Package: jvcoords (via r-universe)

September 17, 2024

Type Package

Title Principal Component Analysis (PCA) and Whitening

Version 1.0.3

Date 2021-06-05

License GPL-3

URL https://github.com/seehuhn/jvcoords

Description Provides functions to standardize and whiten data, and to perform Principal Component Analysis (PCA). The main advantage of this package over alternatives like prcomp() is, that jvcoords makes it easy to convert (additional) data between the original and the transformed coordinates. The package also provides a class coords, which can represent affine coordinate transformations. This class forms the basis of the transformations provided by the package, but can also be used independently. The implementation has been optimized to be of comparable speed (and sometimes even faster) than existing alternatives.

Repository https://seehuhn.r-universe.dev

RemoteUrl https://github.com/seehuhn/jvcoords

RemoteRef HEAD

RemoteSha 5a1a3a2444b14e3bf42b847a4e6cd53725baf29f

Contents

jvcoords-package *Package overview*

Description

The jvcoords package provides functions to standardize and whiten data, and an implementation of Principal Component Analysis (PCA). All three transformations are implemented using a common class `coords` which allows to easily convert data from and to the new coordinate systems.

See the documentation for `standardize`, `whiten`, and `PCA` for information on how to use this package.

Author(s)

Jochen Voss <voss@seehuhn.de>

See Also

`standardize`, `whiten`, `PCA`, `coords`

coords *An S3 class to represent affine coordinate transforms*

Description

Perform affine coordinate transformations.

Usage

```
coords(p, name = NULL, shift = 0)
appendTrfm(trfm, op = c("diag", "orth"), val)
toCoords(trfm, x)
fromCoords(trfm, y, apply.shift = TRUE)
```

Arguments

`p`	The number of variables in the original data.
`name`	A short name for the coordinate transformation (optional).
`shift`	A value subtracted from the data as the first step of the coordinate transformation. Usually, this will be the mean of the data (optional).
`trfm`	An object of class `coords`.
`op`	The type of transformation to append.
`val`	Data for the transformation to append.
`x`	Data matrix, rows are observations, columns are variables.
`y`	Transformed data matrix, rows are observations, columns are variables.
`apply.shift`	Whether to apply the final shift of coordinates. Set this to `FALSE` in order to only apply the linear part of the transformation.

Details

The function coords() creates a new object representing an affine coordinate transformation. Initially, the object represents a shift by the amount shift, mapping p-dimensional vectors x to x-shift. The function appendTrfm() can then be used to modify the transformation. The optional argument name, if set, is used when printing objects of class coords.

The function toCoords() applies the affine transformation trfm to the data x. The data x must either be a vector of length trfm$p, in which case the result is a vector of length trfm$q, or a matrix with trfm$p columns, in which case the transformation is applied to each row of the matrix separately.

The function fromCoords() implements the inverse transform to toCoords(). The output always satisfies toCoords(trfm, fromCoords(trfm, y)) == y. If trfm$p == trfm$q, *i.e.* if the transformation is bijective, the fromCoords(trfm, toCoords(trfm, x)) == x also holds. The argument apply.shift can be set to false to apply only the linear part of the (inverse) transformation, leaving out the final shift.

The function appendTrfm() concatenates trfm with an additional, linear transformation and returns the result. The arguments op and val specify which kind of linear transformation to append. There are two choices for op:

- diag denotes multiplication with a diagonal matrix: an input vector x is mapped to the output x * val. The scaling factor val can either be a vector of length trfm$q (for element-wise scaling), or a number.
- orth denotes multiplication with an orthogonal matrix. val must be a matrix with orthogonal columns (not necessarily square) and trfm$q rows. An input vector x is mapped to the output x %*% orth.

The new transformation is applied after any other transformations already associated with trfm.

