

# Package: PLNmodels (via r-universe)

October 2, 2026

**Title** Poisson Lognormal Models

**Version** 1.3.2.9550

**Description** The Poisson-lognormal model and variants (Chiquet, Mariadassou and Robin, 2021 <[doi:10.3389/fevo.2021.588292](https://doi.org/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)

**Depends** R (>= 4.1.0)

**Imports** cli, corplot, dplyr, ggplot2, grDevices, grid, gridExtra, igraph, magrittr, MASS, Matrix, methods, nloptr, parallel, purrr, R6, Rcpp (>= 1.0.10), rlang, stats, tidyr, torch

**Suggests** factoextra, glassoFast, knitr, pheatmap, rmarkdown, spelling, quarto, testthat

**LinkingTo** nloptr, Rcpp (>= 1.0.10), RcppArmadillo

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**Roxygen** list(markdown = TRUE)

**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'  
 'PLNnetworkfamily-S3methods.R' 'PLNnetworkfamily-class.R'  
 'PLNnetworkfit-S3methods.R' 'PLNnetworkfit-class.R'  
 'RcppExports.R' 'ZIPLNfit-class.R' 'ZIPLN.R'  
 'ZIPLNfit-S3methods.R' 'ZIPLNnetwork.R' 'barents.R'  
 'graphical\_lasso.R' 'import\_utils.R' 'microcosm.R' 'mollusk.R'  
 'oaks.R' 'plot\_utils.R' 'scRNA.R' 'trichoptera.R'  
 'utils-pipe.R' 'utils-zipln.R' 'utils.R' 'zzz.R'

**Config/roxygen2/version** 8.1.0

**Config/pak/sysreqs** cmake libglpk-dev libicu-dev libxml2-dev

**Repository** <https://pln-team.r-universe.dev>

**Date/Publication** 2026-10-02 11:00:34 UTC

**RemoteUrl** <https://github.com/pln-team/plnmodels>

**RemoteRef** HEAD

**RemoteSha** 374dbc8771f72378caa8b5f2c3b72c10c833ed87

## Contents

|                                        |    |
|----------------------------------------|----|
| AIC.PLNfit . . . . .                   | 4  |
| AIC.ZIPLNfit . . . . .                 | 5  |
| barents . . . . .                      | 5  |
| BIC.PLNfit . . . . .                   | 6  |
| BIC.ZIPLNfit . . . . .                 | 7  |
| coef.PLNfit . . . . .                  | 7  |
| coef.PLNLDAfit . . . . .               | 8  |
| coef.PLNmixturefit . . . . .           | 9  |
| coef.ZIPLNfit . . . . .                | 10 |
| coefficient_path . . . . .             | 11 |
| compute_offset . . . . .               | 11 |
| compute_PLN_starting_point . . . . .   | 13 |
| compute_ZIPLN_starting_point . . . . . | 14 |
| extract_probs . . . . .                | 15 |
| fitted.PLNfit . . . . .                | 16 |
| fitted.PLNmixturefit . . . . .         | 17 |
| fitted.ZIPLNfit . . . . .              | 17 |
| getBestModel.PLNPCAfamily . . . . .    | 18 |
| getModel.PLNPCAfamily . . . . .        | 19 |
| graphical_lasso . . . . .              | 20 |
| ICL . . . . .                          | 23 |
| logLik.PLNfit . . . . .                | 24 |
| logLik.ZIPLNfit . . . . .              | 24 |
| microcosm . . . . .                    | 25 |
| mollusk . . . . .                      | 26 |
| Networkfamily . . . . .                | 27 |
| oaks . . . . .                         | 30 |

|                          |     |
|--------------------------|-----|
| PLN                      | 31  |
| PLN_param                | 32  |
| PLNfamily                | 35  |
| PLNfit                   | 37  |
| PLNfit_diagonal          | 43  |
| PLNfit_fixedcov          | 44  |
| PLNfit_genpop            | 46  |
| PLNfit_spherical         | 47  |
| PLNLDA                   | 48  |
| PLNLDA_param             | 49  |
| PLNLDAfit                | 52  |
| PLNLDAfit_diagonal       | 57  |
| PLNLDAfit_spherical      | 58  |
| PLNmixture               | 60  |
| PLNmixture_param         | 61  |
| PLNmixturefamily         | 64  |
| PLNmixturefit            | 66  |
| PLNnetwork               | 70  |
| PLNnetwork_param         | 71  |
| PLNnetworkfamily         | 76  |
| PLNnetworkfit            | 78  |
| PLNPCA                   | 81  |
| PLNPCA_param             | 82  |
| PLNPCAfamily             | 85  |
| PLNPCAfit                | 88  |
| plot.Networkfamily       | 94  |
| plot.PLNfamily           | 96  |
| plot.PLNLDAfit           | 97  |
| plot.PLMixturefamily     | 98  |
| plot.PLMixturefit        | 99  |
| plot.PLNnetworkfit       | 100 |
| plot.PLNPCAfamily        | 101 |
| plot.PLNPCAfit           | 102 |
| plot.ZIPLNfit_sparse     | 104 |
| predict.PLNfit           | 105 |
| predict.PLNLDAfit        | 106 |
| predict.PLMixturefit     | 107 |
| predict.ZIPLNfit         | 108 |
| predict_cond             | 109 |
| prepare_data             | 110 |
| rPLN                     | 112 |
| scRNA                    | 113 |
| sigma.PLNfit             | 114 |
| sigma.PLMixturefit       | 115 |
| sigma.ZIPLNfit           | 116 |
| stability_selection      | 116 |
| standard_error.PLNPCAfit | 117 |
| structural_zeros         | 119 |

|                              |     |
|------------------------------|-----|
| trichoptera . . . . .        | 120 |
| vcov.PLNfit . . . . .        | 121 |
| ZIPLN . . . . .              | 122 |
| ZIPLN_param . . . . .        | 124 |
| ZIPLNfit . . . . .           | 126 |
| ZIPLNfit_diagonal . . . . .  | 130 |
| ZIPLNfit_fixed . . . . .     | 131 |
| ZIPLNfit_sparse . . . . .    | 132 |
| ZIPLNfit_spherical . . . . . | 135 |
| ZIPLNnetwork . . . . .       | 136 |
| ZIPLNnetwork_param . . . . . | 137 |
| ZIPLNnetworkfamily . . . . . | 140 |

|              |            |
|--------------|------------|
| <b>Index</b> | <b>142</b> |
|--------------|------------|

---

|            |                                                            |
|------------|------------------------------------------------------------|
| AIC.PLNfit | <i>Akaike Information Criterion for a fitted PLN model</i> |
|------------|------------------------------------------------------------|

---

## Description

Computes the variational AIC as  $\text{loglik} - \text{nb\_param}$  (larger is better). This follows the maximization convention used throughout PLNmodels.

## Usage

```
## S3 method for class 'PLNfit'
AIC(object, ..., k = 2)
```

## Arguments

|        |                                                      |
|--------|------------------------------------------------------|
| object | an R6 object with class <code>PLNfit</code>          |
| ...    | additional parameters for S3 compatibility. Not used |
| k      | not used, present for S3 compatibility.              |

## Value

A scalar: the variational AIC (larger is better).

## Examples

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
model <- PLN(Abundance ~ 1, data = trichoptera)
AIC(model)
```

AIC.ZIPLNfit

*Akaike Information Criterion for a fitted ZIPLN model***Description**

Computes the variational AIC as  $\text{loglik} - \text{nb\_param}$  (larger is better). This follows the maximization convention used throughout PLNmodels.

**Usage**

```
## S3 method for class 'ZIPLNfit'
AIC(object, ..., k = 2)
```

**Arguments**

|        |                                                      |
|--------|------------------------------------------------------|
| object | an R6 object with class <a href="#">ZIPLNfit</a>     |
| ...    | additional parameters for S3 compatibility. Not used |
| k      | not used, present for S3 compatibility.              |

**Value**

A scalar: the variational AIC (larger is better).

**Examples**

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
model <- ZIPLN(Abundance ~ 1, data = trichoptera)
AIC(model)
```

barents

*Barents fish data set***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 89 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**

Fossheim, Maria, Einar M. Nilssen, and Michaela Aschan. "Fish assemblages in the Barents Sea." Marine Biology Research 2.4 (2006). doi:[10.1080/17451000600815698](https://doi.org/10.1080/17451000600815698)

**Examples**

```
data(barents)
```

---

BIC.PLNfit

*Bayesian Information Criterion for a fitted PLN model*

---

**Description**

Computes the variational BIC as  $\text{loglik} - 0.5 * \log(n) * \text{nb\_param}$  (larger is better). This follows the maximization convention used throughout PLNmodels.

**Usage**

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

**Arguments**

|        |                                                      |
|--------|------------------------------------------------------|
| object | an R6 object with class <a href="#">PLNfit</a>       |
| ...    | additional parameters for S3 compatibility. Not used |

**Value**

A scalar: the variational BIC (larger is better).

**Examples**

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
model <- PLN(Abundance ~ 1, data = trichoptera)
BIC(model)
```

BIC.ZIPLNfit

*Bayesian Information Criterion for a fitted ZIPLN model***Description**

Computes the variational BIC as  $\text{loglik} - 0.5 * \log(n) * \text{nb\_param}$  (larger is better). This follows the maximization convention used throughout PLNmodels.

**Usage**

```
## S3 method for class 'ZIPLNfit'
BIC(object, ...)
```

**Arguments**

`object` an R6 object with class [ZIPLNfit](#)  
`...` additional parameters for S3 compatibility. Not used

**Value**

A scalar: the variational BIC (larger is better).

**Examples**

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
model <- ZIPLN(Abundance ~ 1, data = trichoptera)
BIC(model)
```

coef.PLNfit

*Extract model coefficients***Description**

Extracts model coefficients from objects returned by [PLN\(\)](#) and its variants

**Usage**

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

Arguments

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

$B$

or "covariance" for  

$\Sigma$

  
...               additional parameters for S3 compatibility. Not used

Value

A matrix of coefficients extracted from the PLNfit model.

See Also

[sigma.PLNfit\(\)](#), [vcov.PLNfit\(\)](#), [standard\\_error.PLNfit\(\)](#)

Examples

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

---

|                |                                                                                      |
|----------------|--------------------------------------------------------------------------------------|
| coef.PLNLDAfit | <i>Extracts model coefficients from objects returned by <a href="#">PLNLDA()</a></i> |
|----------------|--------------------------------------------------------------------------------------|

---

Description

The method for objects returned by [PLNLDA\(\)](#) only returns coefficients associated to the

$$\Theta$$

part of the model (see the PLNLDA vignette for mathematical details).

Usage

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

Arguments

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



## 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()
coef(myPLN) ## Theta - empty here
coef(myPLN, type = "mixture") ## pi
coef(myPLN, type = "means") ## mu
coef(myPLN, type = "covariance") ## Sigma
```

---

|               |                                   |
|---------------|-----------------------------------|
| coef.ZIPLNfit | <i>Extract model coefficients</i> |
|---------------|-----------------------------------|

---

## Description

Extracts model coefficients from objects returned by [ZIPLN\(\)](#) and its variants

## Usage

```
## S3 method for class 'ZIPLNfit'
coef(object, type = c("count", "zero", "precision", "covariance"), ...)
```

## Arguments

|        |                                                                                                                                                        |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| object | an R6 object with class <a href="#">ZIPLNfit</a>                                                                                                       |
| type   | type of parameter that should be extracted. Either "count" (default) for $B$ , "zero" for $B_0$ , "precision" for $\Omega$ , "covariance" for $\Sigma$ |
| ...    | additional parameters for S3 compatibility. Not used                                                                                                   |

## Value

A matrix of coefficients extracted from the ZIPLNfit model.

## See Also

[sigma.ZIPLNfit\(\)](#)

## Examples

```
data(scRNA)
# data subsample: only 100 random cell and the 50 most varying transcript
subset <- sample.int(nrow(scRNA), 100)
myPLN <- ZIPLN(counts[, 1:50] ~ 1 + offset(log(total_counts)), subset = subset, data = scRNA)
```

---

|                  |                                                            |
|------------------|------------------------------------------------------------|
| coefficient_path | <i>Extract the regularization path of a PLNnetwork fit</i> |
|------------------|------------------------------------------------------------|

---

**Description**

Extract the regularization path of a PLNnetwork fit

**Usage**

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

**Arguments**

|           |                                                                                                                                |
|-----------|--------------------------------------------------------------------------------------------------------------------------------|
| Robject   | an object with class <a href="#">Networkfamily</a> , i.e. an output from <a href="#">PLNnetwork()</a>                          |
| precision | a logical, should the coefficients of the precision matrix Omega or the covariance matrix Sigma be sent back. Default is TRUE. |
| corr      | a logical, should the correlation (partial in case precision = TRUE) be sent back. Default is TRUE.                            |

**Value**

Sends back a tibble/data.frame.

**Examples**

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

---

|                |                                                                                     |
|----------------|-------------------------------------------------------------------------------------|
| compute_offset | <i>Compute offsets from a count data using one of several normalization schemes</i> |
|----------------|-------------------------------------------------------------------------------------|

---

**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**

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

## 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

- Chen, L., Reeve, J., Zhang, L., Huang, S., Wang, X. and Chen, J. (2018) GMPR: A robust normalization method for zero-inflated count data with application to microbiome sequencing data. *PeerJ*, 6, e4600 [doi:10.7717/peerj.4600](https://doi.org/10.7717/peerj.4600)
- Paulson, J. N., Colin Stine, O., Bravo, H. C. and Pop, M. (2013) Differential abundance analysis for microbial marker-gene surveys. *Nature Methods*, 10, 1200-1202 [doi:10.1038/nmeth.2658](https://doi.org/10.1038/nmeth.2658)
- Anders, S. and Huber, W. (2010) Differential expression analysis for sequence count data. *Genome Biology*, 11, R106 [doi:10.1186/gb20101110r106](https://doi.org/10.1186/gb20101110r106)

Kumar, M., Slud, E., Okrah, K. et al. (2018) Analysis and correction of compositional bias in sparse sequencing count data. BMC Genomics 19, 799 doi:[10.1186/s12864-018-5160-5](https://doi.org/10.1186/s12864-018-5160-5)

Robinson, M.D., Oshlack, A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biol 11, R25 doi:[10.1186/gb2010113r25](https://doi.org/10.1186/gb2010113r25)

## 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 S2 when fitting a PLN. Mostly intended for internal use.

## Usage

```
compute_PLN_starting_point(Y, X, O, w, method = c("LM", "GLM"))
```

## Arguments

|        |                                                                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Y      | Response count matrix                                                                                                                                                                                   |
| X      | Covariate matrix. Note that initialization will fail if the model matrix is singular.                                                                                                                   |
| O      | Offset matrix (in log-scale)                                                                                                                                                                            |
| w      | Weight vector (defaults to 1)                                                                                                                                                                           |
| method | character: strategy used to initialize B. Either "LM" (default, fast weighted log-linear regression) or "GLM" (p independent Poisson GLMs, more accurate for complex or unbalanced designs but slower). |

## Details

- **B**: estimated by weighted LM (method = "LM", default) or p independent Poisson GLMs (method = "GLM"). The GLM option gives better B estimates for factorial or unbalanced designs at the cost of p IRLS fits.
- **M**: initialized to  $\log((1 + Y) / \exp(0))$  ( $M_{full}$  in the  $X*B + M_{res}$  parameterization).
- **S**: initialized element-wise to  $1 / \sqrt{2 + Y}$ , the approximate VE-step optimum at  $\Omega = I$ . This adapts automatically to count levels: high S for zero counts (high uncertainty), low S for large counts.

## Value

a named list of starting values for model parameter B and variational parameters M and S2 used in the iterative optimization algorithm of [PLN\(\)](#)

## Examples

```
## 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)
compute_PLN_starting_point(Y, X, O, w, method = "GLM")

## End(Not run)
```

---

```
compute_ZIPLN_starting_point
```

*Helper function for ZIPLN initialization.*

---

## Description

Fast LM-based starting point for ZIPLN: one multivariate `lm.fit` for the PLN component and empirical zero rates / binomial GLMs for the ZI component. Replaces the previous per-species `pscl::zeroinfl` loop. R is set to the empirical zero rate of each species on its zero counts, and to 0 elsewhere.

## Usage

```
compute_ZIPLN_starting_point(Y, X, X0, O, w = NULL)
```

**Arguments**

|    |                                                                                                     |
|----|-----------------------------------------------------------------------------------------------------|
| Y  | Response count matrix ( $n \times p$ )                                                              |
| X  | Design matrix for the PLN component ( $n \times d$ )                                                |
| X0 | Design matrix for the ZI component ( $n \times d0$ , empty <code>matrix(NA,0,0)</code> when unused) |
| O  | Offset matrix in log-scale ( $n \times p$ )                                                         |
| w  | Weight vector of length $n$ (defaults to uniform weights)                                           |

**Value**

Named list: B ( $d \times p$ ), M ( $n \times p$ ), S2 ( $n \times p$ ), R ( $n \times p$ ), B0 ( $d0 \times p$ )

---

|               |                                                                 |
|---------------|-----------------------------------------------------------------|
| extract_probs | <i>Extract edge selection frequency in bootstrap subsamples</i> |
|---------------|-----------------------------------------------------------------|

---

**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**

