# Package 'penfa'

July 23, 2025

**Title** Single- And Multiple-Group Penalized Factor Analysis

**Version** 0.1.1

**Description** Fits single- and multiple-group penalized factor analysis models via a trust-region algorithm with integrated automatic multiple tuning parameter selection (Geminiani et al., 2021 <doi:10.1007/s11336-021-09751-8>). Available penalties include lasso, adaptive lasso, scad, mcp, and ridge.

**License** GPL-3

**Depends** R(>= 3.5.0)

**Imports** MASS, methods, mgcv, GJRM, stats, trust, utils

**Suggests** cartography, knitr, plotly, rmarkdown

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** true

**URL** https://github.com/egeminiani/penfa

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Elena Geminiani [aut, cre] (ORCID: <https://orcid.org/0000-0001-5992-9728>), Giampiero Marra [aut] (ORCID: <https://orcid.org/0000-0002-9010-2646>), Irini Moustaki [aut] (ORCID: <https://orcid.org/0000-0001-8371-1251>)

**Maintainer** Elena Geminiani <geminianielena@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-07-17 05:50:03 UTC

## Contents

---

`penfa-package` *penfa: Single- and Multiple-Group Penalized Factor Analysis*

---

## Description

The `penfa` package (a short form for *PENalized Factor Analysis*) provides several routines for single- and multiple-group penalized factor analysis for continuous data. The models are estimated via a trust-region algorithm with integrated automatic multiple tuning parameter selection. The available penalties include lasso, adaptive lasso, scad, mcp, and ridge.

The main function of the package is `penfa`. To learn more about it, start with the vignettes and tutorials at `browseVignettes(package = "penfa")` and https://egeminiani.github.io/penfa/articles/.

## Details

Penalized factor analysis allows to produce parsimonious models using largely an automated procedure. In the single-group case, a typical penalty function will automatically shrink a subset of the factor loadings to zero. The use of sparsity-inducing penalty functions leads to optimally sparse factor structures supported by the data. The resulting models are less prone to instability in the estimation process and are easier to interpret and generalize than their unpenalized counterparts.

In the multiple-group scenario, penalized factor analysis can be used to automatically ascertain differences and similarities of parameter estimates across groups. Typical penalties will automatically encourage sparse loading matrices and invariant factor loadings and intercepts.

In `penfa`, estimation is achieved via a penalized likelihood-based framework that builds upon differentiable approximations of non-differentiable penalties, a theoretically founded definition of degrees of freedom, and an algorithm with integrated automatic multiple tuning parameter selection. The estimation is based on a trust-region algorithm approach exploiting second-order analytical

derivative information. The standard errors for the model parameters are derived using a Bayesian approach.

The selection of the tuning parameters is a crucial issue in penalized estimation strategies, as the tuning parameters are responsible for the optimal balance between goodness of fit and sparsity. In penfa, the optimal values of the tuning parameters can be determined through the automatic procedure or grid-searches.

In addition to the fitting function penfa, the package provides several methods for examining the parameter estimates, monitoring the optimization process, and inspecting the structures of the penalty matrices through interactive visualizations.

## Author(s)

Authors: Elena Geminiani, Giampiero Marra, Irini Moustaki

Maintainer: Elena Geminiani. Please address any query or comment to <geminianielena@gmail.com>.

## References

Geminiani, E., Marra, G., & Moustaki, I. (2021). "Single- and Multiple-Group Penalized Factor Analysis: A Trust-Region Algorithm Approach with Integrated Automatic Multiple Tuning Parameter Selection." Psychometrika, 86(1), 65-95. doi: 10.1007/s11336021097518

Geminiani E. (2020), "A penalized likelihood-based framework for single and multiple-group factor analysis models" (Doctoral dissertation, University of Bologna). Available at http://amsdottorato.unibo.it/9355/.

## See Also

penfa, penfa-class

---

ccdata *Data set for cross-cultural analysis*

---

## Description

A data set for cross-cultural analysis containing the standardized ratings to 12 items concerning organizational citizenship behavior. Employees from different countries were asked to rate their attitudes towards helping other employees and giving suggestions for improved work conditions. The items are thought to measure two latent factors: helping behavior (first seven items) and voice behavior (last five items). See below for details.

