### Package ‘PLNmodels’

February 1, 2022

**Title**  Poisson Lognormal Models

**Version**  0.11.6

**Description**  The Poisson-lognormal model and variants (Chiquet, Mariadassou and Robin, 2020 [doi:10.1101/2020.10.07.329383]) can be used for a variety of multivariate problems when count data are at play, including principal component analysis for count data, discriminant analysis, model-based clustering and network inference. Implements variational algorithms to fit such models accompanied with a set of functions for visualization and diagnostic.

**URL**  https://pln-team.github.io/PLNmodels/

**BugReports**  https://github.com/pln-team/PLNmodels/issues

**License**  GPL (>= 3)

**Encoding**  UTF-8

**RoxygenNote**  7.1.2

**Depends**  R (>= 3.4)

**LazyData**  true

**biocViews**

**Imports**  methods, stats, MASS, future, future.apply, R6, glassoFast, Matrix, Rcpp, nloptr, igraph, grid, gridExtra, dplyr, tidyr, purrr, ggplot2, corplot, magrittr, rlang

**Suggests**  knitr, rmarkdown, testthat, covr, pkgdown, biomformat, phyloseq, spelling, factoextra, Wrench

**LinkingTo**  Rcpp, RcppArmadillo, nloptr

**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' 'PLNmixure.R' 'PLNmixture-S3methods.R' 'PLNmixturefamily-S3methods.R' 'PLNmixturefamily-class.R' 'PLNmixturefit-S3methods.R' 'PLNmixturefit-class.R'
'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'
'oaks.R' 'plot_utils.R' 'trichoptera.R' 'utils-pipe.R'
'utils.R' 'zzz.R'

**Language** en-US

**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],
François Gindraud [aut]

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

**Repository** CRAN

**Date/Publication** 2022-02-01 16:00:24 UTC

**R topics documented:**

- coef.PLNfit .................................................. 3
- coef.PLNLDAfit ............................................. 4
- coef.PLNmixturefit ........................................ 5
- coefficient_path ........................................... 6
- compute_offset ............................................ 6
- extract_probs .............................................. 8
- fisher ...................................................... 9
- fitted.PLNfit ............................................. 10
- fitted.PLNmixturefit .................................... 11
- getBestModel.PLNPCAfamily .............................. 11
- getModel.PLNPCAfamily .................................. 12
- mollusk ................................................... 13
- oaks ....................................................... 14
- PLN .......................................................... 16
- PLNfamily .................................................. 17
- PLNfit ...................................................... 20
- PLNLDA ..................................................... 25
- PLNLDAfit .................................................. 27
- PLNmixture ................................................ 31
- PLNmixturefamily ........................................ 33
- PLNmixturefit ............................................ 36
- PLNmodels .................................................. 39
- PLNnetwork ............................................... 40
- PLNnetworkfamily ........................................ 42
- PLNnetworkfit ............................................ 46
- PLNPCA ..................................................... 50
- PLNPCAfamily ............................................. 51
- PLNPCAfit .................................................. 54
### coef.PLNfit

Extract model coefficients

#### 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.PLNfit(), vcov.PLNfit(), standard_error.PLNfit()

Examples

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 PLNLDA().

**Description**

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

**Usage**

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

**Arguments**

- **object**: an R6 object with class PLNLDAfit
- **...**: additional parameters for S3 compatibility. Not used

**Value**

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

**Examples**

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

Extracts model coefficients from objects returned by PLN() and its variants

Usage

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

Arguments

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

Value

A matrix of coefficients extracted from the PLNfit model.

See Also

sigma.PLNmixturefit()

Examples

```r
data(trichoptera)
trichoptera <- prepare_data(trichoptera\$Abundance, trichoptera\$Covariate)
myPLN <- PLNmixture(Abundance ~ 1 + offset(log(Offset)),
data = trichoptera, control_main = list(iterates = 0)) %>% getBestModel()
coef(myPLN) ## Theta - empty here
coeff(myPLN, type = "mixture") ## pi
coeff(myPLN, type = "means") ## mu
coef(myPLN, type = "covariance") ## Sigma
```
coefficient_path  Extract the regularization path of a PLNnetwork fit

Description
Extract the regularization path of a PLNnetwork fit

Usage
coefficient_path(Robject, precision = TRUE, corr = TRUE)

Arguments
Robject  an object with class PLNnetworkfamily, i.e. an output from PLNnetwork()
precision  a logical, should the coefficients of the precision matrix Omega or the covariance matrix Sigma be sent back. Default is TRUE.
corr  a logical, should the correlation (partial in case precision = TRUE) be sent back. Default is TRUE.

Value
Sends 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", "Wrench", "none"),
  ...
)

compute_offset

Arguments

  counts          Required. An abundance count table, preferably with dimensions names and species as columns.
  offset         Optional. Normalization 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), Wrench (introduced in Kumar 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 specific methods (for now CSS and RLE)

Details

RLE has additional pseudocounts and type arguments to add pseudocounts to the observed counts (defaults to 0L) and to compute offsets using only positive counts (if type == "poscounts"). This mimics the behavior of DESeq2::DESeq() when using sfType == "poscounts". 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). Wrench has two additional parameters: groups to specify sample groups and type to either reproduce exactly the default Wrench::wrench() behavior (type = "wrench", default) or to use simpler heuristics (type = "simple"). 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 (unless type == "poscounts" has been specified) 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)
extract_probs <- trichoptera$Abundance
counts <- compute_offset(counts)
## Other normalization schemes
counts <- compute_offset(counts, offset = "RLE", pseudocounts = 1)
counts <- compute_offset(counts, offset = "Wrench", groups = trichoptera$Covariate$Group)
counts <- compute_offset(counts, offset = "GMPR")
## User supplied offsets
my_offset <- setNames(rep(1, nrow(counts)), rownames(counts))
counts <- compute_offset(counts, offset = my_offset)

### Description

**extract_probs**

Extracts edge selection frequency in networks reconstructed from bootstrap subsamples during the stars stability selection procedure, as either a matrix or a named vector. In the latter case, edge names follow igraph naming convention.

### Usage

```r
event | 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 PLNnetwork_family, i.e. an output from 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 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.
- **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 naming 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))]

## End(Not run)

fisher

Fisher information matrix for \( \Theta \)

Description

Extracts Fisher information matrix of \( \Theta \) from objects returned by PLN and its variants. Fisher matrix is computed using one of two approximation scheme: wald (default, conservative, gives large confidence interval) or 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

fisher(object, type)

## S3 method for class 'PLNfit'
fisher(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
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)

- **PLNfit**: Fisher information matrix for PLNfit

See Also

- `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.
fitted.PLNmixturefit  Extracts model fitted values from objects returned by \texttt{PLNmixture()} and its variants

**Description**

Extracts model fitted values from objects returned by \texttt{PLNmixture()} and its variants

**Usage**

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

**Arguments**

- **object**
  - an R6 object with class \texttt{PLNmixturefit}
- **...**
  - additional parameters for S3 compatibility. Not used

**Value**

A matrix of Fitted values extracted from the object \texttt{object}.

---

\texttt{getBestModel.PLNPCAfamily}  \textit{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"), ...)

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

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

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

- **Robject**: an object with class PLNPCAfamilly 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
- **PLNmixturefamily**: Model extraction for PLNmixturefamily
- **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 'PLNmixturefamily'
```
getModel(Robject, var, index = NULL)

## 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 PLNPCAf it or PLNnetworkfit.

Methods (by class)