```
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 <code>PLNnetworkfamily</code> , i.e. an output from <code>PLNnetwork()</code>                                                                                                                                                                                                                                                                                  |
| 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 <code>stability_selection()</code> with default argument. |
| format  | output format. Either a matrix (default) or a named vector.                                                                                                                                                                                                                                                                                                                         |
| tol     | tolerance for rounding error when comparing penalties.                                                                                                                                                                                                                                                                                                                              |

**Value**

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

**Examples**

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

## End(Not run)

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

## End(Not run)
```

---

|               |                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------|
| fitted.PLNfit | <i>Extracts model fitted values from objects returned by <a href="#">PLN()</a> and its variants</i> |
|---------------|-----------------------------------------------------------------------------------------------------|

---

**Description**

Extracts model fitted values from objects returned by [PLN\(\)](#) and its variants

**Usage**

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

**Arguments**

|        |                                                      |
|--------|------------------------------------------------------|
| object | an R6 object with class <a href="#">PLNfit</a>       |
| ...    | additional parameters for S3 compatibility. Not used |

**Value**

A matrix of Fitted values extracted from the object object.

---

|                      |                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------|
| fitted.PLNmixturefit | <i>Extracts model fitted values from objects returned by <a href="#">PLNmixture()</a> and its variants</i> |
|----------------------|------------------------------------------------------------------------------------------------------------|

---

**Description**

Extracts model fitted values from objects returned by [PLNmixture\(\)](#) and its variants

**Usage**

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

**Arguments**

|        |                                                       |
|--------|-------------------------------------------------------|
| object | an R6 object with class <a href="#">PLNmixturefit</a> |
| ...    | additional parameters for S3 compatibility. Not used  |

**Value**

A matrix of Fitted values extracted from the object object.

---

|                 |                                                                                                       |
|-----------------|-------------------------------------------------------------------------------------------------------|
| fitted.ZIPLNfit | <i>Extracts model fitted values from objects returned by <a href="#">ZIPLN()</a> and its variants</i> |
|-----------------|-------------------------------------------------------------------------------------------------------|

---

**Description**

Extracts model fitted values from objects returned by [ZIPLN\(\)](#) and its variants

**Usage**

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

**Arguments**

|        |                                                      |
|--------|------------------------------------------------------|
| object | an R6 object with class <a href="#">ZIPLNfit</a>     |
| ...    | 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

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

getBestModel(Robject, crit, ...)

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

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

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

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

## Arguments

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

## Value

Send back an object with class [PLNPcAfit](#) or [PLNnetworkfit](#)

## Methods (by class)

- `getBestModel(PLNPcAfamily)`: Model extraction for [PLNPcAfamily](#)

- `getBestModel(PLNmixturefamily)`: Model extraction for [PLNmixturefamily](#)
- `getBestModel(Networkfamily)`: Model extraction for [PLNnetworkfamily](#) or [ZIPLNnetworkfamily](#)
- `getBestModel(PLNnetworkfamily)`: Model extraction for [PLNnetworkfamily](#)
- `getBestModel(ZIPLNnetworkfamily)`: Model extraction for [ZIPLNnetworkfamily](#)

## Examples

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

## End(Not run)
```

---

`getModel.PLNPCAfamily` *Model extraction from a collection of models*

---

## Description

Model extraction from a collection of models

## Usage

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

getModel(Robject, var, index)

## S3 method for class 'PLNmixturefamily'
getModel(Robject, var, index = NULL)

## S3 method for class 'Networkfamily'
getModel(Robject, var, index = NULL)

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

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

## Arguments

|                      |                                                                                                                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Robject</code> | an R6 object with class <a href="#">PLNPCAfamily</a> or <a href="#">PLNnetworkfamily</a>                                                                                                                                        |
| <code>var</code>     | 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. |
| <code>index</code>   | 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(Networkfamily)`: Model extraction for `PLNnetworkfamily` or `ZIPLNnetworkfamily`
- `getModel(PLNnetworkfamily)`: Model extraction for `PLNnetworkfamily`
- `getModel(ZIPLNnetworkfamily)`: Model extraction for `ZIPLNnetworkfamily`

**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)
```

---

graphical\_lasso

*Graphical Lasso*


---

**Description**

Sparse estimation of a precision matrix by the graphical Lasso, that is the minimization over positive definite matrices  $\Theta$  of

$$-\log \det(\Theta) + \text{tr}(S\Theta) + \|\rho \circ \Theta\|_1.$$

This is the solver used internally by `PLNnetwork()` and `ZIPLNnetwork()`.

**Usage**

```
graphical_lasso(
  S,
  rho,
  thr = 1e-04,
  maxit = 10000L,
  w_init = NULL,
  wi_init = NULL,
  trace = FALSE,
  stall_patience = 1000L
)
```

## Arguments

|                              |                                                                                                                                                                                                                                                                                                                                             |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>S</code>               | a symmetric $p \times p$ (empirical) covariance matrix.                                                                                                                                                                                                                                                                                     |
| <code>rho</code>             | the penalty: either a non-negative scalar, applied to all entries (the diagonal included), or a symmetric $p \times p$ matrix of non-negative per-entry penalties (e.g. with a zero diagonal to leave it unpenalized).                                                                                                                      |
| <code>thr</code>             | convergence threshold, relative to the average absolute off-diagonal entry of $S$ scaled to a unit diagonal (see Details). Default is $1e-4$ , as in <b>glassoFast</b> .                                                                                                                                                                    |
| <code>maxit</code>           | maximal number of outer sweeps. Default is $10000$ , as in <b>glassoFast</b> .                                                                                                                                                                                                                                                              |
| <code>w_init, wi_init</code> | optional warm start: the $w$ and $w_i$ of a previous solve, typically at a nearby penalty along a regularization path. Both must be given, with the dimensions of $S$ , to be used. Note that a warm start stops closer to the starting point than a cold one at the same <code>thr</code> , so it does not reproduce a cold solve exactly. |
| <code>trace</code>           | if TRUE, also return the per-sweep convergence criterion, to diagnose a solve that does not converge. Default is FALSE.                                                                                                                                                                                                                     |
| <code>stall_patience</code>  | number of consecutive sweeps without progress after which the algorithm concludes that it is cycling and stops (see Details). Inf disables the detection. Default is $1000$ .                                                                                                                                                               |

## Details

The algorithm is the block coordinate descent of Friedman, Hastie and Tibshirani (2008), in the implementation of Sustik and Calderhead (2012): the code is a C++ port of the Fortran routine of the **glassoFast** package. It first puts the problem on a unit diagonal: for any positive diagonal matrix  $D$ ,  $\Theta$  solves the problem for  $(S, \rho)$  if and only if  $D^{-1}\Theta D^{-1}$  solves it for  $(DSD, D\rho D)$ , and  $D = \text{diag}(S + \rho)^{-1/2}$  is used. This change of variables is exact, and on ordinary input the result is the one of **glassoFast** up to the stopping rule (same support, entries within the tolerance). On a covariance matrix whose variances span orders of magnitude, as the residual covariance of `PLNnetwork()` does, it is what makes the descent converge, in a few sweeps where the unscaled one would cycle, fail in its inner loop, or return an indefinite precision matrix. It also departs from **glassoFast** on degenerate input:

- it always terminates: non-finite input, or a coordinate with  $S_{ii} + \rho_{ii} \leq 0$ , is rejected (the result is filled with NA and converged is FALSE), and the inner coordinate descent is bounded, where **glassoFast** can loop forever on a nearly collapsed covariance matrix;
- failure to converge is reported through converged rather than silently;
- it can be interrupted from R;
- when  $S$  has no off-diagonal mass, the (diagonal) solution  $1/(S_{ii} + \rho_{ii})$  is returned, where **glassoFast** returns  $1/\max(\rho_{ii}, \epsilon)$ ;
- it detects when it is cycling rather than converging, and stops.

That last point is a safeguard. Without the scaling to a unit diagonal, the sweeps could settle into a small limit cycle on an ill-conditioned  $S$ : the convergence criterion stops decreasing and oscillates just above its threshold forever, while the solution itself no longer moves. **glassoFast** spends its whole sweep budget on these and reports success regardless; here the cycle would be detected after `stall_patience` sweeps without progress, the solve stopped, and status would report "stalled".

With the scaling, the covariance matrices on which this was observed (a oaks-derived one, residual covariances along `PLNnetwork()` paths) all converge.

The remaining statuses are "max\_iter" (max it reached while still progressing), "inner\_failure" and "degenerate" (numerical trouble, the result may contain NA).

The precision matrix  $w_i$  is backed out of the regression coefficients of the algorithm, so that it is the inverse of  $w$  only at the exact solution. Short of it, on an ill-conditioned  $S$ , it can come out indefinite (whatever the status), and a caller using it as a precision matrix then diverges. Its diagonal is then shifted, which keeps the estimated network, so that its smallest eigenvalue is the one of the inverse of  $w$ . The shift is reported in `shift`, which is 0 otherwise.

## Value

a list with components

- `w`: the estimated covariance matrix,
- `wi`: the estimated precision matrix (symmetric),
- `niter`: the number of outer sweeps performed,
- `converged`: TRUE if the convergence criterion was met,
- `status`: how the solve ended, one of "converged", "stalled", "max\_iter", "inner\_failure" or "degenerate" (see Details),
- `delta`: the best value reached by the convergence criterion, relative to the threshold it had to cross. `delta <= 1` means convergence; a stalled solve typically sits between 1 and 3, that is, just short of it,
- `shift`: the value added to the diagonal of `wi` to make it positive definite, 0 when it already was (see Details),
- `dw_trace`: the per-sweep criterion when `trace = TRUE`.

## References

J. Friedman, T. Hastie and R. Tibshirani (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 432–441.

M. A. Sustik and B. Calderhead (2012). GLASSOFAST: An efficient GLASSO implementation. UTCS Technical Report TR-12-29, The University of Texas at Austin.

## Examples

```
data(trichoptera)
S <- cov(log1p(as.matrix(trichoptera$Abundance)))
fit <- graphical_lasso(S, rho = 0.1)
fit$converged
sum(fit$wi[upper.tri(fit$wi)] != 0) # number of edges

## penalty weights, leaving the diagonal unpenalized
W <- matrix(1, ncol(S), ncol(S)); diag(W) <- 0
fit <- graphical_lasso(S, rho = 0.1 * W)
```

---

|     |                                             |
|-----|---------------------------------------------|
| ICL | <i>Integrated Classification Likelihood</i> |
|-----|---------------------------------------------|

---

**Description**

Generic function to compute the Integrated Classification Likelihood (ICL) of a fitted model. ICL = BIC - entropy of the variational distribution (larger is better).

ICL.PLNfit: ICL for a fitted [PLNfit](#).

ICL.ZIPLNfit: ICL for a fitted [ZIPLNfit](#).

**Usage**

```
ICL(object, ...)  
  
## S3 method for class 'PLNfit'  
ICL(object, ...)  
  
## S3 method for class 'ZIPLNfit'  
ICL(object, ...)
```

**Arguments**

|        |                                                  |
|--------|--------------------------------------------------|
| object | an R6 object with class <a href="#">ZIPLNfit</a> |
| ...    | additional parameters passed to methods          |

**Value**

A scalar: the variational ICL (larger is better).

**Examples**

```
data(trichoptera)  
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)  
model <- PLN(Abundance ~ 1, data = trichoptera)  
ICL(model)  
data(trichoptera)  
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)  
model <- ZIPLN(Abundance ~ 1, data = trichoptera)  
ICL(model)
```

---

|               |                                                     |
|---------------|-----------------------------------------------------|
| logLik.PLNfit | <i>Extract log-likelihood of a fitted PLN model</i> |
|---------------|-----------------------------------------------------|

---

**Description**

Returns the variational lower bound of the log-likelihood as a "logLik" object, compatible with `stats::AIC()` and `stats::BIC()`.

**Usage**

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

**Arguments**

|        |                                                      |
|--------|------------------------------------------------------|
| object | an R6 object with class <code>PLNfit</code>          |
| ...    | additional parameters for S3 compatibility. Not used |

**Value**

An object of class "logLik". The numeric value is the variational ELBO. Attributes `df` and `nobs` hold the number of parameters and observations.

**Examples**

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
model <- PLN(Abundance ~ 1, data = trichoptera)
logLik(model)
```

---

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| logLik.ZIPLNfit | <i>Extract log-likelihood of a fitted ZIPLN model</i> |
|-----------------|-------------------------------------------------------|

---

**Description**

Returns the variational lower bound of the log-likelihood as a "logLik" object, compatible with `stats::AIC()` and `stats::BIC()`.

**Usage**

```
## S3 method for class 'ZIPLNfit'
logLik(object, ...)
```

**Arguments**

|        |                                                      |
|--------|------------------------------------------------------|
| object | an R6 object with class <code>ZIPLNfit</code>        |
| ...    | additional parameters for S3 compatibility. Not used |

**Value**

An object of class "logLik". The numeric value is the variational ELBO. Attributes df and nobs hold the number of parameters and observations.

**Examples**

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
model <- ZIPLN(Abundance ~ 1, data = trichoptera)
logLik(model)
```

---

microcosm

*Cow microbiome data set*


---

**Description**

This data gives the evolution of the microbiota of 45 lactating cows before and after calving in terms of counts of Amplicon Sequence Variants (ASV) in various body sites. Three body sites (vagina, mouth, nose) in addition to the milk of the 4 teats were sampled at 4 time points: 1 week before calving (except for the milk), 1 month, 3 months and 7 months after calving. We present a reduced version of the original data consisting of 880 samples with sequencing depths ranging from 1,001 to 71,881 reads per sample (see [doi:10.1186/s42523-023-00252-w](https://doi.org/10.1186/s42523-023-00252-w) for details). ASVs with prevalence in the dataset lower than 5% were filtered out and samples for which the total depth (after ASV filtering were removed), resulting in a count table of n = 880 samples with p = 259 ASV and a mean proportion of zeroes of 90.3%.

**Usage**

```
microcosm
```

**Format**

A data frame with 6 variables:

- Abundance: A 880 samples by 259 taxa count matrix
- sample: Unique sample id
- Offset: sequencing depth
- site: sampling site (O: oral; N: nasal; V: vaginal; M: milk)
- time: sampling time (-1W: 1 week before calving; 1M: 1 month after calving; 3M: 3 months after calving; 7M: 7 months after calving)
- site\_time: factor of possible pairs of (site, time). The combination M -1W is absent.

**Source**

Data from M. Mariadassou and coauthors.

## References

Mariadassou, M., Nouvel, L.X., Constant, F. et al. Microbiota members from body sites of dairy cows are largely shared within individual hosts throughout lactation but sharing is limited in the herd. *anim microbiome* 5, 32 (2023). doi:[10.1186/s42523-023-00252-w](https://doi.org/10.1186/s42523-023-00252-w)

## See Also

[prepare\\_data\(\)](#)

## Examples

```
data(microcosm)
## Not run:
my_ZIPLN <- ZIPLN(formula = Abundance ~ 0 + site + offset(log(Offset)) | 1, data = microcosm)

## 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

Richardot-Coulet, M., Chessel D. and Bournaud M. (1986) Typological value of the benthos of old beds of a large river. Methodological approach. Archiv für Hydrobiologie, 107, 363–383.

## See Also

[prepare\\_data\(\)](#)

## Examples

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

---

Networkfamily

*An R6 Class to virtually represent a collection of network fits*

---

## Description

The functions [PLNnetwork\(\)](#) and [ZIPLNnetwork\(\)](#) both produce an instance of this class, which can be thought of as a vector of [PLNnetworkfits](#) [ZIPLNfit\\_sparses](#) (indexed by penalty parameter)

This class comes with a set of methods mostly used to compare network fits (in terms of goodness of fit) or extract one from the family (based on penalty parameter and/or goodness of it). See the documentation for [getBestModel\(\)](#), [getModel\(\)](#) and [plot\(\)](#) for the user-facing ones.

## Super class

[PLNfamily](#) -> Networkfamily

## 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

`ebic_gamma` the tuning parameter gamma of the EBIC, between 0 and 1, shared by every fit of the collection. Zero gives back the BIC; the default 0.5 is the value recommended by Foygel and Drton (2010). Assign to it to change the EBIC of the whole collection, and hence the model selected by `getBestModel("EBIC")`.

`criteria` a data frame with the values of some criteria (variational 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,  $\loglik - 0.5 \text{ penalty}$

## Methods

### Public methods:

- `Networkfamily$new()`
- `Networkfamily$optimize()`
- `Networkfamily$postTreatment()`
- `Networkfamily$coefficient_path()`
- `Networkfamily$getBestModel()`
- `Networkfamily$plot()`
- `Networkfamily$plot_stars()`
- `Networkfamily$plot_objective()`
- `Networkfamily$show()`
- `Networkfamily$clone()`

`Networkfamily$new()`: Initialize all models in the collection

*Usage:*

`Networkfamily$new(penalties, data, control)`

*Arguments:*

`penalties` a vector of positive real number controlling the level of sparsity of the underlying network.

`data` a named list used internally to carry the data matrices

`control` a list for controlling the optimization.

