Package ‘PLNmodels’

January 8, 2024

Title  Poisson Lognormal Models

Version  1.1.0

Description  The Poisson-lognormal model and variants (Chiquet, Mariadassou and Robin, 2021 <doi:10.3389/fevo.2021.588292>) 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  7.2.3

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, corrplot, magrittr, torch, rlang

Suggests  knitr, rmarkdown, testthat, pkgdown, spelling, factoextra

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' 'PLNmixture.R' 'PLNmixturefamily-S3methods.R' 'PLNmixturefamily-class.R' 'PLNmixturefit-S3methods.R' 'PLNmixturefit-class.R' 'PLNmodels-package.R' 'PLNnetwork.R'
R topics documented:

'PLNnetworkfamily-S3methods.R' 'PLNnetworkfamily-class.R'
'PLNnetworkfit-S3methods.R' 'PLNnetworkfit-class.R'
'RcppExports.R' 'barents.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 [aut],
François Gindraud [aut],
Julie Aubert [ctb],
Bastien Batardièire [ctb],
Giovanni Poggiato [ctb],
Cole Trapnell [ctb],
Maddy Duran [ctb]

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

Repository CRAN

Date/Publication 2024-01-08 18:30:02 UTC

R topics documented:

barents . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
coef.PLNfit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
coef.PLNLDAfit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
coef.PLNmixturefit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
coefficient_path . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
compute_offset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
compute_PLN_starting_point . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
extract_probs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
fitted.PLNfit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
fitted.PLNmixturefit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
getBestModel.PLNPCAfamily . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
getModel.PLNPCAfamily . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
mollusk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
oaks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
PLN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
PLNfamily . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
PLNfit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
PLNfit_diagonal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
PLNfit_fixedcov . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
PLNfit_spherical . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
PLNLDA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
PLNLDAfit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
PLNLDAfit_diagonal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
PLNLDA_param . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
Description

This data set gives the abundance of 30 fish species observed in 89 sites in the Barents sea. For each site, 4 additional covariates are known. Subsample of the original datasets studied by Fossheim et al, 2006.

Usage

barents
Format

A data frame with 6 variables:

- Abundance: A 30 fish species by 89 sites count matrix
- Offset: A 30 fish species by 116 samples offset matrix, measuring the sampling effort in each site
- 4 covariates for latitude, longitude, depth (in meters), temperature (in Celsius degrees).

Source

Data from M. Fossheim and coauthors.

References


Examples

data(barents)

table(coef.PLNfit)

Description

Extracts model coefficients from objects returned by \texttt{PLN()} and its variants

Usage

\begin{verbatim}
## S3 method for class 'PLNfit'
coef(object, type = c("main", "covariance"), ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{object} an R6 object with class \texttt{PLNfit}
  \item \texttt{type} type of parameter that should be extracted. Either "main" (default) for $B$
  \item or "covariance" for $\Sigma$
  \item \ldots additional parameters for S3 compatibility. Not used
\end{itemize}

Value

A matrix of coefficients extracted from the PLNfit model.
coef.PLNLDAfit

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) ## B
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

Θ

covariance 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)
Extract model coefficients

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 = PLNmixture_param(smoothing = "none")) %>% getBestModel()
coef(myPLN) ## Theta - empty here
coef(myPLN, type = "mixture") ## pi
coef(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**

```r
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**

```r
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, Wrench, TMM, etc) described in the literature.

**Usage**

```r
compute_offset(
  counts,
  offset = c("TSS", "GMPR", "RLE", "CSS", "Wrench", "TMM", "none"),
  scale = c("none", "count"),
  ...)
```
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).

scale Either "none" (default) or "count". Should the offset be normalized to be on the same scale as the counts?

... 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. TMM code between two libraries is simplified and adapted from M. Robinson (edgeR:::calcFactorTMM). The final output is however different from the one produced by edgeR:::calcFactorTMM as they are intended to be used as such in the model (whereas they need to be multiplied by sequencing depths in edgeR).

Value

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

References


Examples

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

compute_PLN_starting_point

Helper function for PLN initialization.

Description

Barebone function to compute starting points for B, M and S when fitting a PLN. Mostly intended for internal use.

Usage

compute_PLN_starting_point(Y, X, O, w, s = 0.1)

Arguments

Y  Response count matrix
X  Covariate matrix
O  Offset matrix (in log-scale)
w  Weight vector (defaults to 1)
s  Scale parameter for S (defaults to 0.1)

Details

The default strategy to estimate B and M is to fit a linear model with covariates X to the response count matrix (after adding a pseudocount of 1, scaling by the offset and taking the log). The regression matrix is used to initialize B and the residuals to initialize M. S is initialized as a constant conformable matrix with value s.
**extract_probs**

**Value**

a named list of starting values for model parameter B and variational parameters M and S used in the iterative optimization algorithm of PLN()

**Examples**

```r
## Not run:
data(barents)
Y <- barents$Abundance
X <- model.matrix(Abundance ~ Latitude + Longitude + Depth + Temperature, data = barents)
O <- log(barents$Offset)
w <- rep(1, nrow(Y))
compute_PLN_starting_point(Y, X, O, w)
## End(Not run)
```

**extract_probs**

*Extract edge selection frequency in bootstrap subsamples*

**Description**

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

**Arguments**

- `Robject` an object with class `PLNnetworkfamily`, i.e. an output from `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.
fitted.PLNfit

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

### Description

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

### Usage

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

### Arguments

- **object**: an R6 object with class \texttt{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 `PLNmixture()` and its variants

---

**Description**

Extracts model fitted values from objects returned by `PLNmixture()` and its variants

**Usage**

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

**Arguments**

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

**Value**

A matrix of Fitted values extracted from the object object.

---

**getBestModel.PLNPCAfamily**

*Best model extraction from a collection of models*

---

**Description**

Best model extraction from a collection of models

**Usage**

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

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

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

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

Model extraction from a collection of models

### Usage

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

getModel(Robject, var, index)
```

```r
## S3 method for class 'PLNmixturefamily'
```

### 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(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 PLNPCAfit or PLNnetworkfit.

Methods (by class)

- getModel(PLNPCAfamily): Model extraction for PLNPCAfamily
- getModel(PLNmixturefamily): Model extraction for PLNmixturefamily
- getModel(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)
```

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

```r
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)
**PLN**

*Poisson lognormal model*

---

**Description**

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

**Usage**

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

**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-like structure for controlling the optimization, with default generated by `PLN_param()`. See the associated documentation for details.

**Value**

an R6 object with class `PLNfit`

**See Also**

The class `PLNfit` and the configuration function `PLN_param()`

**Examples**

```r
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1, data = trichoptera)
```
PLNfamily

Description

An R6 Class to represent a collection of PLNfit super class for PLNPCAfamily and PLNnetworkfamily.

Public fields

- responses: the matrix of responses common to every models
- covariates: the matrix of covariates common to every models
- offsets: the matrix of offsets common to every models
- weights: the vector of observation weights
- 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 list controlling the optimization and the model

Returns: A new PLNfamily object

Method postTreatment(): Update fields after optimization

Usage:
PLNfamily$postTreatment(config_post, config_optim)

Arguments:
config_post a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.).
config_optim a list for controlling the optimization parameters used during post_treatments

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.
PLNfit

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.

Active bindings

- `n` number of samples
- `q` number of dimensions of the latent space
- `p` number of species
- `d` number of covariates
- `nb_param` number of parameters in the current PLN model
- `model_par` a list with the matrices of the model parameters: B (covariates), Sigma (covariance), Omega (precision matrix), plus some others depending on the variant
- `var_par` a list with the matrices of the variational parameters: M (means) and S2 (variances)
- `optim_par` a list with parameters useful for monitoring the optimization
- `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)
- `vcov_coef` matrix of sandwich estimator of the variance-covariance of B (need fixed -ie known- covariance at the moment)
- `vcov_model` character: the model used for the residual covariance
- `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$new()`
- `PLNfit$update()`
- `PLNfit$optimize()
- `PLNfit$optimize_vestep()
- `PLNfit$postTreatment()
- `PLNfit$predict()
- `PLNfit$predict_cond()
- `PLNfit$show()
- `PLNfit$print()
- `PLNfit$clone()

**Method new():** Initialize a PLNfit model

*Usage:*

```r
PLNfit$new(responses, covariates, offsets, weights, formula, 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
- `control` a list-like structure for controlling the fit, see `PLN_param()`.

