Package ‘PLNmodels’

January 27, 2020

Title Poisson Lognormal Models

Version 0.9.5

Description The Poisson-lognormal model and variants can be used for a variety of multivariate problems when count data are at play, including principal component analysis for count data (Chiquet, Mariadassou and Robin, 2018 <doi:10.1214/18-AOAS1177>), discriminant analysis and network inference (Chiquet, Mariadassou and Robin, 2018 <http://proceedings.mlr.press/v97/chiquet19a.html>). Implements variational algorithms to fit such models accompanied with a set of functions for visualization and diagnostic.

URL https://jchiquet.github.io/PLNmodels/

BugReports https://github.com/jchiquet/PLNmodels/issues

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.0.2

Depends R (>= 3.4)

LazyData true

biocViews

Imports methods, stats, MASS, parallel, R6, glassoFast, Matrix, Rcpp, nloptr (>= 1.2.0), igraph, grid, gridExtra, dplyr, tidyr, ggplot2, corrplot, magrittr

Suggests knitr, rmarkdown, testthat, covr, pkgdown, biomformat, phyloseq

LinkingTo Rcpp, RcppArmadillo, nloptr (>= 1.2.0)

VignetteBuilder knitr

Collate 'PLNfit-class.R' 'PLN.R' 'PLNLDA.R' 'PLNLDAfit-S3methods.R'

  'PLNLDAfit-class.R' 'PLNPCA.R' 'PLNPCAfamily-S3methods.R'
  'PLNfamily-class.R' 'PLNPCAfamily-class.R'
  'PLNPCAfit-S3methods.R' 'PLNPCAfit-class.R'
  'PLNfamily-S3methods.R' 'PLNfit-S3methods.R'
Topics documented:

- 'PLNmodels-package.R' 'PLNnetwork.R'
- 'PLNnetworkfamily-S3methods.R' 'PLNnetworkfamily-class.R'
- 'PLNnetworkfit-S3methods.R' 'PLNnetworkfit-class.R'
- 'RcppExports.R' 'deprecated.R' 'import_utils.R' 'mollusk.R'
- 'plot_utils.R' 'trichoptera.R' 'utils.pipe.R' 'utils.R'

NeedsCompilation yes

Author Julien Chiquet [aut, cre] (<https://orcid.org/0000-0002-3629-3429>),
        Mahendra Mariadassou [aut] (<https://orcid.org/0000-0003-2986-354X>),
        Stéphane Robin [ctb]

Maintainer Julien Chiquet <julien.chiquet@inrae.fr>

Repository CRAN

Date/Publication 2020-01-27 21:10:02 UTC

R topics documented:

- coef.PLNfit .................................................. 3
- coef.PLNLDAfit ............................................. 4
- coefficient_path .......................................... 4
- compute_offset ............................................. 5
- extract_probs .............................................. 6
- fisher ....................................................... 8
- fitted.PLNfit .............................................. 9
- getBestModel.PLNPCAfamily .............................. 9
- getModel.PLNPCAfamily .................................. 10
- mollusk ..................................................... 11
- PLN ........................................................... 12
- PLNfamily .................................................. 14
- PLNfit ....................................................... 15
- PLNLDA ...................................................... 18
- PLNLDAfit ................................................... 20
- PLNmodels .................................................. 23
- PLNnetwork ............................................... 24
- PLNnetworkfamily ........................................ 26
- PLNnetworkfit ............................................. 28
- PLNPCA ...................................................... 31
- PLNPCAfamily .............................................. 33
- PLNPCAfit ................................................... 35
- plot.PLNLDAfit ............................................. 38
- plot.PLNnetworkfamily .................................. 39
- plot.PLNnetworkfit ....................................... 41
- plot.PLNPCAfamily ........................................ 42
- plot.PLNPCAfit ............................................. 43
- predict.PLNfit ............................................. 44
- predict.PLNLDAfit ......................................... 45
- prepare_data .............................................. 46
- rPLN ......................................................... 47
**coef.PLNfit**

Extract model coefficients from objects returned by PLN and its variants

### Description

Extracts model coefficients from objects returned by PLN and its variants

### Usage

```r
## S3 method for class 'PLNfit'
coef(object, type = c("main", "covariance"), ...)
```

### Arguments

- `object`: an R6 object with class PLNfit
- `type`: type of parameter that should be extracted. Either "main" (default) for $\Theta$ or "covariance" for $\Sigma$
- `...`: additional parameters for S3 compatibility. Not used

### Value

A matrix of coefficients extracted from the PLNfit model.

### See Also

`sigma, vcov, standard_error`

### Examples

```r
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1 + offset(log(Offset)), data = trichoptera)
coef(myPLN)  # Theta
coef(myPLN, type = "covariance")  # Sigma
```
**coef.PLNLDAfit**

Extracts model coefficients from objects returned by PLNDA

Description

The method for objects returned by PLNDA only returns coefficients associated to the $\Theta$ part of the model (see the PLNDA vignette for mathematical details).

Usage

```r
## S3 method for class 'PLNLDAfit'
coef(object, ...)
```

Arguments

- `object`: an R6 object with class PLNfit
- `...`: additional parameters for S3 compatibility. Not used

Value

Either NULL or a matrix of coefficients extracted from the PLNLDAfit model.

Examples

```r
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ Wind, grouping = Group, data = trichoptera)
coef(myPLNLDA)
```

**coefficient_path**

Extract the regularization path of a PLNnetwork fit

Description

Extract the regularization path of a PLNnetwork fit

Usage

```r
coefficient_path(Robject, precision = TRUE, corr = TRUE)
```
compute_offset

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robject</td>
<td>an object with class PLNnetworkfamily, i.e. an output from PLNnetwork</td>
</tr>
<tr>
<td>precision</td>
<td>a logical, should the coefficients of the precision matrix Omega or the covariance matrix Sigma be sent back. Default is TRUE.</td>
</tr>
<tr>
<td>corr</td>
<td>a logical, should the correlation (partial in case precision = TRUE) be sent back. Default is TRUE.</td>
</tr>
</tbody>
</table>

Value

Send back a tibble/data.frame.

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fits <- PLNnetwork(Abundance ~ 1, data = trichoptera)
head(coefficient_path(fits))

compute_offset

Compute offsets from a count data using one of several normalization schemes

Description

Computes offsets from the count table using one of several normalization schemes (TSS, CSS, RLE, GMPR, etc) described in the literature.

Usage

compute_offset(counts, offset = c("TSS", "GMPR", "RLE", "CSS", "none"), ...)

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>counts</td>
<td>Required. An abundance count table, preferably with dimensions names and species as columns.</td>
</tr>
<tr>
<td>offset</td>
<td>Optional. Normalisation scheme used to compute scaling factors used as offset during PLN inference. Available schemes are &quot;TSS&quot; (Total Sum Scaling, default), &quot;CSS&quot; (Cumulative Sum Scaling, used in metagenomeSeq), &quot;RLE&quot; (Relative Log Expression, used in DESeq2), &quot;GMPR&quot; (Geometric Mean of Pairwise Ratio, introduced in Chen et al., 2018) or &quot;none&quot;. Alternatively the user can supply its own vector or matrix of offsets (see note for specification of the user-supplied offsets).</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters passed on to specific methods (for now CSS and RLE)</td>
</tr>
</tbody>
</table>
Details

RLE has an additional pseudocounts argument to add pseudocounts to the observed counts (defaults to 0). CSS has an additional reference argument to choose the location function used to compute the reference quantiles (defaults to median as in the Nature publication but can be set to mean to reproduce behavior of functions cumNormStat* from metagenomeSeq). Note that (i) CSS normalization fails when the median absolute deviation around quantiles does not become instable for high quantiles (limited count variations both within and across samples) and/or one sample has less than two positive counts, (ii) RLE fails when there are no common species across all samples and (iii) GMPR fails if a sample does not share any species with all other samples.

Value

If offset = “none”, NULL else a vector of length nrow(counts) with one offset per sample.

