(Wang_Chen_Sim)
ls1s1 <- c(2.1, 304.5, 304.5)
us1s1 <- c(2.3, 305.1, 305.1)
targets1 <- c(2.2, 304.8, 304.8)

MultCapability(Wang_Chen_Sim, ls1s = ls1s1, us1s = us1s1, targets = targets1, ncomps = 2)

data(Wang_Chen)
targets2 <- c(177, 53)
ls1s2 <- c(112.7, 32.7)
us1s2 <- c(241.3, 73.3)

MultCapability(Wang_Chen, ls1s = ls1s2, us1s = us1s2, targets = targets2, ncomps = 1)

```

---

MVcis

*Calculate Hotelling's T2 Confidence Intervals*


---

**Description**

Calculate joint confidence intervals (Hotelling's T2 Intervals).

**Usage**

```
MVcis(data, segments = 51, level = .95, Vars2Plot = c(1, 2), include.zero = F)
```

**Arguments**

data	a multivariable dataset to compare to means
segments	number of line-segments used to draw ellipse.
level	draw elliptical contours at these (normal) probability or confidence levels.
Vars2Plot	variables to plot
include.zero	add the zero axis to the graph output

**Details**

This function calculates the Hotelling's T2 Intervals for a mean vector.

Assumption:

Population is a random sample from a multivariate population.

If the confidence ellipse does not cover  $c(0, 0)$ , we reject the NULL that the joint confidence region is equal to zero (at the stated alpha level).

**Value**

This function returns the Hotelling's T2 confidence intervals for the p-variates and its corresponding confidence ellipse at the stated confidence level.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Johnson, R.A., Wichern, D.W. (2002) Applied Multivariate Statistical Analysis. Prentice Hall.

**See Also**

[MVComp](#)

**Examples**

```
data(College)
MVCis(College, Vars2Plot = c(1, 2), include.zero = TRUE)
```

---

MVComp

*Traditional Multivariate Mean Vector Comparison*

---

**Description**

Performs a traditional multivariate comparison of mean vectors drawn from two populations.

**Usage**

```
MVComp(data1, data2, level = .95)
```

**Arguments**

data1	a multivariable dataset to compare to.
data2	a multivariable dataset to compare.
level	draw elliptical contours at these (normal) probability or confidence levels.

**Details**

This function provides a T2-statistic for testing the equality of two mean vectors. This test is appropriate for testing two populations, assuming independence.

Assumptions:

The sample for both populations is a random sample from a multivariate population.

- Both populations are independent
- Both populations are multivariate normal
- Covariance matrices are approximately equal

**Value**

This function returns the simultaneous confidence intervals for the p-variates and its corresponding confidence ellipse at the stated confidence level.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Johnson, R.A., Wichern, D.W. (2002) Applied Multivariate Statistical Analysis. Prentice Hall.

**Examples**

```
data(College)
dat1 <- College
#Generate a 'fake' difference of 15 units
dat2 <- College + matrix(rnorm(nrow(dat1) * ncol(dat1), mean = 15),
  nrow = nrow(dat1), ncol = ncol(dat1))

Comparison <- MVComp(dat1, dat2, level = .95)
Comparison
plot(Comparison, Diff2Plot = c(1, 2), include.zero = FALSE)
plot(Comparison, Diff2Plot = c(1, 2), include.zero = TRUE)

plot(Comparison, Diff2Plot = c(2, 3), include.zero = FALSE)
plot(Comparison, Diff2Plot = c(2, 3), include.zero = TRUE)

data(iris)
dat1b <- iris[, -5]
#Generate a 'fake' difference of .5 units
dat2b <- dat1b + matrix(rnorm(nrow(dat1b) * ncol(dat1b), mean = .5),
  nrow = nrow(dat1b), ncol = ncol(dat1b))

Comparison2 <- MVComp(dat1b, dat2b, level = .90)
plot(Comparison2, Diff2Plot = c(1, 2), include.zero = FALSE)
plot(Comparison2, Diff2Plot = c(1, 2), include.zero = TRUE)

plot(Comparison2, Diff2Plot = c(3, 4), include.zero = FALSE)
plot(Comparison2, Diff2Plot = c(3, 4), include.zero = TRUE)
```

**Description**

When validation = 'oob' this routine effects the bootstrap procedure for mvdareg objects.

**Usage**

```
mvdaboot(X, Y, ncomp, method = "bidiagpls", scale = FALSE, n_cores, parallel,
         boots, ...)
```

**Arguments**

X	a matrix of observations. NAs and Infs are not allowed.
Y	a vector. NAs and Infs are not allowed.
ncomp	the number of components to include in the model (see below).
method	PLS algorithm used.
scale	scaling used.
n_cores	No. of cores to run for parallel processing. Currently set to 2 (4 max).
parallel	should parallelization be used.
boots	No. of bootstrap samples when validation = 'oob'
...	additional arguments. Currently ignored.

**Details**

This function should not be called directly, but through the generic function `plsFit` with the argument `validation = 'oob'`.

**Value**

Provides the following bootstrapped results as a list for `mvdareg` objects:

<code>coefficients</code>	fitted values
<code>weights</code>	weights
<code>loadings</code>	loadings
<code>ncomp</code>	number of latent variables
<code>bootstraps</code>	No. of bootstraps
<code>scores</code>	scores
<code>cvR2</code>	bootstrap estimate of <code>cvR2</code>
<code>PRESS</code>	bootstrap estimate of prediction error sums of squares
<code>MSPRESS</code>	bootstrap estimate of mean squared error prediction sums of squares
<code>boot.means</code>	bootstrap mean of bootstrapped parameters
<code>RMSPRESS</code>	bootstrap estimate of mean squared error prediction sums of squares
<code>D2</code>	bidiag2 matrix
<code>iD2</code>	Inverse of bidiag2 matrix
<code>y.loadings</code>	normalized y-loadings
<code>y.loadings2</code>	non-normalized y-loadings
<code>MSPRESS.632</code>	.632 corrected estimate of <code>MSPRESS</code>
<code>oob.fitted</code>	out-of-bag PLS fitted values
<code>RMSPRESS.632</code>	.632 corrected estimate of <code>RMSPRESS</code>
<code>in.bag</code>	bootstrap samples used for model building at each bootstrap

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>), Thanh Tran (<thanh.tran@mvdalab.com>)

**References**

There are many references explaining the bootstrap and its implementation for confidence interval estimation. Among them are:

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman & Hall.

Hinkley, D.V. (1988) Bootstrap methods (with Discussion). *Journal of the Royal Statistical Society, B*, 50, 312:337, 355:370.

NOTE: This function is adapted from `mvr` in package **pls** with extensive modifications by Nelson Lee Afanador and Thanh Tran.

**See Also**

[plsFit](#), [mvdaloo](#)

**Examples**

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "oob", boots = 300)

## Run line below to see bootstrap results
## mod1$validation
```

---

mvdaloo

*Leave-one-out routine for mvdareg objects*

---

**Description**

When `validation = 'loo'` this routine effects the leave-one-out cross-validation procedure for `mvdareg` objects.

**Usage**

```
mvdaloo(X, Y, ncomp, weights = NULL, method = "bidiagpls",
        scale = FALSE, boots = NULL, ...)
```

**Arguments**

X	a matrix of observations. NAs and Infs are not allowed.
Y	a vector. NAs and Infs are not allowed.
ncomp	the number of components to include in the model (see below).
weights	currently not in use
method	PLS algorithm used
scale	scaling used
boots	not applicable for validation = 'loo'
...	additional arguments. Currently ignored.

**Details**

This function should not be called directly, but through the generic function `plsFit` with the argument `validation = 'loo'`.

**Value**

Provides the following bootstrapped results as a list for `mvdareg` objects:

cvR2	leave-one-out estimate of cvR2.
PRESS	leave-one-out estimate of prediction error sums of squares.
MSPRESS	leave-one-out estimate of mean squared error prediction sums of squares.
RMSPRESS	leave-one-out estimate of mean squared error prediction sums of squares.
in.bag	leave-one-out samples used for model building.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>), Thanh Tran (<thanh.tran@mvdalab.com>)

**References**

NOTE: This function is adapted from `mvr` in package **pls** with extensive modifications by Nelson Lee Afanador and Thanh Tran.

