

# Package ‘pcv’

January 28, 2023

**Version** 1.0.1

**Date** 2023-01-28

**Title** Procrustes Cross-Validation

**Maintainer** Sergey Kucheryavskiy <svkucheryavski@gmail.com>

**Description** Implements Procrustes cross-validation method for Principal Component Analysis, Principal Component Regression and Partial Least Squares regression models. S. Kucheryavskiy (2020) <[doi:10.1016/j.chemolab.2020.103937](https://doi.org/10.1016/j.chemolab.2020.103937)>.

**Encoding** UTF-8

**License** MIT + file LICENSE

**Imports** graphics, grDevices, stats

**RoxygenNote** 7.2.1

**Suggests** testthat

**NeedsCompilation** no

**Depends** R (>= 3.5.0)

**URL** <https://github.com/svkucheryavski/pcv>

**BugReports** <https://github.com/svkucheryavski/pcv/issues>

**Author** Sergey Kucheryavskiy [aut, cre]  
(<<https://orcid.org/0000-0002-3145-7244>>)

**Repository** CRAN

**Date/Publication** 2023-01-28 09:40:09 UTC

## R topics documented:

corn . . . . .	2
getCrossvalParams . . . . .	2
getR . . . . .	3
getxpvorth . . . . .	3
normalize . . . . .	4
pcv . . . . .	4
pcvcrossval . . . . .	5

pcvpca . . . . .	6
pcvpcr . . . . .	7
pcvpls . . . . .	9
pcvreg . . . . .	11
plotD . . . . .	11
rotationMatrixToX1 . . . . .	12
simpls . . . . .	13

<b>Index</b>	<b>14</b>
--------------	-----------

---

corn	<i>Corn data</i>
------	------------------

---

### Description

NIR spectra and moisture of 80 Corn samples.

### Usage

```
data(corn)
```

### Format

A list with two matrices, spectra and moisture.

### Details

This is a part of Corn dataset, which was downloaded from Eigenvector Research, Inc. website (<https://eigenvector.com/resources/data-sets/>), where it is available publicly. This dataset contains several NIR spectra of corn samples recorded using different instruments. For our examples we took "mp5" spectra and corrected them using Standard Normal Variate transformation.

### Source

1. Eigenvector Research, Inc. (<https://eigenvector.com/resources/data-sets/>)

---

getCrossvalParams	<i>Returns parameters for cross-validation based on 'cv' value</i>
-------------------	--

---

### Description

Returns parameters for cross-validation based on 'cv' value

### Usage

```
getCrossvalParams(cv, nobj)
```

**Arguments**

cv	settings for cross-validation provided by user
nobj	number of objects in calibration set

---

getR	<i>Creates rotation matrix to map a set vectors base1 to a set of vectors base2.</i>
------	--

---

**Description**

In both sets vectors should be orthonormal.

**Usage**

```
getR(base1, base2)
```

**Arguments**

base1	Matrix (JxA) with A orthonormal vectors as columns to be rotated ( $A \leq J$ )
base2	Matrix (JxA) with A orthonormal vectors as columns, base1 should be aligned with

**Value**

Rotation matrix (JxJ)

---

getxpvorth	<i>Generates the orthogonal part for Xpv</i>
------------	--

---

**Description**

Generates the orthogonal part for Xpv

**Usage**

```
getxpvorth(q.k, X.k, PRM)
```

**Arguments**

q.k	vector with orthogonal distances for cross-validation set for segment k
X.k	matrix with local validation set for segment k
PRM	projecton matrix for orthogonalization of residuals

**Value**

A matrix with orthogonal part for Xpv

---

normalize	<i>Normalization rows or columns of a matrix</i>
-----------	--

---

**Description**

Normalization rows or columns of a matrix

**Usage**

```
normalize(
  X,
  dim = 1,
  weights = if (dim == 1) 1/sqrt(rowSums(X^2)) else 1/sqrt(colSums(X^2))
)
```

**Arguments**

X	matrix with numeric values
dim	which dimension to normalize (1 for rows, 2 for columns)
weights	vector with normalization weights, by default 2-norm is used

---

pcv	<i>Compute matrix with pseudo-validation set</i>
-----	--

---

**Description**

Compute matrix with pseudo-validation set

**Usage**

```
pcv(
  X,
  ncomp = min(round(nrow(X)/nseg) - 1, col(X), 20),
  nseg = 4,
  scale = FALSE
)
```

**Arguments**

X	matrix with calibration set (IxJ)
ncomp	number of components for PCA decomposition
nseg	number of segments in cross-validation
scale	logical, standardize columns of X prior to decomposition or not

**Details**

This is the old (original) version of PCV algorithm for PCA models. Use `pcvpca` instead. Ane check project web-site for details: <https://github.com/svkucheryavski/pcv>