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

Examples

## 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 covariates are known.

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)
```

---

**oaks** 

*Oaks amplicon data set*

---

**Description**

This data set gives the abundance of 114 taxa (66 bacterial OTU, 48 fungal OTUs) in 116 samples. For each sample, 11 additional covariates are known.

**Usage**

`oaks`
Format

A data frame with 13 variables:

- Abundance: A 114 taxa by 116 samples count matrix
- Offset: A 114 taxa by 116 samples offset matrix
- Sample: Unique sample id
- tree: Tree status with respect to the pathogen (susceptible, intermediate or resistant)
- branch: Unique branch id in each tree (4 branches were sampled in each tree, with 10 leaves per branch)
- leafNO: Unique leaf id in each tree (40 leaves were sampled in each tree)
- distTObase: Distance of the sampled leaf to the base of the branch
- distTOtrunk: Distance of the sampled leaf to the base of the tree trunk
- distTOground: Distance of the sampled leaf to the base of the ground
- pmInfection: Powdery mildew infection, proportion of the upper leaf area displaying mildew symptoms
- orientation: Orientation of the branch (South-West SW or North-East NE)
- readsTOTfun: Total number of ITS1 reads for that leaf
- readsTOTbac: Total number of 16S reads for that leaf

Source

Data from B. Jakuschkin and coauthors.

References


See Also

prepare_data()

Examples

data(oaks)
## Not run:
oaks_networks <- PLNnetwork(formula = Abundance ~ 1 + offset(log(Offset)), data = oaks)
## End(Not run)
Description

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

Usage

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 observation weights to be used in the fitting process.
- `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 0
- "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".

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)

Description

super class for PLNPCAfamily and PLNnetworkfamily.

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 0
• "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".

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
inception a PLNfit object, obtained when no sparsifying penalty is applied.
models a list of PLNfit object, one per penalty.

Active bindings

criteria a data frame with the values of some criteria (approximated log-likelihood, BIC, ICL, etc.) for the collection of models / fits BIC and ICL are defined so that they are on the same scale as the model log-likelihood, i.e. with the form, loglik - 0.5 penalty
convergence sends back a data frame with some convergence diagnostics associated with the optimization process (method, optimal value, etc)

Methods

Public methods:
- PLNfamily$new()
- PLNfamily$postTreatment()
- PLNfamily$getModel()
- PLNfamily$plot()
- PLNfamily$show()
- PLNfamily$print()
- PLNfamily$clone()

Method new(): Create a new PLNfamily object.

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

Arguments:
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
control a list for controlling the optimization. See details.
Returns: A new PLNfamily object

Method postTreatment(): Update fields after optimization

Usage:
PLNfamily$postTreatment()

Method getModel(): Extract a model from a collection of models

Usage:
PLNfamily$getModel(var, index = NULL)

Arguments:
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.

Returns: A PLNfit object

Method plot(): Lineplot of selected criteria for all models in the collection

Usage:
PLNfamily$plot(criteria, reverse)

Arguments:
criteria A valid model selection criteria for the collection of models. Includes loglik, BIC (all), ICL (PLNPCA) and pen_loglik, EBIC (PLNnetwork)
reverse A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e use the natural direction, on the same scale as the log-likelihood.

Returns: A ggplot2 object

Method show(): User friendly print method

Usage:
PLNfamily$show()

Method print(): User friendly print method

Usage:
PLNfamily$print()

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

Usage:
PLNfamily$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
getModel()
An R6 Class to represent a PLNfit in a standard, general framework

Description

The function PLN() fit a model which is an instance of a object with class PLNfit. Objects produced by the functions PLNNetwork(), PLNPCA(), PLNmixture() and PLNLDA() also enjoy the methods 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(), predict(), vcov() and standard_error().

Fields are accessed via active binding and cannot be changed by the user.

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 0
- "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".

Active bindings

n number of samples
q number of dimensions of the latent space
\( p \) number of species
\( d \) number of covariates

`model_par` a list with the matrices of parameters found in the model (\( \Theta \), \( \Sigma \), plus some others depending on the variant)

`fisher` Variational approximation of the Fisher Information matrix

`std_err` Variational approximation of the variance-covariance matrix of model parameters estimates.

`var_par` a list with two matrices, \( M \) and \( S^2 \), which are the estimated parameters in the variational approximation

`latent` a matrix: values of the latent vector (\( Z \) in the model)

`latent_pos` a matrix: values of the latent position vector (\( Z \)) without covariates effects or offset

`fitted` a matrix: fitted values of the observations (\( A \) in the model)

`nb_param` number of parameters in the current PLN model

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

`optim_par` a list with parameters useful for monitoring the optimization

`weights` observational weights

`loglik` (weighted) variational lower bound of the loglikelihood

`loglik_vec` element-wise variational lower bound of the loglikelihood

`BIC` variational lower bound of the BIC

`entropy` Entropy of the variational distribution

`ICL` variational lower bound of the ICL

`R_squared` approximated goodness-of-fit criterion

`criteria` a vector with loglik, BIC, ICL and number of parameters

### Methods

**Public methods:**

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

**Method `update()`**: Update a `PLNfit` object
Usage:
PLNfit$update(
  Theta = NA,
  Sigma = NA,
  M = NA,
  S2 = NA,
  Ji = NA,
  R2 = NA,
  Z = NA,
  A = NA,
  monitoring = NA
)

Arguments:
Theta matrix of regression matrix
Sigma variance-covariance matrix of the latent variables
M matrix of mean vectors for the variational approximation
S2 matrix of variance vectors for the variational approximation
Ji vector of variational lower bounds of the log-likelihoods (one value per sample)
R2 approximate R^2 goodness-of-fit criterion
Z matrix of latent vectors (includes covariates and offset effects)
A matrix of fitted values
monitoring a list with optimization monitoring quantities

Returns: Update the current PLNfit object

Method new(): Initialize a PLNfit model

Usage:
PLNfit$new(responses, covariates, offsets, weights, formula, xlevels, control)

Arguments:
responses the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily-class
covariates design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily-class
offsets offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily-class
weights an optional vector of observation weights to be used in the fitting process.
formula model formula used for fitting, extracted from the formula in the upper-level call
xlevels named listed of factor levels included in the models, extracted from the formula in the upper-level call and used for predictions.
control a list for controlling the optimization. See details.

Method optimize(): Call to the C++ optimizer and update of the relevant fields

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

Arguments:
responses the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily-class
covariates design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily-class
offsets offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily-class
weights an optional vector of observation weights to be used in the fitting process.
control a list for controlling the optimization. See details.

Method VEstep(): Result of one call to the VE step of the optimization procedure: optimal variational parameters (M, S) and corresponding log likelihood values for fixed model parameters (Sigma, Theta). Intended to position new data in the latent space.

Usage:
PLNfit$VEstep(covariates, offsets, responses, weights, control = list())

Arguments:
covariates design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily-class
offsets offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily-class
responses the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily-class
weights an optional vector of observation weights to be used in the fitting process.
control a list for controlling the optimization. See details.

Returns: A list with three components:
- the matrix M of variational means,
- the matrix S2 of variational variances
- the vector log.lik of (variational) log-likelihood of each new observation

Method set_R2(): Update R2 field after optimization

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

Arguments:
responses the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily-class
covariates design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily-class
offsets offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily-class
weights an optional vector of observation weights to be used in the fitting process.
nullModel null model used for approximate R2 computations. Defaults to a GLM model with same design matrix but not latent variable.

Method compute_fisher(): Safely compute the fisher information matrix (FIM)

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

Arguments:
- type: approximation scheme to compute the fisher information matrix. Either wald (default) or louis. type = "louis" results in smaller confidence intervals.
- X: design matrix used to compute the FIM

Returns: a sparse matrix with sensible dimension names

Method compute_standard_error(): Compute univariate standard error for coefficients of Theta from the FIM

Usage:
PLNfit$compute_standard_error()

Returns: a matrix of standard deviations.

Method postTreatment(): Update R2, fisher and std_err fields after optimization

Usage:
PLNfit$postTreatment(
  responses,
  covariates,
  offsets,
  weights = rep(1, nrow(responses)),
  type = c("wald", "louis", "none"),
  nullModel = NULL
)

Arguments:
- responses: the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily-class
- covariates: design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily-class
- offsets: offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily-class
- weights: an optional vector of observation weights to be used in the fitting process.
- type: approximation scheme to compute the fisher information matrix. Either wald (default) or louis. type = "louis" results in smaller confidence intervals.
- nullModel: null model used for approximate R2 computations. Defaults to a GLM model with same design matrix but not latent variable.

Method predict(): Predict position, scores or observations of new data.

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

Arguments:
- newdata: A data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
- type: Scale used for the prediction. Either link (default, predicted positions in the latent space) or response (predicted counts).
envir  Environment in which the prediction is evaluated

Returns: A matrix with predictions scores or counts.

Method show(): User friendly print method

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

Arguments:
model  First line of the print output

Method print(): User friendly print method

Usage:
PLNfit$print()

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

Usage:
PLNfit$clone(deep = FALSE)

Arguments:
deep  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**

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 observation weights to be used in the fitting process.
- `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-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 0
- "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".

Value

An R6 object with class `PLNLDAfit()`

See Also

The class `PLNLDAfit`
Data preparation:
```r
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ 1, grouping = Group, data = trichoptera)
```

**PLNLDAfit**

An R6 Class to represent a PLNfit in a LDA framework

**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()` method for LDA visualization and `predict()` method for prediction.

**Super class**

`PLNmodels::PLNfit` $>$ `PLNLDAfit`

**Active bindings**

- `rank` the dimension of the current model
- `nb_param` number of parameters in the current PLN 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)
- `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
- `group_means` a matrix of group mean vectors in the latent space.

**Methods**

**Public methods:**

- `PLNLDAfit$new()`
- `PLNLDAfit$optimize()`
- `PLNLDAfit$postTreatment()`
- `PLNLDAfit$setVisualization()`
- `PLNLDAfit$plot_individual_map()`
- `PLNLDAfit$plot_correlation_map()`
- `PLNLDAfit$plot_LDA()`
- `PLNLDAfit$predict()`
- `PLNLDAfit$show()`
**Method** `new()` : Initialize a `PLNLDAfit` object

*Usage:*