References


Examples

data(trichoptera)
counts <- trichoptera$Abundance
compute_offset(counts)
## Other normalization schemes
compute_offset(counts, offset = "GMPR")
compute_offset(counts, offset = "RLE", pseudocounts = 1)
## User supplied offsets
my_offset <- setNames(rep(1, nrow(counts)), rownames(counts))
compute_offset(counts, offset = my_offset)
extract_probs

Usage

extract_probs(
  Robject,
  penalty = NULL,
  index = NULL,
  crit = c("StARS", "BIC", "EBIC"),
  format = c("matrix", "vector"),
  tol = 1e-05
)

Arguments

Robject an object with class PLNnetworkfamily, i.e. an output from \texttt{PLNnetwork}
penalty penalty used for the bootstrap subsamples
index Integer index of the model to be returned. Only the first value is taken into account.
crit a character for the criterion used to performed the selection. Either "BIC", "ICL", "EBIC", "StARS", "R_squared". Default is \texttt{ICL} for PLNPCA, and \texttt{BIC} for PLNnetwork. If \texttt{StARS} (Stability Approach to Regularization Selection) is chosen and stability selection was not yet performed, the function will call the method stability_selection with default argument.
format output format. Either a matrix (default) or a named vector.
tol tolerance for rounding error when comparing penalties.

Value

Either a matrix or named vector of edge-wise probabilities. In the latter case, edge names follow igraph convention.

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
nets <- PLNnetwork(Abundance ~ 1 + offset(log(Offset)), data = trichoptera)
## Not run:
stability_selection(nets)
probs <- extract_probs(nets, crit = "StARS", format = "vector")
probs

## End(Not run)

## Not run:
## Add edge attributes to graph using igraph
net_stars <- getBestModel(nets, "StARS")
g <- plot(net_stars, type = "partial_cor", plot=F)
library(igraph)
E(g)$prob <- probs[as_ids(E(g))]
g
Description

Extracts Fisher information matrix of \( \Theta \) from objects returned by \texttt{PLN} and its variants. Fisher matrix is computed using one of two approximation scheme: \textit{wald} (default, conservative, gives large confidence interval) or \textit{louis} (anticonservative). Note that the Fisher information matrix is the full-data version (scaled by the number of observations), usually noted

\[ I_n(\theta) \]

Usage

\texttt{fisher(object, type)}

\texttt{## S3 method for class \textquotesingle{PLNfit}\textquotesingle{}}
\texttt{fisher(object, type = \texttt{c\textquotesingle{\texttt{wald}, \texttt{\textquotesingle{louis}}}}\textquotesingle{})}

Arguments

\begin{itemize}
  \item \texttt{object} \hspace{1cm} an R6 object with class \texttt{PLNfit} \\
  \item \texttt{type} \hspace{1cm} Either \texttt{wald} (default) or \texttt{louis}. Approxomation scheme used to compute the Fisher information matrix
\end{itemize}

Value

A block-diagonal matrix with \( p \) (number of species) blocks of size \( d \) (number of covariates), assuming \( \Theta \) is a matrix of size \( d \times p \).

Methods (by class)

- \texttt{PLNfit}: Fisher information matrix for \texttt{PLNfit}

See Also

\texttt{standard_error} for standard errors
fitted.PLNfit

Extracts model fitted values from objects returned by PLN and its variants

Description
Extracts model fitted values from objects returned by PLN and its variants

Usage
```r
## S3 method for class 'PLNfit'
fitted(object, ...)
```

Arguments
- `object`: an R6 object with class PLNfit
- `...`: additional parameters for S3 compatibility. Not used

Value
A matrix of Fitted values extracted from the object object.

getBestModel.PLNPCAfamily

Best model extraction from a collection of models

Description
Best model extraction from a collection of models

Usage
```r
## S3 method for class 'PLNPCAfamily'
getBestModel(Robject, crit = c("ICL", "BIC", "R_squared"), ...)
```

```r
getBestModel(Robject, crit, ...)
```

```r
## S3 method for class 'PLNnetworkfamily'
getBestModel(  
  Robject,  
  crit = c("BIC", "loglik", "R_squared", "EBIC", "StARS"),  
  ...,  
)
```
Arguments

- **Robject**: an object with class `PLNPCAfamily` or `PLNnetworkfamily`
- **crit**: a character for the criterion used to performed the selection. Either "BIC", "ICL", "EBIC", "StARS", "R_squared". Default is ICL for PLNPCA, and BIC for PLNnetwork. If StARS (Stability Approach to Regularization Selection) is chosen and stability selection was not yet performed, the function will call the method `stability_selection` with default argument.
- ... additional parameters for StARS criterion (only for PLNnetwork). stability, a scalar indicating the target stability (= 1 - 2 beta) at which the network is selected. Default is 0.9.

Value

Send back an object with class `PLNPCAfit` or `PLNnetworkfit`.

Methods (by class)

- `PLNPCAfamily`: Model extraction for `PLNPCAfamily`
- `PLNnetworkfamily`: Model extraction for `PLNnetworkfamily`

Examples

```r
## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPCA <- PLNPCA(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, ranks = 1:4)
myModel <- getBestModel(myPCA)
## End(Not run)
```

getModel.PLNPCAfamily  Model extraction from a collection of models

Description

Model extraction from a collection of models

Usage

```r
## S3 method for class 'PLNPCAfamily'
getModel(Robject, var, index = NULL)

getModel(Robject, var, index)

## S3 method for class 'PLNnetworkfamily'
getModel(Robject, var, index = NULL)
```
Arguments

Robject  an R6 object with class PLNPCAfamily or PLNnetworkfamily

var      value of the parameter (rank for PLNPCA, sparsity for PLNnetwork) that identifies the model to be extracted from the collection. If no exact match is found, the model with closest parameter value is returned with a warning.

index    Integer index of the model to be returned. Only the first value is taken into account.

Value

Sends back an object with class PLNPCAfit or PLNnetworkfit.

Methods (by class)

- PLNPCAfamily: Model extraction for PLNPCAfamily
- PLNnetworkfamily: Model extraction for PLNnetworkfamily

Examples

```r
## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPCA <- PLNPCA(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, ranks = 1:5)
myModel <- getModel(myPCA, 2)
## End(Not run)
```

mollusk

Mollusk data set

Description

This data set gives the abundance of 32 mollusk species in 163 samples. For each sample, 4 additional informations (covariates) are knowne.

Usage

mollusk

Format

A list with 2 two data frames:

- **Abundance** a 163 x 32 data frame of abundancies/counts (163 samples and 32 mollusk species)
- **Covariate** a 163 x 4 data frame of covariates:
  - **site** a factor with 8 levels indicating the sampling site
  - **season** a factor with 4 levels indicating the season
**method**  a factor with 2 levels for the method of sampling - wood or string

**duration**  a numeric with 3 levels for the time of exposure in week

In order to prepare the data for using formula in multivariate analysis (multiple outputs and inputs), use `prepare_data`. Original data set has been extracted from ade4.

**Source**

Data from Richardot-Coulet, Chessel and Bournaud.

**References**


**See Also**

`prepare_data`

**Examples**

```r
data(mollusk)
mollusc <- prepare_data(mollusk$Abundance, mollusk$Covariate)
```

---

**PLN**  

*Poisson lognormal model*

**Description**

Fit the multivariate Poisson lognormal model with a variational algorithm. Use the (g)lm syntax for model specification (covariates, offsets, weights).

**Usage**

```r
PLN(formula, data, subset, weights, control = list())
```

**Arguments**

- **formula**: an object of class "formula": a symbolic description of the model to be fitted.
- **data**: an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which PLN is called.
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **weights**: an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.
- **control**: a list for controlling the optimization. See details.
Details

The parameter control is a list controlling the optimization with the following entries:

- "covariance" character setting the model for the covariance matrix. Either "full", "diagonal" or "spherical". Default is "full".
- "trace" integer for verbosity.
- "inception" Set up the initialization. By default, the model is initialized with a multivariate linear model applied on log-transformed data, and with the same formula as the one provided by the user. However, the user can provide a PLNfit (typically obtained from a previous fit), which sometimes speeds up the inference.
- "ftol_rel" stop when an optimization step changes the objective function by less than ftol multiplied by the absolute value of the parameter. Default is 1e-6 when n < p, 1e-8 otherwise.
- "ftol_abs" stop when an optimization step changes the objective function by less than ftol multiplied by the absolute value of the parameter. Default is 0.
- "xtol_rel" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is 1e-4.
- "xtol_abs" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is 1e-4 for the variational variance parameters, 0 otherwise.
- "lower_bound" the lower bound (box constraint) for the variational variance parameters. Default is 1e-4.
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10000.
- "maxtime" stop when the optimization time (in seconds) exceeds maxtime. Default is -1 (no restriction).
- "algorithm" the optimization method used by NLOPT among LD type, i.e. "CCSAQ", "MMA", "LBFGS", "VAR1", "VAR2". See NLOPT documentation for further details. Default is "CCSAQ".