The method computes pseudo-validation matrix  $X_{pv}$ , based on PCA decomposition of calibration set  $X$  and systematic (venetian blinds) cross-validation. It is assumed that data rows are ordered correctly, so systematic cross-validation can be applied

**Value**

Pseudo-validation matrix (IxJ)

---

pcvcrossval	<i>Generate sequence of indices for cross-validation</i>
-------------	--

---

**Description**

Generates and returns sequence of object indices for each segment in random segmented cross-validation

**Usage**

```
pcvcrossval(cv = 1, nobj = NULL, resp = NULL)
```

**Arguments**

cv	cross-validation settings, can be a number, a list or a vector with integers.
nobj	number of objects in a dataset
resp	vector or matrix with response values to use in case of venetian blinds

**Details**

Parameter ‘cv’ defines how to split the rows of the training set. The split is similar to cross-validation splits, as PCV is based on cross-validation. This parameter can have the following values:

1. A number (e.g. ‘cv = 4’). In this case this number specifies number of segments for random splits, except ‘cv = 1’ which is a special case for leave-one-out (full cross-validation).
2. A list with 2 values: ‘list("name", nseg)’. In this case ‘"name"’ defines the way to make the split, you can select one of the following: ‘"loo"’ for leave-one-out, ‘"rand"’ for random splits or ‘"ven"’ for Venetian blinds (systematic) splits. The second parameter, ‘nseg’, is a number of segments for splitting the rows into. For example, ‘cv = list("ven", 4)’, shown in the code examples above, tells PCV to use Venetian blinds splits with 4 segments.
3. A vector with integer numbers, e.g. ‘cv = c(1, 2, 3, 1, 2, 3, 1, 2, 3)’. In this case number of values in this vector must be the same as number of rows in the training set. The values specify which segment a particular row will belong to. In case of the example shown here, it is assumed that you have 9 rows in the calibration set, which will be split into 3 segments. The first segment will consist of measurements from rows 1, 4 and 7.

**Value**

vector with object indices for each segment

---

pcvpca

*Procrustes cross-validation for PCA models*

---

**Description**

Procrustes cross-validation for PCA models

**Usage**

```
pcvpca(
  X,
  ncomp = min(nrow(X) - 1, col(X), 30),
  cv = list("ven", 4),
  center = TRUE,
  scale = FALSE
)
```

**Arguments**

X	matrix with predictors from the training set.
ncomp	number of components to use (more than the expected optimal number).
cv	which split method to use for cross-validation (see description for details).
center	logical, to center or not the data sets
scale	logical, to scale or not the data sets

**Details**

The method computes pseudo-validation matrix  $X_{pv}$ , based on PCA decomposition of calibration set 'X' and cross-validation. See description of the method in [1].

Parameter 'cv' defines how to split the rows of the training set. The split is similar to cross-validation splits, as PCV is based on cross-validation. This parameter can have the following values:

1. A number (e.g. 'cv = 4'). In this case this number specifies number of segments for random splits, except 'cv = 1' which is a special case for leave-one-out (full cross-validation).
2. A list with 2 values: 'list("name", nseg)'. In this case "name" defines the way to make the split, you can select one of the following: "loo" for leave-one-out, "rand" for random splits or "ven" for Venetian blinds (systematic) splits. The second parameter, 'nseg', is a number of segments for splitting the rows into. For example, 'cv = list("ven", 4)', shown in the code examples above, tells PCV to use Venetian blinds splits with 4 segments.
3. A vector with integer numbers, e.g. 'cv = c(1, 2, 3, 1, 2, 3, 1, 2, 3)'. In this case number of values in this vector must be the same as number of rows in the training set. The values specify which segment a particular row will belong to. In case of the example shown here, it is assumed that you have 9 rows in the calibration set, which will be split into 3 segments. The first segment will consist of measurements from rows 1, 4 and 7.

**Value**

Matrix with PV-set (same size as X)

**References**

1. S. Kucheryavskiy, O. Rodionova, A. Pomerantsev. Procrustes cross-validation of multivariate regression models. Submitted, 2022.