```r
PLNLDAfit$new(
  grouping,
  responses,
  covariates,
  offsets,
  weights,
  formula,
  xlevels,
  control
)
```

*Arguments:*

- `grouping` : a factor specifying the class of each observation used for discriminant analysis.
- `responses` : the matrix of responses (called `Y` in the model). Will usually be extracted from the corresponding field in `PLNfamily-class`.
- `covariates` : design matrix (called `X` in the model). Will usually be extracted from the corresponding field in `PLNfamily-class`.
- `offsets` : offset matrix (called `O` in the model). Will usually be extracted from the corresponding field in `PLNfamily-class`.
- `weights` : an optional vector of observation weights to be used in the fitting process.
- `formula` : model formula used for fitting, extracted from the formula in the upper-level call.
- `xlevels` : named listed of factor levels included in the models, extracted from the formula in the upper-level call and used for predictions.
- `control` : a list for controlling the optimization. See details.

**Method** `optimize()` : Compute group means and axis of the LDA (noted `B` in the model) in the latent space, update corresponding fields

*Usage:*

```r
PLNLDAfit$optimize(grouping, covariates, control)
```

*Arguments:*

- `grouping` : design matrix for the grouping variable.
- `covariates` : design matrix. Automatically built from the covariates and the formula from the call.
- `control` : a list for controlling the optimization. See details.

**Method** `postTreatment()` : Update `R2`, `fisher` and `std_err` fields and visualization after optimization

*Usage:*

```r
PLNLDAfit$postTreatment(grouping, responses, covariates, offsets)
```

*Arguments:*

- `grouping` : design matrix for the grouping variable.
- `responses` : the matrix of responses (called `Y` in the model). Will usually be extracted from the corresponding field in `PLNfamily-class`.
- `covariates` : design matrix for the model formula.
- `offsets` : offset matrix (called `O` in the model). Will usually be extracted from the corresponding field in `PLNfamily-class`.
- `weights` : an optional vector of observation weights to be used in the fitting process.
PLNLDAfit

Method setVisualization(): Compute LDA scores in the latent space and update corresponding fields.

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

Arguments:
- scale.unit Logical. Should LDA scores be rescaled to have unit variance

Method plot_individual_map(): Plot the factorial map of the LDA

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

Arguments:
- axes numeric, the axes to use for the plot when map = "individual" or "variable". Default it c(1, min(rank))
- main character. A title for the single plot (individual or variable factor map). If NULL (the default), an hopefully appropriate title will be used.
- plot logical. Should the plot be displayed or sent back as ggplot object

Returns: a ggplot graphic

Method plot_correlation_map(): Plot the correlation circle of a specified axis for a PLNLDAfit object

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

Arguments:
- axes numeric, the axes to use for the plot when map = "individual" or "variable". Default it c(1, min(rank))
- main character. A title for the single plot (individual or variable factor map). If NULL (the default), an hopefully appropriate title will be used.
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

Returns: a ggplot graphic

Method plot_LDA(): Plot a summary of the PLNLDAfit object

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

Arguments:

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

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

Returns: a grob object

Method predict(): Predict group of new samples

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

Arguments:

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.

envir  Environment in which the prediction is evaluated

Method show(): User friendly print method

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

Usage:

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

Arguments:

depth Whether to make a deep clone.

See Also

The function `PLNLDA`.

Examples

```r
## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ 1, grouping = Group, data = trichoptera)
class(myPLNLDA)
print(myPLNLDA)
## End(Not run)
```

---

**PLNmixutre**

**Poisson lognormal mixture model**

**Description**

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

**Usage**

```r
PLNmixutre(
    formula,
    data,
    subset,
    clusters = 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.
- `clusters`: a vector of integer containing the successive number of clusters (or components) to be considered
- `control_init`: a list for controlling the optimization at initialization. See details.
- `control_main`: a list for controlling the main optimization process. See details.

**Details**

The list of parameters `control_init` and `control_main` control the optimization of the initialization and the main process, with the following entries:

- "covariance" character setting the model for the covariance matrices of the mixture components. Either "full", "diagonal" or "spherical". Default is "spherical".
- "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 `0`.
- "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".
- "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`.
- "smoothing" The smoothing to apply. Either, 'forward', 'backward' or 'both'. Default is 'both'.
- "iterates" number of forward/backward iteration of smoothing. Default is `2`.
PLNmixturefamily

Value

an R6 object with class PLNmixturefamily, which contains a collection of models with class PLNmixturefit

See Also

The classes PLNmixturefamily and PLNmixturefit

Examples

```r
## Use future to dispatch the computations on 2 workers
## Not run:
future::plan("multisession", workers = 2)
## End(Not run)

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myMixtures <- PLNmixture(Abundance ~ 1 + offset(log(Offset)), data = trichoptera, 
control_main = list(smoothing = "forward", iterates = 1))

# Shut down parallel workers
## Not run:
future::plan("sequential")
## End(Not run)
```

---

PLNmixturefamily | An R6 Class to represent a collection of PLNmixturefit

Description

The function PLNmixture() 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 -> PLNmixturefamily

Active bindings

clusters vector indicating the number of clusters considered is the successively fitted models
Methods

Public methods:

- PLNmixturefamily$new()
- PLNmixturefamily$optimize()
- PLNmixturefamily$smooth()
- PLNmixturefamily$plot()
- PLNmixturefamily$plot_objective()
- PLNmixturefamily$getAsBestModel()
- PLNmixturefamily$show()
- PLNmixturefamily$print()
- PLNmixturefamily$clone()

Method new(): Initialize all models in the collection.