Value

an R6 object with class PLNfit

See Also

The class PLNfit

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1, data = trichoptera)
An R6 Class to represent a collection of PLNfit

Description

super class for PLNPCAfamily and PLNnetworkfamily
This class comes with a set of methods, some of them being useful for the user: See the documentation for getBestModel, getModel and plot.

Public fields

  responses  the matrix of responses common to every models
  covariates the matrix of covariates common to every models
  offsets   the matrix of offsets common to every models
  weights   the vector of observation weights
  models    a list of PLNnetworkfit object, one per penalty.
  inception a PLNfit object, obtained when no sparsifying penalty is applied.

Active bindings

  criteria a data frame with the values of some criteria (variational lower bound J, BIC, ICL and R2) for the different models.

Methods

Public methods:

  • PLNfamily$new()
  • PLNfamily$postTreatment()
  • PLNfamily$getModel()
  • PLNfamily$plot()
  • PLNfamily$show()
  • PLNfamily$print()
  • PLNfamily$clone()

Method new():

  Usage:
  PLNfamily$new(responses, covariates, offsets, weights, control)

Method postTreatment():

  Usage:
  PLNfamily$postTreatment()

Method getModel():

**PLNfit**

An R6 Class to represent a PLNfit in a standard, general framework

**Method** `getModel()`:

*Usage:*

```r
PLNfamily$getModel(var, index = NULL)
```

**Method** `plot()`:

*Usage:*

```r
PLNfamily$plot(criteria, annotate = TRUE)
```

**Method** `show()`:

*Usage:*

```r
PLNfamily$show()
```

**Method** `print()`:

*Usage:*

```r
PLNfamily$print()
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```r
PLNfamily$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.

---

**Description**

The function `PLN` fit a model which is an instance of a object with class `PLNfit`. Objects produced by the functions `PLNnetwork`, `PLNPCA` and `PLNLDA` also enjoy the method of `PLNfit` by inheritance. This class comes with a set of R6 methods, some of them being useful for the user and exported as S3 methods. See the documentation for `coef`, `sigma.PLNfit`, `predict`, `vcov` and `standard_error`. Fields are accessed via active binding and cannot be changed by the user.

**Public fields**

- `model` character: the model used for the coavariance (either "spherical", "diagonal" or "full")

**Active bindings**

- `model_par` a list with the matrices of parameters found in the model (Theta, Sigma, plus some others depending on the variant)
- `var_par` a list with two matrices, M and S, which are the estimated parameters in the variational approximation
- `latent` a matrix: values of the latent vector (Z in the model)
- `optim_par` a list with parameters useful for monitoring the optimization
model character: the model used for the coavariance (either "spherical", "diagonal" or "full")
loglik variational lower bound of the loglikelihood
loglik_vec element-wise variational lower bound of the loglikelihood
BIC variational lower bound of the BIC
ICL variational lower bound of the ICL
R_squared approximated goodness-of-fit criterion
nb_param number of parameters in the current PLN model
criteria a vector with loglik, BIC, ICL, R_squared and number of parameters

Methods

**Public methods:**

- `PLNfit$update()`
- `PLNfit$new()`
- `PLNfit$optimize()`
- `PLNfit$set_R2()`
- `PLNfit$postTreatment()`
- `PLNfit$latent_pos()`
- `PLNfit$VEstep()`
- `PLNfit$predict()`
- `PLNfit$compute_fisher()`
- `PLNfit$compute_standard_error()`
- `PLNfit$show()`
- `PLNfit$print()`
- `PLNfit$clone()`

**Method update():**

*Usage:*

```r
PLNfit$update(
  Theta = NA,
  Sigma = NA,
  M = NA,
  S = NA,
  Ji = NA,
  R2 = NA,
  Z = NA,
  A = NA,
  monitoring = NA
)
```

**Method new():**

*Usage:*

```r
PLNfit$new(responses, covariates, offsets, weights, model, control)
```
Method optimize():
Usage:
PLNfit$optimize(responses, covariates, offsets, weights, control)

Method set_R2():
Usage:
PLNfit$set_R2(responses, covariates, offsets, weights, nullModel = NULL)

Method postTreatment():
Usage:
PLNfit$postTreatment(
  responses,
  covariates,
  offsets,
  weights = rep(1, nrow(responses)),
  type = c("wald", "louis"),
  nullModel = NULL
)

Method latent_pos():
Usage:
PLNfit$latent_pos(covariates, offsets)

Method VEstep():
Usage:
PLNfit$VEstep(X, O, Y, control = list())

Method predict():
Usage:
PLNfit$predict(newdata, type = c("link", "response"), envir = parent.frame())

Method compute_fisher():
Usage:
PLNfit$compute_fisher(type = c("wald", "louis"), X = NULL)

Method compute_standard_error():
Usage:
PLNfit$compute_standard_error()

Method show():
Usage:
PLNfit$show(
  model = paste("A multivariate Poisson Lognormal fit with", private$covariance,
                 "covariance model.\n")
)
Method print():

Usage:
PLNfit$print()

Method clone(): The objects of this class are cloneable with this method.

Usage:
PLNfit$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

Examples

## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1, data = trichoptera)
class(myPLN)
print(myPLN)

## End(Not run)

---

**PLNLDA**

**Poisson lognormal model towards Linear Discriminant Analysis**

**Description**

Fit the Poisson lognormal for LDA with a variational algorithm. Use the (g)lm syntax for model specification (covariates, offsets).

**Usage**

