Package ‘maxstablePCA’

September 6, 2024

Title Apply a PCA Like Procedure Suited for Multivariate Extreme Value Distributions

Type Package

Description Dimension reduction for multivariate data of extreme events with a PCA like procedure as described in Reinbott, Janßen, (2024), <doi:10.48550/arXiv.2408.10650>. Tools for necessary transformations of the data are provided.

Version 0.1.1

Maintainer Felix Reinbott <felix.reinbott@ovgu.de>

Depends R (>= 3.0.1)

License MIT + file LICENSE

Imports nloptr

Suggests testthat, evd, mev

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Felix Reinbott [aut, cre]

Repository CRAN

Date/Publication 2024-09-06 16:50:02 UTC

Contents

compress — *Transform data to compact representation given by max-stable PCA*

Description

Turn the given data into a compressed latent representation given by the fit of the max_stable_prcomp function. This is done by taking the max-matrix product of the data and the encoder matrix from the fit.

Usage

```
compress(fit, data)
```

Arguments

`fit`	max_stable_prcomp object. Data should be assumed to follow the same distribution as the data used in max_stable_prcomp.
`data`	array with same number of columns as the data of the fit object.

Value

An array of shape nrow(data), p giving the encoded representation of the data in p components which are also unit Frechet distributed which is to be takin into consideration for further analysis.

See Also

`max_stable_prcomp()`, `maxmatmul()`

Examples

```
# generate some data with the desired margins
dat <- matrix(evd::rfrechet(300), 100, 3)
maxPCA <- max_stable_prcomp(dat, 2)

#  look at summary to obtain further information about
# loadings the space spanned and loss function
summary(maxPCA)

# transfrom data to compressed representation
# for a representation that is p-dimensional,
# preserves the max-stable structure and is numeric solution to
# optimal reconstruction.
compr <- compress(maxPCA, dat)

# For visual examination reconstruct original vector from compressed representation
rec <- reconstruct(maxPCA, dat)
```

maxmatmul — *Multiply two matrices with a matrix product that uses maxima instead of addition*

Description

By calculating the entries with

$$(A \diamond B)_{ij} = \max_{j=1,\dots,l} A_{il} B_{lj}$$

for appropriate dimensions. Note that this operation is particularly useful when working with multivariate exreme value distributions, because, if the margins are standardized to standard Fréchet margins, then the max-matrix product of a matrix A and a multivariate extreme value distribution Z with standard Fréchet margins has the same margins up to scaling.

Usage

```
maxmatmul(A, B)
```

Arguments

`A`	a non-negative array of dim n, k
`B`	a non-negative array of dim k, l

Value

A non netgative array of dim n, l. The entries are given by the maximum of componentwise multiplication of rows from A and columns from B.

Examples

```
# Set up example matrices
A <- matrix(c(1,2,3,4,5,6), 2, 3)
B <- matrix(c(1,2,1,2,1,2), 3, 2)

# calling the function
m1 <- maxmatmul(A, B)

# can be used for matrix-vector multiplication as well
v <- c(7,4,7)
m2 <- maxmatmul(A, v)
m3 <- maxmatmul(v,v)
```

`max_stable_prcomp` *Calculate max-stable PCA with dimension p for given dataset*

Description

Find an optimal encoding of data of extremes using max-linear combinations by a distance minimization approach. Can be used to check if the data follows approximately a generalized max-linear model. For details on the statistical procedure it is advised to consult the articles "F. Reinbott, A. Janßen, Principal component analysis for max-stable distributions (https://arxiv.org/abs/2408.10650)" and "M.Schlather F. Reinbott, A semi-group approach to Principal Component Analysis (https://arxiv.org/abs/2112.04026)".

Usage

```
max_stable_prcomp(data, p, s = 3, n_initial_guesses = 150, norm = "l1", ...)
```

Arguments

`data` array or data.frame of n observations of d variables with unit Frechet margins. The max-stable PCA is fitted to reconstruct this dataset with a rank p approximation.

`p` integer between 1 and ncol(data). Determines the dimension of the encoded state, i.e. the number of max-linear combinations in the compressed representation.

`s` (default = 3), numeric greater than 0. Hyperparameter for the stable tail dependence estimator used in tn the calculation.

`n_initial_guesses` number of guesses to choose a valid initial value for optimization from. This procedure uses a pseudo random number generator so setting a seed is necessary for reproducibility.

`norm` (delfault "l1") which norm to use for the spectral measure estimator, currently only l1 and sup norm "linfty" are available.

`...` additional parameters passed to `link{nloptr::slsqp()}`

Value

object of class max_stable_prcomp with slots p, inserted value of dimension, decoder_matrix, an array of shape (d,p), where the columns represent the basis of the max-linear space for the reconstruction. encoder_matrix, an array of shape (p,d), where the rows represent the loadings as max-linear combinations for the compressed representation. reconstr_matrix, an array of shape (d,d), where the matrix is the mapping of the data to the reconstruction used for the distance minimization. loss_fctn_value, float representing the final loss function value of the fit. optim_conv_status, integer indicating the convergence of the optimizer if greater than 0.