**Examples**

```
# load NIR spectra of Corn samples
data(corn)
X <- corn$spectra

# generate Xpv set based on PCA decomposition with A = 20 and venetian blinds split with 4 segments
Xpv <- pcvpca(X, ncomp = 20, center = TRUE, scale = FALSE, cv = list("ven", 4))

# show the original spectra and the PV-set (as is and mean centered)
oldpar <- par(mfrow = c(2, 2))
matplot(t(X), type = "l", lty = 1, main = "Original data")
matplot(t(Xpv), type = "l", lty = 1, main = "PV-set")
matplot(t(scale(X, scale = FALSE)), type = "l", lty = 1, main = "Original data (mean centered)")
matplot(t(scale(Xpv, scale = FALSE)), type = "l", lty = 1, main = "PV-set (mean centered)")
par(oldpar)
```

---

pcvpcr

*Procrustes cross-validation for PCR models*

---

**Description**

Procrustes cross-validation for PCR models

**Usage**

```
pcvpcr(
  X,
  Y,
  ncomp = min(nrow(X) - 1, ncol(X), 30),
  cv = list("ven", 4),
  center = TRUE,
  scale = FALSE
)
```

## Arguments

X	matrix with predictors from the training set.
Y	vector with response values from the training set.
ncomp	number of components to use (more than the expected optimal number).
cv	which split method to use for cross-validation (see description of method 'pcv-pls()' for details).
center	logical, to center or not the data sets
scale	logical, to scale or not the data sets

## Details

The method computes pseudo-validation matrix  $X_{pv}$ , based on PCR decomposition of calibration set  $X$ ,  $y$  and cross-validation.

Parameter 'cv' defines how to split the rows of the training set. The split is similar to cross-validation splits, as PCV is based on cross-validation. This parameter can have the following values:

1. A number (e.g. 'cv = 4'). In this case this number specifies number of segments for random splits, except 'cv = 1' which is a special case for leave-one-out (full cross-validation).
2. A list with 2 values: 'list("name", nseg)'. In this case "name" defines the way to make the split, you can select one of the following: "loo" for leave-one-out, "rand" for random splits or "ven" for Venetian blinds (systematic) splits. The second parameter, 'nseg', is a number of segments for splitting the rows into. For example, 'cv = list("ven", 4)', shown in the code examples above, tells PCV to use Venetian blinds splits with 4 segments.
3. A vector with integer numbers, e.g. 'cv = c(1, 2, 3, 1, 2, 3, 1, 2, 3)'. In this case number of values in this vector must be the same as number of rows in the training set. The values specify which segment a particular row will belong to. In case of the example shown here, it is assumed that you have 9 rows in the calibration set, which will be split into 3 segments. The first segment will consist of measurements from rows 1, 4 and 7.

## Value

Pseudo-validation matrix (same size as  $X$ ) with an additional attribute, 'D' which contains the scaling coefficients ( $ck/c$ )

## References

1. S. Kucheryavskiy, O. Rodionova, A. Pomerantsev. Procrustes cross-validation of multivariate regression models. Submitted, 2022.

## Examples

```
# load NIR spectra of Corn samples
data(corn)
X <- corn$spectra
Y <- corn$moisture
```

```
# generate Xpv set based on PCA decomposition with A = 20 and venetian blinds split with 4 segments
Xpv <- pcvpcr(X, Y, ncomp = 20, center = TRUE, scale = FALSE, cv = list("ven", 4))

# show the original spectra and the PV-set (as is and mean centered)
oldpar <- par(mfrow = c(2, 2))
matplot(t(X), type = "l", lty = 1, main = "Original data")
matplot(t(Xpv), type = "l", lty = 1, main = "PV-set")
matplot(t(scale(X, scale = FALSE)), type = "l", lty = 1, main = "Original data (mean centered)")
matplot(t(scale(Xpv, scale = FALSE)), type = "l", lty = 1, main = "PV-set (mean centered)")
par(oldpar)
```

---

pcvpls

*Procrustes cross-validation for PLS models*


---

## Description

Procrustes cross-validation for PLS models

## Usage

```
pcvpls(
  X,
  Y,
  ncomp = min(nrow(X) - 1, ncol(X), 30),
  center = TRUE,
  scale = FALSE,
  cv = list("ven", 4)
)
```

## Arguments

X	matrix with predictors from the training set.
Y	vector or matrix with response values from the training set.
ncomp	number of components to use (more than the expected optimal number).
center	logical, to center or not the data sets
scale	logical, to scale or not the data sets
cv	which split method to use for cross-validation (see description for details).

## Details

The method computes pseudo-validation matrix Xpv, based on PLS decomposition of calibration set X, y and cross-validation.

Parameter ‘cv’ defines how to split the rows of the training set. The split is similar to cross-validation splits, as PCV is based on cross-validation. This parameter can have the following values:

1. A number (e.g. 'cv = 4'). In this case this number specifies number of segments for random splits, except 'cv = 1' which is a special case for leave-one-out (full cross-validation).
2. A list with 2 values: 'list("name", nseg)'. In this case 'name' defines the way to make the split, you can select one of the following: "loo" for leave-one-out, "rand" for random splits or "ven" for Venetian blinds (systematic) splits. The second parameter, 'nseg', is a number of segments for splitting the rows into. For example, 'cv = list("ven", 4)', shown in the code examples above, tells PCV to use Venetian blinds splits with 4 segments.
3. A vector with integer numbers, e.g. 'cv = c(1, 2, 3, 1, 2, 3, 1, 2, 3)'. In this case number of values in this vector must be the same as number of rows in the training set. The values specify which segment a particular row will belong to. In case of the example shown here, it is assumed that you have 9 rows in the calibration set, which will be split into 3 segments. The first segment will consist of measurements from rows 1, 4 and 7.