**Method update():** Update a PLNfit object

*Usage:*

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

*Arguments:*

- `B` 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 variational parameters for the mean
S  matrix of variational parameters for the variance
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 optimize():  Call to the NLOpt or TORCH optimizer and update of the relevant fields

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

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.
config  part of the control argument which configures the optimizer

Method optimize_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, B). Intended to position new data in the latent space.

Usage:
PLNfit$optimize_vestep(  covariates,  offsets,  responses,  weights,  B = self$model_par$B,  Omega = self$model_par$Omega,  control = PLN_param(backend = "nlopt")  )

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.
Optional fixed value of the regression parameters

\( \Omega \)  precision matrix of the latent variables. Inverse of \( \Sigma \).

control a list-like structure for controlling the fit, see \texttt{PLN_param()}.

\( \Sigma \) variance-covariance matrix of the latent variables

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 \texttt{postTreatment()}: Update \( R^2 \), fisher and std_err fields after optimization

Usage:
\begin{verbatim}
PLNfit$postTreatment(
  responses,  
covariates,  
offsets,  
weights = rep(1, nrow(responses)),  
config_post,  
config_optim,  
nullModel = NULL
)
\end{verbatim}

Arguments:
- \texttt{responses} the matrix of responses (called \texttt{Y} in the model). Will usually be extracted from the corresponding field in \texttt{PLNfamily-class}
- \texttt{covariates} design matrix (called \texttt{X} in the model). Will usually be extracted from the corresponding field in \texttt{PLNfamily-class}
- \texttt{offsets} offset matrix (called \texttt{O} in the model). Will usually be extracted from the corresponding field in \texttt{PLNfamily-class}
- \texttt{weights} an optional vector of observation weights to be used in the fitting process.
- \texttt{config_post} a list for controlling the post-treatments (optional bootstrap, jackknife, \( R^2 \), etc.). See details
- \texttt{config_optim} a list for controlling the optimization (optional bootstrap, jackknife, \( R^2 \), etc.). See details
- \texttt{nullModel} null model used for approximate \( R^2 \) computations. Defaults to a GLM model with same design matrix but not latent variable.

Details: The list of parameters \texttt{config} controls the post-treatment processing, with the following entries:
- \texttt{jackknife} boolean indicating whether jackknife should be performed to evaluate bias and variance of the model parameters. Default is \texttt{FALSE}.
- \texttt{bootstrap} integer indicating the number of bootstrap resamples generated to evaluate the variance of the model parameters. Default is \texttt{0} (inactivated).
- \texttt{variational_var} boolean indicating whether variational Fisher information matrix should be computed to estimate the variance of the model parameters (highly underestimated). Default is \texttt{FALSE}.
- \texttt{rsquared} boolean indicating whether approximation of \( R^2 \) based on deviance should be computed. Default is \texttt{TRUE}.
• trace integer for verbosity. should be > 1 to see output in post-treatments

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

**Usage**:
```r
PLNfit$predict(
  newdata,
  responses = NULL,
  type = c("link", "response"),
  level = 1,
  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.
- `responses`: Optional data frame containing the count of the observed variables (matching the names of the provided as data in the PLN function), assuming the interest in in testing the model.
- `type`: Scale used for the prediction. Either `link` (default, predicted positions in the latent space) or `response` (predicted counts).
- `level`: Optional integer value the level to be used in obtaining the predictions. Level zero corresponds to the population predictions (default if `responses` is not provided) while level one (default) corresponds to predictions after evaluating the variational parameters for the new data.
- `envir`: Environment in which the prediction is evaluated

**Details**: Note that `level = 1` can only be used if `responses` are provided, as the variational parameters can’t be estimated otherwise. In the absence of `responses`, `level` is ignored and the fitted values are returned

**Returns**: A matrix with predictions scores or counts.

**Method predict_cond()**: Predict position, scores or observations of new data, conditionally on the observation of a (set of) variables

**Usage**:
```r
PLNfit$predict_cond(
  newdata,
  cond_responses,
  type = c("link", "response"),
  var_par = FALSE,
  envir = parent.frame()
)
```

**Arguments**:
- `newdata`: A data frame containing the covariates of the sites where to predict
- `cond_responses`: a data frame containing the count of the observed variables (matching the names of the provided as data in the PLN function)
- `type`: Scale used for the prediction. Either `link` (default, predicted positions in the latent space) or `response` (predicted counts).
var_par  Boolean. Should new estimations of the variational parameters of mean and variance be sent back, as attributes of the matrix of predictions. Default to FALSE.

envir  Environment in which the prediction is evaluated

Returns:  A matrix with predictions scores or counts.

Method show():  User friendly print method

Usage:
PLNfit$show(
  model = paste("A multivariate Poisson Lognormal fit with", self$vcov_model, "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)

---

**PLNfit_diagonal**  An R6 Class to represent a PLNfit in a standard, general framework, with diagonal residual covariance

**Description**

The function *PLNLDA()* produces an instance of an object with class *PLNLDAti*. 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 -> PLNfit_diagonal

Active bindings

nb_param  number of parameters in the current PLN model
vcov_model  character: the model used for the residual covariance

Methods

Public methods:

- PLNfit_diagonal$new()
- PLNfit_diagonal$clone()

Method new(): Initialize a PLNfit model

Usage:
PLNfit_diagonal$new(responses, covariates, offsets, weights, formula, 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
control  a list for controlling the optimization. See details.

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

Usage:
PLNfit_diagonal$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

Super classes

PLNmodels::PLNfit -> PLNmodels::PLNLDaFit -> PLNLDaFit_spherical

Active bindings

vcov_model  character: the model used for the residual covariance
nb_param  number of parameters in the current PLN model
Methods

Public methods:

• PLNLDAfit_spherical$new()
• PLNLDAfit_spherical$clone()

Method new(): Initialize a PLNfit model

Usage:
PLNLDAfit_spherical$new(
  grouping,
  responses,
  covariates,
  offsets,
  weights,
  formula,
  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
- control: a list for controlling the optimization. See details.

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

Usage:
PLNLDAfit_spherical$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)
## Not run:
data(trichoptera)
```r
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ 1, data = trichoptera, control = PLN_param(covariance = "spherical"))
class(myPLNLDA)
print(myPLNLDA)