**See Also**

[plsFit](#), [mvdaboot](#)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, method = "bidiagpls", validation = "loo")

mod1$validation$cvR2
mod1$validation$PRESS
mod1$validation$MSPRESS
```



```
mod1$validation$RMSPRESS
mod1$validation$in.bag
```

---

mvrnorm.svd	<i>Simulate from a Multivariate Normal, Poisson, Exponential, or Skewed Distribution</i>
-------------	--

---

### Description

Produces one or more samples from the specified multivariate distribution.

### Usage

```
mvrnorm.svd(n = 1, mu = NULL, Sigma = NULL, tol = 1e-06, empirical = FALSE,
            Dist = "normal", skew = 5, skew.mean = 0, skew.sd = 1,
            poisson.mean = 5)
```

### Arguments

n	the number of samples required.
mu	a vector giving the means of the variables.
Sigma	a positive-definite symmetric matrix specifying the covariance matrix of the variables.
tol	tolerance (relative to largest variance) for numerical lack of positive-definiteness in Sigma.
empirical	logical. If true, mu and Sigma specify the empirical not population mean and covariance matrix.
Dist	desired distribution.
skew	amount of skew for skewed distributions.
skew.mean	mean for skewed distribution.
skew.sd	standard deviation for skewed distribution.
poisson.mean	mean for poisson distribution.

### Details

"mvrnorm.svd" The matrix decomposition is done via svd

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

### Examples

```
Sigma <- matrix(c(1, .5, .5, .5, 1, .5, .5, .5, 1), 3, 3)
Means <- rep(0, 3)

Sim.dat.norm <- mvrnorm.svd(n = 1000, Means, Sigma, Dist = "normal")
plot(as.data.frame(Sim.dat.norm))

Sim.dat.pois <- mvrnorm.svd(n = 1000, Means, Sigma, Dist = "poisson")
plot(as.data.frame(Sim.dat.pois))

Sim.dat.exp <- mvrnorm.svd(n = 1000, Means, Sigma, Dist = "exp")
plot(as.data.frame(Sim.dat.exp))

Sim.dat.skew <- mvrnorm.svd(n = 1000, Means, Sigma, Dist = "skewnorm")
plot(as.data.frame(Sim.dat.skew))
```

---

my.dummy.df

*Create a Design Matrix with the Desired Constrasts*

---

### Description

This function generates a dummy variable data frame in support various functions.

### Usage

```
my.dummy.df(data, contr = "contr.niets")
```

### Arguments

data	a data frame
contr	an optional list. See the contrasts.arg of model.matrix.default.

### Details

my.dummy.df takes a data.frame with categorical variables, and returns a data.frame in which all the categorical variables columns are expanded as dummy variables.

The argument contr is passed to the default contr.niets; contr.helmert, contr.poly, contr.sum, contr.treatment are also supported.

### Value

For datasets with categorical variables it produces the specified design matrix.

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
data(iris)
my.dummy.df(iris)
```

---

no.intercept	<i>Delete Intercept from Model Matrix</i>
--------------	---

---

**Description**

Deletes the intercept from a model matrix.

**Usage**

```
no.intercept(mm)
```

**Arguments**

mm	Model Matrix
----	--------------

**Value**

A model matrix without intercept column.

**Author(s)**

Nelson Lee Afanador

---

pca.nipals	<i>PCA with the NIPALS algorithm</i>
------------	--------------------------------------

---

**Description**

Implements the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm for computing PCA scores and loadings and intermediate steps to convergence.

**Usage**

```
pca.nipals(data, ncomps = 1, Iters = 500, start.vec = NULL, tol = 1e-08)
```

**Arguments**

data	A dataframe
ncomps	the number of components to include in the analysis.
Iters	Number of iterations
start.vec	option for choosing your own starting vector
tol	tolernace for convergence

**Details**

The NIPALS algorithm is a popular algorithm in multivariate data analysis for computing PCA scores and loadings. This function is specifically designed to help explore the subspace prior to convergence. Currently only mean-centering is employed.

**Value**

Loadings	Loadings obtained via NIPALS
Scores	Scores obtained via NIPALS
Loading.Space	A list containing the intermediate step to convergence for the loadings
Score.Space	A list containing the intermediate step to convergence for the scores

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

There are many good references for the NIPALS algorithm:

Risvik, Henning. "Principal component analysis (PCA) & NIPALS algorithm." (2007).

Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." *Chemometrics and intelligent laboratory systems* 2.1-3 (1987): 37:52.

**Examples**

```
my.nipals <- pca.nipals(iris[, 1:4], ncomps = 4, tol = 1e-08)
names(my.nipals)

#Check results
my.nipals$Loadings
svd(scale(iris[, 1:4], scale = FALSE))$v

nipals.scores <- data.frame(my.nipals$Scores)
names(nipals.scores) <- paste("np", 1:4)
svd.scores <- data.frame(svd(scale(iris[, 1:4], scale = FALSE))$u)
names(svd.scores) <- paste("svd", 1:4)
Scores. <- cbind(nipals.scores, svd.scores)
plot(Scores.)

my.nipals>Loading.Space
my.nipals$Score.Space
```

---

pcaFit *Principal Component Analysis*

---

**Description**

Function to perform principal component analysis.

**Usage**

```
pcaFit(data, scale = TRUE, ncomp = NULL)
```

**Arguments**

data	an data frame containing the variables in the model.
scale	should scaling to unit variance be used.
ncomp	the number of components to include in the model (see below).

**Details**

The calculation is done via singular value decomposition of the data matrix. Dummy variables are automatically created for categorical variables.

**Value**

pcaFit returns a list containing the following components:

loadings	X loadings
scores	X scores
D	eigenvalues
Xdata	X matrix
Percent.Explained	Explained variation in X
PRESS	Prediction Error Sum-of-Squares
ncomp	number of latent variables
method	PLS algorithm used

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Everitt, Brian S. (2005). An R and S-Plus Companion to Multivariate Analysis. Springer-Verlag.  
Edoardo Saccentia, Jos? Camacho, (2015) On the use of the observation-wise k-fold operation in PCA cross-validation, J. Chemometrics 2015; 29: 467-478.

**See Also**

[loadingsplot2D](#), [T2](#), [Xresids](#), [ScoreContrib](#)

**Examples**

```
data(iris)
pc1 <- pcaFit(iris, scale = TRUE, ncomp = NULL)
pc1

print(pc1) #Model summary
plot(pc1) #MSEP
PE(pc1) #X-explained variance

T2(pc1, ncomp = 2) #T2 plot

Xresids(pc1, ncomp = 2) #X-residuals plot

scoresplot(pc1) #scoresplot variable importance

(SC <- ScoreContrib(pc1, obs1 = 1:9, obs2 = 10:11)) #score contribution
plot(SC) #score contribution plot

loadingsplot(pc1, ncomp = 1) #loadings plot
loadingsplot(pc1, ncomp = 1:2) #loadings plot
loadingsplot(pc1, ncomp = 1:3) #loadings plot
loadingsplot(pc1, ncomp = 1:7) #loadings plot
loadingsplot2D(pc1, comps = c(1, 2)) #2-D loadings plot
loadingsplot2D(pc1, comps = c(2, 3)) #2-D loadings plot
```

---

 PE

*Percent Explained Variation of X*

---

**Description**

This function provides both the cumulative and individual percent explained for the X-block for an `mvdareg` and `mvdapca` objects.

**Usage**

```
PE(object, verbose = FALSE)
```

**Arguments**

<code>object</code>	an object of class <code>mvdareg</code> or <code>mvdapca</code> objects.
<code>verbose</code>	output results as a data frame

**Details**

This function provides both the cumulative and individual percent explained for the X-block for an `mvdareg` or `mvdapca` objects.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "none")
PE(mod1)

## Not run:
data(Penta)
mod2 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "loo")
PE(mod2)

## End(Not run)
```

---

Penta

*Penta data set*

---

**Description**

This data is obtained from drug discovery and includes measurements pertaining to size, lipophilicity, and polarity at various sites on a molecule.

**Usage**

Penta

**Format**

A data frame with 30 observations and the following 17 variables.

Obs.Name Categorical ID Variable

S1 numeric predictor vector

L1 numeric predictor vector

P1 numeric predictor vector

S2 numeric predictor vector

L2 numeric predictor vector

P2 numeric predictor vector

S3 numeric predictor vector

L3 numeric predictor vector

P3 numeric predictor vector

S4 numeric predictor vector  
 L4 numeric predictor vector  
 P4 numeric predictor vector  
 S5 numeric predictor vector  
 L5 numeric predictor vector  
 P5 numeric predictor vector  
 log.RAI numeric response vector

### Source

Umetrics, Inc. (1995), Multivariate Analysis (3-day course), Winchester, MA.  
 SAS/STAT(R) 9.22 User's Guide, "The PLS Procedure".

---

perc.cis *Percentile Bootstrap Confidence Intervals*

---

### Description

Computes percentile bootstrap confidence intervals for chosen parameters for `plsfit` models fitted with `validation = "oob"`

### Usage

```
perc.cis(object, ncomp = object$ncomp, conf = 0.95,
         type = c("coefficients", "loadings", "weights"))
```

### Arguments

<code>object</code>	an object of class "mvdareg", i.e., <code>plsfit</code>
<code>ncomp</code>	number of components to extract percentile intervals.
<code>conf</code>	confidence level.
<code>type</code>	input parameter vector.

### Details

The function fits computes the bootstrap percentile confidence intervals for any fitted `mvdareg` model.

### Value

A `perc.cis` object contains component results for the following:

<code>ncomp</code>	number of components in the model
<code>variables</code>	variable names
<code>boot.mean</code>	mean of the bootstrap
<code>percentiles</code>	confidence intervals



**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

There are many references explaining the bootstrap and its implementation for confidence interval estimation. Among them are:

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman & Hall.

Hinkley, D.V. (1988) Bootstrap methods (with Discussion). *Journal of the Royal Statistical Society, B*, 50, 312:337, 355:370.

**Examples**