*Returns:* Update all network fits in the family with smart starting values

`Networkfamily$optimize()`: Call to the C++ optimizer on all models of the collection

*Usage:*

`Networkfamily$optimize(data, config)`

*Arguments:*

`data` a named list used internally to carry the data matrices

`config` a list for controlling the optimization.

`Networkfamily$postTreatment()`: Update fields after optimization

*Usage:*

`Networkfamily$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

`Networkfamily$coefficient_path()`: Extract the regularization path of a [Networkfamily](#)

*Usage:*

`Networkfamily$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

**Networkfamily\$getBestModel():** Extract the best network in the family according to some criteria

*Usage:*

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

*Arguments:*

**crit** character. Criterion used to perform the selection. If "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.

*Details:* For BIC and EBIC criteria, higher is better.

**Networkfamily\$plot():** Display various outputs (goodness-of-fit criteria, robustness, diagnostic) associated with a collection of network fits (a [Networkfamily](#))

*Usage:*

```
Networkfamily$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 [ggplot2::ggplot](#) graph

**Networkfamily\$plot\_stars():** Plot stability path

*Usage:*

```
Networkfamily$plot_stars(stability = 0.9, log.x = TRUE)
```

*Arguments:*

**stability** scalar: the targeted level of stability using stability selection. Default is 0.9.  
**log.x** logical: should the x-axis be represented in log-scale? Default is TRUE.

*Returns:* a [ggplot2::ggplot](#) graph

**Networkfamily\$plot\_objective():** Plot objective value of the optimization problem along the penalty path

*Usage:*

`Networkfamily$plot_objective()`

*Returns:* a [ggplot2::ggplot](#) graph

`Networkfamily$show()`: User friendly print method

*Usage:*

`Networkfamily$show()`

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

*Usage:*

`Networkfamily$clone(deep = FALSE)`

*Arguments:*

`deep` Whether to make a deep clone.

### See Also

The functions [PLNnetwork\(\)](#), [ZIPLNnetwork\(\)](#) and the classes [PLNnetworkfit](#), [ZIPLNfit\\_sparse](#)

---

oaks

*Oaks amplicon data set*

---

### Description

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

### Usage

oaks

### Format

A data frame with 13 variables:

- Abundance: A 114 taxa by 116 samples count matrix
- Offset: A 114 taxa by 116 samples offset matrix
- Sample: Unique sample id
- tree: Tree status with respect to the pathogen (susceptible, intermediate or resistant)
- branch: Unique branch id in each tree (4 branches were sampled in each tree, with 10 leaves per branch)
- leafNO: Unique leaf id in each tree (40 leaves were sampled in each tree)
- distTObase: Distance of the sampled leaf to the base of the branch
- distTOtrunk: Distance of the sampled leaf to the base of the tree trunk
- distTOground: Distance of the sampled leaf to the base of the ground

- pmInfection: Powdery mildew infection, proportion of the upper leaf area displaying mildew symptoms
- orientation: Orientation of the branch (South-West SW or North-East NE)
- readsTOTfun: Total number of ITS1 reads for that leaf
- readsTOTbac: Total number of 16S reads for that leaf

## Source

Data from B. Jakuschkin and coauthors.

## References

Jakuschkin, B., Fievet, V., Schwaller, L. et al. Deciphering the Pathobiome: Intra- and Interkingdom Interactions Involving the Pathogen *Erysiphe alphitoides*. *Microb Ecol* 72, 870–880 (2016). [doi:10.1007/s002480160777x](https://doi.org/10.1007/s002480160777x)

## 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

```
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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the model 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**

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

---

|           |                             |
|-----------|-----------------------------|
| PLN_param | <i>Control of a PLN fit</i> |
|-----------|-----------------------------|

---

**Description**

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

**Usage**

```
PLN_param(
  backend = c("nlopt", "builtin", "torch"),
  trace = 1,
  covariance = c("full", "diagonal", "spherical", "fixed", "genpop"),
  Omega = NULL,
  C = NULL,
  config_post = list(),
  config_optim = list(),
  inception = NULL,
  init_method = c("LM", "GLM")
)
```

**Arguments**

- backend          optimization backend, either "nlopt" (default), "builtin" or "torch". "nlopt" (CCSAQ, with `config_optim$profiled = TRUE`) is consistently faster than "builtin". "builtin" is the built-in envelope-theorem Newton optimizer; it can reach a slightly better optimum, an advantage that tends to grow with  $p$  — prefer it when the last bit of ELBO matters more than speed, typically for large  $p$ . "torch" is **experimental**: it may converge to suboptimal solutions or fail on some datasets.

|              |                                                                                                                                                                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| trace        | a integer for verbosity.                                                                                                                                                                                                                                                                                                          |
| covariance   | character setting the model for the covariance matrix. Either "full", "diagonal", "spherical", "fixed" or "genpop". Default is "full".                                                                                                                                                                                            |
| Omega        | precision matrix of the latent variables. Inverse of Sigma. Must be specified if covariance is "fixed"                                                                                                                                                                                                                            |
| C            | a fixed $p \times p$ correlation matrix (e.g. a genetic relationship matrix). Must be specified if covariance is "genpop": the residual covariance is then modeled as $\text{Sigma} = \text{sigma2} * (\text{rho} * \text{C} + (1 - \text{rho}) * \text{I})$ , with sigma2 and rho estimated.                                     |
| 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. |
| init_method  | character: strategy for the starting-point computation (ignored when inception is a PLNfit). Either "LM" (default) or "GLM" (p independent Poisson GLMs, better for complex or unbalanced designs). See <a href="#">compute_PLN_starting_point()</a> .                                                                            |

## 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)
- "profiled" (full covariance only) if TRUE, profile both B and Omega at every nlopt evaluation instead of running an EM loop (Omega fixed for the duration of each inner nlopt solve, B profiled in closed form at every evaluation). Despite the extra  $O(n \times p^2 + p^3)$  cost per evaluation, benchmarks found it consistently faster than the EM loop (and with a slightly better loglik) across a range of problem sizes. Default is TRUE; set to FALSE to recover the EM loop.

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

When "builtin" backend is used, the following entries are relevant

- "maxeval" stop when the number of Newton steps in the inner loop exceeds maxeval. Default is 10000
- "ftol\_in" stop the inner loop when the objective changes by less than ftol\_in (relative). Default is 1e-8
- "maxit\_em" stop the EM outer loop when the number of EM iterations exceeds maxit\_em. Default is 50
- "ftol\_em" stop the EM outer loop when the ELBO changes by less than ftol\_em (relative). Default is 1e-8

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.

---

PLNfamily

*An R6 Class to represent a collection of PLNfit*

---

### Description

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,  $\text{loglik} - 0.5 \text{ penalty}$   
`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()`

`PLNfamily$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

`PLNfamily$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

`PLNfamily$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

`PLNfamily$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::ggplot` object

`PLNfamily$show()`: User friendly print method

*Usage:*

PLNfamily\$show()

PLNfamily\$print(): User friendly print method

*Usage:*

PLNfamily\$print()

PLNfamily\$clone(): The objects of this class are cloneable with this method.

*Usage:*

PLNfamily\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

## See Also

[getModel\(\)](#)

---

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

`degenerate_species` names of the species whose latent variance is above 100 (a standard deviation of 10 on the log scale, see `options(PLNmodels.latent_variance_threshold = )`). The zeros of such a species are fitted by latent means going to minus infinity. This happens to species that are often absent but abundant when present, notably those absent from a whole group of samples (see `structural_zeros()`), for which the model lacks the covariate that explains the absences. In a network fit, such a species ends up connected to most of the others: these edges are artefacts.

`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

`AIC` variational lower bound of the AIC

`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()`

`PLNfit$new()`: Initialize a `PLNfit` model

*Usage:*

```
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()`.

`PLNfit$update()`: Update a `PLNfit` object

*Usage:*

```

PLNfit$update(
  B = NA,
  Sigma = NA,
  Omega = NA,
  M = NA,
  S2 = 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  
**S2** 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

`PLNfit$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

`PLNfit$optimize_vestep()`: Result of one call to the VE step of the optimization procedure: optimal variational parameters (M, S2) 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.

`B` 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 [PLN\\_param\(\)](#).

`Sigma` variance-covariance matrix of the latent variables

*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

`PLNfit$postTreatment()`: Update R2, fisher and std\_err fields after optimization

*Usage:*

```
PLNfit$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 (optional bootstrap, jackknife, R2, etc.). 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:

- **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 estimator should be computed to estimate the variance of the model parameters (highly underestimated). Default is FALSE.
- **trace** integer for verbosity. should be > 1 to see output in post-treatments

**PLNfit\$predict():** Predict position, scores or observations of new data.

*Usage:*

```

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 is 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.

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

*Usage:*

```
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.

`PLNfit$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

`PLNfit$print()`: User friendly print method

*Usage:*

```
PLNfit$print()
```

PLNfit\$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 | <i>An R6 Class to represent a PLNfit in a standard, general framework, with diagonal residual covariance</i> |
|-----------------|--------------------------------------------------------------------------------------------------------------|

---

## Description

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

## Super class

```
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()`

PLNfit\_diagonal\$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.

`PLNfit_diagonal$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.

**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_fixedcov | <i>An R6 Class to represent a PLNfit in a standard, general framework, with fixed (inverse) residual covariance</i> |
|-----------------|---------------------------------------------------------------------------------------------------------------------|

---

**Description**

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

**Super class**

`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$clone()`

`PLNfit_fixedcov$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.

`PLNfit_fixedcov$optimize()`: Call to the NLOpt or TORCH optimizer and update of the relevant fields

#### *Usage:*

`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

`PLNfit_fixedcov$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)
```

---

PLNfit\_genpop

*An R6 Class to represent a PLNfit with a residual covariance structured by a fixed correlation matrix (e.g. a genetic relationship matrix), motivated by population genetics*

---

## Description

$\text{Sigma} = \text{sigma2} * (\text{rho} * \text{C} + (1 - \text{rho}) * \text{I}_p)$ , where C is a fixed  $p \times p$  correlation matrix supplied by the user (control\$C) and (sigma2, rho) are estimated. See GeneticCovTraits in src/covariance\_pln.h for the C++ side.

## Super class

`PLNfit` -> `PLNfit_genpop`

## Active bindings

`nb_param` number of parameters in the current PLN model  
`vcov_model` character: the model used for the residual covariance  
`gen_par` a list with the two extra parameters of the genpop covariance model: `sigma2` (variance scale) and `rho` (mixing weight / heritability), decoded from `Sigma` and `C`.

## Methods

### Public methods:

- `PLNfit_genpop$new()`
- `PLNfit_genpop$clone()`

`PLNfit_genpop$new()`: Initialize a `PLNfit_genpop` model

*Usage:*

`PLNfit_genpop$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, must include a field `C` (the fixed  $p \times p$  correlation matrix). See details.

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

*Usage:*

```
PLNfit_genpop$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
## Not run:
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
p <- ncol(trichoptera$Abundance)
C <- 0.5^abs(outer(1:p, 1:p, "-")); diag(C) <- 1
myPLN <- PLN(Abundance ~ 1, data = trichoptera, control = PLN_param(covariance = "genpop", C = C))
class(myPLN)
print(myPLN)

## End(Not run)
```

---

|                  |                                                                                                               |
|------------------|---------------------------------------------------------------------------------------------------------------|
| PLNfit_spherical | <i>An R6 Class to represent a PLNfit in a standard, general framework, with spherical residual covariance</i> |
|------------------|---------------------------------------------------------------------------------------------------------------|

---

## Description

An R6 Class to represent a PLNfit in a standard, general framework, with spherical residual covariance

## Super class

```
PLNfit -> PLNfit_spherical
```

## 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_spherical$new()`
- `PLNfit_spherical$clone()`

`PLNfit_spherical$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.

`PLNfit_spherical$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)
```

## Description

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

**Usage**

```
PLNLDA(formula, data, subset, weights, grouping, control = PLNLDA_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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the model is called. |
| subset   | an optional vector specifying a subset of observations to be used in the fitting process.                                                                                                                                                                                                               |
| weights  | an optional vector of observation weights to be used in the fitting process.                                                                                                                                                                                                                            |
| grouping | a factor specifying the class of each observation used for discriminant analysis.                                                                                                                                                                                                                       |
| control  | a list-like structure for controlling the optimization, with default generated by <code>PLNLDA_param()</code> . See the associated documentation.                                                                                                                                                       |

**Details**

See `PLNLDA_param()` for a full description of the optimization parameters. Note that unlike `PLN_param()`, `PLNLDA_param()` does not expose the "fixed" covariance option or the Omega parameter, which are not meaningful in the LDA context.

**Value**

an R6 object with class `PLNLDAfit()`

**See Also**

The class `PLNLDAfit`

**Examples**

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

---

PLNLDA\_param

*Control of a PLNLDA fit*

---

**Description**

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

**Usage**

```
PLNLDA_param(
  backend = c("builtin", "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)

- "profiled" (full covariance only) if TRUE, profile both B and Omega at every nlopt evaluation instead of running an EM loop (Omega fixed for the duration of each inner nlopt solve, B profiled in closed form at every evaluation). Despite the extra  $O(n \times p^2 + p^3)$  cost per evaluation, benchmarks found it consistently faster than the EM loop (and with a slightly better loglik) across a range of problem sizes. Default is TRUE; set to FALSE to recover the EM loop.

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

When "builtin" backend is used, the following entries are relevant

- "maxeval" stop when the number of Newton steps in the inner loop exceeds maxeval. Default is 10000
- "ftol\_in" stop the inner loop when the objective changes by less than ftol\_in (relative). Default is  $1e-8$
- "maxit\_em" stop the EM outer loop when the number of EM iterations exceeds maxit\_em. Default is 50
- "ftol\_em" stop the EM outer loop when the ELBO changes by less than ftol\_em (relative). Default is  $1e-8$

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.

---

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

`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()`
- `PLNLDAfit$plot_LDA()`
- `PLNLDAfit$predict()`
- `PLNLDAfit$show()`
- `PLNLDAfit$clone()`

`PLNLDAfit$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

`PLNLDAfit$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.

PLNLDAfit\$postTreatment(): Update R2, fisher and std\_err fields and visualization

*Usage:*

```
PLNLDAfit$postTreatment(
  grouping,
  responses,
  covariates,
  offsets,
  weights = rep(1, nrow(responses)),
  config_post,
  config_optim
)
```

*Arguments:*

grouping a factor with group memberships

responses the matrix of responses (counts)

covariates the matrix of covariates

offsets the matrix of offsets

weights an optional vector of observation weights. Default is uniform weights.

config\_post a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.).

config\_optim list controlling the optimization parameters

PLNLDAfit\$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

PLNLDAfit\$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 is 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 `ggplot2::ggplot` graphic

`PLNLDAfit$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 is `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 `ggplot2::ggplot` graphic

`PLNLDAfit$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

`PLNLDAfit$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 probabilities are computed from the learning set.

**control** a list for controlling the optimization. See [PLN\(\)](#) for details.

**envir** Environment in which the prediction is evaluated

**PLNLDAfit\$show()**: User friendly print method

*Usage:*

```
PLNLDAfit$show()
```

**PLNLDAfit\$clone()**: The objects of this class are cloneable with this method.

*Usage:*

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

*Arguments:*

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

## See Also

The function [PLNLDA](#).