## End(Not run)
```

**PLNfit_fixedcov**

An R6 Class to represent a PLNfit in a standard, general framework, with fixed (inverse) residual covariance

Description

An R6 Class to represent a PLNfit in a standard, general framework, with fixed (inverse) residual covariance

Super class

`PLNmodels::PLNfit` -> `PLNfit_fixedcov`

Active bindings

- `nb_param` number of parameters in the current PLN model
- `vcov_model` character: the model used for the residual covariance
- `vcov_coef` matrix of sandwich estimator of the variance-covariance of B (needs known covariance at the moment)

Methods

Public methods:

- `PLNfit_fixedcov$new()`
- `PLNfit_fixedcov$optimize()`
- `PLNfit_fixedcov$postTreatment()`
- `PLNfit_fixedcov$clone()`

Method `new()`: Initialize a PLNfit model

Usage:

`PLNfit_fixedcov$new(responses, covariates, offsets, weights, formula, 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.

control  a list for controlling the optimization. See details.

**Method** optimize(): Call to the NLopt or TORCH optimizer and update of the relevant fields

**Usage:**

```r
PLNfit_fixedcov$optimize(responses, covariates, offsets, weights, config)
```

**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.
- **config**  part of the control argument which configures the optimizer.

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

**Usage:**

```r
PLNfit_fixedcov$postTreatment(
  responses, covariates, offsets, weights = rep(1, nrow(responses)),
  config_post, config_optim, 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.
- **config_post**  a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.). See details.
- **config_optim**  a list for controlling the optimization parameter. See details.
- **nullModel**  null model used for approximate R2 computations. Defaults to a GLM model with same design matrix but not latent variable.

**Details:** The list of parameters config controls the post-treatment processing, with the following entries:
• trace integer for verbosity. should be > 1 to see output in post-treatments
• jackknife boolean indicating whether jackknife should be performed to evaluate bias and variance of the model parameters. Default is FALSE.
• bootstrap integer indicating the number of bootstrap resamples generated to evaluate the variance of the model parameters. Default is 0 (inactivated).
• variational_var boolean indicating whether variational Fisher information matrix should be computed to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
• rsquared boolean indicating whether approximation of R2 based on deviance should be computed. Default is TRUE

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

Usage:
PLNfit_fixedcov$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)
Methods

Public methods:

• PLNfit_spherical$new()
• PLNfit_spherical$clone()

Method new(): Initialize a PLNfit model

Usage:
PLNfit_spherical$new(responses, covariates, offsets, weights, formula, 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
control a list for controlling the optimization. See details.

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

Usage:
PLNfit_spherical$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)
Usage

PLNLDA(formula, data, subset, weights, grouping, control = PLN_param())

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 specifying the class of each observation used for discriminant analysis.
- **grouping**: an optional vector of observation weights to be used in the fitting process.
- **control**: a list-like structure for controlling the optimization, with default generated by PLN_param(). See the associated documentation.

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-6
- "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".

Value

an R6 object with class PLNLDAfit()
PLNLDAfit

See Also

The class PLNLDAfit

Examples

```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: `B` (covariates), `Sigma` (latent covariance), `C` (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()`
Method new(): Initialize a PLNLDAfit object

Usage:
PLNLDAfit$new(
  grouping,
  responses,
  covariates,
  offsets,
  weights,
  formula,
  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
- control  list controlling the optimization and the model

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

Usage:
PLNLDAfit$optimize(grouping, responses, covariates, offsets, weights, config)

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. Automatically built from the covariates and the formula from the call
- 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.
- config  list controlling the optimization
- X  Abundance matrix.

Method postTreatment(): Update R2, fisher and std_err fields and visualization
Usage:
PLNLDAfit$postTreatment(
  grouping,
  responses,
  covariates,
  offsets,
  config_post,
  config_optim
)

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
config_post  a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.).
cconfig_optim  list controlling the optimization parameters

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 = PLN_param(backend = "nlopt"),
  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 prob-
abilities 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:
PLNLDAfit$show()

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

Usage:
PLNLDAfit$clone(deep = FALSE)

Arguments:
depth  Whether to make a deep clone.

See Also
The function PLNLDA.

Examples

## 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)

PLNLDAfit_diagonal An R6 Class to represent a PLNfit in a LDA framework with diagonal
covariance

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 docu-
mentation for the methods inherited by PLNfit(), the plot() method for LDA visualization and
predict() method for prediction.
PLNLDAfit_diagonal

Super classes

\texttt{PLNmodels::PLNfit} $\rightarrow$ \texttt{PLNmodels::PLNLDAfit} $\rightarrow$ \texttt{PLNLDAfit_diagonal}

Active bindings

\begin{itemize}
  \item \texttt{vcov_model} character: the model used for the residual covariance
  \item \texttt{nb_param} number of parameters in the current PLN model
\end{itemize}

Methods

Public methods:

\begin{itemize}
  \item \texttt{PLNLDAfit_diagonal$new()}
  \item \texttt{PLNLDAfit_diagonal$clone()}
\end{itemize}

Method \texttt{new()}: Initialize a \texttt{PLNfit} model

\textit{Usage}:

\begin{verbatim}
PLNLDAfit_diagonal$new(
  grouping, responses, covariates, offsets, weights, formula, control
)
\end{verbatim}

\textit{Arguments}:

\begin{itemize}
  \item \texttt{grouping} a factor specifying the class of each observation used for discriminant analysis.
  \item \texttt{responses} the matrix of responses (called Y in the model). Will usually be extracted from the corresponding field in \texttt{PLNfamily-class}
  \item \texttt{covariates} design matrix (called X in the model). Will usually be extracted from the corresponding field in \texttt{PLNfamily-class}
  \item \texttt{offsets} offset matrix (called O in the model). Will usually be extracted from the corresponding field in \texttt{PLNfamily-class}
  \item \texttt{weights} an optional vector of observation weights to be used in the fitting process.
  \item \texttt{formula} model formula used for fitting, extracted from the formula in the upper-level call
  \item \texttt{control} a list for controlling the optimization. See details.
\end{itemize}

Method \texttt{clone()}: The objects of this class are cloneable with this method.

\textit{Usage}:

\begin{verbatim}
PLNLDAfit_diagonal$clone(deep = FALSE)
\end{verbatim}

\textit{Arguments}:

\begin{itemize}
  \item \texttt{deep} Whether to make a deep clone.
Examples

```r
## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ 1, data = trichoptera, control = PLN_param(covariance = "diagonal"))
class(myPLNLDA)
print(myPLNLDA)

## End(Not run)
```

---

**PLNLDA_param**

Control of a PLNLDA fit

**Description**

Helper to define list of parameters to control the PLNLDA fit. All arguments have defaults.

**Usage**