```
data(Penta)
## Number of bootstraps set to 250 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "oob", boots = 250)
perc.cis(mod1, ncomp = 1:2, conf = .95, type = "coefficients")
```

---

plot.cp

*Plotting Function for Score Contributions.*

---

**Description**

This function generates a plot an object of class `score.contribution`

**Usage**

```
## S3 method for class 'cp'
plot(x, ncomp = "Overall", ...)
```

**Arguments**

x	score.contribution object
ncomp	the number of components to include the graph output.
...	additional arguments. Currently ignored.

**Details**

A graph of the score contributions for `ScoreContrib` objects.

**Value**

The output of `plot` is a graph of score contributions for the specified observation(s).

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```

data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, method = "bidiagpls", validation = "loo")
Score.Contributions1 <- ScoreContrib(mod1, obs1 = 1, obs2 = 3)
plot(Score.Contributions1, ncomp = 1)

## Not run:
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod2 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "oob", boots = 300)
Score.Contributions2 <- ScoreContrib(mod2, obs1 = 1, obs2 = 3)
plot(Score.Contributions2, ncomp = 1)

## End(Not run)

#PCA Model
pc1 <- pcaFit(Penta[, -1], ncomp = 3)
Score.Contributions1 <- ScoreContrib(mod1, obs1 = 1, obs2 = 3)
plot(Score.Contributions1, ncomp = 1)

```

---

plot.mvcomp

*Plot of Multivariate Mean Vector Comparison*

---

**Description**

Plot a comparison of mean vectors drawn from two populations.

**Usage**

```

## S3 method for class 'mvcomp'
plot(x, Diff2Plot = c(3, 4), segments = 51, include.zero = FALSE, ...)

```

**Arguments**

x	an plot.mvcomp object.
segments	number of line-segments used to draw ellipse.
Diff2Plot	variable differences to plot.
include.zero	add the zero axis to the graph output.
...	additional arguments. Currently ignored.

**Details**

This function provides a plot of the T2-statistic for testing the equality of two mean vectors. This test is appropriate for testing two populations, assuming independence.

Assumptions:

The sample for both populations is a random sample from a multivariate population.

-Both populations are independent

-Both populations are multivariate normal

-Covariance matrices are approximately equal

If the confidence ellipse does not cover  $c(0, 0)$ , we reject the NULL that the difference between mean vectors is equal to zero (at the stated alpha level).

**Value**

This function returns a plot of the simultaneous confidence intervals for the p-variables and its corresponding confidence ellipse at the stated confidence level.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Johnson, R.A., Wichern, D.W. (2002) Applied Multivariate Statistical Analysis. Prentice Hall.

**Examples**

```
data(College)
dat1 <- College
#Generate a 'fake' difference of 15 units
dat2 <- College + matrix(rnorm(nrow(dat1) * ncol(dat1), mean = 15),
  nrow = nrow(dat1), ncol = ncol(dat1))

Comparison <- MVComp(dat1, dat2, level = .95)
Comparison
plot(Comparison, Diff2Plot = c(1, 2), include.zero = FALSE)
plot(Comparison, Diff2Plot = c(1, 2), include.zero = TRUE)

plot(Comparison, Diff2Plot = c(2, 3), include.zero = FALSE)
plot(Comparison, Diff2Plot = c(2, 3), include.zero = TRUE)

data(iris)
dat1b <- iris[, -5]
#Generate a 'fake' difference of .5 units
dat2b <- dat1b + matrix(rnorm(nrow(dat1b) * ncol(dat1b), mean = .5),
  nrow = nrow(dat1b), ncol = ncol(dat1b))

Comparison2 <- MVComp(dat1b, dat2b, level = .90)
plot(Comparison2, Diff2Plot = c(1, 2), include.zero = FALSE)
```

```
plot(Comparison2, Diff2Plot = c(1, 2), include.zero = TRUE)

plot(Comparison2, Diff2Plot = c(3, 4), include.zero = FALSE)
plot(Comparison2, Diff2Plot = c(3, 4), include.zero = TRUE)
```

---

plot.mvdareg

*General plotting function for mvdareg and mvdapaca objects.*


---

## Description

A general plotting function for a mvdareg and mvdapaca objects.

## Usage

```
## S3 method for class 'mvdareg'
plot(x, plottype = c("PE", "scoresplot", "loadingsplot",
                    "loadingsplot2D", "T2", "Xresids", "coefspplot", "ap.plot",
                    "weightsplot", "weightsplot2D", "acfplot"), ...)
```

## Arguments

`x` an object of class "mvdareg", i.e., a fitted model.  
`plottype` the desired plot from an object of class "mvdareg"  
`...` additional arguments. Currently ignored.

## Details

The following plotting functions are supported:

PE, scoreplot, loadingsplot, loadingsplot2D, T2, Xresids, coefspplot, ap.plot, weightsplot, weightsplot2D, acfplot

## Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

## Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
plot(mod1, plottype = "scoresplot")
## Not run:
plot(mod1, plottype = "loadingsplot2D")
plot(mod1, plottype = "T2", ncomp = 2, phase = 1, conf = c(.95, .99))

## End(Not run)
```

---

plot.plusminus                    *2D Graph of the PCA scores associated with a plusminusFit*

---

## Description

Generates a 2-dimensional graph of the scores for both plusminus objects.

## Usage

```
## S3 method for class 'plusminus'  
plot(x, ncomp = 2, comps = c(1, 2), ...)
```

## Arguments

x	an object of class plusminus, i.e. plusminusFit.
ncomp	the number of components to include in the model (see below).
comps	a vector or length 2 corresponding to the number of components to include.
...	additional arguments. Currently ignored.

## Details

plot.plusminus is used to extract a 2D graphical summary of the PCA scores associated with a plusminus object.

## Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

## Examples

```
### PLUS-Minus CLASSIFIER WITH validation = 'none', i.e. no CV ###  
data(plusMinusDat)  
mod1 <- plusminusFit(Y ~., data = plusMinusDat, validation = "none", n_cores = 2)  
plot(mod1, ncomp = 2, comps = c(1, 2))  
  
### Plus-Minus CLASSIFIER WITH validation = 'loo', i.e. leave-one-out CV ###  
## Not run:  
data(plusMinusDat)  
mod2 <- plusminusFit(Y ~., data = plusMinusDat, validation = "loo", n_cores = 2)  
plot(mod2, ncomp = 2, comps = c(1, 2))  
  
## End(Not run)
```

---

plot.R2s	<i>Plot of R2</i>
----------	-------------------

---

### Description

Plots for the cross-validated R2 (CVR2), explained variance in the predictor variables (R2X), and the reponse (R2Y).

### Usage

```
## S3 method for class 'R2s'  
plot(x, ...)
```

### Arguments

x	An R2s object
...	additional arguments. Currently ignored.

### Details

plot.R2s is used to generates the graph of the cross-validated R2 (CVR2), explained variance in the predictor variables (R2X), and the reponse (R2Y) for PLS models.

### Value

The output of plot.R2s is a graph of the stated explained variance summary.

### Author(s)

Thanh Tran (<thanh.tran@mvdalab.com>)

### Examples

```
data(Penta)  
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],  
              ncomp = 2, validation = "loo")  
plot(R2s(mod1))
```

---

`plot.smc`*Plotting function for Significant Multivariate Correlation*

---

## Description

This function generates a plot an object of class `smc`.

## Usage

```
## S3 method for class 'smc'  
plot(x, variables = "all", ...)
```

## Arguments

<code>x</code>	<code>smc</code> object.
<code>variables</code>	the number of variables to include the graph output.
<code>...</code>	additional arguments. Currently ignored.

## Details

`plot.smc` is used to generates the graph of the significant multivariate correlation from `smc` objects.

## Value

The output of `plot.smc` is a graph of the significant multivariate correlation for the specified observation(s).

## Author(s)

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

## Examples

```
data(Penta)  
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],  
              ncomp = 2, validation = "loo")  
smc(mod1)  
plot(smc(mod1))
```

---

`plot.sr`*Plotting function for Selectivity Ratio.*

---

### Description

This function provides the ability to plot an object of class `sr`

### Usage

```
## S3 method for class 'sr'  
plot(x, variables = "all", ...)
```

### Arguments

<code>x</code>	<code>sr</code> object
<code>variables</code>	the number of variables to include the graph output.
<code>...</code>	additional arguments. Currently ignored.

### Details

`plot.sr` is used to generate the graph of the selectivity ratio from `sr` objects.

### Value

The output of `plot.sr` is a graph of the selectivity ratio for the specified observation(s).

### Author(s)

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

### Examples

```
data(Penta)  
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],  
              ncomp = 2, validation = "loo")  
sr(mod1)  
plot(sr(mod1))
```



---

plot.wrtpls	<i>Plots of the Output of a Permutation Distribution for an mvdaReg Object with method = "bidiagpls"</i>
-------------	--

---

### Description

This takes an mvdaReg object fitted with method = "bidiagpls" and produces a graph of the bootstrap distribution and its corresponding normal quantile plot for a variable of interest.

### Usage

```
## S3 method for class 'wrtpls'
plot(x, comp = 1:object$ncomp, distribution = "log", ...)
```

### Arguments

x	an object of class "mvdaReg", i.e., a plsFit.
comp	number of latent variables to generate the permutation distribution
distribution	plot the "log", or "actual", of the permutation distribution
...	additional arguments. Currently ignored.

### Details

The function generates the permutation distribution and normal quantile plot for a mvdaReg model when method = "bidiagpls" is specified.

### Value

The output of plot.wrtpls is a histogram of the permutation distribution with the following vertical line indicators.

Solid line = Actual Value; Dashed Line = Critical Value from t-distribution at the model specified alpha; Dotted line = Quantile at the model specified alpha

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

### See Also

[bca.cis](#)

### Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              method = "wrtpls", validation = "none")
## Not run
## plot.wrtpls(mod1, distribution = "log")
```

plsFit

*Partial Least Squares Regression***Description**

Functions to perform partial least squares regression with a formula interface. Bootstrapping can be used. Prediction, residuals, model extraction, plot, print and summary methods are also implemented.