## Examples

```

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

## End(Not run)

```

---

|                    |                                                                                      |
|--------------------|--------------------------------------------------------------------------------------|
| PLNLDAfit_diagonal | <i>An R6 Class to represent a PLNfit in a LDA framework with diagonal covariance</i> |
|--------------------|--------------------------------------------------------------------------------------|

---

## 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 classes

`PLNfit` -> `PLNLDAfit` -> `PLNLDAfit_diagonal`

## 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_diagonal$new()`
- `PLNLDAfit_diagonal$clone()`

`PLNLDAfit_diagonal$new()`: Initialize a `PLNfit` model

*Usage:*

```
PLNLDAfit_diagonal$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.

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

*Usage:*

```
PLNLDAfit_diagonal$clone(deep = FALSE)
```

*Arguments:*

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

## Examples

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

## End(Not run)
```

---

|                     |                                                                                       |
|---------------------|---------------------------------------------------------------------------------------|
| PLNLDAfit_spherical | <i>An R6 Class to represent a PLNfit in a LDA framework with spherical covariance</i> |
|---------------------|---------------------------------------------------------------------------------------|

---

## 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 classes

```
PLNfit -> 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()`

`PLNLDAfit_spherical$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.

`PLNLDAfit_spherical$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)
myPLNLDA <- PLNLDA(Abundance ~ 1, grouping = Group, data = trichoptera,
  control = PLNLDA_param(covariance = "spherical"))
class(myPLNLDA)
print(myPLNLDA)

## End(Not run)
```

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**

```
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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the model 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 <code>PLNmixture_param()</code> . 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**

```
## Use parallel to dispatch the computations on 2 workers
## Not run:
options(mc.cores = 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'))
```

```
## Not run:
options(mc.cores = 1)

## End(Not run)
```

---

|                  |                                    |
|------------------|------------------------------------|
| PLNmixture_param | <i>Control of a PLNmixture fit</i> |
|------------------|------------------------------------|

---

## Description

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

## Usage

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

## Arguments

|              |                                                                                                                                                                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| backend      | optimization back used, either "builtin", "nlopt" or "torch". Default is "builtin".                                                                                                                                                                                                                                               |
| 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.). 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)
- "profiled" (full covariance only) if TRUE, profile both B and Omega at every nlopt evaluation instead of running an EM loop (Omega fixed for the duration of each inner nlopt solve, B profiled in closed form at every evaluation). Despite the extra  $O(n \cdot p^2 + p^3)$  cost per evaluation, benchmarks found it consistently faster than the EM loop (and with a slightly better loglik) across a range of problem sizes. Default is TRUE; set to FALSE to recover the EM loop.

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

When "builtin" backend is used, the following entries are relevant

- "maxeval" stop when the number of Newton steps in the inner loop exceeds maxeval. Default is 10000
- "ftol\_in" stop the inner loop when the objective changes by less than ftol\_in (relative). Default is 1e-8
- "maxit\_em" stop the EM outer loop when the number of EM iterations exceeds maxit\_em. Default is 50
- "ftol\_em" stop the EM outer loop when the ELBO changes by less than ftol\_em (relative). Default is 1e-8

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.

## Outer-loop optimization parameters

PLNmixture\_param() adds parameters controlling the EM and smoothing outer loops:

- "ftol\_em" outer EM solver stops when the objective changes by less than ftol\_em (relative). Default is 1e-3
- "maxit\_em" outer EM solver stops when the number of iterations exceeds maxit\_em. Default is 50
- "it\_smooth" number of the iterations of the smoothing procedure. Default is 1.

**See Also**

[PLN\\_param\(\)](#)

---

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**

[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\(\)](#)

[PLNmixturefamily\\$new\(\)](#): 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.

`PLNmixturefamily$optimize()`: Call to the optimizer on all models of the collection

*Usage:*

`PLNmixturefamily$optimize(config)`

*Arguments:*

config a list for controlling the optimization

`PLNmixturefamily$smooth()`: function to restart clustering to avoid local minima by smoothing the loglikelihood values as a function of the number of clusters

*Usage:*

`PLNmixturefamily$smooth(control)`

*Arguments:*

control a list to control the smoothing process

`PLNmixturefamily$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::ggplot` object

`PLNmixturefamily$plot_objective()`: Plot objective value of the optimization problem along the penalty path

*Usage:*

`PLNmixturefamily$plot_objective()`

*Returns:* a `ggplot2::ggplot` graph

`PLNmixturefamily$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

`PLNmixturefamily$show()`: User friendly print method

*Usage:*

`PLNmixturefamily$show()`

`PLNmixturefamily$print()`: User friendly print method

*Usage:*

`PLNmixturefamily$print()`

`PLNmixturefamily$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()`

`PLNmixturefit$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.

PLNmixturefit\$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

PLNmixturefit\$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

PLNmixturefit\$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 `ggplot2::ggplot` object

`log_scale` logical. Should the color scale values be log-transform before plotting? Default is TRUE.

*Returns:* a `ggplot2::ggplot` graphic

`PLNmixturefit$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 `ggplot2::ggplot` object

*Returns:* a `ggplot2::ggplot` graphic

`PLNmixturefit$postTreatment()`: Update fields after optimization

*Usage:*

```
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.

`PLNmixturefit$show()`: User friendly print method

*Usage:*

```
PLNmixturefit$show()
```

PLNmixturefit\$print(): User friendly print method

*Usage:*

```
PLNmixturefit$print()
```

PLNmixturefit\$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
PLNmixturefit$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### See Also

The function [PLNmixture](#), the class [PLNmixturefamily](#)

---

PLNnetwork

*Sparse Poisson lognormal model for network inference*

---

### Description

Perform sparse inverse covariance estimation for the Zero Inflated Poisson lognormal model using a variational algorithm. Iterate over a range of logarithmically spaced sparsity parameter values. Use the (g)lm syntax to specify the model (including covariates and 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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the model 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 <code>PLNnetwork_param()</code> for additional tuning of the penalty. With the default <code>penalty_scale = "correlation"</code> , the penalties apply on the scale of the correlations and lie between 0 and 1; until version 1.3.2 they were on the covariance scale, which <code>PLNnetwork_param(penalty_scale = "covariance")</code> gives back. When penalties above 1 are given without choosing the scale, they are taken as penalties on the covariance scale and converted, with a warning (see <code>PLNnetwork_param()</code> ). |
| control   | a list-like structure for controlling the optimization, with default generated by <code>PLNnetwork_param()</code> . 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 and the configuration function `PLNnetwork_param()`.

**Examples**

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

---

|                  |                                  |
|------------------|----------------------------------|
| PLNnetwork_param | <i>Control of PLNnetwork fit</i> |
|------------------|----------------------------------|

---

**Description**

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

**Usage**

```
PLNnetwork_param(
  backend = c("builtin", "nlopt", "torch"),
  inception_cov = c("diagonal", "full", "spherical"),
  inception_backend = NULL,
  inception_niter = NULL,
  maxit_ve = NULL,
  trace = 1,
  n_penalties = 30,
  min_ratio = 0.1,
  penalize_diagonal = FALSE,
  penalty_weights = NULL,
```

```

penalty_scale = c("correlation", "covariance"),
latent_floor = 0.001,
config_post = list(),
config_optim = list(),
inception = NULL
)

```

## Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| backend           | optimization backend, either "builtin" (Newton, default) or "nlopt" (CC-SAQ) or "torch". The default combines "builtin" with <code>maxit_ve = 1</code> and <code>inception_niter = 5</code> (see <code>maxit_ve</code> and <code>inception_backend</code> ): this consistently finds a better ELBO than plain "nlopt", at essentially the same speed. Without a good inception, "builtin" alone ( <code>maxit_ve = NULL</code> ) can converge to a poor basin on large datasets — use "nlopt" if you want to opt out of the whole combination.                                                                                     |
| inception_cov     | Covariance structure used for the inception PLN, which starts the penalty path and sets its top: "diagonal" (default), "full" or "spherical". The top of the grid of penalties is the largest off-diagonal entry of the residual covariance of the inception, $\text{crossprod}(M - XB)/n$ , the penalty above which the graphical Lasso returns the empty network (weighted by <code>penalty_weights</code> , and on the diagonal too when <code>penalize_diagonal = TRUE</code> ). With a diagonal inception and an unpenalized diagonal (the defaults), the inception is itself the empty network, and the path starts from it. |
| inception_backend | character or NULL (default, i.e. same as <code>backend</code> ). Backend for the inception PLN only; the penalty grid models always use <code>backend</code> . Ignored when <code>inception</code> is supplied by the user.                                                                                                                                                                                                                                                                                                                                                                                                        |
| inception_niter   | integer or NULL. Limits the inception PLN to at most this many iterations (EM iterations for "builtin", function evaluations $\times 10$ for "nlopt"). Default is 5L when <code>backend = "builtin"</code> and <code>inception_cov = "full"</code> , NULL (full convergence) otherwise: for a full inception, fewer iterations keep the latent mean $M$ from over-converging toward the unconstrained optimum, which would make it harder to warm-start the sparse penalty models (values above $\sim 20$ typically hurt); a diagonal inception is the empty network, which is better converged.                                   |
| maxit_ve          | integer or NULL. Maximum number of inner VE-step iterations per outer GLASSO alternation turn. Default is 1L when <code>backend = "builtin"</code> (NULL, i.e. full convergence — <code>maxit_em</code> for "builtin", <code>maxeval</code> for "nlopt" — otherwise). <code>maxit_ve = 1</code> implements a <b>partial E-step</b> (generalized EM): one Newton step per outer turn prevents over-convergence that causes oscillations with the GLASSO M-step — see <code>backend</code> for the full default combination and its benchmark.                                                                                       |
| trace             | a integer for verbosity.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| n_penalties       | an integer that specifies the number of values for the penalty grid when internally generated. Ignored when <code>penalties</code> is non NULL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| min_ratio         | the penalty grid ranges from the minimal value that produces a sparse to this value multiplied by <code>min_ratio</code> . Default is 0.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| penalize_diagonal | boolean: should the diagonal terms be penalized in the graphical-Lasso? Default is FALSE. Penalizing the diagonal inflates the latent variances ( $\Sigma_{ii} = S_{ii} + \rho$ on the covariance scale, $\Sigma_{ii} = S_{ii}(1 + \rho)$ on the correlation scale), which the VE step then feeds back into the residual covariance $S$ : along the path, the network may then never become empty, and the latent variances diverge (#180).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| penalty_weights   | either a single or a list of $p \times p$ matrix of weights (default: all weights equal to 1) to adapt the amount of shrinkage to each pairs of node. Must be symmetric with positive values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| penalty_scale     | character, the scale on which the l1 penalty applies. "correlation" (default) penalizes the entries of the precision matrix on the scale of the variables, with a penalty $\lambda \sqrt{S_{ii}S_{jj}}$ on the pair $(i, j)$ , where $S$ is the current residual covariance. This amounts to applying the graphical-Lasso to the residual <i>correlation</i> matrix and rescaling the result, the correlation-based estimator of Rothman, Bickel, Levina and Zhu (2008), and makes the penalties dimensionless, between 0 and 1. "covariance", the only behavior until version 1.3.2, penalizes the entries of the precision matrix as they are. These are not scale invariant: a species with a large latent variance then has nearly free edges, and one that is often absent but abundant when present ends up connected to most of the others (see the field <code>degenerate_species</code> of a <code>PLNfit</code> ). See the section on the scale of the penalty. |
| latent_floor      | a positive number $\epsilon$ (default $1e-3$ ), or NULL for no floor: a floor on the variational means of the degenerate species. As soon as the latent variance of a species exceeds the threshold of the field <code>degenerate_species</code> of a <code>PLNfit</code> (100, see <code>options(PLNmodels.latent_variance_threshold = )</code> ) during the optimization, its variational means are kept above $\log \epsilon - O$ , that is $\exp(O + M) \geq \epsilon$ , in this fit and in the following ones along the penalty path. The species concerned are in the field <code>floored_species</code> of the fits. This needs no preparation of the data: the species are found by the optimization itself. The floor restricts the variational family, not the model, and stops the latent variances from diverging. It leaves a fit without degenerate species exactly as it is.                                                                               |
| 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 <code>PLNfit</code> (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)
- "profiled" (full covariance only) if TRUE, profile both B and Omega at every nlopt evaluation instead of running an EM loop (Omega fixed for the duration of each inner nlopt solve, B profiled in closed form at every evaluation). Despite the extra  $O(n \times p^2 + p^3)$  cost per evaluation, benchmarks found it consistently faster than the EM loop (and with a slightly better loglik) across a range of problem sizes. Default is TRUE; set to FALSE to recover the EM loop.

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

When "builtin" backend is used, the following entries are relevant

- "maxeval" stop when the number of Newton steps in the inner loop exceeds maxeval. Default is 10000
- "ftol\_in" stop the inner loop when the objective changes by less than ftol\_in (relative). Default is 1e-8
- "maxit\_em" stop the EM outer loop when the number of EM iterations exceeds maxit\_em. Default is 50
- "ftol\_em" stop the EM outer loop when the ELBO changes by less than ftol\_em (relative). Default is 1e-8

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.

### Outer-loop optimization parameters

`PLNnetwork_param()` adds two parameters controlling the alternating GLASSO/VEM loop:

- "ftol\_em" outer alternating solver stops when the objective changes by less than ftol\_em (relative). Default is 1e-5
- "maxit\_em" outer alternating solver stops when the number of iterations exceeds maxit\_em. Default is 20

### Scale of the penalty

With `penalty_scale = "correlation"`, the penalty on the pair  $(i, j)$  is  $\lambda w_{ij} \sqrt{S_{ii} S_{jj}}$ , recomputed at each M step from the current residual covariance  $S$ . Without weights, the precision matrix is  $\Omega = D^{-1/2} K D^{-1/2}$ , where  $D$  is the diagonal of  $S$  and  $K$  the graphical-Lasso estimate on the correlation matrix  $D^{-1/2} S D^{-1/2}$ ; this is the correlation-based estimator of Rothman et al. (2008), who show that it has a better rate of convergence in operator norm than the estimator on the covariance scale. The grid of penalties is built on the residual correlation of the inception, and lies between 0 and 1. Since the weights depend on  $S$ , the alternating optimization does not maximize a fixed penalized criterion: it looks for a fixed point.

It is the default since version 1.3.3. In simulations with a known network (a hundred configurations of graph, sample size, dimension, latent variances and abundances, with or without species absent from groups of samples), the edges were recovered better on the correlation scale in every configuration, the more so as the latent variances differed between species, and the time was the same.

**To get the behavior of version 1.3.2 back**, use `PLNnetwork_param(penalty_scale = "covariance")`; penalties are then on the covariance scale. Penalties given through `penalties` are otherwise taken on the correlation scale. Since values above 1 cannot be on that scale (they all give the empty network), when some are given while `penalty_scale` is left to its default they are taken as penalties on the covariance scale and converted, with a warning: they are divided by the ratio of the largest residual covariance of the inception to its largest residual correlation, so that the penalty giving the empty network on one scale gives it on the other. This is exact when all latent variances are equal, and only a guide otherwise: the two scales do not give the same networks. Setting `penalty_scale` explicitly, to either value, leaves the penalties as given.

On either scale, the latent variance of a species absent from whole groups of samples diverges along the path without `latent_floor`. On the covariance scale, the floor stabilizes the fit without repairing the network: the degenerate species have fewer edges, but the edges still concentrate on them.

## References

Rothman, A. J., Bickel, P. J., Levina, E. and Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2, 494–515. doi:[10.1214/08EJS176](https://doi.org/10.1214/08EJS176)

## See Also

[PLN\\_param\(\)](#)

---

PLNnetworkfamily

An R6 Class to represent a collection of [PLNnetworkfits](#)

---

## Description

The function [PLNnetwork\(\)](#) produces an instance of this class.

This class comes with a set of methods mostly used to compare network fits (in terms of goodness of fit) or extract one from the family (based on penalty parameter and/or goodness of it). See the documentation for [getBestModel\(\)](#), [getModel\(\)](#) and [plot\(\)](#) for the user-facing ones.

## Super classes

[PLNfamily](#) -> [Networkfamily](#) -> PLNnetworkfamily

## Methods

### Public methods:

- [PLNnetworkfamily\\$new\(\)](#)
- [PLNnetworkfamily\\$stability\\_selection\(\)](#)
- [PLNnetworkfamily\\$clone\(\)](#)

[PLNnetworkfamily\\$new\(\)](#): Initialize all models in the collection

*Usage:*

```
PLNnetworkfamily$new(penalties, data, control)
```

*Arguments:*

`penalties` a vector of positive real number controlling the level of sparsity of the underlying network.

`data` a named list used internally to carry the data matrices

`control` a list for controlling the optimization.

*Returns:* Update current [PLNnetworkfit](#) with smart starting values

[PLNnetworkfamily\\$stability\\_selection\(\)](#): Compute the stability path by stability selection

*Usage:*

```
PLNnetworkfamily$stability_selection(
  subsamples = NULL,
  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 controlling the main optimization process in each call to [PLNnetwork\(\)](#). See [PLNnetwork\(\)](#) and [PLN\\_param\(\)](#) for details.

[PLNnetworkfamily\\$clone\(\)](#): The objects of this class are cloneable with this method.

*Usage:*

```
PLNnetworkfamily$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

The function [PLNnetwork\(\)](#), the class [PLNnetworkfit](#)

## Examples

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

---

|               |                                                                                   |
|---------------|-----------------------------------------------------------------------------------|
| PLNnetworkfit | <i>An R6 Class to represent a PLNfit in a sparse inverse covariance framework</i> |
|---------------|-----------------------------------------------------------------------------------|

---

## 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

`PLNfit` -> `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.

`penalty_scale` the scale on which the penalty applies, "covariance" or "correlation" (see `PLNnetwork_param()`)

`latent_floor` the lower bound on  $\exp(0 + M)$  for the species in `floored_species`, NULL if none (see `PLNnetwork_param()`)

`floored_species` names of the species whose variational means are bounded by `latent_floor`: those whose latent variance has exceeded the threshold of `degenerate_species` during the optimization, of this fit or of the previous ones along the penalty path

`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, the criterion that the M step of the last iteration maximizes:  $\text{loglik} - n/2 * \sum(\text{abs}(\rho * \Omega))$ , where  $\rho$  is the penalty matrix of the graphical Lasso, that is the penalty times the penalty weights and, on the correlation scale, times  $\sqrt{S_{ii}S_{jj}}$

`ebic_gamma` the tuning parameter gamma of the EBIC, between 0 and 1. Zero gives back the BIC; the default 0.5 is the value recommended by Foygel and Drton (2010). Assign to it to change the EBIC of this fit.

`EBIC` variational lower bound of the extended BIC of Foygel and Drton (2010), that is the BIC with the additional penalty  $2 \gamma |\text{el}| \log(p)$  on the edge set. Their criterion was designed for the graphical Lasso; up to and including version 1.3.2, PLNmodels used instead the approximation  $\gamma |\text{el}| \log(p - 1) / (2 |\text{el}|)$  of the original EBIC of Chen and Chen (2008).