```r
PLNLDA_param(
  backend = c("nlopt", "torch"),
  trace = 1,
  covariance = c("full", "diagonal", "spherical"),
  config_post = list(),
  config_optim = list(),
  inception = NULL
)
```

**Arguments**

- `backend` : optimization back used, either "nlopt" or "torch". Default is "nlopt"
- `trace` : a integer for verbosity.
- `covariance` : character setting the model for the covariance matrix. Either "full", "diagonal" or "spherical". Default is "full".
- `config_post` : a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.). See details
- `config_optim` : a list for controlling the optimizer (either "nlopt" or "torch" backend). See details
- `inception` : set up the parameters 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.
Details

The list of parameters `config_optim` controls the optimizers. When "nlopt" is chosen the following entries are relevant:

- "algorithm" the optimization method used by NLOPT among LD type, e.g. "CCSAQ", "MMA", "LBFGS". See NLOPT documentation for further details. Default is "CCSAQ".
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10000
- "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
- "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-6
- "ftol_abs" stop when an optimization step changes the objective function by less than ftol_abs. Default is 0.0 (disabled)
- "xtol_abs" stop when an optimization step changes every parameters by less than xtol_abs. Default is 0.0 (disabled)
- "maxtime" stop when the optimization time (in seconds) exceeds maxtime. Default is -1 (disabled)

When "torch" backend is used (only for PLN and PLNLDA for now), the following entries are relevant:

- "algorithm" the optimizer used by torch among RPROP (default), RMSPROP, ADAM and ADAGRAD
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10 000
- "numepoch" stop training once this number of epochs exceeds numepoch. Set to -1 to enable infinite training. Default is 1 000
- "num_batch" number of batches to use during training. Defaults to 1 (use full dataset at each epoch)
- "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
- "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-6
- "lr" learning rate. Default is 0.1.
- "momentum" momentum factor. Default is 0 (no momentum). Only used in RMSPROP
- "weight_decay" Weight decay penalty. Default is 0 (no decay). Not used in RPROP
- "step_sizes" pair of minimal (default: 1e-6) and maximal (default: 50) allowed step sizes. Only used in RPROP
- "etas" pair of multiplicative increase and decrease factors. Default is (0.5, 1.2). Only used in RPROP
- "centered" if TRUE, compute the centered RMSProp where the gradient is normalized by an estimation of its variance weight_decay (L2 penalty). Default to FALSE. Only used in RMSPROP
The list of parameters `config_post` controls the post-treatment processing (for most `PLN*()` functions), with the following entries (defaults may vary depending on the specific function, check `config_post_default_*` for defaults values):

- `jackknife` boolean indicating whether jackknife should be performed to evaluate bias and variance of the model parameters. Default is FALSE.
- `bootstrap` integer indicating the number of bootstrap resamples generated to evaluate the variance of the model parameters. Default is 0 (inactivated).
- `variational_var` boolean indicating whether variational Fisher information matrix should be computed to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
- `sandwich_var` boolean indicating whether sandwich estimation should be used to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
- `rsquared` boolean indicating whether approximation of R2 based on deviance should be computed. Default is TRUE.

### Value

List of parameters configuring the fit.

---

**PLNmixture**

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
PLNmixture(formula, data, subset, clusters = 1:5, control = PLNmixture_param())
```

**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`: a list-like structure for controlling the optimization, with default generated by `PLNmixture_param()`. See the associated documentation for details.
Value

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

See Also

The classes `PLNmixturefamily`, `PLNmixturefit` and `PLNmixture_param()`.

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)), clusters = 1:4, data = trichoptera,
control = PLNmixture_param(smoothing = 'none'))

# 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$plot_objective()
• PLNmixturefamily$getBestModel()
• PLNmixturefamily$show()
• PLNmixturefamily$print()
• PLNmixturefamily$clone()

Method new(): helper function for forward smoothing: split a group
Initialize all models in the collection.

Usage:
PLNmixturefamily$new(
  clusters,
  responses,
  covariates,
  offsets,
  formula,
  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
control a list for controlling the optimization. See details.
control a list for controlling the optimization. See details.

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

Usage:
PLNmixturefamily$optimize(config)

Arguments:
config a list for controlling the optimization

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`
**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:
PLNmixturefit$new(
  responses,
  covariates,
  offsets,
  posteriorProb,
  formula,
  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
control a list for controlling the optimization.

Method optimize(): Optimize a PLNmixturefit model

Usage:
PLNmixturefit$optimize(responses, covariates, offsets, config)

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
config a list for controlling the optimization

Method predict(): Predict group of new samples

Usage:
PLNmixturefit$predict(
  newdata,
  type = c("posterior", "response", "position"),
  prior = matrix(rep(1/self$k, self$k), nrow(newdata), self$k, byrow = TRUE),
  control = PLNmixture_param(),
  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-like structure for controlling the fit. See PLNmixture_param() 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:
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:
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:
```r
PLNmixturefit$postTreatment(
    responses,
    covariates,
    offsets,
    weights,
    config_post,
    config_optim,
    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.
- `config_post` a list for controlling the post-treatment
- `config_optim` a list for controlling the optimization during the post-treatment computations
- `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:
```r
PLNmixturefit$show()
```

Method `print()`: User friendly print method

Usage:
```r
PLNmixturefit$print()
```

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

Usage:
```r
PLNmixturefit$clone(deep = FALSE)
```

Arguments:
- `deep` Whether to make a deep clone.

See Also

The function `PLNmixture`, the class `PLNmixturefamily`
PLNmixture_param

Control of a PLNmixture fit

Description

Helper to define list of parameters to control the PLNmixture fit. All arguments have defaults.

Usage

```r
PLNmixture_param(
  backend = "nlopt",
  trace = 1,
  covariance = "spherical",
  init_cl = "kmeans",
  smoothing = "both",
  config_optim = list(),
  config_post = list(),
  inception = NULL
)
```

Arguments

- `backend`: optimization back used, either "nlopt" or "torch". Default is "nlopt"
- `trace`: a integer for verbosity.
- `covariance`: character setting the model for the covariance matrices of the mixture components. Either "full", "diagonal" or "spherical". Default is "spherical".
- `init_cl`: The initial clustering to apply. Either, 'kmeans', CAH' or a user defined clustering given as a list of clusterings, the size of which is equal to the number of clusters considered. Default is 'kmeans'.
- `smoothing`: The smoothing to apply. Either, 'none', 'forward', 'backward' or 'both'. Default is 'both'.
- `config_optim`: a list for controlling the optimizer (either "nlopt" or "torch" backend). See details
- `config_post`: a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.).
- `inception`: Set up the parameters 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.

Details

See `PLN_param()` for a full description of the generic optimization parameters. PLNmixture_param() also has additional parameters controlling the optimization due the inner-outer loop structure of the optimizer.
• "ftol_out" outer solver stops when an optimization step changes the objective function by less than xtol multiplied by the absolute value of the parameter. Default is 1e-6
• "maxit_out" outer solver stops when the number of iteration exceeds maxit_out. Default is 50
• "it_smoothing" number of the iterations of the smoothing procedure. Default is 1.

Value
list of parameters configuring the fit.

See Also
PLN_param()

PLNnetwork

Poisson lognormal model towards sparse network inference

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

Usage
PLNnetwork(
  formula,
  data,
  subset,
  weights,
  penalties = NULL,
  control = PLNnetwork_param()
)

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 PLNnetwork_param() for additional tuning of the penalty.
  control a list-like structure for controlling the optimization, with default generated by PLNnetwork_param(). See the corresponding documentation for details;
Value

An R6 object with class `PLNnetworkfamily`, which contains a collection of models with class `PLNnetworkfit`.

See Also

The classes `PLNnetworkfamily` and `PLNnetworkfit`, and the configuration function `PLNnetwork_param()`.

Examples

```r
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` \(\rightarrow\) `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**:

```r
PLNnetworkfamily$new(
  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
  control      # a list for controlling the optimization.
)
```

**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
- `control` a list for controlling the optimization.

**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(config)
```

**Arguments**:

- `config` a list for controlling the optimization.

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

**Usage**:

```r
PLNnetworkfamily$stability_selection(
  subsamples = NULL,  # 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 \(0.8 \times n\) otherwise following Liu et al. (2010) recommendations.
  control = PLNnetwork_param()
)
```

**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 \(10 \times \sqrt{n}\) if \(n \geq 144\) and \(0.8 \times n\) otherwise following Liu et al. (2010) recommendations.
- `control` a list for controlling the optimization.
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 $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:**

```r
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:**

```r
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:
depth 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 classes

PLNmodels::PLNfit -> PLNmodels::PLNfit_fixedcov -> PLNnetworkfit
Active bindings

vcov_model character: the model used for the residual covariance
penalty the global level of sparsity in the current model
penalty_weights a matrix of weights controlling the amount of penalty element-wise.
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
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$optimize()
- PLNnetworkfit$latent_network()
- PLNnetworkfit$plot_network()
- PLNnetworkfit$show()
- PLNnetworkfit$clone()

Method new(): Initialize a PLNnetworkfit object

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

Arguments:

penalty a positive real number controlling the level of sparsity of the underlying network.
penalty_weights either a single or a list of 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.
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

c control a list for controlling the optimization.

Method update(): Update fields of a PLNnetworkfit object

Usage:
PLNnetworkfit$update(
  penalty = NA,
  B = NA,
  Sigma = NA,
  Omega = NA,
  M = NA,
  S = 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.
B 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
S 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, config)

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.
config a list for controlling the optimization

**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:**
depth Whether to make a deep clone.
See Also

The function `PLNnetwork()`, the class `PLNnetworkfamily`

Examples

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

## End(Not run)
```

---

**PLNnetwork_model**

Control of PLNnetwork fit

Description

Helper to define list of parameters to control the PLN fit. All arguments have defaults.

Usage

```r
PLNnetwork_param(
  backend = c("nlopt", "torch"),
  inception_cov = c("full", "spherical", "diagonal"),
  trace = 1,
  n_penalties = 30,
  min_ratio = 0.1,
  penalize_diagonal = TRUE,
  penalty_weights = NULL,
  config_post = list(),
  config_optim = list(),
  inception = NULL
)
```

Arguments

- **backend**
  - Optimization back used, either "nlopt" or "torch". Default is "nlopt".
- **inception_cov**
  - Covariance structure used for the inception model used to initialize the PLN-family. Defaults to "full" and can be constrained to "diagonal" and "spherical".
- **trace**
  - A integer for verbosity.
- **n_penalties**
  - An integer that specifies 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.
penalize_diagonal

boolean: should the diagonal terms be penalized in the graphical-Lasso? Default is TRUE

penalty_weights

either a single or a list of 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.

config_post

a list for controlling the post-treatment (optional bootstrap, jackknife, R2, etc).

config_optim

a list for controlling the optimizer (either "nlopt" or "torch" backend). See de-
tails

inception

Set up the parameters 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 infer-
ence.

Details

See PLN_param() for a full description of the generic optimization parameters. PLNnetwork_param() also has two additional parameters controlling the optimization due the inner-outer loop structure
of the optimizer:

• "ftol_out" outer solver stops when an optimization step changes the objective function by less
than xtol multiplied by the absolute value of the parameter. Default is 1e-6

• "maxit_out" outer solver stops when the number of iteration exceeds maxit_out. Default is 50

Value

list of parameters configuring the fit.

See Also

PLN_param()
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 a list-like structure for controlling the optimization, with default generated by PLNPCA_param(). See the associated documentation for details.

Value

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

See Also

The classes PLNPCAfamily and PLNPCAfit, and the configuration function PLPCA_param().

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 PLPCA() produces an instance of this class.

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

PLNmodels::PLNfamily -> PLNPCAfamily

Active bindings

ranks the dimensions of the successively fitted models

Methods

Public methods:

- PLNPCAfamily$new()
- PLNPCAfamily$optimize()
- PLNPCAfamily$getModel()
- PLNPCAfamily$getBestModel()
- PLNPCAfamily$plot()
- PLNPCAfamily$show()
- PLNPCAfamily$clone()

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

Usage:

```
PLNPCAfamily$new(
  ranks,
  responses,
  covariates,
  offsets,
  weights,
  formula,
  control
)
```

Arguments:

- ranks 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
- weights the vector of observation weights
- formula model formula used for fitting, extracted from the formula in the upper-level call
- control list controlling the optimization and the model

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

Usage:

```
PLNPCAfamily$optimize(config)
```

Arguments:

- config list controlling the optimization.
Method `getModel()`: Extract model from collection and add "PCA" class for compatibility with `factoextra::fviz()`

Usage:
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 `PLNPCAfit` object

Method `getBestModel()`: Extract best model in the collection

Usage:
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 `PLNPCAfit` object

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

Usage:
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 `ggplot2` object

Method `show()`: User friendly print method

Usage:
PLNPCAfamily$show()

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

Usage:
PLNPCAfamily$clone(deep = FALSE)

Arguments:
- `deep` Whether to make a deep clone.

See Also
The function `PLNPCA()`, the class `PLNPCAfit()`
PLNPCAf it

Examples

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

PLNPCAf it

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 `PLNPCAf it`. 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` \rightarrow `PLNPCAf it`

Active bindings

- `rank` the dimension of the current model
- `vcov_model` character: the model used for the residual covariance
- `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: B (covariates), Sigma (covariance), Omega (precision) and C (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$optimize()
- PLNPCAfit$optimize_vestep()
- PLNPCAfit$project()
- PLNPCAfit$postTreatment()
- PLNPCAfit$setVisualization()
- PLNPCAfit$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, 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 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.
- formula: model formula used for fitting, extracted from the formula in the upper-level call
- control: a list for controlling the optimization. See details.

Method update(): Update a PLNPCAfit object

Usage:

PLNPCAfit$update(
  B = NA,
  Sigma = NA,
  Omega = NA,
  C = NA,
  M = NA,
  S = NA,
  Z = NA,
  A = NA,
  Ji = NA,
  R2 = NA,
  monitoring = NA
)
Arguments:

B  matrix of regression matrix
Sigma variance-covariance matrix of the latent variables
Omega precision matrix of the latent variables. Inverse of Sigma.
C  matrix of PCA loadings (in the latent space)
M  matrix of mean vectors for the variational approximation
S  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)
R^2 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, config)

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.
config part of the control argument which configures the optimizer

Method optimize_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 (C, B). Intended to position new data in the latent space for further use with PCA.

Usage:
PLNPCAfit$optimize_vestep(
covariates,
offsets,
responses,
weights = rep(1, self$n),
control = PLNPCA_param(backend = "nlopt")
)

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 S2 of variational variances
- the vector log.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 = PLNPCA_param(), 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

Usage:
PLNPCAfit$postTreatment(
  responses,
covariates,
offsets,
weights,
config_post,
config_optim,
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.

config_post a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.). See details.

config_optim a list for controlling the optimizer (either "nlopt" or "torch" backend). See details.

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

Details: The list of parameters config_post controls the post-treatment processing, with the following entries:

- jackknife boolean indicating whether jackknife should be performed to evaluate bias and variance of the model parameters. Default is FALSE.
- bootstrap integer indicating the number of bootstrap resamples generated to evaluate the variance of the model parameters. Default is 0 (inactivated).
- variational_var boolean indicating whether variational Fisher information matrix should be computed to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
- rsquared boolean indicating whether approximation of R2 based on deviance should be computed. Default is TRUE.
- trace integer for verbosity. should be > 1 to see output in post-treatments.