Usage:
PLNmixturefamily$new(
  clusters,
  responses,
  covariates,
  offsets,
  formula,
  xlevels,
  control
)

Arguments:
clusters the dimensions of the successively fitted models
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
formula model formula used for fitting, extracted from the formula in the upper-level call
xlevels named listed of factor levels included in the models, extracted from the formula in the
call levels
control a list for controlling the optimization. See details.

Method optimize(): Call to the optimizer on all models of the collection

Usage:
PLNmixturefamily$optimize(control)

Arguments:
control a list for controlling the optimization. See details.

Method smooth(): function to restart clustering to avoid local minima by smoothing the log-likelihood values as a function of the number of clusters
Usage:
PLNmixturefamily$smooth(control)

Arguments:
control a list to control the smoothing process

Method plot(): Lineplot of selected criteria for all models in the collection

Usage:
PLNmixturefamily$plot(criteria = c("loglik", "BIC", "ICL"), reverse = FALSE)

Arguments:
criteria A valid model selection criteria for the collection of models. Any of "loglik", "BIC" or "ICL" (all).
reverse A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - 0.5 penalty) or in the "reverse" direction (~2 loglik + penalty). Default to FALSE, i.e. use the natural direction, on the same scale as the log-likelihood.

Returns: A ggplot2 object

Method plot_objective(): Plot objective value of the optimization problem along the penalty path

Usage:
PLNmixturefamily$plot_objective()

Returns: a ggplot graph

Method getBestModel(): Extract best model in the collection

Usage:
PLNmixturefamily$getBestModel(crit = c("BIC", "ICL", "loglik"))

Arguments:
crit a character for the criterion used to performed the selection. Either "BIC", "ICL" or "loglik". Default is ICL

Returns: a PLNmixturefit object

Method show(): User friendly print method

Usage:
PLNmixturefamily$show()

Method print(): User friendly print method

Usage:
PLNmixturefamily$print()

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

Usage:
PLNmixturefamily$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

The function PLNmixture, the class PLNmixturefit
An R6 Class to represent a PLNfit in a mixture framework

Description

The function `PLNmixture` produces a collection of models which are instances of object with class `PLNmixturefit`. A `PLNmixturefit` (say, with k components) is itself a collection of k `PLNfit`.

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

Active bindings

- `n` number of samples
- `p` number of dimensions of the latent space
- `k` number of components
- `d` number of covariates
- `components` components of the mixture (PLNfits)
- `latent` a matrix: values of the latent vector (Z in the model)
- `latent_pos` a matrix: values of the latent position vector (Z) without covariates effects or offset
- `posteriorProb` matrix of posterior probability for cluster belonging
- `memberships` vector for cluster index
- `mixtureParam` vector of cluster proportions
- `optim_par` a list with parameters useful for monitoring the optimization
- `nb_param` number of parameters in the current PLN model
- `entropy_clustering` Entropy of the variational distribution of the cluster (multinomial)
- `entropy_latent` Entropy of the variational distribution of the latent vector (Gaussian)
- `entropy` Full entropy of the variational distribution (latent vector + clustering)
- `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 (include entropy of both the clustering and latent distributions)
- `R_squared` approximated goodness-of-fit criterion
- `criteria` a vector with loglik, BIC, ICL, and number of parameters
- `model_par` a list with the matrices of parameters found in the model (Theta, Sigma, Mu and Pi)
- `vcov_model` character: the model used for the covariance (either "spherical", "diagonal" or "full")
- `fitted` a matrix: fitted values of the observations (A in the model)
- `group_means` a matrix of group mean vectors in the latent space.
Methods

**Public methods:**

- `PLNmixturefit$new()`
- `PLNmixturefit$optimize()`
- `PLNmixturefit$predict()`
- `PLNmixturefit$plot_clustering_data()`
- `PLNmixturefit$plot_clustering_pca()`
- `PLNmixturefit$postTreatment()`
- `PLNmixturefit$show()`
- `PLNmixturefit$print()`
- `PLNmixturefit$clone()`

**Method** `new()`: Optimize a the

Initialize a `PLNmixturefit` model

*Usage:*

```r
PLNmixturefit$new(
  responses,
  covariates,
  offsets,
  posteriorProb,
  formula,
  xlevels,
  control
)
```

*Arguments:*

- **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
- **posteriorProb**: matrix of posterior probability for cluster belonging
- **formula**: model formula used for fitting, extracted from the formula in the upper-level call
- **xlevels**: named listed of factor levels included in the models, extracted from the formula in the upper-level call
- **control**: a list for controlling the optimization. See details.

**Method** `optimize()`: Optimize a `PLNmixturefit` model

*Usage:*

```r
PLNmixturefit$optimize(responses, covariates, offsets, control)
```

*Arguments:*

- **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
- **control**: a list for controlling the optimization. See details.
control a list for controlling the optimization. See details.

Method predict(): Predict group of new samples

Usage:
```r
PLNmixturefit$predict(
  newdata,
  type = c("posterior", "response", "position"),
  prior = matrix(rep(1/self$k, self$k), nrow(newdata), self$k, byrow = TRUE),
  control = list(),
  envir = parent.frame()
)
```

Arguments:
- **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 posterior are posterior probabilities for each group, response is the group with maximal posterior probability and latent is the averaged latent coordinate (without offset and nor covariate effects), with weights equal to the posterior probabilities.
- **prior**: User-specified prior group probabilities in the new data. The default uses a uniform prior.
- **control**: a list for controlling the optimization. See `PLN()` for details.
- **envir**: Environment in which the prediction is evaluated

Method plot_clustering_data(): Plot the matrix of expected mean counts (without offsets, without covariate effects) reordered according the inferred clustering

Usage:
```r
PLNmixturefit$plot_clustering_data(
  main = "Expected counts reorder by clustering",
  plot = TRUE,
  log_scale = TRUE
)
```

Arguments:
- **main**: character. A title for the plot. An hopefully appropriate title will be used by default.
- **plot**: logical. Should the plot be displayed or sent back as `ggplot` object
- **log_scale**: logical. Should the color scale values be log-transform before plotting? Default is TRUE.

Returns: a `ggplot` graphic

Method plot_clustering_pca(): Plot the individual map of a PCA performed on the latent coordinates, where individuals are colored according to the memberships

Usage:
```r
PLNmixturefit$plot_clustering_pca(
  main = "Clustering labels in Individual Factor Map",
  plot = TRUE
)
```

Arguments:
main character. A title for the plot. An hopefully appropriate title will be used by default.
plot logical. Should the plot be displayed or sent back as ggplot object

Returns: a ggplot graphic

Method postTreatment(): Update fields after optimization
Usage:
PLNmixturefit$postTreatment(responses, covariates, offsets, weights, nullModel)
Arguments:
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 an optional vector of observation weights to be used in the fitting process.
nullModel null model used for approximate R2 computations. Defaults to a GLM model with
same design matrix but not latent variable.

Method show(): User friendly print method
Usage:
PLNmixturefit$show()

Method print(): User friendly print method
Usage:
PLNmixturefit$print()

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

See Also
The function PLNmixture, the class PLNmixturefamily

PLNmodels

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 visualization and diagnostic.
Multivariate 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 Constrained Poisson lognormal for Poisson Principal Component Analysis (aka 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 el 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.