```r
PLNLDA(formula, data, subset, weights, grouping, control = list())
```

**Arguments**

- **formula**: an object of class "formula": a symbolic description of the model to be fitted.
- **data**: an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **weights**: an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.
- **grouping**: a factor specifying the class of each observation used for discriminant analysis.
- **control**: a list for controlling the optimization process. See details.
Details

The parameter control is a list controlling the optimization with the following entries:

- "covariance" character setting the model for the covariance matrix. Either "full" or "spherical". Default is "full".
- "trace" integer for verbosity.
- "inception" Set up the initialization. By default, the model is initialized with a multivariate linear model applied on log-transformed data. However, the user can provide a PLNfit (typically obtained from a previous fit), which often speed up the inference.
- "ftol_rel" stop when an optimization step changes the objective function by less than ftol multiplied by the absolute value of the parameter. Default is 1e-6
- "ftol_abs" stop when an optimization step changes the objective function by less than ftol multiplied by the absolute value of the parameter. Default is 1e-6
- "xtol_rel" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is 1e-4
- "xtol_abs" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is 1e-4
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10000
- "maxtime" stop when the optimization time (in seconds) exceeds maxtime. Default is -1 (no restriction)
- "algorithm" the optimization method used by NLOPT among LD type, i.e. "CCSAQ", "MMA", "LBFGS", "VAR1", "VAR2". See NLOPT documentation for further details. Default is "CC-SAQ".
- "lower_bound" the lower bound (box constraint) for the variational variance parameters. Default is 1e-4.

Value

an R6 object with class PLNLDAfit

See Also

The class PLNLDAfit

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ 1, grouping = Group, data = trichoptera)
PLNLDAfit

**Description**

The function `PLNLDA` produces an instance of an object with class `PLNLDAfit`. This class comes with a set of methods, some of them being useful for the user: See the documentation for the methods inherited by `PLNfit`, the `plot.PLNPCAfit` method for LDA visualization and `predict.PLNPCAfit` method for prediction.

**Super class**

`PLNmodels::PLNfit` -> `PLNLDAfit`

**Public fields**

- `var_par` a list with two matrices, M and S, which are the estimated parameters in the variational approximation.
- `optim_par` a list with parameters useful for monitoring the optimization.
- `loglik` variational lower bound of the loglikelihood.
- `BIC` variational lower bound of the BIC.
- `ICL` variational lower bound of the ICL.
- `R_squared` approximated goodness-of-fit criterion.
- `criteria` a vector with loglik, BIC, ICL, R_squared and number of parameters.

**Active bindings**

- `rank` the dimension of the current model.
- `model_par` a list with the matrices associated with the estimated parameters of the PLN model: Theta (covariates), Sigma (latent covariance), B (latent loadings), P (latent position) and Mu (group means).
- `var_par` a list with two matrices, M and S, which are the estimated parameters in the variational approximation.
- `optim_par` a list with parameters useful for monitoring the optimization.
- `loglik` variational lower bound of the loglikelihood.
- `BIC` variational lower bound of the BIC.
- `ICL` variational lower bound of the ICL.
- `R_squared` approximated goodness-of-fit criterion.
- `criteria` a vector with loglik, BIC, ICL, R_squared and number of parameters.
- `nb_param` number of parameters in the current PLN model.
- `percent_var` the percent of variance explained by each axis.
- `corr_map` a matrix of correlations to plot the correlation circles.
- `scores` a matrix of scores to plot the individual factor maps.
Methods

Public methods:

• PLNLDAfit$new()
• PLNLDAfit$optimize()
• PLNLDAfit$setVisualization()
• PLNLDAfit$postTreatment()
• PLNLDAfit$plot_individual_map()
• PLNLDAfit$plot_correlation_map()
• PLNLDAfit$plot_LDA()
• PLNLDAfit$predict()
• PLNLDAfit$show()
• PLNLDAfit$clone()

Method new():

Usage:
PLNLDAfit$new(
  grouping,
  responses,
  covariates,
  offsets,
  weights,
  model,
  control
)

Method optimize():

Usage:
PLNLDAfit$optimize(X, covar, design_group, control)

Method setVisualization():

Usage:
PLNLDAfit$setVisualization(scale.unit = FALSE)

Method postTreatment():

Usage:
PLNLDAfit$postTreatment(responses, covariates, offsets)

Method plot_individual_map():

Usage:
PLNLDAfit$plot_individual_map(
  axes = 1:min(2, self$rank),
  main = "Individual Factor Map",
  plot = TRUE
)
Method plot_correlation_map():
Usage:
PLNLDAfit$plot_correlation_map(
    axes = 1:min(2, self$rank),
    main = "Variable Factor Map",
    cols = "default",
    plot = TRUE
)

Method plot_LDA():
Usage:
PLNLDAfit$plot_LDA(
    nb_axes = min(3, self$rank),
    var_cols = "default",
    plot = TRUE
)

Method predict():
Usage:
PLNLDAfit$predict(
    newdata,
    type = c("posterior", "response", "scores"),
    scale = c("log", "prob"),
    prior = NULL,
    control = list(),
    envir = parent.frame()
)

Method show():
Usage:
PLNLDAfit$show()

Method clone(): The objects of this class are cloneable with this method.
Usage:
PLNLDAfit$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

See Also
The function PLNLDA.

Examples
## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
PLNmodels

myPLNLDA <- PLNLDA(Abundance ~ 1, grouping = Group, data = trichoptera)
class(myPLNLDA)
print(myPLNLDA)

## End(Not run)

**Description**

The Poisson lognormal model and variants can be used for a variety of multivariate problems when count data are at play (including PCA or LDA for count data, network inference). This package implements efficient variational algorithms to fit such models accompanied with a set of functions for visualisation and diagnostic.

**Unpenalized Poisson lognormal model (aka PLN)**

See the main function `PLN` and the associated methods for manipulation.

Also try vignette("PLN_trichoptera", package="PLNmodels") for an overview.

**Rank Constraint Poisson lognormal for Poisson Principal Component Analysis (ala PLNPCA)**

See the main function `PLNPCA` and the associated methods for manipulation.

The Poisson PCA and the associated variational inference is fully explained in CHiquet et al (2018), see reference below.

Also try vignette("PLNPCA_trichoptera", package="PLNmodels") for an overview.

**Sparse Poisson lognormal model for sparse covariance inference for counts (aka PLNnetwork)**

See the main function `PLNnetwork` and the associated methods for manipulation.

Also try vignette("PLNnetwork_trichoptera", package="PLNmodels") for an overview.

**Poisson lognormal discriminant analysis (aka PLNLDA)**

See the main function `PLNLDA` and the associated methods for manipulation.

Also try vignette("PLNLDA_trichoptera", package="PLNmodels") for an overview.

**Author(s)**

Julien Chiquet <julien.chiquet@inra.fr>
Mahendra Mariadassou <mahendra.mariadassou@inra.fr>
Stéphane Robin <stephane.robin@inra.fr>
PLNnetwork

Poisson lognormal model towards sparse network inference

Description

Fit the sparse inverse covariance variant of the Poisson lognormal with a variational algorithm. Use the (g)lm syntax for model specification (covariates, offsets).

Usage

PLNnetwork(
  formula, data, subset, weights, penalties = NULL, control_init = list(), control_main = list()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>an object of class &quot;formula&quot;: a symbolic description of the model to be fitted.</td>
</tr>
<tr>
<td>data</td>
<td>an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.</td>
</tr>
<tr>
<td>subset</td>
<td>an optional vector specifying a subset of observations to be used in the fitting process.</td>
</tr>
<tr>
<td>weights</td>
<td>an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.</td>
</tr>
<tr>
<td>penalties</td>
<td>an optional vector of positive real number controlling the level of sparsity of the underlying network. If NULL (the default), will be set internally.</td>
</tr>
<tr>
<td>control_init</td>
<td>a list for controlling the optimization of the PLN model used at initialization. See details.</td>
</tr>
<tr>
<td>control_main</td>
<td>a list for controlling the main optimization process. See details.</td>
</tr>
</tbody>
</table>

Details

The list of parameters control_main controls the optimization of the main process, with the following entries:

- "ftol_rel": stop when an optimization step changes the objective function by less than ftol_rel multiplied by the absolute value of the parameter.
- "ftol_abs": stop when an optimization step changes the objective function by less than ftol_abs.
- "xtol_rel" stop when an optimization step changes every parameters by less than xtol_rel multiplied by the absolute value of the parameter.
- "xtol_abs" stop when an optimization step changes every parameters by less than xtol_abs.
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10000
- "algorithm" the optimization method used by NLOPT among LD type, i.e. "CCSAQ", "MMA", "LBFGS", "VAR1", "VAR2". See NLOPT documentation for further details. Default is "CCSAQ".
- "cores" integer for number of cores used. Default is 1.
- "trace" integer for verbosity. Useless when cores > 1
- "ftol_out" outer solver stops when an optimization step changes the objective function by less than xtol multiply by the absolute value of the parameter. Default is 1e-6
- "maxit_out" outer solver stops when the number of iteration exceeds out.maxit. Default is 50
- "penalize_diagonal" boolean: should the diagonal terms be penalized in the graphical-Lasso? Default is FALSE.
- "penalty_weights" p x p matrix of weights (default filled with 1) to adapt the amount of shrinkage to each pairs of node. Must be symmetric with positive values.

The list of parameters control_init controls the optimization process in the initialization and in the function PLN, plus two additional parameters:

- "nPEnalties" an integer that specified the number of values for the penalty grid when internally generated. Ignored when penalties is non NULL
- "min.ratio" the penalty grid ranges from the minimal value that produces a sparse to this value multiplied by min_ratio. Default is 0.1.

Value

an R6 object with class PLNnetworkfamily, which contains a collection of models with class PLNnetworkfit

See Also

The classes PLNnetworkfamily and PLNnetworkfit

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fits <- PLNnetwork(Abundance ~ 1, data = trichoptera)
An R6 Class to represent a collection of PLNnetworkfit

Description

The function `PLNnetwork` produces an instance of this class.

This class comes with a set of methods, some of them being useful for the user: See the documentation for `getBestModel`, `getModel` and `plot`.

Super class

`PLNmodels::PLNfamily` -> `PLNnetworkfamily`

Public fields

- responses: the matrix of responses common to every models
- covariates: the matrix of covariates common to every models
- offsets: the matrix of offsets common to every models
- weights: the vector of observation weights
- models: a list of `PLNnetworkfit` object, one per penalty.
- inception: a `PLNfit` object, obtained when no sparsifying penalty is applied.

Active bindings

- responses: the matrix of responses common to every models
- covariates: the matrix of covariates common to every models
- offsets: the matrix of offsets common to every models
- weights: the vector of observation weights
- penalties: the sparsity level of the network in the successively fitted models
- models: a list of `PLNnetworkfit` object, one per penalty.
- inception: a `PLNfit` object, obtained when no sparsifying penalty is applied.
- criteria: a data frame with the values of some criteria (variational lower bound J, BIC, ICL and R2) for the different models.

Methods

Public methods:

- `PLNnetworkfamily$new()`
- `PLNnetworkfamily$optimize()`
- `PLNnetworkfamily$stability_selection()`
- `PLNnetworkfamily$coefficient_path()`
- `PLNnetworkfamily$getBestModel()`
• `PLNnetworkfamily$plot()`
• `PLNnetworkfamily$plot_stars()`
• `PLNnetworkfamily$plot_objective()`
• `PLNnetworkfamily$show()`
• `PLNnetworkfamily$clone()`

**Method** `new()`:

*Usage:*