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 PLNLDAfit 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.
PLNPCA_param

See Also

The function PLNPCA, the class PLNPCAfamily

Examples

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

PLNPCA_param

Control of PLNPCA fit

Description

Helper to define list of parameters to control the PLNPCA fit. All arguments have defaults.

Usage

PLNPCA_param(
  backend = "nlopt",
  trace = 1,
  config_optim = list(),
  config_post = list(),
  inception = NULL
)

Arguments

backend optimization back used, either "nlopt" or "torch". Default is "nlopt"
trace a integer for verbosity.
config_optim a list for controlling the optimizer (either "nlopt" or "torch" backend). See details
cconfig_post a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.). See details
inception Set up the parameters 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.
Details

The list of parameters `config_optim` controls the optimizers. When "nlopt" is chosen the following entries are relevant:

- "algorithm" the optimization method used by NLOPT among LD type, e.g. "CCSAQ", "MMA", "LBFGS". See NLOPT documentation for further details. Default is "CCSAQ".
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10000
- "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 $1e^{-8}$
- "xtol_rel" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is $1e^{-6}$
- "ftol_abs" stop when an optimization step changes the objective function by less than ftol_abs. Default is 0.0 (disabled)
- "xtol_abs" stop when an optimization step changes every parameters by less than xtol_abs. Default is 0.0 (disabled)
- "maxtime" stop when the optimization time (in seconds) exceeds maxtime. Default is -1 (disabled)

When "torch" backend is used (only for PLN and PLNLDA for now), the following entries are relevant:

- "algorithm" the optimizer used by torch among RPROP (default), RMSPROP, ADAM and ADAGRAD
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10 000
- "numepoch" stop training once this number of epochs exceeds numepoch. Set to -1 to enable infinite training. Default is 1 000
- "num_batch" number of batches to use during training. Defaults to 1 (use full dataset at each epoch)
- "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 $1e^{-8}$
- "xtol_rel" stop when an optimization step changes every parameters by less than xtol multiplied by the absolute value of the parameter. Default is $1e^{-6}$
- "lr" learning rate. Default is 0.1.
- "momentum" momentum factor. Default is 0 (no momentum). Only used in RMSPROP
- "weight_decay" Weight decay penalty. Default is 0 (no decay). Not used in RPROP
- "step_sizes" pair of minimal (default: 1e-6) and maximal (default: 50) allowed step sizes. Only used in RPROP
- "etas" pair of multiplicative increase and decrease factors. Default is (0.5, 1.2). Only used in RPROP
- "centered" if TRUE, compute the centered RMSProp where the gradient is normalized by an estimation of its variance weight_decay (L2 penalty). Default to FALSE. Only used in RMSPROP
The list of parameters `config_post` controls the post-treatment processing (for most `PLN*()` functions), with the following entries (defaults may vary depending on the specific function, check `config_post_default_*` for defaults values):

- **jackknife** boolean indicating whether jackknife should be performed to evaluate bias and variance of the model parameters. Default is FALSE.
- **bootstrap** integer indicating the number of bootstrap resamples generated to evaluate the variance of the model parameters. Default is 0 (inactivated).
- **variational_var** boolean indicating whether variational Fisher information matrix should be computed to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
- **sandwich_var** boolean indicating whether sandwich estimation should be used to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
- **rsquared** boolean indicating whether approximation of R2 based on deviance should be computed. Default is TRUE

Value

list of parameters configuring the fit.

**PLN_param**

*Control of a PLN fit*

**Description**

Helper to define list of parameters to control the PLN fit. All arguments have defaults.

**Usage**

```r
PLN_param(
  backend = c("nlopt", "torch"),
  trace = 1,
  covariance = c("full", "diagonal", "spherical", "fixed"),
  Omega = NULL,
  config_post = list(),
  config_optim = list(),
  inception = NULL
)
```

**Arguments**

- **backend** optimization back used, either "nlopt" or "torch". Default is "nlopt"
- **trace** a integer for verbosity.
- **covariance** character setting the model for the covariance matrix. Either "full", "diagonal", "spherical" or "fixed". Default is "full".
Omega precision matrix of the latent variables. Inverse of Sigma. Must be specified if covariance is "fixed"

config_post a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.). See details

config_optim a list for controlling the optimizer (either "nlopt" or "torch" backend). See details

inception Set up the parameters 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.

Details

The list of parameters config_optim controls the optimizers. When "nlopt" is chosen the following entries are relevant

- "algorithm" the optimization method used by NLOPT among LD type, e.g. "CCSAQ", "MMA", "LBFGS". See NLOPT documentation for further details. Default is "CCSAQ".
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10000
- "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
- "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-6
- "ftol_abs" stop when an optimization step changes the objective function by less than ftol_abs. Default is 0.0 (disabled)
- "xtol_abs" stop when an optimization step changes every parameters by less than xtol_abs. Default is 0.0 (disabled)
- "maxtime" stop when the optimization time (in seconds) exceeds maxtime. Default is -1 (disabled)

When "torch" backend is used (only for PLN and PLNLDA for now), the following entries are relevant:

- "algorithm" the optimizer used by torch among RPROP (default), RMSPROP, ADAM and ADAGRAD
- "maxeval" stop when the number of iteration exceeds maxeval. Default is 10 000
- "numepoch" stop training once this number of epochs exceeds numepoch. Set to -1 to enable infinite training. Default is 1 000
- "num_batch" number of batches to use during training. Defaults to 1 (use full dataset at each epoch)
- "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
- "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-6
• "lr" learning rate. Default is 0.1.
• "momentum" momentum factor. Default is 0 (no momentum). Only used in RMSPROP
• "weight_decay" Weight decay penalty. Default is 0 (no decay). Not used in RPROP
• "step_sizes" pair of minimal (default: 1e-6) and maximal (default: 50) allowed step sizes. Only used in RPROP
• "etas" pair of multiplicative increase and decrease factors. Default is (0.5, 1.2). Only used in RPROP
• "centered" if TRUE, compute the centered RMSProp where the gradient is normalized by an estimation of its variance weight_decay (L2 penalty). Default to FALSE. Only used in RMSPROP

The list of parameters config_post controls the post-treatment processing (for most PLN*() functions), with the following entries (defaults may vary depending on the specific function, check config_post_default_* for defaults values):

• jackknife boolean indicating whether jackknife should be performed to evaluate bias and variance of the model parameters. Default is FALSE.
• bootstrap integer indicating the number of bootstrap resamples generated to evaluate the variance of the model parameters. Default is 0 (inactivated).
• variational_var boolean indicating whether variational Fisher information matrix should be computed to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
• sandwich_var boolean indicating whether sandwich estimation should be used to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
• rsquared boolean indicating whether approximation of R2 based on deviance should be computed. Default is TRUE

Value

list of parameters configuring the fit.

---

plot.PLNfamily

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

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

```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)
```

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
plot.PLNLDAfit

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 PLNPCAf
  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).