**Usage**

```
plsFit(formula, data, subset, ncomp = NULL, na.action,
method = c("bidiagpls", "wrtppls"), scale = TRUE, n_cores = 2,
alpha = 0.05, perms = 2000, validation = c("none", "oob", "loo"),
boots = 1000, model = TRUE, parallel = FALSE,
x = FALSE, y = FALSE, ...)
## S3 method for class 'mvdareg'
summary(object, ncomp = object$ncomp, digits = 3, ...)
```

**Arguments**

formula	a model formula (see below).
data	an optional data frame containing the variables in the model.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
ncomp	the number of components to include in the model (see below).
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
method	the multivariate regression algorithm to be used.
scale	should scaling to unit variance be used.
n_cores	Number of cores to run for parallel processing. Currently set to 2 with the max being 4.
alpha	the significance level for wrtppls
perms	the number of permutations to run for wrtppls
validation	character. What kind of (internal) validation to use. See below.
boots	Number of bootstrap samples when validation = 'oob'
model	an optional data frame containing the variables in the model.
parallel	should parallelization be used.
x	a logical. If TRUE, the model matrix is returned.
y	a logical. If TRUE, the response is returned.

object	an object of class "mvdareg", i.e., a fitted model.
digits	the number of decimal place to output with <code>summary.mvdareg</code>
...	additional arguments, passed to the underlying fit functions, and <code>mvdareg</code> . Currently not in use.

## Details

The function fits a partial least squares (PLS) model with 1, ..., `ncomp` number of latent variables. Multi-response models are not supported.

The type of model to fit is specified with the `method` argument. Currently two PLS algorithms are available: the `bigdiag2` algorithm ("`bigdiagpls`" and "`wrtpls`").

The `formula` argument should be a symbolic formula of the form `response ~ terms`, where `response` is the name of the response vector and `terms` is the name of one or more predictor matrices, usually separated by `+`, e.g., `y ~ X + Z`. See [lm](#) for a detailed description. The named variables should exist in the supplied data data frame or in the global environment. The chapter Statistical models in R of the manual *An Introduction to R* distributed with R is a good reference on formulas in R.

The number of components to fit is specified with the argument `ncomp`. If this is not supplied, the maximal number of components is used.

Note that if the number of samples is  $\leq 15$ , oob validation may fail. It is recommended that you PLS with `validation = "loo"`.

If `method = "bidiagpls"` and `validation = "oob"`, bootstrap cross-validation is performed. Bootstrap confidence intervals are provided for `coefficients`, `weights`, `loadings`, and `y.loadings`. The number of bootstrap samples is specified with the argument `boots`. See `mvdaboot` for details.

If `method = "bidiagpls"` and `validation = "loo"`, leave-one-out cross-validation is performed.

If `method = "bidiagpls"` and `validation = "none"`, no cross-validation is performed. Note that the number of components, `ncomp`, is set to  $\min(\text{nobj} - 1, \text{npred})$

If `method = "wrtpls"` and `validation = "none"`, The Weight Randomization Test for the selection of the number of components is performed. Note that the number of components, `ncomp`, is set to  $\min(\text{nobj} - 1, \text{npred})$

## Value

An object of class `mvdareg` is returned. The object contains all components returned by the underlying fit function. In addition, it contains the following:

<code>loadings</code>	X loadings
<code>weights</code>	weights
<code>D2.values</code>	bidiag2 matrix
<code>iD2</code>	inverse of bidiag2 matrix
<code>Ymean</code>	mean of reponse variable
<code>Xmeans</code>	mean of predictor variables
<code>coefficients</code>	PLS regression coefficients
<code>y.loadings</code>	y-loadings
<code>scores</code>	X scores

R	orthogonal weights
Y.values	scaled response values
Yactual	actual response values
fitted	fitted values
residuals	residuals
Xdata	X matrix
iPreds	predicted values
y.loadings2	scaled y-loadings
ncomp	number of latent variables
method	PLS algorithm used
scale	scaling used
validation	validation method
call	model call
terms	model terms
model	fitted model

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>), Thanh Tran (<thanh.tran@mvdalab.com>)

### References

NOTE: This function is adapted from `mvr` in package **pls** with extensive modifications by Nelson Lee Afanador and Thanh Tran.

### See Also

[bidiagpls.fit](#), [mvdaboot](#), [boot.plots](#), [R2s](#), [PE](#), [ap.plot](#), [T2](#), [Xresids](#), [smc](#), [scoresplot](#), [ScoreContrib](#), [sr](#), [loadingsplot](#), [weightsplot](#), [coefsplot](#), [coefficientsplot2D](#), [loadingsplot2D](#), [weightsplot2D](#), [bca.cis](#), [coefficients.boots](#), [loadings.boots](#), [weight.boots](#), [coefficients](#), [loadings](#), [weights](#), [BiPlot](#), [jk.after.boot](#)

### Examples

```
### PLS MODEL FIT WITH method = 'bidiagpls' and validation = 'oob', i.e. bootstrapping ###
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1], method = "bidiagpls",
               ncomp = 2, validation = "oob", boots = 300)
summary(mod1) #Model summary

### PLS MODEL FIT WITH method = 'bidiagpls' and validation = 'loo', i.e. leave-one-out CV ###
## Not run:
mod2 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1], method = "bidiagpls",
               ncomp = 2, validation = "loo")
```

```

summary(mod2) #Model summary

## End(Not run)

### PLS MODEL FIT WITH method = 'bidiagpls' and validation = 'none', i.e. no CV is performed ###
## Not run:
mod3 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1], method = "bidiagpls",
               ncomp = 2, validation = "none")
summary(mod3) #Model summary

## End(Not run)
### PLS MODEL FIT WITH method = 'wrtpls' and validation = 'none', i.e. WRT-PLS is performed ###
## Not run:
mod4 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               method = "wrtpls", validation = "none")
summary(mod4) #Model summary
plot.wrtpls(mod4)

## End(Not run)

```

---

plusminus.fit

*PlusMinus (Mas-o-Menos)*


---

## Description

Plus-Minus classifier

## Usage

```
plusminus.fit(XX, YY, ...)
```

## Arguments

XX	a matrix of observations. NAs and Infs are not allowed.
YY	a vector. NAs and Infs are not allowed.
...	additional arguments. Currently ignored.

## Details

This function should not be called directly, but through `plusminusFit` with the argument `method="plusminus"`. It implements the Plus-Minus algorithm.

## Value

An object of class `plusminus` is returned. The object contains all components returned by the underlying fit function. In addition, it contains the following:

<code>coefficients</code>	regression coefficients
<code>Y</code>	response values
<code>X</code>	scaled predictors

**Author(s)**

Richard Baumgartner (<richard\_baumgartner@merck.com>), Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Zhao et al. (2014) Mas-o-menos: a simple sign averaging method for discrimination in genomic data analysis. *Bioinformatics*, 30(21):3062-3069.

**See Also**

[plusminusFit](#)

---

plusminus.loo      *Leave-one-out routine for plusminus objects*

---

**Description**

When `validation = 'loo'` this routine effects the leave-one-out cross-validation procedure for plusminus objects.

**Usage**

```
plusminus.loo(X, Y, method = "plusminus", n_cores, ...)
```

**Arguments**

X	a matrix of observations. NAs and Infs are not allowed.
Y	a vector. NAs and Infs are not allowed.
method	PlusMinus algorithm used
n_cores	number of cores
...	additional arguments. Currently ignored.

**Details**

This function should not be called directly, but through the generic function `plusminusFit` with the argument `validation = 'loo'`.

**Value**

Provides the following crossvalidated results as a list for plusminus objects:

cvError	leave-one-out estimate of cv error.
in.bag	leave-one-out samples used for model building.