```r
PLNnetworkfamily$new(
  penalties,
  responses,
  covariates,
  offsets,
  weights,
  model,
  control
)
```

**Method** `optimize()`:

*Usage:*

```r
PLNnetworkfamily$optimize(control)
```

**Method** `stability_selection()`:

*Usage:*

```r
PLNnetworkfamily$stability_selection(
  subsamples = NULL,
  control = list(),
  mc.cores = 1
)
```

**Method** `coefficient_path()`:

*Usage:*

```r
PLNnetworkfamily$coefficient_path(precision = TRUE, corr = TRUE)
```

**Method** `getBestModel()`:

*Usage:*

```r
PLNnetworkfamily$getBestModel(
  crit = c("BIC", "loglik", "R_squared", "EBIC", "StARS"),
  stability = 0.9
)
```

**Method** `plot()`:

*Usage:*

```r
PLNnetworkfamily$plot(
  criteria = c("loglik", "pen_loglik", "BIC", "EBIC"),
  log.x = TRUE,
  annotate
)
```
Method plot_stars():
Usage:
PLNnetworkfamily$plot_stars(stability = 0.9, log.x = TRUE)

Method plot_objective():
Usage:
PLNnetworkfamily$plot_objective()

Method show():
Usage:
PLNnetworkfamily$show()

Method clone(): The objects of this class are cloneable with this method.
Usage:
PLNnetworkfamily$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

See Also
The function PLNnetwork, the class PLNnetworkfit

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fits <- PLNnetwork(Abundance ~ 1, data = trichoptera)
class(fits)

PLNnetworkfit An R6 Class to represent a PLNfit in a sparse inverse covariance framework

Description
The function PLNnetwork produces a collection of models which are instances of object with class PLNnetworkfit.

This class comes with a set of methods, some of them being useful for the user: See the documentation for plot.PLNnetworkfit + methods inherited from PLNfit.

Super class
PLNmodels::PLNfit -> PLNnetworkfit
Public fields

- **var_par**: a list with two matrices, M and S, which are the estimated parameters in the variational approximation
- **latent**: a matrix: values of the latent vector (Z in the model)
- **optim_par**: a list with parameters useful for monitoring the optimization
- **loglik**: variational lower bound of the loglikelihood
- **BIC**: variational lower bound of the BIC
- **ICL**: variational lower bound of the ICL
- **R_squared**: approximated goodness-of-fit criterion

Active bindings

- **penalty**: the level of sparsity in the current model
- **model_par**: a list with the matrices associated with the estimated parameters of the pPCA model: Theta (covariates), Sigma (latent covariance) and Theta (latent precision matrix). Note Omega and Sigma are inverse of each other.
- **var_par**: a list with two matrices, M and S, which are the estimated parameters in the variational approximation
- **latent**: a matrix: values of the latent vector (Z in the model)
- **optim_par**: a list with parameters useful for monitoring the optimization
- **loglik**: variational lower bound of the loglikelihood
- **pen_loglik**: variational lower bound of the l1-penalized loglikelihood
- **BIC**: variational lower bound of the BIC
- **EBIC**: variational lower bound of the EBIC
- **ICL**: variational lower bound of the ICL
- **R_squared**: approximated goodness-of-fit criterion
- **nb_param**: number of parameters in the current PLN model
- **density**: proportion of non-null edges in the network
- **criteria**: a vector with loglik, BIC, ICL, R_squared and number of parameters

Methods

Public methods:

- `PLNnetworkfit$new()`
- `PLNnetworkfit$update()`
- `PLNnetworkfit$optimize()`
- `PLNnetworkfit$postTreatment()`
- `PLNnetworkfit$latent_network()`
- `PLNnetworkfit$plot_network()`
- `PLNnetworkfit$show()`
- `PLNnetworkfit$clone()`
Method new():
Usage:
PLNnetworkfit$new(
  penalty,
  responses,
  covariates,
  offsets,
  weights,
  model,
  control
)

Method update():
Usage:
PLNnetworkfit$update(
  penalty = NA,
  Theta = NA,
  Sigma = NA,
  Omega = NA,
  M = NA,
  S = NA,
  Z = NA,
  A = NA,
  Ji = NA,
  R2 = NA,
  monitoring = NA
)

Method optimize():
Usage:
PLNnetworkfit$optimize(responses, covariates, offsets, weights, control)

Method postTreatment():
Usage:
PLNnetworkfit$postTreatment(responses, covariates, offsets, weights, nullModel)

Method latent_network():
Usage:
PLNnetworkfit$latent_network(type = c("partial_cor", "support", "precision"))

Method plot_network():
Usage:
PLNnetworkfit$plot_network(
  type = c("partial_cor", "support"),
  output = c("igraph", "corrplot"),
  edge.color = c("#F8766D", "#00BFC4"),
  remove.isolated = FALSE,
node.labels = NULL,
layout = layout_in_circle,
plot = TRUE
)

**Method** `show()`:

*Usage:*  
`PLNnetworkfit$show()`

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*  
`PLNnetworkfit$clone(deep = FALSE)`

*Arguments:*  
deep Whether to make a deep clone.

**See Also**

The function `PLNnetwork`, the class `PLNnetworkfamily`

**Examples**

```r
## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
nets <- PLNnetwork(Abundance ~ 1, data = trichoptera)
myPLNnet <- getBestModel(nets)
class(myPLNnet)
print(myPLNnet)
## End(Not run)
```

---

**PLNPCA**

**Poisson lognormal model towards Principal Component Analysis**

**Description**

Fit the PCA variants of the Poisson lognormal with a variational algorithm. Use the (g)lm syntax for model specification (covariates, offsets).

**Usage**