Examples

```
# generate some data with the desired margins
dat <- matrix(evd::rfrechet(300), 100, 3)
maxPCA <- max_stable_prcomp(dat, 2)

# look at summary to obtain further information about
# loadings the space spanned and loss function
summary(maxPCA)

# transfrom data to compressed representation
# for a representation that is p-dimensional,
# preserves the max-stable structure and is numeric solution to
# optimal reconstruction.
compr <- compress(maxPCA, dat)

# For visual examination reconstruct original vector from compressed representation
rec <- reconstruct(maxPCA, dat)
```

---

reconstruct *Obtain reconstructed data for PCA*

Description

Map the data to the reconstruction given by the fit of the max_stable_prcomp function. This is done by taking the max-matrix product of the data and the reconstruction matrix from the fit.

Usage

```
reconstruct(fit, data)
```

Arguments

`fit`	max_stable_prcomp object. Data should be assumed to follow the same distribution as the data used in max_stable_prcomp.
`data`	array with same number of columns as the data of the fit object.

Value

An array of shape nrow(data), p giving the encoded representation of the data in p components which are also unit Frechet distributed which is to be takin into consideration for further analysis.

See Also

`max_stable_prcomp()`, `maxmatmul()`

Examples

```
# generate some data with the desired margins
dat <- matrix(evd::rfrechet(300), 100, 3)
maxPCA <- max_stable_prcomp(dat, 2)

#  look at summary to obtain further information about
# loadings the space spanned and loss function
summary(maxPCA)

# transfrom data to compressed representation
# for a representation that is p-dimensional,
# preserves the max-stable structure and is numeric solution to
# optimal reconstruction.
compr <- compress(maxPCA, dat)

# For visual examination reconstruct original vector from compressed representation
rec <- reconstruct(maxPCA, dat)
```

`summary.max_stable_prcomp`

Print summary of a max_stable_prcomp object.

Description

Print summary of a max_stable_prcomp object.

Usage

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

Arguments

`object`	max_stable_prcomp object. Data should be assumed to follow the same distribution as the data used in max_stable_prcomp.
`...`	additional unused arguments.

Value

Same as `base::print()`.

See Also

`max_stable_prcomp()`

`transform_orig_margins` *Transform the columns of a transformed dataset to original margins*

Description

Since the dataset is intended to be transformed for PCA, this function takes a dataset transformed_data and transforms the margins to the marginal distribution of the dataset orig_data.

Usage

```
transform_orig_margins(transformed_data, orig_data)
```

Arguments

`transformed_data`	arraylike data of dimension n, d
`orig_data`	arraylike data of dimension n , d

Value

array of dimension n,d with transformed columns of transformed_data that follow approximately the same marginal distribution of orig_data.

See Also

`max_stable_prcomp()`, `transform_unitfrechet()`, [mev::fit.gev()] for information about why to transform data

[mev::fit.gev()]: R:mev::fit.gev()

Examples

```
# create a sample
dat <- rnorm(1000)
transformed_dat <- transform_unitpareto(dat)
```

`transform_unitfrechet` *Transform the columns of a dataset to (approximately) unit Frechet margins*

Description

Transforms columns of dataset to unit Frechet margins, to ensure the theoretical requirements are satisfied for the application of `max_stable_prcomp` using the empirical distribution function.

Usage

```
transform_unitfrechet(data)
```

Arguments

`data`	array or vector with the data which columns are to be transformed

Value

array or vector of same shape and type as data with the transformed data with unit Frechet margins-

See Also

`max_stable_prcomp()`, `transform_orig_margins()`, [mev::fit.gev())] for information about why to transform data.

[mev::fit.gev())]: R:mev::fit.gev())

Examples

```
# sample some data
dat <- rnorm(1000)
transformed_dat <- transform_unitfrechet(dat)

# Look at a plot of distribution
boxplot(transformed_dat)
plot(stats::ecdf(transformed_dat))
```

`transform_unitpareto` *Transform the columns of a dataset to unit Pareto*

Description

Transforms columns of dataset to unit Pareto margins, to ensure the theoretical requirements are satisfied for the application of `max_stable_prcomp` using the empirical distribution function.

Usage

```
transform_unitpareto(data)
```

Arguments

`data`	array or vector with the data which columns are to be transformed

Value

array or vector of same shape and type as data with the transformed data with unit Frechet margins-

See Also

max_stable_prcomp(), transform_orig_margins(), [mev::fit.gev())] for information about why to transform data.

[mev::fit.gev())]: R:mev::fit.gev())

Examples

```
# sample some data
dat <- rnorm(1000)
transformed_dat <- transform_unitfrechet(dat)

# Look at a plot of distribution
boxplot(transformed_dat)
plot(stats::ecdf(transformed_dat))
```

Index