**Author(s)**

Richard Baumgartner (<richard\_baumgartner@merck.com>), Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

## References

NOTE: This function is adapted from `mvr` in package **pls** with extensive modifications by Nelson Lee Afanador and Thanh Tran.

## See Also

[plusminusFit](#)

## Examples

```
data(plusMinusDat)
mod1 <- plusminusFit(Y ~., data = plusMinusDat, validation = "loo", n_cores = 2)
## Not run:
summary(mod1)
mod1$validation$cvError
mod1$validation$in.bag

## End(Not run)
```

---

plusMinusDat	<i>plusMinusDat data set</i>
--------------	------------------------------

---

## Description

A simulated dataset for demonstrating the performance of a `plusminusFit` analysis.

## Usage

```
plusMinusDat
```

## Format

A data frame with 201 observations, 200 input variables (X) and one response variable (Y).

## Source

Richard Baumgartner (<[richard\\_baumgartner@merck.com](mailto:richard_baumgartner@merck.com)>)

plusminusFit

*Plus-Minus (Mas-o-Menos) Classifier***Description**

Functions to perform plus-minus classifier with a formula interface. Leave one out crossvalidation also implemented. Model extraction, plot, print and summary methods are also implemented.

**Usage**

```
plusminusFit(formula, data, subset, na.action, method = "plusminus", n_cores = 2,
              validation = c("loo", "none"), model = TRUE,
              x = FALSE, y = FALSE, ...)
```

```
## S3 method for class 'plusminus'
summary(object, ...)
```

**Arguments**

formula	a model formula (see below).
data	an optional data frame containing the variables in the model.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
method	the classification algorithm to be used.
n_cores	Number of cores to run for parallel processing. Currently set to 2 with the max being 4.
validation	character. What kind of (internal) validation to use. See below.
model	an optional data frame containing the variables in the model.
x	a logical. If TRUE, the model matrix is returned.
y	a logical. If TRUE, the response is returned.
object	an object of class "plusminus", i.e., a fitted model.
...	additional arguments, passed to the underlying fit functions, and plusminus. Currently not in use.

**Details**

The function fits a Plus-Minus classifier.

The formula argument should be a symbolic formula of the form response ~ terms, where response is the name of the response vector and terms is the name of one or more predictor matrices, usually



separated by +, e.g.,  $y \sim X + Z$ . See [lm](#) for a detailed description. The named variables should exist in the supplied data data frame or in the global environment. The chapter Statistical models in R of the manual An Introduction to R distributed with R is a good reference on formulas in R.

If `validation = "loo"`, leave-one-out cross-validation is performed. If `validation = "none"`, no cross-validation is performed.

## Value

An object of class `plusminus` is returned. The object contains all components returned by the underlying fit function. In addition, it contains the following:

<code>coefficients</code>	Plus-Minus regression coefficients
<code>X</code>	X matrix
<code>Y</code>	actual response values (class labels)
<code>val.method</code>	validation method
<code>call</code>	model call
<code>terms</code>	model terms
<code>mm</code>	model matrix
<code>model</code>	fitted model

## Author(s)

Richard Baumgartner (<[richard\\_baumgartner@merck.com](mailto:richard_baumgartner@merck.com)>), Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

## References

Zhao et al.: Mas-o-menos: a simple sign averaging method for discrimination in genomic data analysis. *Bioinformatics*, 30(21):3062-3069,2014.

## See Also

[plusminus.fit](#), [plusminus.loo](#)

## Examples

```
### PLUS-Minus CLASSIFIER WITH validation = 'none', i.e. no CV ###
data(plusMinusDat)
mod1 <- plusminusFit(Y ~., data = plusMinusDat, validation = "none", n_cores = 2)
summary(mod1)

### Plus-Minus CLASSIFIER WITH validation = 'loo', i.e. leave-one-out CV ###
## Not run:
data(plusMinusDat)
mod2 <- plusminusFit(Y ~., data = plusMinusDat, validation = "loo", n_cores = 2)
summary(mod2)

## End(Not run)
```

---

predict.mvdareg      *Model Predictions From a plsFit Model*

---

### Description

predict provides predictions from the results of a pls model.

### Usage

```
## S3 method for class 'mvdareg'  
predict(object, newdata, ncomp = object$ncomp,  
        na.action = na.pass, ...)
```

### Arguments

object	A plsFit model.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
ncomp	the number of components to include in the model (see below).
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	additional arguments. Currently ignored.

### Details

predict.mvdareg produces predicted values, obtained by evaluating the regression function in the frame newdata (which defaults to model.frame(object)). If newdata is omitted the predictions are based on the data used for the fit.

If comps is missing (or is NULL), predictions of the number of latent variables is provided. Otherwise, if comps is given parameters for a model with only the requested components is returned. The generic function residuals return the model residuals for all the components specified for the model. If the model was fitted with na.action = na.exclude (or after setting the default na.action to na.exclude with options), the residuals corresponding to excluded observations are returned as NA; otherwise, they are omitted.

### Value

predict.mvdareg produces a vector of predictions or a matrix of predictions

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

### References

NOTE: This function is adapted from mvr in package **pls** with extensive modifications by Nelson Lee Afanador.

**See Also**

[coef](#), [coefficients.boots](#), [coefficients](#), [loadings](#), [loadings.boots](#), [weights](#), [weight.boots](#)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
predict.mvdareg(mod1)
## Not run:
residuals(mod1)

## End(Not run)
```

---

print.mvdalab

*Print Methods for mvdalab Objects*


---

**Description**

Summary and print methods for mvdalab objects.

**Usage**

```
## S3 method for class 'mvdareg'
print(x, ...)
```

**Arguments**

x                    an mvdalab object  
...                    additional arguments. Currently ignored.

**Details**

print.mvdalab Is a generic function used to print mvdalab objects, such as print.empca for imputeEM, print.mvdapca for mvdapca objects, and summary.mvdareg for mvdareg objects.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
print(mod1, ncomp = 2)
summary(mod1, ncomp = 2)
```

---

print.plusminus      *Print Methods for plusminus Objects*

---

### Description

Summary and print methods for plusminus objects.

### Usage

```
## S3 method for class 'plusminus'  
print(x, ...)
```

### Arguments

x                    an plusminus object  
...                  additional arguments. Currently ignored.

### Details

print.plusminus Is a generic function used to print plusminus objects, such as print.plusminus for plusminus objects.

### Author(s)

Richard Baumgartner (<richard\_baumgartner@merck.com>), Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

### Examples

```
## Not run:  
data(plusMinusDat)  
mod1 <- plusminusFit(Y ~., data = plusMinusDat, validation = "loo", n_cores = 2)  
print(mod1)  
  
## End(Not run)
```

---

proCrustes              *Comparison of n-point Configurations vis Procrustes Analysis*

---

### Description

Implementation of Procrustes Analysis in the spirit of multidimensional scaling.

### Usage

```
proCrustes(X, Y, scaling = TRUE, standardize = FALSE, scale.unit = F, ...)
```

**Arguments**

X	Target configuration
Y	Matching configuration
scaling	Scale Y-axis
standardize	Standardize configurations
scale.unit	Scale to unit variance
...	additional arguments. Currently ignored.

**Details**

This function implements Procrustes Analysis as described in the reference below. That is to say:

Translation: Fixed displacement of points through a constant distance in a common direction

Rotation: Fixed displacement of all points through a constant angle

Dilation: Stretching or shrinking by a constant amount

**Value**

Rotation.Matrix	The matrix, Q, that rotates Y towards X; obtained via svd of X'Y
Residuals	residuals after fitting
M2_min	Residual Sums of Squares
Xmeans	Column Means of X
Ymeans	Column Means of Y
PRMSE	Procrustes Root Mean Square Error
Yproj	Projected Y-values
scale	logical. Should Y be scaled.
Translation	Scaling through a common distance based on rotation of Y and scaling parameter, c
residuals.	residual sum-of-squares
Anova.MSS	Explained Variance w.r.t. Y
Anova.ESS	Unexplained Variance w.r.t. Y
Anova.TSS	Total Sums of Squares w.r.t. X

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Krzanowski, Wojtek. Principles of multivariate analysis. OUP Oxford, 2000.

**Examples**

```
X <- iris[, 1:2]
Y <- iris[, 3:4]

proc <- proCrustes(X, Y)
proc
names(proc)
```

---

R2s

*Cross-validated R2, R2 for X, and R2 for Y for PLS models*

---

**Description**

Functions to report the cross-validated R2 (CVR2), explained variance in the predictor variables (R2X), and the reponse (R2Y) for PLS models.

**Usage**

```
R2s(object)
```

**Arguments**

object            an mvdareg object, i.e., plsFit.