```r
PLNPCA(
  formula,
  data,
  subset,
  weights,
  ranks = 1:5,
  control_init = list(),
  control_main = list()
)
```
Arguments

- **formula**: an object of class "formula": a symbolic description of the model to be fitted.
- **data**: an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
- **subset**: an optional vector specifying a subset of observations to be used in the fitting process.
- **weights**: an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.
- **ranks**: a vector of integer containing the successive ranks (or number of axes to be considered)
- **control_init**: a list for controlling the optimization at initialization. See details of function PLN.
- **control_main**: a list for controlling the main optimization process. See details.

Details

The list of parameters control_main controls the optimization of the main process, with the following entries:

- "ftol_rel" stop when an optimization step changes the objective function by less than ftol multiplied by the absolute value of the parameter. Default is 1e-8
- "ftol_abs" stop when an optimization step changes the objective function by less than ftol multiplied by the absolute value of the parameter. Default is 0
- "xtol_rel" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is 1e-4
- "xtol_abs" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is 1e-4
- "lower_bound" the lower bound (box constraint) for the variational variance parameters. Default is 1e-4.
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10000
- "maxtime" stop when the optimization time (in seconds) exceeds maxtime. Default is -1 (no restriction)
- "algorithm" the optimization method used by NLOPT among LD type, i.e. "CCSAQ", "MMA", "LBFGS", "VAR1", "VAR2". See NLOPT documentation for further details. Default is "CCSAQ".
- "trace" integer for verbosity. Useless when cores > 1
- "cores" The number of core used to parallelize jobs over the ranks vector. Default is 1.

Value

an R6 object with class PLNPCAfamily, which contains a collection of models with class PLPCAfit
PLNPCAfamily

See Also

The classes PLNPCAfamily and PLNPCAfit

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPCA <- PLNPCA(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, ranks = 1:5)

---

PLNPCAfamily

An R6 Class to represent a collection of PLNPCAfit

Description

The function PLNPCA produces an instance of this class.
This class comes with a set of methods, some of them being useful for the user: See the documentation for getBestModel, getModel and plot.

Super class

PLNmodels::PLNfamily -> PLNPCAfamily

Public fields

responses  the matrix of responses common to every models
covariates  the matrix of covariates common to every models
offsets    the matrix of offsets common to every models
weights    the vector of observation weights
models   a list of PLNPCAfit object, one per rank.
inception  a PLNfit object, obtained when full rank is considered.
criteria  a data frame with the values of some criteria (variational lower bound J, BIC, ICL and R2) for the different models.

Active bindings

responses  the matrix of responses common to every models
covariates  the matrix of covariates common to every models
offsets    the matrix of offsets common to every models
weights    the vector of observation weights
ranks      the dimensions of the successively fitted models
models   a list of PLNPCAfit object, one per rank.
inception  a PLNfit object, obtained when full rank is considered.
criteria  a data frame with the values of some criteria (variational lower bound J, BIC, ICL and R2) for the different models.
Methods

Public methods:

• PLNPCAfamily$new()
• PLNPCAfamily$optimize()
• PLNPCAfamily$getBestModel()
• PLNPCAfamily$plot()
• PLNPCAfamily$show()
• PLNPCAfamily$clone()

Method new():

Usage:
PLNPCAfamily$new(
  ranks,
  responses,
  covariates,
  offsets,
  weights,
  model,
  control
)

Method optimize():

Usage:
PLNPCAfamily$optimize(control)

Method getBestModel():

Usage:
PLNPCAfamily$getBestModel(crit = c("BIC", "ICL", "R_squared"))

Method plot():

Usage:
PLNPCAfamily$plot(criteria = c("loglik", "BIC", "ICL"), annotate = TRUE)

Method show():

Usage:
PLNPCAfamily$show()

Method clone(): The objects of this class are cloneable with this method.

Usage:
PLNPCAfamily$clone(deep = FALSE)

Arguments:

deep  Whether to make a deep clone.

See Also

The function PLNPCA, the class PLNPCAf


Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPCAs <- PLNPCA(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, ranks = 1:5)
class(myPCAs)

Description

The function **PLNPCA** produces a collection of models which are instances of object with class **PLNPCAfit**.

This class comes with a set of methods, some of them being useful for the user: See the documentation for the methods inherited by **PLNfit** and the **plot.PLNPCAfit** methods for PCA visualization.

Super class

**PLNmodels::PLNfit** -> **PLNPCAfit**

Public fields

- var_par: a list with two matrices, M and S, which are the estimated parameters in the variational approximation
- latent: a matrix: values of the latent vector (Z in the model)
- optim_par: a list with parameters useful for monitoring the optimization
- loglik: variational lower bound of the loglikelihood
- BIC: variational lower bound of the BIC
- ICL: variational lower bound of the ICL
- R_squared: approximated goodness-of-fit criterion
- criteria: a vector with loglik, BIC, ICL, R_squared and number of parameters

Active bindings

- rank: the dimension of the current model
- model_par: a list with the matrices associated with the estimated parameters of the pPCA model: Theta (covariates), Sigma (latent covariance) and B (latent loadings)
- var_par: a list with two matrices, M and S, which are the estimated parameters in the variational approximation
- latent: a matrix: values of the latent vector (Z in the model)
- optim_par: a list with parameters useful for monitoring the optimization
- loglik: variational lower bound of the loglikelihood
- BIC: variational lower bound of the BIC
ICL  variational lower bound of the ICL
R_squared approximated goodness-of-fit criterion
criteria a vector with loglik, BIC, ICL, R_squared and number of parameters
nb_param number of parameters in the current PLN model
percent_var the percent of variance explained by each axis
corr_circle a matrix of correlations to plot the correlation circles
scores a matrix of scores to plot the individual factor maps (a.k.a. principal components)
rotation a matrix of rotation of the latent space

Methods

Public methods:

• PLNPCAfit$new()
• PLNPCAfit$update()
• PLNPCAfit$setVisualization()
• PLNPCAfit$optimize()
• PLNPCAfit$postTreatment()
• PLNPCAfit$latent_pos()
• PLNPCAfit$plot_individual_map()
• PLNPCAfit$plot_correlation_circle()
• PLNPCAfit$plot_PCA()
• PLNPCAfit$compute_fisher()
• PLNPCAfit$show()
• PLNPCAfit$clone()

Method new():

Usage:
PLNPCAfit$new(rank, responses, covariates, offsets, weights, model, control)

Method update():

Usage:
PLNPCAfit$update(
  Theta = NA,
  Sigma = NA,
  B = NA,
  M = NA,
  S = NA,
  Z = NA,
  A = NA,
  Ji = NA,
  R2 = NA,
  monitoring = NA
)
Method setVisualization:
  Usage:
  PLNPCAfit$setVisualization(scale.unit = FALSE)

Method optimize:
  Usage:
  PLNPCAfit$optimize(responses, covariates, offsets, weights, control)

Method postTreatment:
  Usage:
  PLNPCAfit$postTreatment(responses, covariates, offsets, weights, nullModel)

Method latent_pos:
  Usage:
  PLNPCAfit$latent_pos(covariates, offsets)

Method plot_individual_map:
  Usage:
  PLNPCAfit$plot_individual_map(
    axes = 1:min(2, self$rank),
    main = "Individual Factor Map",
    plot = TRUE,
    cols = "default"
  )

Method plot_correlation_circle:
  Usage:
  PLNPCAfit$plot_correlation_circle(
    axes = 1:min(2, self$rank),
    main = "Variable Factor Map",
    cols = "default",
    plot = TRUE
  )

Method plot_PCA:
  Usage:
  PLNPCAfit$plot_PCA(
    nb_axes = min(3, self$rank),
    ind_cols = "ind_cols",
    var_cols = "var_cols",
    plot = TRUE
  )

Method compute_fisher:
  Usage:
  PLNPCAfit$compute_fisher(type = c("wald", "louis"), X = NULL)
Method show():

Usage:
PLNPCAfit$show()

Method clone(): The objects of this class are cloneable with this method.

Usage:
PLNPCAfit$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

See Also
The function PLNPCA, the class PLNPCAfamily

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPCAs <- PLNPCA(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, ranks = 1:5)
myPCA <- getBestModel(myPCAs)
class(myPCA)
print(myPCA)

plot.PLNLDAfit LDA visualization (individual and/or variable factor map(s)) for a PLNPCAfit object

Description
LDA visualization (individual and/or variable factor map(s)) for a PLNPCAfit object

Usage

## S3 method for class 'PLNLDAfit'
plot(
  x,
  map = c("both", "individual", "variable"),
  nb_axes = min(3, x$rank),
  axes = seq.int(min(2, x$rank)),
  var_cols = "var_colors",
  plot = TRUE,
  main = NULL,
  ... )
plot.PLNnetworkfamily

Display various outputs (goodness-of-fit criteria, robustness, diagnostic) associated with a collection of PLNnetwork fits (a PLNnetworkfamily)

Description

Display various outputs (goodness-of-fit criteria, robustness, diagnostic) associated with a collection of PLNnetwork fits (a PLNnetworkfamily)

Arguments

x an R6 object with class PLNPCAfit

map the type of output for the PCA visualization: either "individual", "variable" or "both". Default is "both".

nb_axes scalar: the number of axes to be considered when map = "both". The default is min(3,rank).

axes numeric, the axes to use for the plot when map = "individual" or "variable". Default it c(1,min(rank))

var_cols a character or factor to define the color associated with the variables. By default, all variables receive the default color of the current palette.

plot logical. Should the plot be displayed or sent back as ggplot object

main character. A title for the single plot (individual or variable factor map). If NULL (the default), an hopefully appropriate title will be used.

... Not used (S3 compatibility).

Value

displays an individual and/or variable factor maps for the corresponding axes, and/or sends back a ggplot2 or gtable object

Examples

data(trichoptera)
trichoptera <- prepare.data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ 1, grouping = Group, data = trichoptera)
## Not run:
plot(myPLNLDA, map = "individual", nb_axes = 2)
## End(Not run)

plot.PLNnetworkfamily
Usage

```r
## S3 method for class 'PLNnetworkfamily'
plot(
  x,
  type = c("criteria", "stability", "diagnostic"),
  criteria = c("loglik", "pen_loglik", "BIC", "EBIC"),
  log.x = TRUE,
  stability = 0.9,
  annotate = TRUE,
  ...
)
```

Arguments

- `x`: an R6 object with class PLNfamily
- `type`: a character, either "criteria", "stability" or "diagnostic" for the type of plot.
- `criteria`: vector of characters. The criteria to plot in c("loglik", "BIC", "ICL", "R_squared", "EBIC", "pen_loglik"). Default is c("loglik", "pen_loglik", "BIC", "EBIC"). Only relevant when type equals "criteria".
- `log.x`: logical: should the x-axis be represented in log-scale? Default is TRUE.
- `stability`: scalar: the targeted level of stability in stability plot. Default is 0.9.
- `annotate`: logical: should the value of approximated R squared be added to the plot of criteria? Default is TRUE.
- `...`: additional parameters for S3 compatibility. Not used

Value

Produces a plot representing the evolution of the criteria of the different models considered, highlighting the best model in terms of BIC and EBIC (the greater, the better). These criteria have the form 'loglik - 1/2 * penalty' so that they are on the same scale as the model loglikelihood.

Examples