Value

An object of class coords, as a list with the following components:

p	the number of variables in the original data set
q	the number of variables in the transformed data set
shift	the affine part of the transformation
name	the name of the transformation
cmds	a representation of the transformation (internal use only)

Author(s)

Jochen Voss <voss@seehuhn.de>

See Also

standardize, whiten, PCA

Examples

```
pc <- PCA(iris[, 1:4], n.comp = 3)
toCoords(pc, c(5, 3, 4, 1))
fromCoords(pc, c(1, 0, 0))
```

PCA *Perform Principal Component Analysis (PCA)*

Description

Perform principal components analysis on a data matrix and return the results as an object of class `coords`.

Usage

```
PCA(x, n.comp, scale = FALSE, compute.scores = TRUE)
```

Arguments

`x`	A data matrix, rows are observations, columns are variables.
`n.comp`	How many principal components to compute.
`scale`	Whether to standardize the columns before doing PCA.
`compute.scores`	Whether to compute the scores (i.e. x in the new basis).

Details

This function performs Principal Component Analysis (PCA) on the data. Variables are always centred before the PCA is performed and, if `scale` is set, the variables will also be rescaled to unit variance.

If `compute.scores` is set to `FALSE`, only the information required for the `toPC()` and `fromPC()` to work is stored in the returned `coords` object; otherwise the scores will be stored in the `$y` field of the `coords` object.

The `PCA()` function is an alternative to the `prcomp()` command from the standard library. The main advantage of `PCA()` is that the `coords` class provides functions to convert between the original basis and the principal component basis.

Value

An object of class `coords`, with the following additional components added:

`loadings`	the loadings, each column is one of the new basis vectors
`y`	if `compute.scores==TRUE`, this is `x` expressed in the new basis
`var`	the variance of the data along each of the new basis vectors
`total.var`	the total variance of the data

Author(s)

Jochen Voss <voss@seehuhn.de>

See Also

coords; alternative implementations: prcomp, princomp

Examples

```
pc <- PCA(iris[, 1:4], scale = TRUE, n.comp = 2)
pc
plot(pc$y, col=iris$Species)
```

---

standardize *Standardize data*

Description

Standardize each column of a data matrix and return the results as an object of class coords.

Usage

```
standardize(x, compute.scores = TRUE)
```

Arguments

x	A data matrix, rows are observations, columns are variables.
compute.scores	Whether to compute the scores (i.e. x in the new basis).

Details

This function standardizes the columns of x by subtracting the mean of each column and then dividing by the standard deviation. The transformed data is stored in the $y field of the returned coords object.

If compute.scores is set to FALSE, only the information required for the toCoords() and fromCoords() to work is stored in the returned coords object; otherwise the scores (transformed data) will be stored in the $y field of the coords object.

Value

An object of class coords, with the following additional components added:

y	if compute.scores==TRUE, this is x expressed in the new basis

Author(s)

Jochen Voss <voss@seehuhn.de>

See Also

coords; alternative implementation scale

Examples

```
w <- standardize(iris[, 1:4])
colMeans(w$y)
apply(w$y, 2, sd)
```

---

whiten *Whiten data*

Description

Whiten data and return the results as an object of class coords.

Usage

```
whiten(x, compute.scores = TRUE)
```

Arguments

x	A data matrix, rows are observations, columns are variables.
compute.scores	Whether to compute the scores (i.e. x in the new basis).

Details

This function whitens the data by finding an affine transformation such that the transformed data has mean 0 and identity covariance matrix.

If compute.scores is set to FALSE, only the information required for the toCoords() and fromCoords() to work is stored in the returned coords object; otherwise the scores (transformed data) will be stored in the $y field of the coords object.

Value

An object of class coords, with the following additional components added:

loadings	the loadings, each column is one of the new basis vectors
y	if compute.scores==TRUE, this is x expressed in the new basis

Author(s)

Jochen Voss <voss@seehuhn.de>

See Also

coords

Examples

```
w <- whiten(iris[, 1:4])
colMeans(w$y)
round(cov(w$y), 3)
```

Index