**Details**

R2s is used to extract a summary of the cross-validated R2 (CVR2), explained variance in the predictor variables (R2X), and the reponse (R2Y) for PLS models.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "loo")

R2s(mod1)
## Not run:
plot(R2s(mod1))

## End(Not run)
```

---

ScoreContrib	<i>Generates a score contribution plot</i>
--------------	--

---

**Description**

Generates a the Score Contribution Graph both `mvdaereg` and `mvdapca` objects.

**Usage**

```
ScoreContrib(object, ncomp = 1:object$ncomp, obs1 = 1, obs2 = NULL)
```

**Arguments**

<code>object</code>	an object of class <code>mvdaereg</code> or <code>mvdapca</code> .
<code>ncomp</code>	the number of components to include in the model (see below).
<code>obs1</code>	the first observaion(s) in the score(s) comparison.
<code>obs2</code>	the second observaion(s) in the score(s) comparison.

**Details**

`ScoreContrib` is used to generates the score contributions for both PLS and PCA models. Up to two groups of score(s) can be selected. If only one group is selected, the contribution is measured to the model average. For PLS models the PCA loadings are replaced with the PLS weights.

**Value**

The output of `ScoreContrib` is a matrix of score contributions for the specified observation(s).

**Author(s)**

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

**References**

MacGregor, Process Monitoring and Diagnosis by Multiblock PLS Methods, May 1994 Vol. 40, No. 5 AIChE Journal.

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "none")
Score.Contributions1 <- ScoreContrib(mod1, ncomp = 1:2, obs1 = 1, obs2 = 3)
plot(Score.Contributions1, ncomp = 2)

## Not run:
data(Penta)
mod2 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
```

```

        ncomp = 2, validation = "none")
Score.Contributions2 <- ScoreContrib(mod2, obs1 = 1, obs2 = 3)
plot(Score.Contributions2)
Score.Contributions3 <- ScoreContrib(mod1, obs1 = c(1, 3), obs2 = c(5:10))
plot(Score.Contributions3)

## End(Not run)

### PLS MODEL FIT WITH method = 'wrtpls' and validation = 'none', i.e. WRT-PLS is performed ###
## Not run:
mod3 <- plsFit(Sepal.Length ~., scale = TRUE, data = iris,
              method = "wrtpls", validation = "none") #ncomp is ignored
Score.Contributions4 <- ScoreContrib(mod3, ncomp = 1:5, obs1 = 1, obs2 = 3)
plot(Score.Contributions4, ncomp = 5)

## End(Not run)

#PCA Model
pc1 <- pcaFit(Penta[, -1], ncomp = 2)
Score.Contributions1 <- ScoreContrib(pc1, obs1 = 1, obs2 = 3)
plot(Score.Contributions1)

```

---

scoresplot

*2D Graph of the scores*

---

## Description

Generates a 2-dimensional graph of the scores for both `mvdareg` and `mvdapca` objects.

## Usage

```
scoresplot(object, comps = c(1, 2), alphas = c(.95, .99),
           segments = 51, verbose = FALSE)
```

## Arguments

<code>object</code>	an object of class <code>mvdareg</code> , i.e. <code>plsFit</code> .
<code>comps</code>	a vector or length 2 corresponding to the number of components to include.
<code>alphas</code>	draw elliptical contours at these confidence levels.
<code>segments</code>	number of line-segments used to draw ellipse.
<code>verbose</code>	output results as a data frame

## Details

`scoresplot` is used to extract a 2D graphical summary of the scores of PLS and PCA models.

## Author(s)

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)



**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
scoresplot(mod1, comp = c(1, 2))
```

---

SeqimputeEM	<i>Sequential Expectation Maximization (EM) for imputation of missing values.</i>
-------------	---

---

**Description**

Missing values are sequentially updated via an EM algorithm.

**Usage**

```
SeqimputeEM(data, max.ncomps = 5, max.ssq = 0.99, Init = "mean",
            adjmean = FALSE, max.iters = 200,
            tol = .Machine$double.eps^0.25)
```

**Arguments**

data	a dataset with missing values.
max.ncomps	integer corresponding to the maximum number of components to test
max.ssq	maximal SSQ for final number of components. This will be improved by automation.
Init	For continuous variables impute either the mean or median.
adjmean	Adjust (recalculate) mean after each iteration.
max.iters	maximum number of iterations for the algorithm.
tol	the threshold for assessing convergence.

**Details**

A completed data frame is returned that mirrors the model matrix. NAs are replaced with convergence values as obtained via Sequential EM algorithm. If object contains no NAs, it is returned unaltered.

**Value**

Imputed.DataFrames	A list of imputed data frames across impute.comps
ncomps	number of components to test

**Author(s)**

Thanh Tran (<thanh.tran@mvdalab.com>), Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

NOTE: Publication Pending

**Examples**

```
dat <- introNAs(iris, percent = 25)
SeqimputeEM(dat)
```

---

 smc

*Significant Multivariate Correlation*


---

**Description**

This function calculates the significant multivariate correlation (smc) metric for an mvdareg object

**Usage**

```
smc(object, ncomps = object$ncomp, corrected = F)
```

**Arguments**

object	an mvdareg or mvdapaca object, i.e. plsFit.
ncomps	the number of components to include in the model (see below).
corrected	whether there should be a correction of 1st order auto-correlation in the residuals.

Note that hidden objects include the smc modeled matrix and error matrices

**Details**

smc is used to extract a summary of the significant multivariate correlation of a PLS model.

If comps is missing (or is NULL), summaries for all smc estimates are returned. Otherwise, if comps are given parameters for a model with only the requested component comps is returned.

**Value**

The output of smc is an smc summary detailing the following:

smc	significant multivariate correlation statistic (smc).
p.value	p-value of the smc statistic.
f.value	f-value of the smc statistic.
Significant	Assessment of statistical significance.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

## References

Thanh N. Tran, Nelson Lee Afanador, Lutgarde M.C. Buydens, Lionel Blanchet, Interpretation of variable importance in Partial Least Squares with Significance Multivariate Correlation (sMC). Chemom. Intell. Lab. Syst. 2014; 138: 153:160.

Nelson Lee Afanador, Thanh N. Tran, Lionel Blanchet, Lutgarde M.C. Buydens, Variable importance in PLS in the presence of autocorrelated data - Case studies in manufacturing processes. Chemom. Intell. Lab. Syst. 2014; 139: 139:145.

## See Also

[smc.acfTest](#), [sr](#)

## Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "loo")
smc(mod1)
plot(smc(mod1))

### PLS MODEL FIT WITH method = 'wrtpls' and validation = 'none', i.e. WRT-PLS is performed ###
## Not run:
mod2 <- plsFit(Sepal.Length ~., scale = TRUE, data = iris,
               method = "wrtpls", validation = "none") #ncomp is ignored
plot(smc(mod2, ncomps = 2))

## End(Not run)
```

---

smc.acfTest

*Test of the Residual Significant Multivariate Correlation Matrix for the presence of Autocorrelation*

---

## Description

This function performs a 1st order test of the Residual Significant Multivariate Correlation Matrix in order to help determine if the smc should be performed correcting for 1st order autocorrelation.

## Usage

```
smc.acfTest(object, ncomp = object$ncomp)
```

## Arguments

object	an object of class mvdareg, i.e. plsFit.
ncomp	the number of components to include in the acf assessment

### Details

This function computes a test for 1st order auto correlation in the smc residual matrix.

### Value

The output of `smc.acfTest` is a list detailing the following:

variable	variable for whom the test is being performed
ACF	value of the 1st lag of the ACF
Significant	Assessment of the statistical significance of the 1st order lag

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

### References

Thanh N. Tran, Nelson Lee Afanador, Lutgarde M.C. Buydens, Lionel Blanchet, Interpretation of variable importance in Partial Least Squares with Significance Multivariate Correlation (sMC). Chemom. Intell. Lab. Syst. 2014; 138: 153:160.

Nelson Lee Afanador, Thanh N. Tran, Lionel Blanchet, Lutgarde M.C. Buydens, Variable importance in PLS in the presence of autocorrelated data - Case studies in manufacturing processes. Chemom. Intell. Lab. Syst. 2014; 139: 139:145.

### Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
smc.acfTest(mod1, ncomp = 2)
```

---

sr	<i>Selectivity Ratio</i>
----	--------------------------

---

### Description

This function calculates the Selectivity Ratio (sr) metric for an `mvdareg` object

### Usage

```
sr(object, ncomps = object$ncomp)
```

### Arguments

object	an <code>mvdareg</code> or <code>mvdapaca</code> object, i.e. <code>plsFit</code> .
ncomps	the number of components to include in the model (see below).

## Details

sr is used to extract a summary of the significant multivariate correlation of a PLS model.

If comps is missing (or is NULL), summaries for all sr estimates are returned. Otherwise, if comps are given parameters for a model with only the requested component comps is returned.

## Value

The output of sr is an sr summary detailing the following:

sr	selectivity ratio statistic (sr).
p.value	p-value of the sr statistic.
f.value	f-value of the sr statistic.
Significant	Assessment of statistical significance.

Note that hidden objects include the SR modeled matrix and error matrices.

## Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

## References

O.M. Kvalheim, T.V. Karstang, Interpretation of latent-variable regression models. Chemom. Intell. Lab. Syst., 7 (1989), pp. 39:51

O.M. Kvalheim, Interpretation of partial least squares regression models by means of target projection and selectivity ratio plots. J. Chemom., 24 (2010), pp. 496:504

## See Also

[smc](#)

## Examples

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
sr(mod1)
plot(sr(mod1))

## Not run:
mod2 <- plsFit(Sepal.Length ~., scale = TRUE, data = iris,
              method = "wrtpls", validation = "none") #ncomp is ignored
plot(sr(mod2, ncomps = 2))

## End(Not run)
```

---

**T2** *Generates a Hotelling's T2 Graph*

---

**Description**

Generates a Hotelling's T2 Graph both `mvdaReg` and `mvdaPCA` objects.