```r
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fits <- PLNnetwork(Abundance ~ 1, data = trichoptera)
## Not run:
plot(fits)
## End(Not run)```
plot.PLNnetworkfit

Extract and plot the network (apartial correlation, support or inverse covariance) from a PLNnetworkfit object

Description

Extract and plot the network (apartial correlation, support or inverse covariance) from a PLNnetworkfit object

Usage

## S3 method for class 'PLNnetworkfit'
plot(
  x,
  type = c("partial_cor", "support"),
  output = c("igraph", "corrplot"),
  edge.color = c("#F8766D", "#00BFC4"),
  remove.isolated = FALSE,
  node.labels = NULL,
  layout = layout_in_circle,
  plot = TRUE,
  ...
)

Arguments

x an R6 object with class PLNnetworkfit
type character. Value of the weight of the edges in the network, either "partial_cor" (partial correlation) or "support" (binary). Default is "partial_cor".
output the type of output used: either 'igraph' or 'corrplot'. Default is 'igraph'.
edge.color 2D numeric. Color for positive/negative edges. Default is c("#F8766D", "#00BFC4"). Only relevant for igraph output.
remove.isolated if TRUE, isolated node are remove before plotting. Only relevant for igraph output.
node.labels vector of character. The labels of the nodes. The Default will use the column names of the response matrix.
layout an optional igraph layout. Only relevant for igraph output.
plot logical. Should the final network be displayed or only sent back to the user. Default is TRUE.
...

Value

Send back an invisible object (igraph or Matrix, depending on the output chosen) and optionally displays a graph (via igraph or corrplot for large ones)
Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fits <- PLNnetwork(Abundance ~ 1, data = trichoptera)
myNet <- getBestModel(fits)
## Not run:
plot(myNet)
## End(Not run)

### plot.PLNPCAfamily

Display the criteria associated with a collection of PLNPCA fits (a PLNPCAfamily)

Description

Display the criteria associated with a collection of PLNPCA fits (a PLNPCAfamily)

Usage

## S3 method for class 'PLNPCAfamily'
plot(x, criteria = c("loglik", "BIC", "ICL"), annotate = TRUE, ...)

Arguments

x an R6 object with class PLNfamily
criteria vector of characters. The criteria to plot in c("loglik", "BIC", "ICL", "R_squared"). Default is c("loglik", "BIC", "ICL").
annotate logical: should the value of approximated R squared be added to the plot?
... additional parameters for S3 compatibility. Not used

Value

Produces a plot representing the evolution of the criteria of the different models considered, highlighting the best model in terms of BIC and ICL (the greater, the better). These criteria have the form ‘loglik - 1/2 * penalty’ so that they are on the same scale as the model loglikelihood.

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPCAs <- PLNPCA(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, ranks = 1:5)
## Not run:
plot(myPCAs)
## End(Not run)
plot.PLNPCAfit

PCA visualization (individual and/or variable factor map(s)) for a PLNPCAfit object

Description

PCA visualization (individual and/or variable factor map(s)) for a PLNPCAfit object

Usage

## S3 method for class 'PLNPCAfit'
plot(
  x,
  map = c("both", "individual", "variable"),
  nb_axes = min(3, x$rank),
  axes = seq.int(min(2, x$rank)),
  ind_cols = "ind_colors",
  var_cols = "var_colors",
  plot = TRUE,
  main = NULL,
  ...
)

Arguments

x an R6 object with class PLNPCAfit
map the type of output for the PCA visualization: either "individual", "variable" or "both". Default is "both".
nb_axes scalar: the number of axes to be considered when map = "both". The default is min(3,rank).
axes numeric, the axes to use for the plot when map = "individual" or "variable". Default it c(1,min(rank))
ind_cols a character, factor or numeric to define the color associated with the individuals. By default, all variables receive the default color of the current palette.
var_cols a character, factor or numeric to define the color associated with the variables. By default, all variables receive the default color of the current palette.
plot logical. Should the plot be displayed or sent back as ggplot object
main character. A title for the single plot (individual or variable factor map). If NULL (the default), an hopefully appropriate title will be used.
... Not used (S3 compatibility).

Value

displays an individual and/or variable factor maps for the corresponding axes, and/or sends back a ggplot2 or gtable object
Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPCAs <- PLNPCA(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, ranks = 1:5)
myPCA <- getBestModel(myPCAs)
## Not run:
plot(myPCA, map = "individual", nb_axes=2, ind_cols = trichoptera$Group)
plot(myPCA, map = "variable", nb_axes=2)
plot(myPCA, map = "both", nb_axes=2, ind_cols = trichoptera$Group)
## End(Not run)

predict.PLNfit  

Description

Predict counts of a new sample

Usage

## S3 method for class 'PLNfit'
predict(object, newdata, type = c("link", "response"), ...)

Arguments

object      an R6 object with class PLNfit
newdata     A data frame in which to look for variables and offsets with which to predict
type        The type of prediction required. The default is on the scale of the linear predictors (i.e. log average count)
...          additional parameters for S3 compatibility. Not used

Value

A matrix of predicted log-counts (if type = "link") or predicted counts (if type = "response").
predict.PLNLDAfit

**Predict group of new samples**

**Description**

Predict group of new samples

**Usage**

```r
## S3 method for class 'PLNLDAfit'
predict(
  object,
  newdata,
  type = c("posterior", "response", "scores"),
  scale = c("log", "prob"),
  prior = NULL,
  control = list(),
  ...
)
```

**Arguments**

- `object` an R6 object with class PLNLDAfit
- `newdata` A data frame in which to look for variables, offsets and counts with which to predict.
- `type` The type of prediction required. The default are posterior probabilities for each group (in either unnormalized log-scale or natural probabilities, see "scale" for details), "response" is the group with maximal posterior probability and "scores" is the average score along each separation axis in the latent space, with weights equal to the posterior probabilities.
- `scale` The scale used for the posterior probability. Either log-scale ("log", default) or natural probabilities summing up to 1 ("prob").
- `prior` User-specified prior group probabilities in the new data. If NULL (default), prior probabilities are computed from the learning set.
- `control` a list for controlling the optimization. See PLN for details.
- `...` additional parameters for S3 compatibility. Not used

**Value**

A matrix of posterior probabilities for each group (if type = "posterior"), a matrix of (average) scores in the latent space (if type = "scores") or a vector of predicted groups (if type = "response").
Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myLDA <- PLNLDA(Abundance ~ 0 + offset(log(Offset)),
              grouping = Group,
              data = trichoptera)
## Not run:
post_probs <- predict(myLDA, newdata = trichoptera, type = "posterior", scale = "prob")
head(round(post_probs, digits = 3))
predicted_group <- predict(myLDA, newdata = trichoptera, type = "response")
table(predicted_group, trichoptera$Group, dnn = c("predicted", "true"))
## End(Not run)

prepare_data  Prepare data for use in PLN models

Description

Prepare data in proper format for use in PLN model and its variants. The function (i) merges a count table and a covariate data frame in the most comprehensive way and (ii) computes offsets from the count table using one of several normalization schemes (TSS, CSS, RLE, GMPR, etc). The function fails with informative messages when the heuristics used for sample matching fail.

Usage

prepare_data(counts, covariates, offset = "TSS", ...)

Arguments

counts  Required. An abundance count table, preferably with dimensions names and species as columns.
covariates  Required. A covariates data frame, preferably with row names.
offset  Optional. Normalisation scheme used to compute scaling factors used as offset during PLN inference. Available schemes are "TSS" (Total Sum Scaling, default), "CSS" (Cumulative Sum Scaling, used in metagenomeSeq), "RLE" (Relative Log Expression, used in DESeq2), "GMPR" (Geometric Mean of Pairwise Ratio, introduced in Chen et al., 2018) or "none". Alternatively the user can supply its own vector or matrix of offsets (see note for specification of the user-supplied offsets).

...  Additional parameters passed on to compute_offset

Value

A data.frame suited for use in PLN and its variants with two specials components: an abundance count matrix (in component "Abundance") and an offset vector/matrix (in component "Offset", only if offset is not set to "none")
Note

User supplied offsets should be either vectors/column-matrices or have the same number of column as the original count matrix and either (i) dimension names or (ii) the same dimensions as the count matrix. Samples are trimmed in exactly the same way to remove empty samples.