### Value

Pseudo-validation matrix (same size as X) with an additional attribute, 'D' which contains the scaling coefficients (ck/c)

### References

1. S. Kucheryavskiy, O. Rodionova, A. Pomerantsev. Procrustes cross-validation of multivariate regression models. Submitted, 2022.

### Examples

```
# load NIR spectra of Corn samples
data(corn)
X <- corn$spectra
Y <- corn$moisture

# generate Xpv set based on PCA decomposition with A = 20 and venetian blinds split with 4 segments
Xpv <- pcvpls(X, Y, ncomp = 20, center = TRUE, scale = FALSE, cv = list("ven", 4))

# show the original spectra and the PV-set (as is and mean centered)
oldpar <- par(mfrow = c(2, 2))
matplot(t(X), type = "l", lty = 1, main = "Original data")
matplot(t(Xpv), type = "l", lty = 1, main = "PV-set")
matplot(t(scale(X, scale = FALSE)), type = "l", lty = 1, main = "Original data (mean centered)")
matplot(t(scale(Xpv, scale = FALSE)), type = "l", lty = 1, main = "PV-set (mean centered)")
par(oldpar)

# show the heatmap with the scaling coefficients
plotD(Xpv)
```

---

pcvreg	<i>Procrustes cross-validation for multivariate regression models</i>
--------	---

---

### Description

This is a generic method, use ‘pcvpls()’ or ‘pcvpcr()’ instead.

### Usage

```
pcvreg(
  X,
  Y,
  ncomp = min(nrow(X) - 1, ncol(X), 30),
  cv = list("ven", 4),
  center = TRUE,
  scale = FALSE,
  funlist = list()
)
```

### Arguments

X	matrix with predictors from the training set.
Y	vector with response values from the training set.
ncomp	number of components to use (more than the expected optimal number).
cv	which split method to use for cross-validation (see description of method ‘pcvpls()’ for details).
center	logical, to center or not the data sets
scale	logical, to scale or not the data sets
funlist	list with functions for particular implementatio

---

plotD	<i>Plots heatmap for scaling coefficients obtained when generating PV-set for PCR or PLS</i>
-------	--

---

### Description

Plots heatmap for scaling coefficients obtained when generating PV-set for PCR or PLS

**Usage**

```
plotD(
  Xpv,
  colmap = colorRampPalette(c("blue", "white", "red"))(256),
  lim = c(-2, 4),
  xlab = "Components",
  ylab = "Segments",
  ...
)
```

**Arguments**

Xpv	PV-set generated by 'pcvpcr()' or 'pcvpls()'.
colmap	colormap - any with 256 colors.
lim	limits for color map (smallest/largest expected value), centered around 1.
xlab	label for x-axis
ylab	label for y-axis
...	any other parameters for method 'image'

**Value**

No return value, just creates a plot.

---

rotationMatrixToX1      *Creates a rotation matrix to map a vector x to [1 0 0 ... 0]*

---

**Description**

Creates a rotation matrix to map a vector x to [1 0 0 ... 0]

**Usage**

```
rotationMatrixToX1(x)
```

**Arguments**

x	Vector (sequence with J coordinates)
---	--------------------------------------

**Value**

Rotation matrix (JxJ)

---

`simpls`*SIMPLS algorithm*

---

**Description**

SIMPLS algorithm for calibration of PLS model

**Usage**

```
simpls(X, Y, ncomp)
```

**Arguments**

<code>X</code>	a matrix with x values (predictors)
<code>Y</code>	a matrix with y values (responses)
<code>ncomp</code>	number of components to calculate

**Value**

a list with computed weights, x- and y-loadings for PLS regression model.

**References**

[1]. S. de Jong. SIMPLS: An Alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18, 1993 (251-263).

# Index

## \* datasets

corn, [2](#)

corn, [2](#)

getCrossvalParams, [2](#)

getR, [3](#)

getxpvorth, [3](#)

normalize, [4](#)

pcv, [4](#)

pcvcrossval, [5](#)

pcvpca, [5](#), [6](#)

pcvpcr, [7](#)

pcvpls, [9](#)

pcvreg, [11](#)

plotD, [11](#)

rotationMatrixToX1, [12](#)

simpls, [13](#)