**Usage**

```
T2(object, ncomp = object$ncomp, phase = 1, conf = c(.95, .99), verbose = FALSE)
```

**Arguments**

<code>object</code>	an object of class <code>mvdaReg</code> or <code>mvdaPCA</code> .
<code>ncomp</code>	the number of components to include in the calculation of Hotelling's T2.
<code>phase</code>	designates whether the confidence limits should reflect the current data frame, <code>phase = 1</code> or future observations, <code>phase = 2</code> .
<code>conf</code>	the confidence level(s) to use for upper control limit.
<code>verbose</code>	output results as a data frame

**Details**

T2 is used to generate a Hotelling's T2 graph both PLS and PCA models.

**Value**

The output of T2 is a graph of Hotelling's T2 and a data frame listing the T2 values.

**Author(s)**

Nelson Lee Afanador (<[nelson.afanador@mvdaLab.com](mailto:nelson.afanador@mvdaLab.com)>)

**References**

Hotelling, H. (1931). "The generalization of Student's ratio". *Annals of Mathematical Statistics* 2 (3): 360:378.

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
T2(mod1, ncomp = 2)
```

---

Wang_Chen	<i>Bivariate process data.</i>
-----------	--------------------------------

---

**Description**

Twenty-five observations where 'H' represents brinell hardness and 'S' represents tensile strength.

**Usage**

Wang\_Chen

**Format**

A data frame with 25 observations and the following 2 variables.

H brinell hardness

S tensile strength

**Source**

Wang F, Chen J (1998). "Capability index using principal components analysis." *Quality Engineering*, 11, 21-27.

---

Wang_Chen_Sim	<i>Simulated process data from a plastics manufacturer.</i>
---------------	---

---

**Description**

Fifty observations where 'D' represents depth, 'L' represents length, and 'W' represents width.

**Usage**

Wang\_Chen\_Sim

**Format**

A simulated data frame with 50 observations and the following 3 variables.

D depth

L length

W width

**Source**

Data simulated by Nelson Lee Afanador from average and covariance estimates provided in Wang F, Chen J (1998). "Capability index using principal components analysis." *Quality Engineering*, 11, 21-27.

---

weight.boots

*BCa Summaries for the weights of an mvdareg object*


---

### Description

Computes weights bootstrap BCa confidence intervals, along with expanded bootstrap summaries.

### Usage

```
weight.boots(object, ncomp = object$ncomp, conf = .95)
```

### Arguments

object	an object of class mvdareg, i.e. plsFit.
ncomp	number of components in the model.
conf	desired confidence level.

### Details

The function fits computes the bootstrap BCa confidence intervals for fitted mvdareg models where `validation = "oob"`. Should be used in instances in which there is reason to suspect the percentile intervals. Results provided across all latent variables or for specific latent variables via `ncomp`.

### Value

A weight.boots object contains component results for the following:

variable	variable names.
actual	Actual loading estimate using all the data.
BCa percentiles	confidence intervals.
boot.mean	mean of the bootstrap.
skewness	skewness of the bootstrap distribution.
bias	estimate of bias w.r.t. the loading estimate.
Bootstrap Error	estimate of bootstrap standard error.
t value	approximate 't-value' based on the Bootstrap Error.
bias t value	approximate 'bias t-value' based on the Bootstrap Error.

### Author(s)

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)



## References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society, B*, 54, 83:127.

## Examples

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "oob", boots = 300)
weight.boots(mod1, ncomp = 2, conf = .95)
```

---

weights

*Extract Summary Information Pertaining to the Bootstrapped weights*

---

## Description

Functions to extract weights bootstrap information from mvdalab objects.

## Usage

```
## S3 method for class 'mvdareg'
weights(object, ncomp = object$ncomp, conf = .95, ...)
```

## Arguments

object	an mvdareg or mvdapaca object, i.e. plsFit.
ncomp	the number of components to include in the model (see below).
conf	for a bootstrapped model, the confidence level to use.
...	additional arguments. Currently ignored.

## Details

weights is used to extract a summary of the weights of a PLS. If ncomps is missing (or is NULL), summaries for all regression estimates are returned. Otherwise, if comps is given parameters for a model with only the requested component comps is returned.

For mvdareg objects only, bootstrap summaries provided are for actual regression weights, bootstrap percentiles, bootstrap mean, skewness, and bias. These summaries can also be extracted using weight.boots

**Value**

A weights object contains a data frame with columns:

variable	variable names.
Actual	Actual loading estimate using all the data.
BCa percentiles	confidence intervals.
boot.mean	mean of the bootstrap.
skewness	skewness of the bootstrap distribution.
bias	estimate of bias w.r.t. the loading estimate.
Bootstrap Error	estimate of bootstrap standard error.
t value	approximate 't-value' based on the Bootstrap Error.
bias t value	approximate 'bias t-value' based on the Bootstrap Error.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society, B*, 54, 83:127.

**See Also**

[weightsplot](#), [weight.boots](#), [weightsplot2D](#)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
               ncomp = 2, validation = "loo")
weights(mod1, ncomp = 2, conf = .95)
```

---

`weightsplot`*Extract Graphical Summary Information Pertaining to the Weights*

---

## Description

Functions to extract regression coefficient bootstrap information from `mvdaLab` objects.

## Usage

```
weightsplot(object, ncomp = object$ncomp, conf = .95, verbose = FALSE)
```

## Arguments

<code>object</code>	an <code>mvdaReg</code> object, i.e. <code>plsFit</code>
<code>ncomp</code>	the number of components to include.
<code>conf</code>	for a bootstrapped model, the confidence level to use.
<code>verbose</code>	output results as a data frame

## Details

`weightsplot` is used to extract a graphical summary of the weights of a PLS model.

If `comps` is missing (or is `NULL`), a graphical summary for the `n`th component regression estimates are returned. Otherwise, if `comps` is given parameters for a model with only the requested component `comps` is returned.

Bootstrap graphical summaries provided are when `method = oob`.

## Author(s)

Nelson Lee Afanador (<[nelson.afanador@mvdaLab.com](mailto:nelson.afanador@mvdaLab.com)>)

## Examples

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "oob", boots = 300)
weightsplot(mod1, ncomp = 1:2)
```

---

weightsplot2D	<i>Extract a 2-Dimensional Graphical Summary Information Pertaining to the weights of a PLS Analysis</i>
---------------	--

---

**Description**

Functions to extract 2D graphical weights information from mvdalab objects.

**Usage**

```
weightsplot2D(object, comps = c(1, 2), verbose = FALSE)
```

**Arguments**

object	an mvdareg object, i.e. plsFit.
comps	a vector or length 2 corresponding to the number of components to include.
verbose	output results as a data frame

**Details**

weightsplot2D is used to extract a graphical summary of the weights of a PLS model.

If comp is missing (or is NULL), a graphical summary for the 1st and 2nd components are returned.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
weightsplot2D(mod1, comp = c(1, 2))
```

---

wrtpls.fit	<i>Weight Randomization Test PLS</i>
------------	--------------------------------------

---

**Description**

Weight Randomization Test algorithm for PLS1

**Usage**

```
wrtpls.fit(X, Y, ncomp, perms, alpha, ...)
```

**Arguments**

X	a matrix of observations. NAs and Inf s are not allowed.
Y	a vector. NAs and Inf s are not allowed.
ncomp	the number of components to include in the model (see below).
alpha	the significance level for wrtpls
perms	the number of permutations to run for wrtpls
...	additional arguments. Currently ignored.

**Details**

This function should not be called directly, but through `plsFit` with the argument `method="wrtpls"`. It implements the Bidiag2 scores algorithm with a permutation test for selecting the statistically significant components.

**Value**

An object of class `mvdaReg` is returned. The object contains all components returned by the underlying fit function. In addition, it contains the following:

loadings	X loadings
weights	weights
D2	bidiag2 matrix
iD2	inverse of bidiag2 matrix
Ymean	mean of reponse variable
Xmeans	mean of predictor variables
coefficients	regression coefficients
y.loadings	y-loadings
scores	X scores
R	orthogonal weights
Y	scaled response values
Yactual	actual response values
fitted	fitted values
residuals	residuals
Xdata	X matrix
iPreds	predicted values
y.loadings2	scaled y-loadings
wrtpls	permutations effected
wrtpls.out.Sig	Significant LVs
wrtpls.crit	weight critical values
actual.normwobs	normed weights

fit.time	model fitting time
val.method	validation method
ncomp	number of latent variables
perms	number of permutations performed
alpha	permutation alpha value
method	PLS algorithm
scale	scaling used
scaled	was scaling performed
call	model call
terms	model terms
mm	model matrix
model	fitted model

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>), Thanh Tran (<thanh.tran@mvdalab.com>)

**References**

Indahl, Ulf G., (2014) The geometry of PLS1 explained properly: 10 key notes on mathematical properties of and some alternative algorithmic approaches to PLS1 modeling. *Journal of Chemometrics*, 28, 168:180.

Manne R., Analysis of two partial-least-squares algorithms for multi-variate calibration. *Chemom. Intell. Lab. Syst.* 1987; 2: 187:197.

Thanh Tran, Ewa Szymanska, Jan Gerretzen, Lutgarde Buydens, Nelson Lee Afanador, Lionel Blanchet, Weight Randomization Test for the Selection of the Number of Components in PLS Models. *Chemom. Intell. Lab. Syst.*, accepted for publication - Jan 2017.

**See Also**

[plsFit](#)

---

Xresids

*Generates a Graph of the X-residuals*

---

**Description**

Generates a graph of the X-residuals for both mvdareg and mvdapca objects.

**Usage**