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

`PLNnetworkfit$new()`: Initialize a `PLNnetworkfit` object

*Usage:*

```
PLNnetworkfit$new(data, control)
```

*Arguments:*

`data` a named list used internally to carry the data matrices  
`control` a list for controlling the optimization.

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

*Usage:*

```
PLNnetworkfit$optimize(data, config)
```

*Arguments:*

`data` a named list used internally to carry the data matrices  
`config` a list for controlling the optimization

`PLNnetworkfit$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`

`PLNnetworkfit$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,
  edge.alpha = 0.2,
  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.

`edge.alpha` opacity of the weakest edge, the strongest one being fully opaque, so that the strength of an edge can be read off a dense network. Default is 0.2. Set it to 1 for uniformly opaque edges. Only relevant for igraph output with `type = "partial_cor"`.

`plot` logical. Should the final network be displayed or only sent back to the user. Default is TRUE.

`PLNnetworkfit$show()`: User friendly print method

*Usage:*

```
PLNnetworkfit$show()
```

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

*Usage:*

```
PLNnetworkfit$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**See Also**

The function [PLNnetwork\(\)](#), the class [PLNnetworkfamily](#)

**Examples**

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

## End(Not run)
```

**Description**

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

**Usage**

```
PLNPCA(formula, data, subset, weights, ranks = 1:5, control = PLNPCA_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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the model 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 <code>PLNPCA_param()</code> . 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 `PLNPCA_param()`.

**Examples**

```
## Use parallel to dispatch the computations on 2 workers
## Not run:
options(mc.cores = 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)
```

```
## Not run:
options(mc.cores = 1)

## End(Not run)
```

---

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 = c("nlopt", "builtin", "torch"),
  trace = 1,
  config_optim = list(),
  config_post = list(),
  inception = NULL,
  init_method = c("LM", "GLM", "EM"),
  init_niter = 5L,
  sequential = FALSE
)
```

## Arguments

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| backend | optimization backend, either "nlopt" (default, NLOPT/CCSAQ, recommended for PLNPCA: conservative per-variable steps reliably find the global basin even when the singular-value ratio $d1/\sqrt{n}$ is large), "builtin" (the home-made optimizer, a profiled trust-region Newton), or "torch" ( <b>experimental</b> : automatic differentiation via the torch package; tends to find lower loglik than "nlopt" and "builtin" on most datasets — not recommended). The "builtin" backend profiles out the variational parameters (M, S) with a per-observation Newton VE-step and optimises the loadings (B, C) with a saddle-aware trust-region Newton on the resulting objective (analytic Schur Hessian-vector products, Jacobi-preconditioned Steihaug-CG). It reliably reaches a higher variational bound than "nlopt" on small/moderate data at comparable speed, and is faster on large data; tuning keys in config_optim: cg_maxit, maxit_out, ftol_out, gtol, delta0. On large data (many samples and variables) the reduced-Hessian landscape is indefinite and the outer trust-region iteration converges slowly but keeps improving the bound, so it is capped by maxit_out (default 150) rather than by the gradient tolerance: the higher ranks then trade quality for time. For the best variational bound on large data, raise it, e.g. config_optim = list(maxit_out = 300). |
| trace   | a integer for verbosity.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>config_optim</code> | a list for controlling the optimizer (either "nlopt" or "torch" backend). See details                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>config_post</code>  | a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.). See details                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>inception</code>    | an optional pre-fitted <code>PLNfit</code> object. When provided, its variational means $M$ and regression coefficients $B$ are used to compute the shared SVD $\text{svd}(M - X*B)$ that initialises all ranks simultaneously. This replaces the default LM-based starting point and gives the highest-quality initialisation for large ranks. When NULL (default), see <code>init_method</code> . <code>init_method</code> is ignored when <code>inception</code> is set. <b>Recommendation:</b> for large ranks ( $\text{rank} > \sqrt{p}$ ) on complex datasets, prefer <code>init_method = "EM"</code> (cheap, most gains) or <code>inception = PLN(formula, data)</code> (maximum quality, full convergence cost).                                                                                                                                                                                                                           |
| <code>init_method</code>  | character: strategy used to compute the starting point for the shared SVD. <ul style="list-style-type: none"> <li>• "LM" (default): fast multivariate <code>lm.fit</code> on log-transformed counts. Good for small to moderate ranks (<math>\text{rank} \leq \sqrt{p}</math>). Recommended default.</li> <li>• "GLM": <math>p</math> independent Poisson GLMs. Rarely better than "LM" for PLNPCA; not recommended.</li> <li>• "EM": run <code>init_niter</code> variational EM iterations of a full PLN (builtin backend) before computing the SVD, giving a latent mean <math>M</math> closer to the true posterior. <b>Recommended for large ranks</b> (<math>\text{rank} &gt; \sqrt{p}</math>), where it noticeably improves the loglik at negligible extra cost. Caution: hurts for small ranks, because an over-converged <math>M</math> reduces low-rank SVD discriminability. Ignored when <code>inception</code> is provided.</li> </ul> |
| <code>init_niter</code>   | integer: number of PLN EM iterations when <code>init_method = "EM"</code> . Default is 5. The sweet spot is 2–10: beyond ~10 the $M$ over-converges toward the full PLN optimum, degrading the low-rank SVD and hurting small-rank models. Ignored for "LM" and "GLM".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>sequential</code>   | logical. If TRUE, ranks are fitted in ascending order and each model is warm-started from the converged solution of the previous rank: loadings $C$ are augmented with new columns from the inception SVD, while latent scores $M$ and variances $S^2$ are padded with zeros / 0.01 respectively. Disables parallel fitting across ranks. Default is FALSE.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## 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)
- "profiled" (full covariance only) if TRUE, profile both B and Omega at every nlopt evaluation instead of running an EM loop (Omega fixed for the duration of each inner nlopt solve, B profiled in closed form at every evaluation). Despite the extra  $O(n \times p^2 + p^3)$  cost per evaluation, benchmarks found it consistently faster than the EM loop (and with a slightly better loglik) across a range of problem sizes. Default is TRUE; set to FALSE to recover the EM loop.

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

When "builtin" backend is used, the following entries are relevant

- "maxeval" stop when the number of Newton steps in the inner loop exceeds maxeval. Default is 10000
- "ftol\_in" stop the inner loop when the objective changes by less than ftol\_in (relative). Default is 1e-8

- "maxit\_em" stop the EM outer loop when the number of EM iterations exceeds maxit\_em. Default is 50
- "ftol\_em" stop the EM outer loop when the ELBO changes by less than ftol\_em (relative). Default is 1e-8

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.

---

PLNPCAfamily

*An R6 Class to represent a collection of PLNPCAfit*

---

### Description

The function `PLNPCA()` produces an instance of this class.

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

### Super class

`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()`

`PLNPCAfamily$new()`: Initialize all models in the collection. A single SVD of the residual matrix  $M - X*B$  is computed once and shared across all ranks.  $M$  and  $B$  come from either a user-provided `PLNfit` inception or a fast LM on log-transformed counts (default, controlled by `init_method`).

#### 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

`PLNPCAfamily$optimize()`: Call to the C++ optimizer on all models of the collection

#### Usage:

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

#### Arguments:

`config` list controlling the optimization.

`PLNPCAfamily$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

`PLNPCAfamily$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

`PLNPCAfamily$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::ggplot](#) object

`PLNPCAfamily$show()`: User friendly print method

*Usage:*

```
PLNPCAfamily$show()
```

`PLNPCAfamily$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\(\)](#)

**Examples**

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

---

PLNPCAfit

An R6 Class to represent a PLNfit in a PCA framework

---

## Description

The function `PLNPCA()` produces a collection of models which are instances of object with class `PLNPCAfit`. This class comes with a set of methods, some of them being useful for the user: See the documentation for the methods inherited by `PLNfit` and the `plot()` methods for PCA visualization

## Super class

`PLNfit` -> `PLNPCAfit`

## Active bindings

`var_par` variational parameters (M, S2) in the rank-q latent space  
`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$new()`
- `PLNPCAfit$warm_start_from()`
- `PLNPCAfit$update()`

- `PLNPCAfit$optimize()`
- `PLNPCAfit$optimize_vestep()`
- `PLNPCAfit$project()`
- `PLNPCAfit$setVisualization()`
- `PLNPCAfit$postTreatment()`
- `PLNPCAfit$plot_individual_map()`
- `PLNPCAfit$plot_correlation_circle()`
- `PLNPCAfit$plot_PCA()`
- `PLNPCAfit$show()`
- `PLNPCAfit$clone()`

`PLNPCAfit$new()`: Initialize a `PLNPCAfit` object. Uses the shared SVD from `control$svdM` (computed once in `PLNPCAfamily`) to set the starting loadings  $C$  and scores  $M$ . The regression coefficients  $B$  are initialised by the parent `PLNfit` constructor (LM or user-provided inception).

*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.

`PLNPCAfit$warm_start_from()`: Reinitialize parameters for sequential warm-starting from a lower-rank fit. Fitted loadings  $C$ , scores  $M$ , variances  $S$ , and regression coefficients  $B$  from `prev_fit` are carried over; new columns are padded using the inception SVD ( $C$ ) or zeros/0.1 ( $M/S$ ).

*Usage:*

```
PLNPCAfit$warm_start_from(prev_fit, svdM)
```

*Arguments:*

`prev_fit` a converged `PLNPCAfit` of rank `self$rank - k` ( $k \geq 1$ )

`svdM` the inception SVD (from `PLNPCAfamily`)

`PLNPCAfit$update()`: Update a `PLNPCAfit` object

*Usage:*

```
PLNPCAfit$update(
  B = NA,
  Sigma = NA,
  Omega = NA,
```

```

    C = NA,
    M = NA,
    S2 = 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

S2 matrix of variational variances ( $n \times q$ )

Z matrix of latent vectors (includes covariates and offset effects)

A matrix of fitted values

Ji vector of variational lower bounds of the log-likelihoods (one value per sample)

R2 approximate  $R^2$  goodness-of-fit criterion

monitoring a list with optimization monitoring quantities

*Returns:* Update the current `PLNPCAfit` object

`PLNPCAfit$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

`PLNPCAfit$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

**PLNPCAfit\$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

**PLNPCAfit\$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

**PLNPCAfit\$postTreatment():** Update R2, fisher, std\_err fields and set up visualization

*Usage:*

```

PLNPCAfit$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](#)

**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

**PLNPCAfit\$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 is `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 `ggplot2::ggplot` graphic

`PLNPCAfit$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 is `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 `ggplot2::ggplot` graphic

`PLNPCAfit$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

`PLNPCAfit$show()`: User friendly print method

*Usage:*

```
PLNPCAfit$show()
```

PLNPCAfit\$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
PLNPCAfit$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## See Also

The function [PLNPCA](#), the class [PLNPCAfamily](#)

## Examples

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

---

|                    |                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| plot.Networkfamily | <i>Display various outputs (goodness-of-fit criteria, robustness, diagnostic) associated with a collection of network fits (either <a href="#">PLNnetworkfamily</a> or <a href="#">ZIPLNnetworkfamily</a>)</i> |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

## Description

Display various outputs (goodness-of-fit criteria, robustness, diagnostic) associated with a collection of network fits (either [PLNnetworkfamily](#) or [ZIPLNnetworkfamily](#))

## Usage

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

## 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,
    ...
)

## S3 method for class 'ZIPLNnetworkfamily'
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 <a href="#">PLNnetworkfamily</a> or <a href="#">ZIPLNnetworkfamily</a>                                                                                                                                                                  |
| type      | a character, either "criteria", "stability" or "diagnostic" for the type of plot.                                                                                                                                                                               |
| criteria  | Vector of criteria to plot, to be selected among "loglik" (log-likelihood), "BIC", "ICL", "R_squared", "EBIC" and "pen_loglik" (penalized log-likelihood). Default is c("loglik", "pen_loglik", "BIC", "EBIC"). Only used when type = "criteria".               |
| reverse   | A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - 0.5 penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e use the natural direction, on the same scale as the log-likelihood. |
| log.x     | logical: should the x-axis be represented in log-scale? Default is TRUE.                                                                                                                                                                                        |
| stability | scalar: the targeted level of stability in stability plot. Default is .9.                                                                                                                                                                                       |
| ...       | additional parameters for S3 compatibility. Not used                                                                                                                                                                                                            |

### Details

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

### Value

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

### Functions

- plot(PLNnetworkfamily): Display various outputs associated with a collection of network fits

- `plot(ZIPLNnetworkfamily)`: Display various outputs associated with a collection of network fits

### Examples

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

## End(Not run)
```

---

|                             |                                                                                     |
|-----------------------------|-------------------------------------------------------------------------------------|
| <code>plot.PLNfamily</code> | <i>Display the criteria associated with a collection of PLN fits (a PLN-family)</i> |
|-----------------------------|-------------------------------------------------------------------------------------|

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

### Value

Produces a plot representing the evolution of the criteria of the different models considered, highlighting the best model in terms of BIC and ICL (see details).

**See Also**

[plot.PLNPCAfamily\(\)](#) and [plot.PLNnetworkfamily\(\)](#)

**Examples**

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

## End(Not run)
```

---

|                |                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------|
| plot.PLNLDAfit | <i>LDA visualization (individual and/or variable factor map(s)) for a <a href="#">PLNPCAfit</a> object</i> |
|----------------|------------------------------------------------------------------------------------------------------------|

---

**Description**

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

**Usage**

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

**Arguments**

|          |                                                                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| x        | an R6 object with class PLNPCAfit                                                                                                                    |
| map      | the type of output for the PCA visualization: either "individual", "variable" or "both". Default is "both".                                          |
| nb_axes  | scalar: the number of axes to be considered when map = "both". The default is min(3,rank).                                                           |
| axes     | numeric, the axes to use for the plot when map = "individual" or "variable". Default is 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 <code>ggplot2::ggplot</code> 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::ggplot` 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 <code>PLNmixturefamily</code>                                                                                      |
| type     | a character, either "criteria" or "diagnostic" for the type of plot.                                                                       |
| criteria | vector of characters. The criteria to plot in <code>c("loglik", "BIC", "ICL")</code> . Default is <code>c("loglik", "BIC", "ICL")</code> . |

|         |                                                                                                                                                                                                                                                                 |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 | <i>Mixture visualization of a <a href="#">PLNmixturefit</a> object</i> |
|--------------------|------------------------------------------------------------------------|

---

### 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 <a href="#">PLNmixturefit</a>                                                |
| 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 <a href="#">ggplot2::ggplot</a> object         |
| ...  | Not used (S3 compatibility).                                                                         |

**Value**

a `ggplot2::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.PLNnetworkfit | <i>Extract and plot the network (partial correlation, support or inverse covariance) from a <code>PLNnetworkfit</code> object</i> |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------|

---

**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,
  edge.alpha = 0.2,
  plot = TRUE,
  ...
)
```

**Arguments**

|        |                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| x      | an R6 object with class <code>PLNnetworkfit</code>                                                                                                      |
| type   | character. Value of the weight of the edges in the network, either "partial_cor" (partial correlation) or "support" (binary). Default is "partial_cor". |
| output | the type of output used: either 'igraph' or 'corrplot'. Default is 'igraph'.                                                                            |

|                 |                                                                                                                                                                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| edge.color      | Length 2 color vector. Color for positive/negative edges. Default is c("#F8766D", "#00BFC4"). Only relevant for igraph output.                                                                                                                         |
| remove.isolated | if TRUE, isolated node are remove before plotting. Only relevant for igraph output.                                                                                                                                                                    |
| node.labels     | vector of character. The labels of the nodes. The default will use the column names of the response matrix.                                                                                                                                            |
| layout          | an optional igraph layout. Only relevant for igraph output.                                                                                                                                                                                            |
| edge.alpha      | opacity of the weakest edge, the strongest one being fully opaque, so that the strength of an edge can be read off a dense network. Default is 0.2. Set it to 1 for uniformly opaque edges. Only relevant for igraph output with type = "partial_cor". |
| plot            | logical. Should the final network be displayed or only sent back to the user. Default is TRUE.                                                                                                                                                         |
| ...             | Not used (S3 compatibility).                                                                                                                                                                                                                           |

**Value**

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

**Examples**

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

## End(Not run)
```

---

|                   |                                                                                          |
|-------------------|------------------------------------------------------------------------------------------|
| plot.PLNPCAfamily | <i>Display the criteria associated with a collection of PLNPCA fits (a PLNPCAfamily)</i> |
|-------------------|------------------------------------------------------------------------------------------|

---

**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 <a href="#">PLNPCAfamily</a>                                                                                                                                                                                                            |
| criteria | vector of characters. The criteria to plot in c("loglik", "BIC", "ICL"). Default is c("loglik", "BIC", "ICL").                                                                                                                                                  |
| reverse  | A logical indicating whether to plot the value of the criteria in the "natural" direction (loglik - 0.5 penalty) or in the "reverse" direction (-2 loglik + penalty). Default to FALSE, i.e use the natural direction, on the same scale as the log-likelihood. |
| ...      | additional parameters for S3 compatibility. Not used                                                                                                                                                                                                            |

**Details**

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

**Value**

Produces a plot representing the evolution of the criteria of the different models considered, highlighting the best model in terms of BIC and ICL (see details).