Mixtures of Poisson lognormal models for model-based clustering (aka PLNmixture)

See the main function PLNmixture() and the associated methods for manipulation.
Also try vignette("PLNmixture_trichoptera", package="PLNmodels") for an overview.

Author(s)

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

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

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 observation weights to be used in the fitting process.
penalties an optional vector of positive real number controlling the level of sparsity of the underlying network. if NULL (the default), will be set internally. See control_init and control_main options for additional tuning of the penalty.
control_init a list for controlling the optimization of the PLN model used at initialization, and how the vector of penalties is generated. See details.
control_main a list for controlling the main optimization process. Can be used to specify adaptive penalty weights. 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-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_rel 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_abs. Default is 0
- "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".
PLNnetworkfamily

- "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 TRUE
- "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)

PLNnetworkfamily 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
Active bindings

penalties  the sparsity level of the network in the successively fitted models
stability_path  the stability path of each edge as returned by the stars procedure
stability  mean edge stability along the penalty path
criteria  a data frame with the values of some criteria (approximated log-likelihood, (E)BIC, ICL and R2, stability) for the collection of models / fits BIC, ICL and EBIC are defined so that they are on the same scale as the model log-likelihood, i.e. with the form, loglik - 0.5 penalty

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(): Initialize all models in the collection

Usage:

PLNnetworkfamily$new(
  penalties,
  responses,
  covariates,
  offsets,
  weights,
  formula,
  xlevels,
  control
)

Arguments:

penalties  a vector of positive real number controlling the level of sparsity of the underlying network.
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
formula  model formula used for fitting, extracted from the formula in the upper-level call
xlevels  named listed of factor levels included in the models, extracted from the formula in the upper-level call and used for predictions.
control a list for controlling the optimization. See details.

**Returns:** Update current **PLNnetworkfit** with smart starting values

**Method** optimize(): Call to the C++ optimizer on all models of the collection

**Usage:**

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

**Arguments:**

`control` a list for controlling the optimization. See details.

**Method** stability_selection(): Compute the stability path by stability selection

**Usage:**

```r
PLNnetworkfamily$stability_selection(subsamples = NULL, control = list())
```

**Arguments:**

- `subsamples` a list of vectors describing the subsamples. The number of vectors (or list length) determines the 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) recommendations.
- `control` a list controlling the main optimization process in each call to **PLNnetwork**. See **PLNnetwork()** for details.

**Method** coefficient_path(): Extract the regularization path of a **PLNnetworkfamily**

**Usage:**

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

**Arguments:**

- `precision` Logical. Should the regularization path be extracted from the precision matrix $\Omega$ (TRUE, default) or from the variance matrix $\Sigma$ (FALSE)
- `corr` Logical. Should the matrix be transformed to (partial) correlation matrix before extraction? Defaults to TRUE

**Method** getBestModel(): Extract the best network in the family according to some criteria

**Usage:**

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

**Arguments:**

- `crit` character. Criterion used to perform the selection. Is "StARS" is chosen but $\text{Stability}$ field is empty, will compute stability path.
- `stability` Only used for "StARS" criterion. A scalar indicating the target stability ($= 1 - 2 \beta$) at which the network is selected. Default is 0.9.

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

**Usage:**
PLNnetworkfamily$plot(
  criteria = c("loglik", "pen_loglik", "BIC", "EBIC"),
  reverse = FALSE,
  log.x = TRUE
)

Arguments:
criteria vector of characters. The criteria to plot in c("loglik","pen_loglik","BIC","EBIC").
  Defaults to all of them.
reverse A logical indicating whether to plot the value of the criteria in the "natural" direction
  (loglik - 0.5 penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE,
  i.e use the natural direction, on the same scale as the log-likelihood..
log.x logical: should the x-axis be represented in log-scale? Default is TRUE.
Returns: a ggplot graph

Method plot_stars(): Plot stability path
  Usage:
  PLNnetworkfamily$plot_stars(stability = 0.9, log.x = TRUE)
  Arguments:
  stability scalar: the targeted level of stability in stability plot. Default is 0.9.
  log.x logical: should the x-axis be represented in log-scale? Default is TRUE.
  Returns: a ggplot graph

Method plot_objective(): Plot objective value of the optimization problem along the penalty path
  Usage:
  PLNnetworkfamily$plot_objective()
  Returns: a ggplot graph

Method show(): User friendly print method
  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()` and methods inherited from `PLNfit`.

Super class

`PLNmodels::PLNfit` -> `PLNnetworkfit`

Active bindings

- `penalty` the level of sparsity in the current model
- `n_edges` number of edges if the network (non null coefficient of the sparse precision matrix)
- `nb_param` number of parameters in the current PLN model
- `pen_loglik` variational lower bound of the l1-penalized loglikelihood
- `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.
- `EBIC` variational lower bound of the EBIC
- `density` proportion of non-null edges in the network
- `criteria` a vector with loglik, penalized loglik, BIC, EBIC, ICL, R_squared, number of parameters, number of edges, and graph density

Methods

**Public methods:**

- `PLNnetworkfit$new()`
- `PLNnetworkfit$update()`
- `PLNnetworkfit$optimize()`
- `PLNnetworkfit$postTreatment()`
- `PLNnetworkfit$latent_network()`
- `PLNnetworkfit$plot_network()`
- `PLNnetworkfit$show()`
- `PLNnetworkfit$clone()`

**Method new():** Initialize a `PLNnetworkfit` object

*Usage:*
PLNnetworkfit$new(
  penalty,
  responses,
  covariates,
  offsets,
  weights,
  formula,
  xlevels,
  control
)

**Arguments:**
- **penalty**: a positive real number controlling the level of sparsity of the underlying network.
- **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**: an optional vector of observation weights to be used in the fitting process.
- **formula**: model formula used for fitting, extracted from the formula in the upper-level call
- **xlevels**: named listed of factor levels included in the models, extracted from the formula in PLNnetwork() call
- **control**: a list for controlling the optimization of the PLN model used at initialization. See PLNnetwork() for details.

**Method** update(): Update fields of a PLNnetworkfit object

**Usage:**
PLNnetworkfit$update(
  penalty = NA,
  Theta = NA,
  Sigma = NA,
  Omega = NA,
  M = NA,
  S2 = NA,
  Z = NA,
  A = NA,
  Ji = NA,
  R2 = NA,
  monitoring = NA
)

**Arguments:**
- **penalty**: a positive real number controlling the level of sparsity of the underlying network.
- **Theta**: matrix of regression matrix
- **Sigma**: variance-covariance matrix of the latent variables
- **Omega**: precision matrix of the latent variables. Inverse of Sigma.
- **M**: matrix of mean vectors for the variational approximation
- **S2**: matrix of variance vectors for the variational approximation
- **Z**: matrix of latent vectors (includes covariates and offset effects)
A matrix of fitted values
Ji vector of variational lower bounds of the log-likelihoods (one value per sample)
R2 approximate R^2 goodness-of-fit criterion
monitoring a list with optimization monitoring quantities

Method `optimize()`: Call to the C++ optimizer and update of the relevant fields

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

Arguments:
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 an optional vector of observation weights to be used in the fitting process.
control a list for controlling the optimization of the PLN model used at initialization. See `PLNnetwork()` for details.

Method `postTreatment()`: Compute PCA scores in the latent space and update corresponding fields.

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

Arguments:
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 an optional vector of observation weights to be used in the fitting process.
nullModel null model used for approximate R2 computations. Defaults to a GLM model with same design matrix but not latent variable.