References


See Also

compute_offset for details on the different normalisation schemes

Examples

data(trichoptera)
proper_data <- prepare_data(
  counts = trichoptera$Abundance,
  covariates = trichoptera$Covariate,
  offset = "TSS"
)
proper_data$Abundance
proper_data$Offset

rPLN

Description

Random generation for the PLN model with latent mean equal to mu, latent covariance matrix equal to Sigma and average depths (sum of counts in a sample) equal to depths

Usage

rPLN(
  n = 10,
  mu = rep(0, ncol(Sigma)),
  Sigma = diag(1, 5, 5),
  depths = rep(10000, n)
)
Arguments

- **n** the sample size
- **mu** vectors of means of the latent variable
- **Sigma** covariance matrix of the latent variable
- **depths** Numeric vector of target depths. The first is recycled if there are not 'n' values

Details

The default value for mu and Sigma assume equal abundances and no correlation between the different species.

Value

A n * p count matrix, with row-sums close to depths

Examples

```r
## 10 samples of 5 species with equal abundances, no covariance and target depths of 10,000
rPLN()
## 2 samples of 10 highly correlated species with target depths 1,000 and 100,000
## very different abundances
mu <- rep(c(1, -1), each = 5)
Sigma <- matrix(0.8, 10, 10); diag(Sigma) <- 1
rPLN(n=2, mu = mu, Sigma = Sigma, depths = c(1e3, 1e5))
```

---

**sigma.PLNfit**

*Extract variance-covariance of residuals 'Sigma'*

Description

Extract the variance-covariance matrix of the residuals, usually noted

\[ \Sigma \]

in PLN models. This captures the correlation between the species in the latent space.

Usage

```r
## S3 method for class 'PLNfit'
sigma(object, ...)
```

Arguments

- **object** an R6 object with class PLNfit
- **...** additional parameters for S3 compatibility. Not used
Value

A semi definite positive matrix of size p, assuming there are p species in the model.

See Also

coeff, standard_error and vcov for other ways to access
Σ
.

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1 + offset(log(Offset)), data = trichoptera)
sigma(myPLN) # # Sigma

stability_selection

Compute the stability path by stability selection

Description

This function computes the StARS stability criteria over a path of penalties. If a path has already been computed, the functions stops with a message unless force = TRUE has been specified.

Usage

stability_selection(
  Robject, 
  subsamples = NULL, 
  control = list(), 
  mc.cores = 1, 
  force = FALSE
)

Arguments

Robject an object with class PLNnetworkfamily, i.e. an output from PLNnetwork
subsamples a list of vectors describing the subsamples. The number of vectors (or list length) determines th number of subsamples used in the stability selection. Automatically set to 20 subsamples with size 10*sqrt(n) if n >= 144 and 0.8*n otherwise following Liu et al. (2010) recommandations.
control a list controling the main optimization process in each call to PLNnetwork. See PLNnetwork for details.
mc.cores the number of cores to used. Default is 1.
force force computation of the stability path, even if a previous one has been detected.
standard_error

Value
the list of subsamples. The estimated probabilities of selection of the edges are stored in the fields
stability_path of the initial Robject with class PLNnetworkfamily

Examples
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fits <- PLNnetwork(Abundance ~ 1, data = trichoptera)
## Not run:
  n <- nrow(trichoptera)
  subs <- replicate(10, sample.int(n, size = n/2), simplify = FALSE)
stability_selection(nets, subsamples = subs)
## End(Not run)

standard_error Component-wise standard errors of Theta

Description
Extracts univariate standard errors for the estimated coefficient of Theta. Standard errors are com-
puted from the (approximate) Fisher information matrix. See fisher for more details on the ap-
proximations.

Usage
standard_error(object, type)
## S3 method for class 'PLNfit'
standard_error(object, type = c("wald", "louis"))

Arguments
object an R6 object with class PLNfit
type Either ‘Wald’ (default) or ‘Louis’. Approximation scheme used to compute the
Fisher information matrix

Value
A p * d positive matrix (same size as Θ) with standard errors for the coefficients of Θ

Methods (by class)
• PLNfit: Component-wise standard errors of Theta in PLNfit

See Also
vcov for the complete Fisher information matrix
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1 + offset(log(Offset)), data = trichoptera)
standard_error(myPLN, "wald")

trichoptera

Trichoptera data set

Description
Data gathered between 1959 and 1960 during 49 insect trapping nights. For each trapping night, the abundance of 17 Trichoptera species is recorded as well as 6 meteorological variables which may influence the abundance of each species. Finally, the observations (that is to say, the trapping nights), have been classified into 12 groups corresponding to contiguous nights between summer 1959 and summer 1960.

Usage
trichoptera

Format
A list with 2 two data frames:
    Abundance a 49 x 17 matrix of abundancies/counts (49 trapping nights and 17 trichoptera species)
    Covariate a 49 x 7 data frame of covariates:
        Temperature Evening Temperature in Celsius
        Wind Wind in m/s
        Pressure Pressure in mm Hg
        Humidity relative to evening humidity in percent
        Cloudiness proportion of sky coverage at 9pm
        Precipitation Nighttime precipitation in mm
    Group a factor of 12 levels for the definition of the consecutive night groups

In order to prepare the data for using formula in multivariate analysis (multiple outputs and inputs), use prepare_data. We only kept a subset of the original meteorological covariates for illustration purposes.

Source
Data from P. Usseglio-Polatera.

References
See Also

prepare_data

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)

vcov.PLNfit
Calculate Variance-Covariance Matrix for a fitted PLN model object

Description

Returns the variance-covariance matrix of the main parameters of a fitted PLN model object. The main parameters of the model correspond to

\[ \Theta \]

as returned by coef. The function can also be used to return the variance-covariance matrix of the residuals. The latter matrix can also be accessed via sigma

Usage

## S3 method for class 'PLNfit'
vcov(object, type = c("main", "covariance"), ...)

Arguments

object an R6 object with class PLNfit
type type of parameter that should be extracted. Either "main" (default) for

\[ \Theta \]

or "covariance" for

\[ \Sigma \]

... additional parameters for S3 compatibility. Not used

Value

A matrix of variance/covariance extracted from the PLNfit model. If type="main" and \( \Theta \) is a matrix of size d * p, the result is a block-diagonal matrix with p (number of species) blocks of size d (number of covariates). if type="main", it is a symmetric matrix of size p.

See Also

sigma, coef, standard_error
Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1 + offset(log(Offset)), data = trichoptera)
vcov(myPLN) ## variance-covariance of Theta
vcov(myPLN, type = "covariance") ## Sigma
Index

*Topic datasets
  mollusk, 11
  trichoptera, 51

coeff. 15, 49, 52
coeff.PLNfit, 3
coeff.PLNLDAfit, 4
coefficient_path, 4
compute_offset, 5, 46, 47

eextract_probs, 6

fisher, 8, 50
fitted.PLNfit, 9

getBestModel, 14, 26, 33
getBestModel
  (getBestModel.PLNPCAfamily), 9
getBestModel.PLNPCAfamily, 9
getModel, 14, 26, 33
getModel.PLNPCAfamily, 10
getModel.PLNPCAfamily, 10

mollusk, 11

PLN, 3, 8, 9, 12, 15, 23, 25, 32, 45, 46, 52
PLNfamily, 14
PLNfit, 13–15, 15, 20, 26, 33, 35
PLNLDA, 4, 15, 18, 20, 22, 23
PLNLDAfit, 19, 20
PLNmodels, 23
PLNmodels::PLNfamily, 26, 33
PLNmodels::PLNfit, 20, 28, 35
PLNnetwork, 5, 7, 15, 23, 24, 26, 28, 31, 49
PLNnetworkfamily, 14, 25, 26, 31
PLNnetworkfit, 10, 11, 14, 25, 26, 28, 28
PLNPCA, 15, 23, 31, 33–35, 38
PLNPCAfamily, 14, 32, 33, 33, 38
PLNPCAfit, 10, 11, 33, 34, 35
plot, 14, 26, 33
plot.PLNLDAfit, 38
plot.PLNnetworkfamily, 39
plot.PLNnetworkfit, 28, 41
plot.PLNPCAfamily, 42
plot.PLNPCAfit, 20, 35, 43
PLPCAfit, 32
predict, 15
predict.PLNfit, 44
predict.PLNLDAfit, 45
predict.PLNPCAfit, 20
prepare_data, 12, 46, 51, 52

rPLN, 47

sigma, 3, 52
sigma.PLNfit, 15, 48
stability_selection, 49
standard_error, 3, 8, 15, 49, 50, 52

trichoptera, 51

vcov, 3, 15, 49, 50
vcov.PLNfit, 52