**Examples**

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

## End(Not run)
```

---

|                |                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------|
| plot.PLNPCAfit | <i>PCA visualization (individual and/or variable factor map(s)) for a <a href="#">PLNPCAfit</a> object</i> |
|----------------|------------------------------------------------------------------------------------------------------------|

---

**Description**

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

**Usage**

```
## S3 method for class 'PLNPCAfit'
plot(
  x,
  map = c("both", "individual", "variable"),
  nb_axes = min(3, x$rank),
  axes = seq.int(min(2, x$rank)),
```

```

    ind_cols = "ind_colors",
    var_cols = "var_colors",
    plot = TRUE,
    main = NULL,
    ...
  )

```

## Arguments

|          |                                                                                                                                                                 |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x        | an R6 object with class PLNPCAfit                                                                                                                               |
| map      | the type of output for the PCA visualization: either "individual", "variable" or "both". Default is "both".                                                     |
| nb_axes  | scalar: the number of axes to be considered when map = "both". The default is min(3,rank).                                                                      |
| axes     | numeric, the axes to use for the plot when map = "individual" or map = "variable". Default is 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 <code>ggplot2::ggplot</code> 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::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)

```

---

`plot.ZIPLNfit_sparse`    *Extract and plot the network (partial correlation, support or inverse covariance) from a [ZIPLNfit\\_sparse](#) object*

---

## Description

Extract and plot the network (partial correlation, support or inverse covariance) from a [ZIPLNfit\\_sparse](#) object

## Usage

```
## S3 method for class 'ZIPLNfit_sparse'
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,
  edge.alpha = 0.2,
  plot = TRUE,
  ...
)
```

## Arguments

|                              |                                                                                                                                                                                                                                                                      |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>               | an R6 object with class <a href="#">ZIPLNfit_sparse</a>                                                                                                                                                                                                              |
| <code>type</code>            | character. Value of the weight of the edges in the network, either "partial_cor" (partial correlation) or "support" (binary). Default is "partial_cor".                                                                                                              |
| <code>output</code>          | the type of output used: either 'igraph' or 'corrplot'. Default is 'igraph'.                                                                                                                                                                                         |
| <code>edge.color</code>      | Length 2 color vector. Color for positive/negative edges. Default is <code>c("#F8766D", "#00BFC4")</code> . Only relevant for igraph output.                                                                                                                         |
| <code>remove.isolated</code> | if TRUE, isolated node are remove before plotting. Only relevant for igraph output.                                                                                                                                                                                  |
| <code>node.labels</code>     | vector of character. The labels of the nodes. The default will use the column names of the response matrix.                                                                                                                                                          |
| <code>layout</code>          | an optional igraph layout. Only relevant for igraph output.                                                                                                                                                                                                          |
| <code>edge.alpha</code>      | opacity of the weakest edge, the strongest one being fully opaque, so that the strength of an edge can be read off a dense network. Default is 0.2. Set it to 1 for uniformly opaque edges. Only relevant for igraph output with <code>type = "partial_cor"</code> . |
| <code>plot</code>            | logical. Should the final network be displayed or only sent back to the user. Default is TRUE.                                                                                                                                                                       |
| <code>...</code>             | Not used (S3 compatibility).                                                                                                                                                                                                                                         |

**Value**

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

**Examples**

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
fit <- ZIPLN(Abundance ~ 1, data = trichoptera, control = ZIPLN_param(penalty = 0.1))
## Not run:
plot(fit)

## End(Not run)
```

---

|                |                                       |
|----------------|---------------------------------------|
| predict.PLNfit | <i>Predict counts of a new sample</i> |
|----------------|---------------------------------------|

---

**Description**

Predict counts of a new sample

**Usage**

```
## S3 method for class 'PLNfit'
predict(
  object,
  newdata,
  responses = NULL,
  level = 1,
  type = c("link", "response"),
  ...
)
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                          |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object    | an R6 object with class <a href="#">PLNfit</a>                                                                                                                                                                                                                                           |
| 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 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)                                                                                                                                                                           |
| ...       | additional parameters for S3 compatibility. Not used                                                                                                                                                                                                                                     |

**Value**

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

---

|                   |                                     |
|-------------------|-------------------------------------|
| predict.PLNLDAfit | <i>Predict group of new samples</i> |
|-------------------|-------------------------------------|

---

**Description**

Predict group of new samples

**Usage**

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

**Arguments**

|         |                                                                                                                                                                                                                                                                                                                                                                               |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object  | an R6 object with class <a href="#">PLNLDAfit</a>                                                                                                                                                                                                                                                                                                                             |
| 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 <a href="#">PLN()</a> for details.                                                                                                                                                                                                                                                                                               |
| ...     | additional parameters for S3 compatibility. Not used                                                                                                                                                                                                                                                                                                                          |

**Value**

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

**Examples**

```

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

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

## End(Not run)

```

---

predict.PLNmixturefit *Prediction for a [PLNmixturefit](#) object*

---

**Description**

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

**Usage**

```

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

```

**Arguments**

|         |                                                                                                                                                                                                                                                                       |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object  | an R6 object with class <a href="#">PLNmixturefit</a>                                                                                                                                                                                                                 |
| 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 <a href="#">PLNmixture_param()</a> for details.                                                                                                                                                                    |
| ...     | additional parameters for S3 compatibility. Not used                                                                                                                                                                                                                  |

**Value**

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

**Examples**

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

---

|                  |                                       |
|------------------|---------------------------------------|
| predict.ZIPLNfit | <i>Predict counts of a new sample</i> |
|------------------|---------------------------------------|

---

**Description**

Predict counts of a new sample

**Usage**

```
## S3 method for class 'ZIPLNfit'
predict(
  object,
  newdata,
  responses = NULL,
  level = 1,
  type = c("link", "response", "deflated"),
  ...
)
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                          |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object    | an R6 object with class <a href="#">ZIPLNfit</a>                                                                                                                                                                                                                                         |
| 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 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 | Scale used for the prediction. Either "link" (default, predicted positions in the latent space), "response" (predicted average counts, accounting for zero-inflation) or "deflated" (predicted average counts, not accounting for zero-inflation and using only the PLN part of the model). |
| ...  | additional parameters for S3 compatibility. Not used                                                                                                                                                                                                                                        |

### 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

Note also that when `type = "response"` corresponds to predicting values with  $(1 - \pi)A$ , where  $A$  is the average count in the PLN part of the model and  $\pi$  the probability of zero-inflation, whereas `type = "deflated"` corresponds to  $A$ .

### Value

A matrix of predicted log-counts (if `type = "link"`) or predicted counts, accounting for zero-inflation (if `type = "response"`) or not (if `type = "deflated"`).

---

|              |                                     |
|--------------|-------------------------------------|
| predict_cond | <i>Predict counts conditionally</i> |
|--------------|-------------------------------------|

---

### Description

Predict counts of a new sample conditionally on a (set of) observed variables

### Usage

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

## S3 method for class 'PLNfit'
predict_cond(
  object,
  newdata,
  cond_responses,
  type = c("link", "response"),
  var_par = FALSE
)
```

**Arguments**

|                |                                                                                                                                                                |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object         | an R6 object with class <code>PLNfit</code>                                                                                                                    |
| 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 $E(Z_{\text{uncond}}   Y_c)$ for each given site.                                                      |
| S    | A matrix containing $\text{Var}(Z_{\text{uncond}}   Y_c)$ for each given site (sites are the third dimension of the array) |

**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",
  call = rlang::caller_env(),
  ...
)
```

**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). |
| call       | Optional. The execution environment in which to set the local error call.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| ...        | Additional parameters passed on to <a href="#">compute_offset()</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

**Details**

The function also reports, in a message, the species that are absent from all the samples of a level of a factor covariate while present elsewhere, beyond what chance would explain: see [structural\\_zeros\(\)](#), which is run on all the factor, character and logical columns of covariates. Nothing is removed from the data.

**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**

Chen, L., Reeve, J., Zhang, L., Huang, S., Wang, X. and Chen, J. (2018) GMPR: A robust normalization method for zero-inflated count data with application to microbiome sequencing data. PeerJ, 6, e4600 [doi:10.7717/peerj.4600](https://doi.org/10.7717/peerj.4600)

Paulson, J. N., Colin Stine, O., Bravo, H. C. and Pop, M. (2013) Differential abundance analysis for microbial marker-gene surveys. *Nature Methods*, 10, 1200-1202 [doi:10.1038/nmeth.2658](https://doi.org/10.1038/nmeth.2658)

Anders, S. and Huber, W. (2010) Differential expression analysis for sequence count data. *Genome Biology*, 11, R106 [doi:10.1186/gb20101110r106](https://doi.org/10.1186/gb20101110r106)

Kumar, M., Slud, E., Okrah, K. et al. (2018) Analysis and correction of compositional bias in sparse sequencing count data. *BMC Genomics* 19, 799 [doi:10.1186/s12864-018-5160-5](https://doi.org/10.1186/s12864-018-5160-5)

Robinson, M.D., Oshlack, A. (2010) A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol* 11, R25 [doi:10.1186/gb20101113r25](https://doi.org/10.1186/gb20101113r25)

### See Also

`compute_offset()` for details on the different normalization schemes, `structural_zeros()`

### 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

```
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  $n \times p$  count matrix, with row-sums close to depths, with an attribute "offsets" corresponding to the true generated offsets (in log-scale).

### Examples

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

---

scRNA

*Single cell RNA-seq data*


---

### Description

A dataset containing the counts of the 500 most varying transcripts in the mixtures of 5 cell lines in human liver (obtained with standard 10x scRNAseq Chromium protocol).

### Usage

```
scRNA
```

### Format

A data frame named 'scRNA' with 3918 rows (the cells) and 3 variables:

**counts** a 500 transcript by 3918 count matrix

**cell\_line** factor, the cell line of the current row (among 5)

**total\_counts** Total number of reads for that cell ...

### Source

[https://github.com/LuyiTian/sc\\_mixology/](https://github.com/LuyiTian/sc_mixology/)

---

|              |                                                         |
|--------------|---------------------------------------------------------|
| sigma.PLNfit | <i>Extract variance-covariance of residuals 'Sigma'</i> |
|--------------|---------------------------------------------------------|

---

## 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 <a href="#">PLNfit</a>       |
| ...    | 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 | <i>Extract variance-covariance of residuals 'Sigma'</i> |
|---------------------|---------------------------------------------------------|

---

## 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 PLNmixture, 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 <code>PLNmixturefit</code>   |
| ...    | 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.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
```

---

|                             |                                                         |
|-----------------------------|---------------------------------------------------------|
| <code>sigma.ZIPLNfit</code> | <i>Extract variance-covariance of residuals 'Sigma'</i> |
|-----------------------------|---------------------------------------------------------|

---

### Description

Extract the variance-covariance matrix of the residuals, usually noted  $\Sigma$  in ZIPLN models.

### Usage

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

### Arguments

|                     |                                                      |
|---------------------|------------------------------------------------------|
| <code>object</code> | an R6 object with class <code>ZIPLNfit</code>        |
| <code>...</code>    | 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.ZIPLNfit()`

---

|                                  |                                                          |
|----------------------------------|----------------------------------------------------------|
| <code>stability_selection</code> | <i>Compute the stability path by stability selection</i> |
|----------------------------------|----------------------------------------------------------|

---

### Description

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

### Usage

```
stability_selection(
  Robject,
  subsamples = NULL,
  control = PLNnetwork_param(),
  force = FALSE
)
```

**Arguments**

|            |                                                                                                                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Robjct     | an object with class <code>PLNnetworkfamily</code> or <code>ZIPLNnetworkfamily</code> , i.e. an output from <code>PLNnetwork()</code> or <code>ZIPLNnetwork()</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. |
| control    | a list controlling the main optimization process in each call to <code>PLNnetwork()</code> or <code>ZIPLNnetwork()</code> . See <code>PLN_param()</code> or <code>ZIPLN_param()</code> for details.                                                                                                                  |
| force      | force computation of the stability path, even if a previous one has been detected.                                                                                                                                                                                                                                   |

**Value**

the list of subsamples. The estimated probabilities of selection of the edges are stored in the fields `stability_path` of the initial Robjct with class `Networkfamily`

**Examples**

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

## End(Not run)
```

---

standard\_error.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("sandwich", "variational", "jackknife"),
  parameter = c("B", "Omega")
)

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

## S3 method for class 'PLNfit_fixedcov'
standard_error(
  object,
  type = c("sandwich", "variational", "jackknife", "bootstrap"),
  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". Default is "sandwich". |
| parameter | string describing the target parameter: either B (regression coefficients) or Omega (inverse residual covariance)    |

### Value

A  $p \times d$  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 im-

plemented 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(sandwich_var = TRUE)))
standard_error(myPLN)
```

---

structural\_zeros

*Species absent from a whole level of a factor*

---

### Description

Finds the species that are absent from all the samples of a level of a factor covariate while present in the other samples, beyond what chance alone would explain. Such structural zeros are what a PLN model without the corresponding covariate cannot represent: it fits them by sending the latent means to minus infinity, the latent variance of the species blows up, and in a network the species ends up connected to most of the others (see the field `degenerate_species` of a [PLNfit](#)). [prepare\\_data\(\)](#) runs this check and reports its result in a message.

### Usage

```
structural_zeros(counts, covariates, alpha = 0.05)
```

### Arguments

|            |                                                                                                                                |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| counts     | An abundance count table, with species as columns.                                                                             |
| covariates | A covariates data frame, with the samples of counts as rows. Only its factor, character and logical columns are used.          |
| alpha      | Level of the test, after a Bonferroni correction over all the pairs of a species and a level that are tested. Default is 0.05. |

Details

For a species present in K of the N samples, the probability that none of the n samples of a level is among them, were the presences spread at random, is hypergeometric:  $\text{phyper}(0, K, N - K, n)$ . A rare species is easily absent from a level by chance, and is not reported; a species present in most of the other samples is.

Value

A data frame with one row per reported pair, sorted by p-value, with columns species, covariate, level, n\_samples (the number of samples of the level), prevalence\_elsewhere (the proportion of the other samples where the species is present) and p\_value (Bonferroni-adjusted). It has no row when nothing is reported.

See Also

`prepare_data()`

Examples

```
data(oaks)
## species that are absent from a whole tree, or from a whole type of branch
structural_zeros(oaks$Abundance, oaks[c("tree", "branch")])
```

---

|             |                             |
|-------------|-----------------------------|
| trichoptera | <i>Trichoptera data set</i> |
|-------------|-----------------------------|

---

Description

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

Usage

trichoptera

Format

- A list with 2 two data frames:
- Abundance** a 49 x 17 matrix of abundancies/counts (49 trapping nights and 17 trichoptera species)
  - Covariate** a 49 x 7 data frame of covariates:
    - Temperature** Evening Temperature in Celsius
    - Wind** Wind in m/s
    - Pressure** Pressure in mm Hg

**Humidity** relative to evening humidity in percent

**Cloudiness** proportion of sky coverage at 9pm

**Precipitation** Nighttime precipitation in mm

**Group** a factor of 12 levels for the definition of the consecutive night groups

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

### Source

Data from P. Usseglio-Polatera.

### References

Usseglio-Polatera, P. and Auda, Y. (1987) Influence des facteurs météorologiques sur les résultats de piégeage lumineux. Annales de Limnologie, 23, 65–79. (code des espèces p. 76) See a data description at <http://pbil.univ-lyon1.fr/R/pdf/pps034.pdf> (in French)

### 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 <code>PLNfit</code>                                                              |
| 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  $d * p$ , 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
```

---

 ZIPLN

---

*Zero Inflated Poisson lognormal model*


---

**Description**

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

**Usage**

```
ZIPLN(
  formula,
  data,
  subset,
  zi = c("single", "row", "col"),
  control = ZIPLN_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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the model is called.                                                                                  |
| subset  | an optional vector specifying a subset of observations to be used in the fitting process.                                                                                                                                                                                                                                                                                                |
| zi      | a character describing the model used for zero inflation, either of <ul style="list-style-type: none"> <li>• "single" (default, one parameter shared by all counts)</li> <li>• "col" (one parameter per variable / feature)</li> <li>• "row" (one parameter per sample / individual). If covariates are specified in the formula RHS (see details) this parameter is ignored.</li> </ul> |
| control | a list-like structure for controlling the optimization, with default generated by <code>ZIPLN_param()</code> . See the associated documentation for details.                                                                                                                                                                                                                             |

**Details**

Covariates for the Zero-Inflation parameter (using a logistic regression model) can be specified in the formula RHS using the pipe (`~ PLN effect | ZI effect`) to separate covariates for the PLN part of the model from those for the Zero-Inflation part. Note that different covariates can be used for each part.

**Value**

an R6 object with class `ZIPLNfit`

**See Also**

The class `ZIPLNfit`

**Examples**

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
## Use different models for zero-inflation...
myZIPLN_single <- ZIPLN(Abundance ~ 1, data = trichoptera, zi = "single")
## Not run:
myZIPLN_row    <- ZIPLN(Abundance ~ 1, data = trichoptera, zi = "row")
myZIPLN_col    <- ZIPLN(Abundance ~ 1, data = trichoptera, zi = "col")
## ...including logistic regression on covariates
myZIPLN_covar  <- ZIPLN(Abundance ~ 1 | 1 + Wind, data = trichoptera)

## End(Not run)
```