Method `latent_network()`: Extract interaction network in the latent space

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

Arguments:
type edge value in the network. Can be "support" (binary edges), "precision" (coefficient of the precision matrix) or "partial_cor" (partial correlation between species)

Returns: a square matrix of size PLNnetworkfit$n

Method `plot_network()`: plot the latent 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
)

Arguments:
type edge value in the network. Either "precision" (coefficient of the precision matrix) or "partial_cor" (partial correlation between species).
output Output type. Either igraph (for the network) or corrplot (for the adjacency matrix)
edge.color Length 2 color vector. 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.

Method show(): User friendly print method

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

## 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)
Description

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

Usage

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 observation weights to be used in the fitting process.
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 0
• "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.

Value

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

See Also

The classes PLNPCAfamily and PLNPCAfit

Examples

```r
#' ## Use future to dispatch the computations on 2 workers
#' Not run:
future::plan("multisession", workers = 2)

## End(Not run)

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

# Shut down parallel workers
#' Not run:
future::plan("sequential")

## End(Not run)
```

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

\texttt{PLNmodels::PLNfamily -> PLNPCAfamily}

Active bindings

\begin{itemize}
  \item \texttt{ranks} the dimensions of the successively fitted models
\end{itemize}

Methods

\textbf{Public methods:}

\begin{itemize}
  \item \texttt{PLNPCAfamily$new()}
  \item \texttt{PLNPCAfamily$optimize()}
  \item \texttt{PLNPCAfamily$getModel()}
  \item \texttt{PLNPCAfamily$getBestModel()}
  \item \texttt{PLNPCAfamily$plot()}
  \item \texttt{PLNPCAfamily$show()}
  \item \texttt{PLNPCAfamily$clone()}
\end{itemize}

\textbf{Method \texttt{new()}}: Initialize all models in the collection.

\textit{Usage:}

\begin{verbatim}
PLNPCAfamily$new(
  ranks, 
  responses, 
  covariates, 
  offsets, 
  weights, 
  formula, 
  xlevels, 
  control 
)
\end{verbatim}

\textit{Arguments:}

\begin{itemize}
  \item \texttt{ranks} the dimensions of the successively fitted models
  \item \texttt{responses} the matrix of responses common to every models
  \item \texttt{covariates} the matrix of covariates common to every models
  \item \texttt{offsets} the matrix of offsets common to every models
  \item \texttt{weights} the vector of observation weights
  \item \texttt{formula} model formula used for fitting, extracted from the formula in the upper-level call
  \item \texttt{xlevels} named listed of factor levels included in the models, extracted from the formula in the upper-level call and used for predictions.
  \item \texttt{control} a list for controlling the optimization. See details.
\end{itemize}

\textbf{Method \texttt{optimize()}}: Call to the C++ optimizer on all models of the collection

\textit{Usage:}

\begin{verbatim}
PLNPCAfamily$optimize(control)
\end{verbatim}

\textit{Arguments:}
control a list for controlling the optimization. See details.

Method getModel(): Extract model from collection and add "PCA" class for compatibility with 
\texttt{factoextra::fviz()}

Usage:
\texttt{PLNPCAfamily$getModel(var, index = NULL)}

Arguments:
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.

Returns: a \texttt{PLNPCAfit} object

Method getBestModel(): Extract best model in the collection

Usage:
\texttt{PLNPCAfamily$getBestModel(crit = c("ICL", "BIC"))}

Arguments:
crit a character for the criterion used to performed the selection. Either "ICL", "BIC". Default
is ICL

Returns: a \texttt{PLNPCAfit} object

Method plot(): Lineplot of selected criteria for all models in the collection

Usage:
\texttt{PLNPCAfamily$plot(criteria = c("loglik", "BIC", "ICL"), reverse = FALSE)}

Arguments:
criteria A valid model selection criteria for the collection of models. Any of "loglik", "BIC"
or "ICL" (all).
reverse A logical indicating whether to plot the value of the criteria in the "natural" direction
(loglik - penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e
use the natural direction, on the same scale as the log-likelihood.

Returns: A \texttt{ggplot2} object

Method show(): User friendly print method

Usage:
\texttt{PLNPCAfamily$show()}

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

Usage:
\texttt{PLNPCAfamily$clone(deep = FALSE)}

Arguments:
deep Whether to make a deep clone.

See Also
The function \texttt{PLNPCA()}, the class \texttt{PLNPCAfith()}
Examples

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

PLNPCAfit

An R6 Class to represent a PLNfit in a PCA framework

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() methods for PCA visualization.

Super class

PLNmodels::PLNfit -> PLNPCAfit

Active bindings

rank  the dimension of the current model
nb_param number of parameters in the current PLN model
entropy entropy of the variational distribution
latent_pos a matrix: values of the latent position vector (Z) without covariates effects or offset
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)
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
eig  description of the eigenvalues, similar to percent_var but for use with external methods
var  a list of data frames with PCA results for the variables: coord (coordinates of the variables), cor (correlation between variables and dimensions), cos2 (Cosine of the variables) and contrib (contributions of the variable to the axes)
ind  a list of data frames with PCA results for the individuals: coord (coordinates of the individuals), cos2 (Cosine of the individuals), contrib (contributions of individuals to an axis inertia) and dist (distance of individuals to the origin).
call  Hacky binding for compatibility with factoextra functions
Methods

Public methods:

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

Method new(): Initialize a PLNPCAfit object

Usage:

PLNPCAfit$new(
  rank,
  responses,
  covariates,
  offsets,
  weights,
  formula,
  xlevels,
  control
)

Arguments:

rank rank of the PCA (or equivalently, dimension of the latent space)
responses the matrix of responses (called Y in the model). Will usually be extracted from the
corresponding field in PLNfamily
covariates design matrix (called X in the model). Will usually be extracted from the corre-
sponding field in PLNfamily
offsets offset matrix (called O in the model). Will usually be extracted from the correspond-
ing field in PLNfamily
weights an optional vector of observation weights to be used in the fitting process.
formula model formula used for fitting, extracted from the formula in the upper-level call
xlevels named listed of factor levels included in the models, extracted from the formula in the
upper-level call and used for predictions.
control a list for controlling the optimization. See details.

Method update(): Update a PLNPCAfit object

Usage:
PLNPCAfit$update(
  Theta = NA,
  Sigma = NA,
  B = NA,
  M = NA,
  S2 = NA,
  Z = NA,
  A = NA,
  Ji = NA,
  R2 = NA,
  monitoring = NA
)

Arguments:
Theta  matrix of regression matrix
Sigma  variance-covariance matrix of the latent variables
B      matrix of PCA loadings (in the latent space)
M      matrix of mean vectors for the variational approximation
S2     matrix of variance vectors for the variational approximation
Z      matrix of latent vectors (includes covariates and offset effects)
A      matrix of fitted values
Ji     vector of variational lower bounds of the log-likelihoods (one value per sample)
R2     approximate R^2 goodness-of-fit criterion
monitoring  a list with optimization monitoring quantities

Returns:  Update the current PLNPCAfit object

Method optimize(): Call to the C++ optimizer and update of the relevant fields

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

Arguments:
responses  the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily
covariates design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily
offsets    offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily
weights    an optional vector of observation weights to be used in the fitting process.
control     a list for controlling the optimization. See details.

Method VEstep(): Result of one call to the VE step of the optimization procedure: optimal variational parameters (M, S) and corresponding log likelihood values for fixed model parameters (B, Theta). Intended to position new data in the latent space for further use with PCA.