```
Xresids(object, ncomp = object$ncomp, conf = c(.95, .99),
        normalized = TRUE, verbose = FALSE)
```

**Arguments**

object	an object of class mvdaereg or mvdapca.
ncomp	the number of components to include in the calculation of the X-residuals.
conf	the confidence level(s) to use for upper control limit.
normalized	should residuals be normalized
verbose	output results as a data frame

**Details**

Xresids is used to generates a graph of the X-residuals for both PLS and PCA models.

**Value**

The output of Xresids is a graph of X-residuals and a data frame listing the X-residuals values.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

MacGregor, Process Monitoring and Diagnosis by Multiblock PLS Methods, May 1994 Vol. 40, No. 5 AIChE Journal.

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
Xresids(mod1, ncomp = 2)
```

---

XresidualContrib	<i>Generates the squared prediction error contributions and contribution plot</i>
------------------	---

---

**Description**

Generates the squared prediction error (SPE) contributions and graph both mvdaereg and mvdapca objects.

**Usage**

```
XresidualContrib(object, ncomp = object$ncomp, obs1 = 1)
```

**Arguments**

object            an object of class `mvdareg` or `mvdapca`.  
ncomp            the number of components to include in the SPE calculation.  
obs1             the observaion in SPE assessment.

**Details**

`XresidualContrib` is used to generates the squared prediction error (SPE) contributions and graph for both PLS and PCA models. Only one observation at a time is supported.

**Value**

The output of `XresidualContrib` is a matrix of score contributions for a specified observation and the corresponding graph.

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**References**

MacGregor, Process Monitoring and Diagnosis by Multiblock PLS Methods, May 1994 Vol. 40, No. 5 AICHE Journal

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
XresidualContrib(mod1, ncomp = 2, obs1 = 3)

## Not run:
##PCA Model
pc1 <- pcaFit(Penta[, -1], ncomp = 4)
XresidualContrib(pc1, ncomp = 3, obs1 = 3)

## End(Not run)
```

---

y.loadings

*Extract Summary Information Pertaining to the y-loadings*

---

**Description**

Functions to extract the y-loadings from `mvdareg` and `mvdapca` objects.

**Usage**

```
y.loadings(object, conf = .95)
```



**Arguments**

object            an mvdaereg or mvdapaca object, i.e. plsFit.  
 conf             for a bootstrapped model, the confidence level to use.

**Details**

y.loadings is used to extract a summary of the y-loadings from a PLS or PCA model.

If comps is missing (or is NULL), summaries for all regression estimates are returned. Otherwise, if comps is provided the requested component comps are returned.

For mvdaereg objects only, bootstrap summaries provided are for actual regression y.loadings, bootstrap percentiles, bootstrap mean, skewness, and bias. These summaries can also be extracted using y.loadings.boots

**Author(s)**

Nelson Lee Afanador (<nelson.afanador@mvdalab.com>)

**Examples**

```
data(Penta)
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "loo")
y.loadings(mod1)
```

---

y.loadings.boots            *Extract Summary Information Pertaining to the y-loadings*

---

**Description**

Functions to extract the y-loadings from mvdaereg and mvdapaca objects.

**Usage**

```
y.loadings.boots(object, ncomp = object$ncomp, conf = 0.95)
```

**Arguments**

object            an mvdaereg or mvdapaca object, i.e. plsFit.  
 ncomp            the number of components to include in the model (see below).  
 conf             for a bootstrapped model, the confidence level to use.

**Details**

`y.loadings.boots` is used to extract a summary of the y-loadings from a PLS or PCA model.

If `comps` is missing (or is `NULL`), summaries for all regression estimates are returned. Otherwise, if `comps` is provided the requested component comps are returned.

For `mvdaReg` objects only, bootstrap summaries provided are for actual regression `y.loadings`, bootstrap percentiles, bootstrap mean, skewness, and bias. These summaries can also be extracted using `y.loadings.boots`

**Author(s)**

Nelson Lee Afanador (<[nelson.afanador@mvdalab.com](mailto:nelson.afanador@mvdalab.com)>)

**Examples**

```
data(Penta)
## Number of bootstraps set to 300 to demonstrate flexibility
## Use a minimum of 1000 (default) for results that support bootstrapping
mod1 <- plsFit(log.RAI ~., scale = TRUE, data = Penta[, -1],
              ncomp = 2, validation = "oob", boots = 300)
y.loadings(mod1)
y.loadings.boots(mod1)
```

# Index

acfplot, 4  
ap.plot, 5, 60  
  
bca.cis, 6, 11, 57, 60  
bidiagpls.fit, 7, 60  
BiPlot, 9, 60  
boot.plots, 7, 11, 60  
  
coef, 12, 14, 15, 67  
coef.mvdareg, 12  
coefficients, 12, 14, 15, 59, 60, 67  
coefficients.boots, 12, 13, 15, 60, 67  
coefficients.mvdareg, 14  
coefficientsplot2D, 16, 30, 60  
coefsplot, 14, 17, 60  
College, 18  
contr.niets, 18  
  
ellipse.mvdalab, 19  
  
imputeBasic, 20  
imputeEM, 21  
imputeQs, 22  
imputeRough, 23  
introNAs, 24  
  
jk.after.boot, 25, 60  
  
lm, 59, 65  
loadings, 12, 26, 30, 60, 67  
loadings.boots, 12, 27, 28, 30, 60, 67  
loadingsplot, 27, 29, 60  
loadingsplot2D, 16, 27, 30, 30, 46, 60  
  
mewma, 31  
model.matrix, 32  
MultCapability, 33  
MVCis, 35  
MVComp, 36, 36  
mvdaboot, 7, 37, 40, 60  
mvdalab (mvdalab-package), 3  
  
mvdalab-package, 3  
mvdaloo, 39, 39  
mvdareg (plsFit), 58  
mvrnorm.svd, 41  
mvrnormBase.svd (mvrnorm.svd), 41  
my.dummy.df, 42  
  
no.intercept, 43  
  
pca.nipals, 43  
pcaFit, 45  
PE, 46, 60  
Penta, 47  
perc.cis, 48  
plot.cp, 49  
plot.mvcomp, 50  
plot.mvdapca (pcaFit), 45  
plot.mvdareg, 52  
plot.plusminus, 53  
plot.R2s, 54  
plot.smc, 55  
plot.sr, 56  
plot.wrtpls, 57  
plsFit, 5, 7, 9, 39, 40, 58, 86  
plusminus.fit, 61, 65  
plusminus.loo, 62, 65  
plusMinusDat, 63  
plusminusFit, 62, 63, 64  
predict.mvdareg, 66  
print.empca (imputeEM), 21  
print.mvcomp (MVComp), 36  
print.mvdalab, 67  
print.mvdapca (pcaFit), 45  
print.mvdareg (print.mvdalab), 67  
print.npca (pca.nipals), 43  
print.plusminus, 68  
print.proC (proCrustes), 68  
print.R2s (R2s), 70  
print.roughImputation (imputeRough), 23  
print.seqem (SeqimputeEM), 73

print.smc (smc), 74  
print.sr (sr), 76  
proCrustes, 68

R2s, 60, 70

ScoreContrib, 46, 60, 71  
scoresplot, 60, 72  
SeqimputeEM, 73  
smc, 4, 60, 74, 77  
smc.acfTest, 4, 75, 75  
sr, 60, 75, 76  
summary.mvdareg (plsFit), 58  
summary.plusminus (plusminusFit), 64

T2, 46, 60, 78

Wang\_Chen, 79  
Wang\_Chen\_Sim, 79  
weight.boots, 12, 60, 67, 80, 82  
weights, 12, 60, 67, 81  
weightsplot, 60, 82, 83  
weightsplot2D, 16, 30, 60, 82, 84  
wrtps.fit, 84

Xresids, 46, 60, 86  
XresidualContrib, 87

y.loadings, 88  
y.loadings.boots, 89