ZIPLN\_param

*Control of a ZIPLN fit***Description**

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

**Usage**

```
ZIPLN_param(
  backend = c("builtin", "nlopt"),
  trace = 1,
  covariance = c("full", "diagonal", "spherical", "fixed", "sparse"),
  Omega = NULL,
  penalty = 0,
  penalize_diagonal = FALSE,
  penalty_weights = NULL,
  penalty_scale = c("correlation", "covariance"),
  latent_floor = 0.001,
  config_post = list(),
  config_optim = list(),
  inception = NULL
)
```

**Arguments**

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| backend           | optimization backend, either "builtin" (default, built-in Newton optimizer for the joint VE step) or "nlopt" (NLOPT-based CCSAQ).                                                                                                                                                                                                                                                                                                           |
| trace             | a integer for verbosity.                                                                                                                                                                                                                                                                                                                                                                                                                    |
| covariance        | character setting the model for the covariance matrix. Either "full", "diagonal", "spherical", "fixed" or "sparse". Default is "full".                                                                                                                                                                                                                                                                                                      |
| Omega             | precision matrix of the latent variables. Inverse of Sigma. Must be specified if covariance is "fixed"                                                                                                                                                                                                                                                                                                                                      |
| penalty           | a user-defined penalty to sparsify the residual covariance. Defaults to 0 (no sparsity). With the default penalty_scale = "correlation", it applies on the scale of the correlations and lies between 0 and 1 (a penalty above 1 gives the empty network); penalty_scale = "covariance" gives back the behavior of version 1.3.2.                                                                                                           |
| penalize_diagonal | boolean: should the diagonal terms be penalized in the graphical-Lasso? Default is FALSE. Penalizing the diagonal inflates the latent variances ( $\Sigma_{ii} = S_{ii} + \rho$ on the covariance scale, $\Sigma_{ii} = S_{ii}(1 + \rho)$ on the correlation scale), which the VE step then feeds back into the residual covariance $S$ : along the path, the network may then never become empty, and the latent variances diverge (#180). |

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| penalty_weights | either a single or a list of $p \times p$ matrix of weights (default: all weights equal to 1) to adapt the amount of shrinkage to each pairs of node. Must be symmetric with positive values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| penalty_scale   | character, the scale on which the l1 penalty applies. "correlation" (default) penalizes the entries of the precision matrix on the scale of the variables, with a penalty $\lambda \sqrt{S_{ii}S_{jj}}$ on the pair $(i, j)$ , where $S$ is the current residual covariance. This amounts to applying the graphical-Lasso to the residual <i>correlation</i> matrix and rescaling the result, the correlation-based estimator of Rothman, Bickel, Levina and Zhu (2008), and makes the penalties dimensionless, between 0 and 1. "covariance", the only behavior until version 1.3.2, penalizes the entries of the precision matrix as they are. These are not scale invariant: a species with a large latent variance then has nearly free edges, and one that is often absent but abundant when present ends up connected to most of the others (see the field <code>degenerate_species</code> of a <code>PLNfit</code> ). See the section on the scale of the penalty. |
| latent_floor    | a positive number (default 1e-3), or NULL for no floor: a floor on the variational means of the degenerate species, as in <code>PLNnetwork_param()</code> . Only used with a sparse covariance ( <code>penalty &gt; 0</code> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 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 <code>PLNfit</code> (typically obtained from a previous fit), which sometimes speeds up the inference.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

## Details

See `PLN_param()` for a description of the generic `config_optim` entries (`ftol_rel`, `xtol_rel`, etc.). Like `PLNnetwork_param()`, `ZIPLN_param()` has two parameters controlling the outer EM loop:

- "ftol\_out" outer solver stops when an optimization step changes the objective function by less than `ftol_out` 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 200 for "builtin", 100 for "nlopt"

## Value

list of parameters used during the fit and post-processing steps

---

 ZIPLNfit

*An R6 Class to represent a ZIPLNfit*


---

## Description

The function `ZIPLN()` fits a model which is an instance of an object with class `ZIPLNfit`.

This class comes with a set of R6 methods, some of which are useful for the end-user and exported as S3 methods. See the documentation for `coef()`, `sigma()`, `predict()`.

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

## Details

Covariates for the Zero-Inflation parameter (using a logistic regression model) can be specified in the formula RHS using the pipe (`~ PLN effect | ZI effect`) to separate covariates for the PLN part of the model from those for the Zero-Inflation part. Note that different covariates can be used for each part.

## Active bindings

`n` number of samples/sites

`q` number of dimensions of the latent space

`p` number of variables/species

`d` number of covariates in the PLN part

`d0` number of covariates in the ZI part

`nb_param_zi` number of parameters in the ZI part of the model

`nb_param_pln` number of parameters in the PLN part of the model

`nb_param` number of parameters in the ZIPLN model

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

`var_par` a list with two matrices, M and S2, which are the estimated parameters in the variational approximation

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

`degenerate_species` names of the species whose latent variance is above 100, see the field of the same name in `PLNfit`

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

`zi_model` character: the model used for the zero inflation (either "single", "row", "col" or "covar")

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

loglik\_vec element-wise variational lower bound of the loglikelihood  
 AIC variational lower bound of the AIC  
 BIC variational lower bound of the BIC  
 entropy Entropy of the variational distribution  
 entropy\_ZI Entropy of the variational distribution  
 entropy\_PLN Entropy of the Gaussian variational distribution in the PLN component  
 ICL variational lower bound of the ICL  
 criteria a vector with loglik, BIC, ICL and number of parameters

## Methods

### Public methods:

- `ZIPLNfit$update()`
- `ZIPLNfit$new()`
- `ZIPLNfit$optimize()`
- `ZIPLNfit$optimize_vestep()`
- `ZIPLNfit$predict()`
- `ZIPLNfit$show()`
- `ZIPLNfit$print()`
- `ZIPLNfit$clone()`

`ZIPLNfit$update()`: Update a `ZIPLNfit` object

#### Usage:

```
ZIPLNfit$update(
  B = NA,
  B0 = NA,
  Pi = NA,
  Omega = NA,
  Sigma = NA,
  M = NA,
  S2 = NA,
  R = NA,
  Ji = NA,
  Z = NA,
  A = NA,
  monitoring = NA
)
```

#### Arguments:

B matrix of regression parameters in the Poisson lognormal component  
 B0 matrix of regression parameters in the zero inflated component  
 Pi Zero inflated probability parameter (either scalar, row-vector, col-vector or matrix)  
 Omega precision matrix of the latent variables  
 Sigma covariance matrix of the latent variables

**M** matrix of mean vectors for the variational approximation  
**S2** matrix of variance parameters for the variational approximation  
**R** matrix of probabilities for the variational approximation  
**Ji** vector of variational lower bounds of the log-likelihoods (one value per sample)  
**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 **ZIPLNfit** object

**ZIPLNfit\$new():** Initialize a **ZIPLNfit** model

*Usage:*

```
ZIPLNfit$new(data, control)
```

*Arguments:*

**data** a named list used internally to carry the data matrices  
**control** a list for controlling the optimization. See details.

**ZIPLNfit\$optimize():** Call to the Cpp optimizer and update of the relevant fields

*Usage:*

```
ZIPLNfit$optimize(data, control)
```

*Arguments:*

**data** a named list used internally to carry the data matrices  
**control** a list for controlling the optimization. See details.

**ZIPLNfit\$optimize\_vestep():** Result of one call to the VE step of the optimization procedure: optimal variational parameters (**M**, **S2**, **R**) and corresponding log likelihood values for fixed model parameters (**Sigma**, **B**, **B0**). Intended to position new data in the latent space.

*Usage:*

```

ZIPLNfit$optimize_vestep(
  data,
  B = self$model_par$B,
  B0 = self$model_par$B0,
  Omega = self$model_par$Omega,
  control = ZIPLN_param(backend = "nlopt")$config_optim
)

```

*Arguments:*

**data** a named list used internally to carry the data matrices  
**B** Optional fixed value of the regression parameters in the PLN component  
**B0** Optional fixed value of the regression parameters in the ZI component  
**Omega** inverse variance-covariance matrix of the latent variables  
**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 matrix R of variational ZI probabilities
- the vector  $J_i$  of (variational) log-likelihood of each new observation
- a list monitoring with information about convergence status

`ZIPLNfit$predict()`: Predict position, scores or observations of new data. See [predict.ZIPLNfit\(\)](#) for the S3 method and additional details

*Usage:*

```
ZIPLNfit$predict(
  newdata,
  responses = NULL,
  type = c("link", "response", "deflated"),
  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), "response" (predicted average counts, accounting for zero-inflation) or "deflated" (predicted average counts, not accounting for zero-inflation and using only the PLN part of 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.

`envir` Environment in which the prediction is evaluated

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

`ZIPLNfit$show()`: User friendly print method

*Usage:*

```
ZIPLNfit$show(
  model = paste("A multivariate Zero Inflated Poisson Lognormal fit with",
    self$vcov_model, "covariance model.\n")
)
```

*Arguments:*

`model` First line of the print output

`ZIPLNfit$print()`: User friendly print method

*Usage:*

```
ZIPLNfit$print()
```

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

*Usage:*

```
ZIPLNfit$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```
## Not run:
# See other examples in function ZIPLN
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- ZIPLN(Abundance ~ 1, data = trichoptera)
class(myPLN)
print(myPLN)

## End(Not run)
```

---

ZIPLNfit\_diagonal

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

---

**Description**

An R6 Class to represent a ZIPLNfit in a standard, general framework, with diagonal residual covariance

**Super class**

[ZIPLNfit](#) -> ZIPLNfit\_diagonal

**Active bindings**

nb\_param\_pln number of parameters in the PLN part of the current model  
vcov\_model character: the model used for the residual covariance

**Methods****Public methods:**

- [ZIPLNfit\\_diagonal\\$new\(\)](#)
- [ZIPLNfit\\_diagonal\\$clone\(\)](#)

[ZIPLNfit\\_diagonal\\$new\(\)](#): Initialize a [ZIPLNfit\\_diagonal](#) model

*Usage:*

```
ZIPLNfit_diagonal$new(data, control)
```

*Arguments:*

`data` a named list used internally to carry the data matrices  
`control` a list for controlling the optimization. See details.

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

*Usage:*

```
ZIPLNfit_diagonal$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
## Not run:
# See other examples in function ZIPLN
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- ZIPLN(Abundance ~ 1, data = trichoptera, control = ZIPLN_param(covariance = "diagonal"))
class(myPLN)
print(myPLN)

## End(Not run)
```

---

|                |                                                                                                                       |
|----------------|-----------------------------------------------------------------------------------------------------------------------|
| ZIPLNfit_fixed | <i>An R6 Class to represent a ZIPLNfit in a standard, general framework, with fixed (inverse) residual covariance</i> |
|----------------|-----------------------------------------------------------------------------------------------------------------------|

---

## Description

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

## Super class

[ZIPLNfit](#) -> ZIPLNfit\_fixed

## Active bindings

`nb_param_pln` number of parameters in the PLN part of the current model  
`vcov_model` character: the model used for the residual covariance

## Methods

### Public methods:

- [ZIPLNfit\\_fixed\\$new\(\)](#)
- [ZIPLNfit\\_fixed\\$clone\(\)](#)

`ZIPLNfit_fixed$new()`: Initialize a [ZIPLNfit\\_fixed](#) model

*Usage:*

```
ZIPLNfit_fixed$new(data, control)
```

*Arguments:*

`data` a named list used internally to carry the data matrices

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

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

*Usage:*

```
ZIPLNfit_fixed$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**Examples**

```
## Not run:
# See other examples in function ZIPLN
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- ZIPLN(Abundance ~ 1, data = trichoptera,
  control = ZIPLN_param(Omega = diag(ncol(trichoptera$Abundance))))
class(myPLN)
print(myPLN)

## End(Not run)
```

---

ZIPLNfit\_sparse

*An R6 Class to represent a ZIPLNfit in a standard, general framework,  
with sparse inverse residual covariance*

---

**Description**

An R6 Class to represent a ZIPLNfit in a standard, general framework, with sparse inverse residual covariance

**Super class**

[ZIPLNfit](#) -> ZIPLNfit\_sparse

**Active bindings**

`penalty` the global level of sparsity in the current model

`penalty_weights` a matrix of weights controlling the amount of penalty element-wise.

`penalty_scale` the scale on which the penalty applies, "covariance" or "correlation" (see [PLNnetwork\\_param\(\)](#))

`latent_floor` the lower bound on  $\exp(0 + M)$  for the species in `floored_species`, NULL if none (see [PLNnetwork\\_param\(\)](#))

`floored_species` names of the species whose variational means are bounded by `latent_floor`, as in [PLNnetworkfit](#)

`n_edges` number of edges if the network (non null coefficient of the sparse precision matrix)

`nb_param_pln` number of parameters in the PLN part of the current model

`vcov_model` character: the model used for the residual covariance

`pen_loglik` variational lower bound of the l1-penalized loglikelihood, as in [PLNnetworkfit](#)

`ebic_gamma` the tuning parameter gamma of the EBIC, as in [PLNnetworkfit](#)

`EBIC` variational lower bound of the EBIC of Foygel and Drton (2010), that is the BIC with the additional penalty  $2 \gamma |\mathcal{E}| \log(p)$  on the edge set

`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:

- [ZIPLNfit\\_sparse\\$new\(\)](#)
- [ZIPLNfit\\_sparse\\$optimize\(\)](#)
- [ZIPLNfit\\_sparse\\$latent\\_network\(\)](#)
- [ZIPLNfit\\_sparse\\$plot\\_network\(\)](#)
- [ZIPLNfit\\_sparse\\$clone\(\)](#)

`ZIPLNfit_sparse$new()`: Initialize a [ZIPLNfit\\_fixed](#) model

*Usage:*

```
ZIPLNfit_sparse$new(data, control)
```

*Arguments:*

`data` a named list used internally to carry the data matrices

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

`ZIPLNfit_sparse$optimize()`: Call to the optimizer and update of the relevant fields. Reports non-convergence of the graphical Lasso on top of [ZIPLNfit](#)'s method.

*Usage:*

```
ZIPLNfit_sparse$optimize(data, control)
```

*Arguments:*

`data` a named list used internally to carry the data matrices

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

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

*Usage:*

```
ZIPLNfit_sparse$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 ZIPLNfit\_sparse\$n

ZIPLNfit\_sparse\$plot\_network(): plot the latent network.

*Usage:*

```
ZIPLNfit_sparse$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,
  edge.alpha = 0.2,
  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.

edge.alpha opacity of the weakest edge, the strongest one being fully opaque, so that the strength of an edge can be read off a dense network. Default is 0.2. Set it to 1 for uniformly opaque edges. Only relevant for igraph output with type = "partial\_cor".

plot logical. Should the final network be displayed or only sent back to the user. Default is TRUE.

ZIPLNfit\_sparse\$clone(): The objects of this class are cloneable with this method.

*Usage:*

```
ZIPLNfit_sparse$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
## Not run:
# See other examples in function ZIPLN
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- ZIPLN(Abundance ~ 1, data = trichoptera, control= ZIPLN_param(penalty = 1))
class(myPLN)
print(myPLN)
```

```
plot(myPLN)

## End(Not run)
```

---

|                    |                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------|
| ZIPLNfit_spherical | <i>An R6 Class to represent a ZIPLNfit in a standard, general framework, with spherical residual covariance</i> |
|--------------------|-----------------------------------------------------------------------------------------------------------------|

---

## Description

An R6 Class to represent a ZIPLNfit in a standard, general framework, with spherical residual covariance

## Super class

[ZIPLNfit](#) -> ZIPLNfit\_spherical

## Active bindings

`nb_param_pln` number of parameters in the PLN part of the current model  
`vcov_model` character: the model used for the residual covariance

## Methods

### Public methods:

- [ZIPLNfit\\_spherical\\$new\(\)](#)
- [ZIPLNfit\\_spherical\\$clone\(\)](#)

`ZIPLNfit_spherical$new()`: Initialize a [ZIPLNfit\\_spherical](#) model

*Usage:*

```
ZIPLNfit_spherical$new(data, control)
```

*Arguments:*

`data` a named list used internally to carry the data matrices  
`control` a list for controlling the optimization. See details.

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

*Usage:*

```
ZIPLNfit_spherical$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
## Not run:
# See other examples in function ZIPLN
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myPLN <- ZIPLN(Abundance ~ 1, data = trichoptera, control = ZIPLN_param(covariance = "spherical"))
class(myPLN)
print(myPLN)

## End(Not run)
```

---

ZIPLNnetwork

---

*Zero Inflated Sparse Poisson lognormal model for network inference*


---

## Description

Perform sparse inverse covariance estimation for the Zero Inflated Poisson lognormal model using a variational algorithm. Iterate over a range of logarithmically spaced sparsity parameter values. Use the (g)lm syntax to specify the model (including covariates and offsets).