Usage:
PLNPCAfit$VEstep(
  covariates,
offsets,
responses,
weights = rep(1, nrow(responses)),
control = list()
)

Arguments:
covariates design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily
offsets offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily
responses the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily
weights an optional vector of observation weights to be used in the fitting process.
control a list for controlling the optimization. See details.

Returns: A list with three components:
• the matrix $M$ of variational means,
• the matrix $S^2$ of variational variances
• the vector $\log.\text{lik}$ of (variational) log-likelihood of each new observation

Method project(): Project new samples into the PCA space using one VE step

Usage:
PLNPCAfit$project(newdata, control = list(), envir = parent.frame())

Arguments:
newdata A data frame in which to look for variables, offsets and counts with which to predict.
control a list for controlling the optimization. See PLN() for details.
envir Environment in which the projection is evaluated

Returns:
• the named matrix of scores for the newdata, expressed in the same coordinate system as self$scores

Method setVisualization(): Compute PCA scores in the latent space and update corresponding fields.

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

Arguments:
scale.unit Logical. Should PCA scores be rescaled to have unit variance

Method postTreatment(): Update R2, fisher, std_err fields and set up visualization after optimization

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

Arguments:
responses the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in PLNfamily

covariates design matrix (called X in the model). Will usually be extracted from the corresponding field in PLNfamily

offsets offset matrix (called O in the model). Will usually be extracted from the corresponding field in PLNfamily

weights an optional vector of observation weights to be used in the fitting process.

nullModel null model used for approximate R2 computations. Defaults to a GLM model with same design matrix but not latent variable.

Method compute_fisher(): Safely compute the fisher information matrix (FIM)

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

Arguments:
type approximation scheme to compute the fisher information matrix. Either wald (default) or louis. type = "louis" results in smaller confidence intervals.

X design matrix used to compute the FIM

Returns: a sparse matrix with sensible dimension names

Method plot_individual_map(): Plot the factorial map of the PCA

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

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

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

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

cols a character, factor or numeric to define the color associated with the individuals. By default, all individuals receive the default color of the current palette.

Returns: a ggplot graphic

Method plot_correlation_circle(): Plot the correlation circle of a specified axis for a PLNLDAdFit object

Usage:
PLNPCAfit$plot_correlation_circle(axes = 1:min(2, self$rank),
    main = "Variable Factor Map",
    cols = "default",
    plot = TRUE
)
Arguments:
axes numeric, the axes to use for the plot when map = "individual" or "variable". Default it c(1,min(rank))
main character. A title for the single plot (individual or variable factor map). If NULL (the default), an hopefully appropriate title will be used.
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

Returns: a ggplot graphic

Method plot_PCA(): Plot a summary of the PLNPCAfit object

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

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

Returns: a grob object

Method show(): User friendly print method

Usage:
PLNPCAfit$show()

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

Usage:
PLNPCAfit$clone(deep = FALSE)

Arguments:
  deep Whether to make a deep clone.

See Also
The function PLNPCA, the class PLNPCAfamily
Examples

```r
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.PLNfamily

Display the criteria associated with a collection of PLN fits (a PLN-family)

Description

Display the criteria associated with a collection of PLN fits (a PLNfamily)

Usage

```r
## S3 method for class 'PLNfamily'
plot(x, criteria = c("loglik", "BIC", "ICL"), reverse = FALSE, ...)
```

Arguments

- `x`: an R6 object with class `PLNfamily`
- `criteria`: vector of characters. The criteria to plot in c("loglik", "BIC", "ICL"). Default is c("loglik", "BIC", "ICL").
- `reverse`: A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - 0.5 penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e use the natural direction, on the same scale as the log-likelihood.
- `...`: additional parameters for S3 compatibility. Not used

Details

The BIC and ICL criteria have the form 'loglik - 1/2 * penalty' so that they are on the same scale as the model log-likelihood. You can change this direction and use the alternate form '-2*loglik + penalty', as some authors do, by setting `reverse = TRUE`.

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 (see details).

See Also

`plot.PLNPCAfamily()` and `plot.PLNnetworkfamily()`
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.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,
  ...
)

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 ggplot2 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).
Display the criteria associated with a collection of PLN mixture fits (a PLNmixturefamily)

Description

Display the criteria associated with a collection of PLN mixture fits (a PLNmixturefamily)

Usage

```r
## S3 method for class 'PLNmixturefamily'
plot(
  x,
  type = c("criteria", "diagnostic"),
  criteria = c("loglik", "BIC", "ICL"),
  reverse = FALSE,
  ...
)
```

Arguments

- `x`: an R6 object with class `PLNmixturefamily`
- `type`: a character, either "criteria" or "diagnostic" for the type of plot.
- `criteria`: vector of characters. The criteria to plot in `c("loglik", "BIC", "ICL")`. Default is `c("loglik", "BIC", "ICL")`.
- `reverse`: A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - 0.5 penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e use the natural direction, on the same scale as the log-likelihood.
- `...`: additional parameters for S3 compatibility. Not used
Details

The BIC and ICL criteria have the form \( \log \text{lik} - 1/2 \times \text{penalty} \) so that they are on the same scale as the model log-likelihood. You can change this direction and use the alternate form \(-2\log \text{lik} + \text{penalty}\), as some authors do, by setting `reverse = TRUE`.

Value

Produces either a diagnostic plot (with `type = 'diagnostic'`) or the evolution of the criteria of the different models considered (with `type = 'criteria'`, the default).

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myMixtures <- PLNmixture(Abundance ~ 1 + offset(log(Offset)),
                          data = trichoptera, control_main = list(smoothing = "forward", iterates = 1))
plot(myMixtures, reverse = TRUE)

Description

Represent the result of the clustering either by coloring the individual in a two-dimension PCA factor map, or by representing the expected matrix of count reorder according to the clustering.

Usage

```r
## S3 method for class 'PLNmixturefit'
plot(x, type = c("pca", "matrix"), main = NULL, plot = TRUE, ...)
```

Arguments

- `x` an R6 object with class `PLNmixturefit`
- `type` character for the type of plot, either "pca", for or "matrix". Default is "pca".
- `main` character. A title for the plot. If NULL (the default), an hopefully appropriate title will be used.
- `plot` logical. Should the plot be displayed or sent back as `ggplot` object
- `...` Not used (S3 compatibility).

Value

a `ggplot` graphic
Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLNmixture(Abundance ~ 1 + offset(log(Offset)),
  data = trichoptera, control_main = list(iterates = 0)) %>% getBestModel()
## Not run:
plot(myPLN, "pca")
plot(myPLN, "matrix")
## End(Not run)

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)

Usage

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

Arguments

x  an R6 object with class PLNnetworkfamily
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 = "criteria".
reverse  A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - 0.5 penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e use the natural direction, on the same scale as the log-likelihood.
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 .9.
...  additional parameters for S3 compatibility. Not used
Details

The BIC and ICL criteria have the form 'loglik - 1/2 * penalty' so that they are on the same scale as the model log-likelihood. You can change this direction and use the alternate form '-2*loglik + penalty', as some authors do, by setting reverse = TRUE.

Value

Produces either a diagnostic plot (with type = 'diagnostic'), a stability plot (with type = 'stability') or the evolution of the criteria of the different models considered (with type = 'criteria', the default).

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fits <- PLNnetwork(Abundance ~ 1, data = trichoptera)
## Not run:
plot(fits)
## 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"), reverse = FALSE, ...)

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 Length 2 color vector. 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 ot 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.
... Not used (S3 compatibility).

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)
Arguments

- **x**: an R6 object with class `PLNPCAfamily`
- **criteria**: vector of characters. The criteria to plot in c("loglik", "BIC", "ICL"). Default is c("loglik", "BIC", "ICL").
- **reverse**: A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - 0.5 penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e use the natural direction, on the same scale as the log-likelihood.
- **...**: additional parameters for S3 compatibility. Not used

Details

The BIC and ICL criteria have the form 'loglik - 1/2 * penalty' so that they are on the same scale as the model log-likelihood. You can change this direction and use the alternate form '-2*loglik + penalty', as some authors do, by setting `reverse = TRUE`.

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 (see details).

Examples