Value

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

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLNLDA <- PLNLDA(Abundance ~ 1, grouping = Group, data = trichoptera)
## Not run:
plot(myPLNLDA, map = "individual", nb_axes = 2)
## End(Not run)
plot.PLNmixturefamily  
Display the criteria associated with a collection of PLNmixture fits (a PLNmixturefamily)

Description
Display the criteria associated with a collection of PLNmixture fits (a PLNmixturefamily)

Usage
## 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 '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') 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 = PLNmixture_param(smoothing = "none"))
plot(myMixtures, reverse = TRUE)

plot.PLNmixturefit  Mixture visualization of a PLNmixturefit object

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

## 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 = PLNmixture_param(smoothing = "none")) %>% 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.
plot.PLNnetworkfit

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

Description

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

Usage

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

Arguments

x  an R6 object with class PLNnetworkfit

type  character. Value of the weight of the edges in the network, either "partial_cor" (partial correlation) or "support" (binary). Default is "partial_cor".

output  the type of output used: either 'igraph' or 'corrplot'. Default is 'igraph'.

edge.color  Length 2 color vector. Color for positive/negative edges. Default is c("#F8766D", 
          "#00BFC4"). Only relevant for igraph output.
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 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 loglikelihood.

... 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

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)
**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
predict.PLNLDAfit

Usage

## S3 method for class 'PLNfit'
predict(
  object,
  newdata,
  responses = NULL,
  level = 1,
  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
responses Optional data frame containing the count of the observed variables (matching the names of the provided as data in the PLN function), assuming the interest in testing the model.
level Optional integer value the level to be used in obtaining the predictions. Level zero corresponds to the population predictions (default if responses is not provided) while level one (default) corresponds to predictions after evaluating the variational parameters for the new data.
type The type of prediction required. The default is on the scale of the linear predictors (i.e. log average count)

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

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

prior = NULL,
control = PLN_param(backend = "nlopt"),

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.

Value

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

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myLDA <- PLNLDA(Abundance ~ 0 + offset(log(Offset)),
       grouping = Group,
       data = trichoptera)

## Not run:
post_probs <- predict(myLDA, newdata = trichoptera, type = "posterior", scale = "prob")
head(round(post_probs, digits = 3))
predicted_group <- predict(myLDA, newdata = trichoptera, type = "response")
table(predicted_group, trichoptera$Group, dnn = c("predicted", "true"))

## End(Not run)
Description

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

Usage

```r
## 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 = PLNmixture_param(),
  ...)
```

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-like structure for controlling the fit. See `PLNmixture_param()` 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

```r
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLNmixture(Abundance ~ 1 + offset(log(Offset)),
  data = trichoptera, control = PLNmixture_param(smoothing = "none")) %>% getBestModel()
predict(myPLN, trichoptera, "posterior")
predict(myPLN, trichoptera, "position")
predict(myPLN, trichoptera, "response")
```
**predict_cond**

**Description**

Predict counts conditionally on a (set of) observed variables.

**Usage**

```r
predict_cond(
  object,
  newdata,
  cond_responses,
  type = c("link", "response"),
  var_par = FALSE
)
```

## S3 method for class 'PLNfit'

```r
predict_cond(
  object,
  newdata,
  cond_responses,
  type = c("link", "response"),
  var_par = FALSE
)
```

**Arguments**

- `object`: an R6 object with class `PLNfit`
- `newdata`: A data frame in which to look for variables and offsets with which to predict
- `cond_responses`: a data frame containing the counts of the observed variables (matching the names provided as data in the PLN function)
- `type`: The type of prediction required. The default is on the scale of the linear predictors (i.e. log average count)
- `var_par`: Boolean. Should new estimations of the variational parameters of mean and variance be sent back, as attributes of the matrix of predictions. Default to FALSE.

**Value**

A list containing:

- `pred`: A matrix of predicted log-counts (if `type = "link"`) or predicted counts (if `type = "response"`)
- `M`: A matrix containing E(Z_uncond | Y_c) for each given site.
- `S`: A matrix containing Var(Z_uncond | Y_c) for each given site (sites are the third dimension of the array)
prepare_data

Methods (by class)

- `predict_cond(PLNfit)`: Predict counts of a new sample conditionally on a (set of) observed variables for a `PLNfit`

Examples

data(trichoptera)
trichoptera_prep <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ Temperature + Wind, trichoptera_prep)
# Condition on the set of the first two species in the dataset (Hym, Hys) at the ten first sites
Yc <- trichoptera$Abundance[1:10, c(1, 2), drop=FALSE]
newX <- cbind(1, trichoptera$Covariate[1:10, c("Temperature", "Wind")])
pred <- predict_cond(myPLN, newX, Yc, type = "response")

`prepare_data` Prepare data for use in PLN models

Description

Prepare data in proper format for use in PLN model and its variants. The function (i) merges a count table and a covariate data frame in the most comprehensive way and (ii) computes offsets from the count table using one of several normalization schemes (TSS, CSS, RLE, GMPR, 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 = "GMPR",
  scale = "count"
)
proper_data$Abundance
proper_data$Offset