## Usage

```
ZIPLNnetwork(
  formula,
  data,
  subset,
  weights,
  zi = c("single", "row", "col"),
  penalties = NULL,
  control = ZIPLNnetwork_param()
)
```

## Arguments

|                      |                                                                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>formula</code> | an object of class "formula": a symbolic description of the model to be fitted.                                                                                                                                                                                                                         |
| <code>data</code>    | an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the model is called. |
| <code>subset</code>  | an optional vector specifying a subset of observations to be used in the fitting process.                                                                                                                                                                                                               |
| <code>weights</code> | an optional vector of observation weights to be used in the fitting process.                                                                                                                                                                                                                            |
| <code>zi</code>      | a character describing the model used for zero inflation, either of <ul style="list-style-type: none"> <li>• "single" (default, one parameter shared by all counts)</li> <li>• "col" (one parameter per variable / feature)</li> </ul>                                                                  |

- "row" (one parameter per sample / individual). If covariates are specified in the formula RHS (see details) this parameter is ignored.
- 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. With the default `penalty_scale = "correlation"`, the penalties apply on the scale of the correlations and lie between 0 and 1; until version 1.3.2 they were on the covariance scale, which `PLNnetwork_param(penalty_scale = "covariance")` gives back. When penalties above 1 are given without choosing the scale, they are taken as penalties on the covariance scale and converted, with a warning (see [PLNnetwork\\_param\(\)](#)).
- control        a list-like structure for controlling the optimization, with default generated by [ZIPLNnetwork\\_param\(\)](#). See the associated documentation for details.

### Details

Covariates for the Zero-Inflation parameter (using a logistic regression model) can be specified in the formula RHS using the pipe (`~ PLN effect | ZI effect`) to separate covariates for the PLN part of the model from those for the Zero-Inflation part. Note that different covariates can be used for each part.

### Value

an R6 object with class [ZIPLNnetworkfamily](#)

### See Also

The classes [ZIPLNfit](#) and [ZIPLNnetworkfamily](#)

### Examples

```
data(trichoptera)
trichoptera <- prepare_data(trichoptera$Abundance, trichoptera$Covariate)
myZIPLNs <- ZIPLNnetwork(Abundance ~ 1, data = trichoptera, zi = "single")
```

---

|                    |                                    |
|--------------------|------------------------------------|
| ZIPLNnetwork_param | <i>Control of ZIPLNnetwork fit</i> |
|--------------------|------------------------------------|

---

### Description

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

**Usage**

```
ZIPLNnetwork_param(
  backend = c("builtin", "nlopt"),
  inception_cov = c("full", "spherical", "diagonal"),
  trace = 1,
  n_penalties = 30,
  min_ratio = 0.1,
  penalize_diagonal = FALSE,
  penalty_weights = NULL,
  penalty_scale = c("correlation", "covariance"),
  latent_floor = 0.001,
  config_post = list(),
  config_optim = list(),
  inception = NULL
)
```

**Arguments**

- |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| backend           | optimization backend, either "builtin" (default, joint Newton on $(M, \psi, R)$ , combined with the partial E-step $\text{maxit\_ve} = 1$ ) or "nlopt" (CCSAQ). "builtin" consistently finds a better ELBO across the penalty path at the cost of being slower.                                                                                                                                                                                         |
| inception_cov     | Covariance structure used for the inception ZIPLN, which starts the penalty path and sets its top: "full" (default), "diagonal" or "spherical". The top of the grid of penalties is computed as in <code>PLNnetwork_param()</code> . Unlike for <code>PLNnetwork()</code> , a full inception is the default: a diagonal one makes the top of the path exactly the empty network, but was found to lead to a lower BIC along the path, at a higher cost. |
| 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 FALSE. Penalizing the diagonal inflates the latent variances ( $\Sigma_{ii} = S_{ii} + \rho$ on the covariance scale, $\Sigma_{ii} = S_{ii}(1 + \rho)$ on the correlation scale), which the VE step then feeds back into the residual covariance $S$ : along the path, the network may then never become empty, and the latent variances diverge (#180).             |
| penalty_weights   | either a single or a list of $p \times p$ matrix of weights (default: all weights equal to 1) to adapt the amount of shrinkage to each pairs of node. Must be symmetric with positive values.                                                                                                                                                                                                                                                           |
| penalty_scale     | character, the scale on which the l1 penalty applies. "correlation" (default) penalizes the entries of the precision matrix on the scale of the variables, with a penalty $\lambda \sqrt{S_{ii}S_{jj}}$ on the pair $(i, j)$ , where $S$ is the current residual covariance.                                                                                                                                                                            |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <p>This amounts to applying the graphical-Lasso to the residual <i>correlation</i> matrix and rescaling the result, the correlation-based estimator of Rothman, Bickel, Levina and Zhu (2008), and makes the penalties dimensionless, between 0 and 1. "covariance", the only behavior until version 1.3.2, penalizes the entries of the precision matrix as they are. These are not scale invariant: a species with a large latent variance then has nearly free edges, and one that is often absent but abundant when present ends up connected to most of the others (see the field <code>degenerate_species</code> of a <code>PLNfit</code>). See the section on the scale of the penalty.</p>                                                                                                                                                                                                                                         |
| <code>latent_floor</code> | <p>a positive number <math>\epsilon</math> (default <math>1e-3</math>), or NULL for no floor: a floor on the variational means of the degenerate species. As soon as the latent variance of a species exceeds the threshold of the field <code>degenerate_species</code> of a <code>PLNfit</code> (100, see <code>options(PLNmodels.latent_variance_threshold = )</code>) during the optimization, its variational means are kept above <math>\log \epsilon - O</math>, that is <math>\exp(O + M) \geq \epsilon</math>, in this fit and in the following ones along the penalty path. The species concerned are in the field <code>floored_species</code> of the fits. This needs no preparation of the data: the species are found by the optimization itself. The floor restricts the variational family, not the model, and stops the latent variances from diverging. It leaves a fit without degenerate species exactly as it is.</p> |
| <code>config_post</code>  | <p>a list for controlling the post-treatments (optional bootstrap, jackknife, R2, etc.). See details</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>config_optim</code> | <p>a list for controlling the optimizer (either "nlopt" or "torch" backend). See details</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>inception</code>    | <p>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 <code>PLNfit</code> (typically obtained from a previous fit), which sometimes speeds up the inference.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## Details

See `PLNnetwork_param()` for a full description of the optimization parameters. Note that some defaults values are different than those used in `PLNnetwork_param()`:

- "ftol\_out" (outer loop convergence tolerance the objective function) is set by default to  $1e-6$
- "maxit\_out" (max number of iterations for the outer loop) is set by default to 50

## Value

list of parameters configuring the fit.

## References

Rothman, A. J., Bickel, P. J., Levina, E. and Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2, 494–515. doi:10.1214/08EJS176

## See Also

`PLNnetwork_param()` and `PLN_param()`

---

ZIPLNnetworkfamily      *An R6 Class to represent a collection of ZIPLNnetwork*

---

## Description

The function `ZIPLNnetwork()` 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 classes

`PLNfamily` -> `Networkfamily` -> `ZIPLNnetworkfamily`

## Public fields

`covariates0` the matrix of covariates included in the ZI component

## Methods

### Public methods:

- `ZIPLNnetworkfamily$new()`
- `ZIPLNnetworkfamily$stability_selection()`
- `ZIPLNnetworkfamily$clone()`

`ZIPLNnetworkfamily$new()`: Initialize all models in the collection

*Usage:*

```
ZIPLNnetworkfamily$new(penalties, data, control)
```

*Arguments:*

`penalties` a vector of positive real number controlling the level of sparsity of the underlying network.

`data` a named list used internally to carry the data matrices

`control` a list for controlling the optimization.

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

`ZIPLNnetworkfamily$stability_selection()`: Compute the stability path by stability selection

*Usage:*

```
ZIPLNnetworkfamily$stability_selection(
  subsamples = NULL,
  control = ZIPLNnetwork_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 controlling the main optimization process in each call to `PLNnetwork()`. See `ZIPLNnetwork()` and `ZIPLN_param()` for details.

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

*Usage:*

```
ZIPLNnetworkfamily$clone(deep = FALSE)
```

*Arguments:*

*deep* Whether to make a deep clone.

### See Also

The function `ZIPLNnetwork()`, the class `ZIPLNfit_sparse`

### Examples

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

# Index

- \* **datasets**
  - barents, [5](#)
  - microcosm, [25](#)
  - mollusk, [26](#)
  - oaks, [30](#)
  - scRNA, [113](#)
  - trichoptera, [120](#)
- AIC.PLNfit, [4](#)
- AIC.ZIPLNfit, [5](#)
- barents, [5](#)
- BIC.PLNfit, [6](#)
- BIC.ZIPLNfit, [7](#)
- coef(), [37](#), [126](#)
- coef.PLNfit, [7](#)
- coef.PLNfit(), [114](#), [121](#), [122](#)
- coef.PLNLDAfit, [8](#)
- coef.PLNmixturefit, [9](#)
- coef.PLNmixturefit(), [115](#)
- coef.ZIPLNfit, [10](#)
- coef.ZIPLNfit(), [116](#)
- coefficient\_path, [11](#)
- compute\_offset, [11](#)
- compute\_offset(), [111](#), [112](#)
- compute\_PLN\_starting\_point, [13](#)
- compute\_PLN\_starting\_point(), [33](#)
- compute\_ZIPLN\_starting\_point, [14](#)
- extract\_probs, [15](#)
- factoextra::fviz(), [86](#)
- fitted.PLNfit, [16](#)
- fitted.PLNmixturefit, [17](#)
- fitted.ZIPLNfit, [17](#)
- getBestModel
  - (getBestModel.PLNPCAfamily), [18](#)
- getBestModel(), [27](#), [64](#), [76](#), [85](#), [140](#)
- getBestModel.PLNPCAfamily, [18](#)
- getModel(getModel.PLNPCAfamily), [19](#)
- getModel(), [27](#), [37](#), [64](#), [76](#), [85](#), [140](#)
- getModel.PLNPCAfamily, [19](#)
- ggplot2::ggplot, [29](#), [30](#), [36](#), [55](#), [65](#), [69](#), [87](#), [93](#), [98–100](#), [103](#)
- graphical\_lasso, [20](#)
- grob, [55](#), [93](#)
- ICL, [23](#)
- logLik.PLNfit, [24](#)
- logLik.ZIPLNfit, [24](#)
- microcosm, [25](#)
- mollusk, [26](#)
- Networkfamily, [11](#), [27](#), [28](#), [29](#), [76](#), [117](#), [140](#)
- oaks, [30](#)
- PLN, [31](#)
- PLN(), [7](#), [9](#), [14](#), [16](#), [37](#), [56](#), [91](#), [106](#), [111](#), [121](#)
- PLN\_param, [32](#)
- PLN\_param(), [32](#), [39](#), [40](#), [49](#), [64](#), [76](#), [77](#), [117](#), [125](#), [139](#)
- PLNfamily, [27](#), [35](#), [36](#), [64](#), [76](#), [85](#), [89–92](#), [96](#), [140](#)
- PLNfit, [4](#), [6](#), [8](#), [16](#), [23](#), [24](#), [32](#), [35](#), [36](#), [37](#), [37–39](#), [43–48](#), [52](#), [57–59](#), [73](#), [78](#), [83](#), [86](#), [88](#), [89](#), [105](#), [110](#), [114](#), [119](#), [122](#), [125](#), [126](#), [139](#)
- PLNfit(), [37](#), [52](#), [57](#), [58](#)
- PLNfit\_diagonal, [43](#)
- PLNfit\_fixedcov, [44](#), [78](#), [119](#)
- PLNfit\_genpop, [46](#), [46](#)
- PLNfit\_spherical, [47](#)
- PLNLDA, [48](#), [56](#)
- PLNLDA(), [8](#), [37](#), [52](#), [57](#), [58](#)
- PLNLDA\_param, [49](#)
- PLNLDA\_param(), [49](#)
- PLNLDAfit, [49](#), [52](#), [52](#), [53](#), [55](#), [57](#), [58](#), [93](#), [106](#)

- PLNLDAfit(), [49](#)
- PLNLDAfit\_diagonal, [57](#)
- PLNLDAfit\_spherical, [58](#)
- PLNmixture, [60](#), [66](#), [70](#)
- PLNmixture(), [17](#), [37](#), [64](#)
- PLNmixture\_param, [61](#)
- PLNmixture\_param(), [60](#), [68](#), [107](#)
- PLNmixturefamily, [19](#), [20](#), [60](#), [64](#), [70](#), [98](#)
- PLNmixturefit, [9](#), [17](#), [60](#), [66](#), [66–68](#), [99](#), [107](#), [115](#), [119](#)
- PLNnetwork, [70](#)
- PLNnetwork(), [11](#), [15](#), [20–22](#), [27](#), [30](#), [37](#), [76–78](#), [80](#), [117](#), [138](#), [141](#)
- PLNnetwork\_param, [71](#)
- PLNnetwork\_param(), [71](#), [78](#), [125](#), [132](#), [137–139](#)
- PLNnetworkfamily, [15](#), [19](#), [20](#), [35](#), [71](#), [76](#), [80](#), [94](#), [95](#), [117](#)
- PLNnetworkfit, [18](#), [20](#), [27](#), [30](#), [71](#), [76](#), [77](#), [78](#), [78](#), [79](#), [100](#), [119](#), [133](#), [140](#)
- PLNPCA, [81](#), [94](#)
- PLNPCA(), [37](#), [85](#), [87](#), [88](#)
- PLNPCA\_param, [82](#)
- PLNPCA\_param(), [81](#)
- PLNPCAfamily, [18–20](#), [35](#), [81](#), [85](#), [89](#), [94](#), [102](#)
- PLNPCAfit, [18](#), [20](#), [81](#), [87](#), [88](#), [88–90](#), [93](#), [97](#), [102](#), [118](#)
- PLNPCAfit(), [87](#)
- plot(), [27](#), [52](#), [57](#), [58](#), [64](#), [76](#), [78](#), [85](#), [88](#), [140](#)
- plot.Networkfamily, [94](#)
- plot.PLNfamily, [96](#)
- plot.PLNLDAfit, [97](#)
- plot.PLNmixturefamily, [98](#)
- plot.PLNmixturefit, [99](#)
- plot.PLNnetworkfamily (plot.Networkfamily), [94](#)
- plot.PLNnetworkfamily(), [97](#)
- plot.PLNnetworkfit, [100](#)
- plot.PLNPCAfamily, [101](#)
- plot.PLNPCAfamily(), [97](#)
- plot.PLNPCAfit, [102](#)
- plot.ZIPLNfit\_sparse, [104](#)
- plot.ZIPLNnetworkfamily (plot.Networkfamily), [94](#)
- predict(), [37](#), [52](#), [57](#), [58](#), [126](#)
- predict.PLNfit, [105](#)
- predict.PLNLDAfit, [106](#)
- predict.PLNmixturefit, [107](#)
- predict.ZIPLNfit, [108](#)
- predict.ZIPLNfit(), [129](#)
- predict\_cond, [109](#)
- prepare\_data, [110](#)
- prepare\_data(), [26](#), [27](#), [31](#), [119–121](#)
- rPLN, [112](#)
- scRNA, [113](#)
- sigma(), [37](#), [126](#)
- sigma.PLNfit, [114](#)
- sigma.PLNfit(), [8](#), [121](#), [122](#)
- sigma.PLNmixturefit, [115](#)
- sigma.PLNmixturefit(), [9](#)
- sigma.ZIPLNfit, [116](#)
- sigma.ZIPLNfit(), [10](#)
- stability\_selection, [116](#)
- stability\_selection(), [15](#), [18](#)
- standard\_error (standard\_error.PLNPCAfit), [117](#)
- standard\_error(), [37](#)
- standard\_error.PLNfit(), [8](#), [114](#), [122](#)
- standard\_error.PLNPCAfit, [117](#)
- stats::AIC(), [24](#)
- stats::BIC(), [24](#)
- structural\_zeros, [119](#)
- structural\_zeros(), [38](#), [111](#), [112](#)
- trichoptera, [120](#)
- vcov(), [37](#)
- vcov.PLNfit, [121](#)
- vcov.PLNfit(), [8](#), [114](#), [119](#)
- ZIPLN, [122](#)
- ZIPLN(), [10](#), [17](#), [126](#)
- ZIPLN\_param, [124](#)
- ZIPLN\_param(), [117](#), [123](#), [141](#)
- ZIPLNfit, [5](#), [7](#), [10](#), [17](#), [23](#), [24](#), [108](#), [116](#), [123](#), [126](#), [126–128](#), [130–133](#), [135](#), [137](#)
- ZIPLNfit\_diagonal, [130](#), [130](#)
- ZIPLNfit\_fixed, [131](#), [131](#), [133](#)
- ZIPLNfit\_sparse, [27](#), [30](#), [104](#), [132](#), [141](#)
- ZIPLNfit\_spherical, [135](#), [135](#)
- ZIPLNnetwork, [136](#)
- ZIPLNnetwork(), [20](#), [27](#), [30](#), [117](#), [140](#), [141](#)
- ZIPLNnetwork\_param, [137](#)
- ZIPLNnetwork\_param(), [137](#)
- ZIPLNnetworkfamily, [19](#), [20](#), [94](#), [95](#), [117](#), [137](#), [140](#)