```r
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)
```

Description

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

Usage

```r
## 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 map = "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 ggplot 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**  
*Predict counts of a new sample*

**Description**

Predict counts of a new sample

**Usage**

```r
## 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(), 
  ...
)
```

---
predict.PLNmixturefit

**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**

```r
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)
```

**Description**

Predict either posterior probabilities for each group or latent positions based on new samples
Usage

## S3 method for class 'PLNmixturefit'
predict(
  object,
  newdata,
  type = c("posterior", "response", "position"),
  prior = matrix(rep(1/object$k, object$k), nrow(newdata), object$k, byrow = TRUE),
  control = list(),
  ...
)

Arguments

object an R6 object with class PLNmixturefit
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 posterior are posterior probabilities for each group, response is the group with maximal posterior probability and latent is the averaged latent in the latent space, with weights equal to the posterior probabilities.
prior User-specified prior group probabilities in the new data. The default uses a uniform prior.
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) position in the latent space (if type = "position") or a vector of predicted groups (if type = "response").

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLNmixture(Abundance ~ 1 + offset(log(Offset)),
  data = trichoptera, control_main = list(iterates = 0)) %>% getBestModel()
predict(myPLN, trichoptera, "posterior")
predict(myPLN, trichoptera, "position")
predict(myPLN, trichoptera, "response")
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, Wrench, 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. Normalization 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), Wrench (introduced in Kumar 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 normalization 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

## 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

Σ

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

Usage

## 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

cocf.PLNfit(), standard_error.PLNfit() and vcov.PLNfit() 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

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. For PLNmix-
ture, it is a weighted mean of the variance-covariance matrices of each component.

Usage

## S3 method for class 'PLNmixturefit'
sigma(object, ...)

Arguments

object an R6 object with class PLNmixturefit
...
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.PLNmixturefit() for other ways to access

\[ \Sigma \]

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1 + offset(log(Offset)),
data = trichoptera, control_main = list(iterates = 0)) %>% getBestModel()
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 function stops with a message unless force = TRUE has been specified.

Usage
```
stability_selection(
  Robject, 
  subsamples = NULL, 
  control = list(), 
  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 the number of subsamples used in the stability selection. Automatically set to 20 subsamples with size $10\times\sqrt{n}$ if $n \geq 144$ and $0.8n$ otherwise following Liu et al. (2010) recommendations.
- **control**
  a list controlling the main optimization process in each call to PLNnetwork. See `PLNnetwork()` for details.
- **force**
  force computation of the stability path, even if a previous one has been detected.

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(fits, 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 computed from the (approximate) Fisher information matrix. See `fisher.PLNfit()` for more details on the approximations.

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.PLNfit()` for the complete Fisher information matrix

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1 + offset(log(Offset)), data = trichoptera)
standard_error(myPLN, "wald")
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)
- Covariates: 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.PLNfit(). The function can also be used to return the variance-covariance matrix of the residuals. The latter matrix can also be accessed via sigma.PLNfit()

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 \( d \times p \), 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.PLNfit(), coef.PLNfit(), standard_error.PLNfit()

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

* datasets
  - mollusk, 13
  - oaks, 14
  - trichoptera, 78

coef(), 20
coef.PLNfit, 3
coeff, 74, 79
cor, 74, 79
correlation, 74, 79
correlation coefficients, 74, 79
correlation table, 74, 79
correlation matrix, 74, 79
correlation matrixes, 74, 79
correlation plots, 74, 79
correlations, 74, 79
covariance, 74, 79

correlation plot, 74, 79

covariance matrix, 74, 79

covariance matrixes, 74, 79

covariance plots, 74, 79

correlation analysis, 74, 79

correlation coefficients, 74, 79

correlation matrices, 74, 79

correlation plots, 74, 79

correlation tables, 74, 79

covariance matrices, 74, 79

covariance plots, 74, 79

covariance tables, 74, 79

correlation and covariance, 74, 79

correlation and covariance matrices, 74, 79

correlation and covariance plots, 74, 79

correlation and covariance tables, 74, 79

extract_probs, 8

grob, 30

fisher, 9
fisher.PLNfit, 77

fitted.PLNfit, 10
fitted.PLNmixturefit, 11

getBestModel
  - (getBestModel.PLNPCAfamily), 11
getBestModel, 33, 42, 51
getBestModel .PLNPCAfamily, 11
getModel (getModel.PLNPCAfamily), 12
getModel, 19, 33, 42, 51

ggplot, 29, 30, 35, 38, 39, 43, 58, 59, 63, 68

ggplot2, 19, 35, 53, 61, 62
grob, 30, 59

mollusk, 13

oaks, 14

PLN, 9, 16

PLN, 3, 5, 10, 20, 30, 38, 40, 42, 50, 57, 70–72, 79

PLNfamily, 17, 18, 19, 55–58, 60
PLNfit, 3, 10, 17–20, 21, 22, 46, 54, 69, 74, 77, 79
PLNfit, 20, 27
PLNLDA, 25, 31
PLNLDA, 4, 20, 27, 40
PLNLDAfit, 26, 27, 28–30, 38, 58, 70
PLNLDAfit, 26
PLNmixtur, 31, 35, 36, 39
PLNmixtur, 11, 20, 33, 40
PLNmixturfamily, 12, 13, 33, 39, 62
PLNmixturfit, 5, 11, 33, 35, 36, 37, 63, 70, 71, 75
PLNmodels, 39
PLNmodels::PLNfamily, 33, 42, 52
PLNmodels::PLNfit, 27, 46, 54
PLNetwork, 40
PLNetwork, 6, 8, 20, 40, 42, 44–49, 76
PLNetworkfamily, 6, 8, 12, 13, 17, 42, 42, 44, 49, 64, 76
PLNetworkfit, 12, 13, 42, 44–46, 46, 47, 65, 66
PLNPCA, 50, 59
PLNPCA, 20, 40, 51, 53, 54
PLNPCAfamily, 12, 13, 17, 51, 51, 59, 67
PLNPCAfit, 12, 13, 51, 53, 54, 54, 55, 56, 59, 61, 67
PLNPCAfit, 53
plot, 27, 33, 42, 46, 51, 54
plot.PLNfamily, 60
plot.PLNLDAfit, 61
plot.PLNMixturefamily, 62
plot.PLNMixturefit, 63
plot.PLNetworkfamily, 64
plot.PLNetworkfit, 65
plot.PLNPCAfamily, 66
plot.PLNPCAfit, 60
plot.PLNPCAfit, 67
predict(), 20, 27
predict.PLNfit, 69
predict.PLNLDAfit, 69
predict.PLNmixurefit, 70
prepare_data, 72
prepare_data(), 14, 15, 78
rPLN, 73

sigma(), 20
sigma.PLNfit, 74
sigma.PLNfit(), 4, 79
sigma.PLNmixurefit, 75
sigma.PLNmixurefit(), 5
stability_selection, 76
stability_selection(), 8, 12
standard_error, 10, 77
standard_error(), 20
standard_error.PLNfit(), 4, 74, 79

trichoptera, 78

vcov(), 20
vcov.PLNfit, 79
vcov.PLNfit(), 4, 74, 77

Wrench::wrench(), 7