rPLN

PLN RNG

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

```r
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, with an attribute "offsets" corresponding to the true generated offsets (in log-scale).

Examples

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

---

**sigma.PLNfit**

*Extract variance-covariance of residuals 'Sigma'*

Description

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

\[ \Sigma \]

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

## 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

 coef.PLNfit(), standard_error.PLNfit() and vcov.PLNfit() 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)
sigma(myPLN) ## Sigma

sigma.PLNmixturefit

Extract variance-covariance of residuals 'Sigma'

Description

Extract the variance-covariance matrix of the residuals, usually noted \( \Sigma \)

in PLN models. This captures the correlation between the species in the latent space. or PLN mixture, 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
stability_selection

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

See Also

\code{coef.PLNmixturefit()} for other ways to access \( \Sigma \).

Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLNmixture(Abundance ~ 1 + offset(log(Offset)),
                      data = trichoptera, control = PLNmixture_param(smoothing = "none")) %>% getBestModel()
sigma(myPLN) ## Sigma

---

\textbf{stability_selection} \hspace{1cm} \textit{Compute the stability path by stability selection}

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

Usage

\begin{verbatim}
stability_selection(
  Robject, 
  subsamples = NULL, 
  control = PLNnetwork_param(), 
  force = FALSE 
)
\end{verbatim}

Arguments

\begin{itemize}
  \item \code{Robject} an object with class \code{PLNnetworkfamily}, i.e. an output from \code{PLNnetwork()}
  \item \code{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 \( 0.8 \times n \) otherwise following Liu et al. (2010) recommendations.
  \item \code{control} a list controlling the main optimization process in each call to \code{PLNnetwork}. See \code{PLNnetwork()} for details.
  \item \code{force} force computation of the stability path, even if a previous one has been detected.
\end{itemize}
standard_error.PLNPCAfit

Value

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

Examples

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

standard_error.PLNPCAfit

Component-wise standard errors of B

Description

Extracts univariate standard errors for the estimated coefficient of B. Standard errors are computed from the (approximate) Fisher information matrix.

Usage

## S3 method for class 'PLNPCAfit'
standard_error(
  object,
  type = c("variational", "jackknife", "sandwich"),
  parameter = c("B", "Omega")
)

standard_error(
  object,
  type = c("variational", "jackknife", "sandwich"),
  parameter = c("B", "Omega")
)

## S3 method for class 'PLNfit'
standard_error(
  object,
  type = c("variational", "jackknife", "bootstrap", "sandwich"),
  parameter = c("B", "Omega")
)
## S3 method for class 'PLNPCAfit'
standard_error(
  object,
  type = c("variational", "jackknife", "bootstrap", "sandwich"),
  parameter = c("B", "Omega")
)

## S3 method for class 'PLNfit'
standard_error(
  object,
  type = c("variational", "jackknife", "sandwich"),
  parameter = c("B", "Omega")
)

## S3 method for class 'PLNfit_fixedcov'
standard_error(
  object,
  type = c("variational", "jackknife", "sandwich"),
  parameter = c("B", "Omega")
)

## S3 method for class 'PLNmixturefit'
standard_error(
  object,
  type = c("variational", "jackknife", "sandwich"),
  parameter = c("B", "Omega")
)

## S3 method for class 'PLNnetworkfit'
standard_error(
  object,
  type = c("variational", "jackknife", "sandwich"),
  parameter = c("B", "Omega")
)

### Arguments

- **object**: an R6 object with class PLNfit
- **type**: string describing the type of variance approximation: "variational", "jackknife", "sandwich" (only for fixed covariance). Default is "variational".
- **parameter**: string describing the target parameter: either B (regression coefficients) or Omega (inverse residual covariance)

### Value

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

### Methods (by class)

- standard_error(PLNPCAfit): Component-wise standard errors of B in PLNPCAfit (not implemented yet)
- standard_error(PLNfit): Component-wise standard errors of B in PLNfit
- standard_error(PLNfit_fixedcov): Component-wise standard errors of B in PLNfit_fixedcov
- standard_error(PLNmixturefit): Component-wise standard errors of B in PLNmixturefit (not implemented yet)
- standard_error(PLNnetworkfit): Component-wise standard errors of B in PLNnetworkfit (not implemented yet)

### See Also

vcov.PLNfit() for the complete variance covariance estimation of the coefficient
Examples

data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- PLN(Abundance ~ 1 + offset(log(Offset)), data = trichoptera,
control = PLN_param(config_post = list(variational_var = TRUE)))
standard_error(myPLN)

trichoptera  

Trichoptera data set

Description

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

Usage

trichoptera

Format

A list with 2 two data frames:

Abundance  a 49 x 17 matrix of abundancies/counts (49 trapping nights and 17 trichoptera species)
Covariate  a 49 x 7 data frame of covariates:

Temperature  Evening Temperature in Celsius
Wind  Wind in m/s
Pressure  Pressure in mm Hg
Humidity  relative to evening humidity in percent
Cloudiness  proportion of sky coverage at 9pm
Precipitation  Nighttime precipitation in mm
Group  a factor of 12 levels for the definition of the consecutive night groups

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

Source

Data from P. Usseglio-Polatera.
vcov.PLNfit

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

\[ B \]

, 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

\[ B \]

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 \( B \) 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, type = "covariance")  ## Sigma
Index

* datasets
  barents, 3
  mollusk, 14
  oaks, 15
  trichoptera, 94

barents, 3
coef(), 20
coef.PLNfit, 4
coef.PLNfit(), 90, 95, 96
coef.PLNLDAfit, 5
coef.PLNmixturefit, 6
coef.PLNmixturefit(), 91
coefficient_path, 7
compute_offset, 7
compute_offset(), 87, 88
compute_PLN_starting_point, 9

eextract_probs, 10
f factoextra::fviz(), 62
fitted.PLNfit, 11
fitted.PLNmixturefit, 12

g getBestModel
  (getBestModel.PLNPCAfamily), 12
g getBestModel(), 42, 51, 60
g getBestModel.PLNPCAfamily, 12
g getModel(getModel.PLNPCAfamily), 13
g getModel(), 20, 42, 51, 60
g getModel.PLNPCAfamily, 13
ggplot, 35, 36, 44, 47, 53, 54, 67, 68, 77, 82
ggplot2, 19, 44, 62, 75
grob, 36, 68

mollusk, 14
oaks, 15
PLN, 17

PLN(), 4, 6, 10, 11, 20, 37, 66, 84, 87, 95
PLN_param, 71
PLN_param(), 17, 21, 23, 32, 49, 50, 59
PLNfamily, 18, 18, 19, 64–67, 74
PLNfit, 4, 11, 17–20, 21, 22, 26–28, 31, 38, 54, 63, 83, 86, 87, 90, 93, 95
PLNfit(), 20, 25, 33, 37
PLNfit_diagonal, 25
PLNfit_fixedcov, 28, 93
PLNfit_spherical, 30
PLNLDA, 31, 37
PLNLDA(), 5, 20, 25, 33, 37
PLNLDA_param, 39
PLNLDAfit, 25, 33, 33, 34–37, 67, 84
PLNLDAfit(), 32
PLNLDAfit_diagonal, 37
PLNLDAfit_spherical(PLNfit_diagonal), 25
PLNmixture, 41, 44, 45, 48
PLNmixture(), 12, 20, 42
PLNmixture_param, 49
PLNmixture_param(), 41, 42, 47, 85
PLNmixture_family, 13, 14, 42, 42, 48, 76
PLNmixturefit, 6, 12, 42, 44, 45, 46, 77, 85, 90, 93
PLNmodels::PLNfamily, 42, 51, 61
PLNmodels::PLNfit, 26, 28, 30, 33, 38, 54, 63
PLNmodels::PLNfit_fixedcov, 54
PLNmodels::PLNLDAfit, 26, 38
PLNnetwork, 50
PLNnetwork(), 7, 10, 20, 51, 53, 54, 58, 91
PLNnetwork_param, 58
PLNnetwork_param(), 50, 51
PLNnetwork_family, 7, 10, 13, 14, 18, 51, 51, 53, 58, 78, 91, 92
PLNnetworkfit, 13, 14, 51, 52, 54, 54, 55, 56, 79, 93
PLNPCA, 59, 69
PLNPCA(), 20, 60, 62, 63
PLNPCA_param, 69
PLNPCA_param(), 60
PLNPCAfamily, 13, 14, 18, 60, 60, 60, 80
PLNPCAfit, 13, 14, 60, 62, 63, 63, 64, 65, 68, 
74, 81, 93
PLNPCAfit(), 62
plot(), 25, 33, 37, 42, 51, 54, 60, 63
plot.PLNfamily, 73
plot.PLNLDAfit, 74
plot.PLNmixurefamily, 76
plot.PLNmiurefit, 77
plot.PLNnetworkfamily, 78
plot.PLNnetworkfamily(), 74
plot.PLNnetworkfit, 79
plot.PLNPCAfamily, 80
plot.PLNPCAfamily(), 74
plot.PLNPCAfit, 81
predict(), 20, 25, 33, 37
predict.PLNfit, 82
predict.PLNLDAfit, 83
predict.PLNmiurefit, 85
predict_cond, 86
prepare_data, 87
prepare_data(), 15, 16, 94, 95

rPLN, 88

sigma(), 20
sigma.PLNfit, 89
sigma.PLNfit(), 5, 95, 96
sigma.PLNmiurefit, 90
sigma.PLNmiurefit(), 6
stability_selection, 91
stability_selection(), 10, 13
standard_error

(standard_error.PLNPCAfit), 92
standard_error(), 20
standard_error.PLNfit(), 5, 90, 96
standard_error.PLNPCAfit, 92

trichoptera, 94

vcov(), 20
vcov.PLNfit, 95
vcov.PLNfit(), 5, 90, 93