## Usage

```
ccdata
```

## Format

A data frame with 767 rows and 13 variables:

**country** Character. Country of origin of the employee: Lebanon ("LEB") or Taiwan ("TAIW").

**h1** Numeric. Standardized ratings to the item "*I volunteer to do things for this organization.*"

**h2** Numeric. Standardized ratings to the item "*I help orient new employees in this organization.*"

**h3** Numeric. Standardized ratings to the item "*I attend functions that help this organization.*"

**h4** Numeric. Standardized ratings to the item "*I help others in this work group with their work for the benefit of the group.*"

**h5** Numeric. Standardized ratings to the item "*I get involved to benefit this organization.*"

**h6** Numeric. Standardized ratings to the item "*I help others in this organization learn about the work.*"

**h7** Numeric. Standardized ratings to the item "*I help others in this organization with their work responsibilities.*"

**v1** Numeric. Standardized ratings to the item "*I develop and make recommendations concerning issues that affect this organization.*"

**v2** Numeric. Standardized ratings to the item "*I speak up and encourage other in this organization to get involved in issues that affect the group.*"

**v3** Numeric. Standardized ratings to the item "*I communicate my opinions about work issues to others in this organization even if my opinion is different and others in the organization disagree with me.*"

**v4** Numeric. Standardized ratings to the item "*I keep well informed about issues where my opinion might be useful to this organization.*"

**v5** Numeric. Standardized ratings to the item "*I speak up in this organization with ideas for new projects or changes in procedures.*"

## Details

The original data come from the `ccpsyc` package. For convenience, the following pre-processing has been applied:

- The data were filtered to only include employees from Lebanon and Taiwan.
- The answers, originally on a 7-point Likert scale, were standardized.
- The items were renamed as described above.

## Source

The original data set is available from the `ccpsyc` package. Please refer to Fischer and Karl (2019) and Fischer et al. (2019) for a description and analysis of these data.

coef,penfa-method *Coefficients from a* penfa *object*

## Description

An S4 method returning the estimates of the model parameters.

## Usage

```
## S4 method for signature 'penfa'
coef(object, type = "free", labels = TRUE)
```

## Arguments

| | |
|---|---|
| object | An object of class penfa, found as a result of a call to penfa. |
| type | Character. If type="free", only the estimated parameters (both penalized and unpenalized) are returned. If type="user", all parameters listed in the parameter table are returned, including fixed parameters. |
| labels | Logical. If TRUE, parameters are returned with their names. |

## Value

A numeric vector of class penfa.vector containing the estimated model parameters.

## See Also

penfa, penfa-class

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto")

coef(alasso_fit)
```

`fitted,penfa-method` *Model-implied moments for a* `penfa` *object*

## Description

An S4 method returning the model-implied moments for an object of class `penfa`. For every group, a list with the model-implied moments is returned.

## Usage

```
## S4 method for signature 'penfa'
fitted(object, labels = TRUE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `penfa`, found as a result of a call to `penfa`. |
| `labels` | Logical. If `TRUE`, the model-implied moments are named according to the item names used in the model syntax. |

## Value

A list of the model-implied moments for each group: `cov` contains the implied covariance matrix, and `mean` the implied mean vector. If just the covariance matrix is analyzed, only the `cov` argument is returned.

## See Also

`penfa`, `penfa-class`

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
```

```
                    strategy = "auto")

fitted(alasso_fit)
```

---

penfa *Single- and multiple-group penalized factor analysis*

---

## Description

The function penfa fits single- and multiple-group *PENalized Factor Analysis* models via a trust-region algorithm with integrated automatic multiple tuning parameter selection.

In a single-group analysis, penfa can automatically shrink a subset of the factor loadings to zero. In a multiple-group analysis, it can encourage sparse loading matrices and invariant factor loadings and intercepts. The currently supported penalty functions are lasso, adaptive lasso, scad, mcp, and ridge. Except for the latter, all penalties can achieve sparsity.

## Usage

```
penfa(
  model = NULL,
  data = NULL,
  group = NULL,
  pen.shrink = "alasso",
  pen.diff = "none",
  eta = list(shrink = c(lambda = 0.01), diff = c(none = 0)),
  strategy = "auto",
  ...
)
```

## Arguments

| | |
|---|---|
| model | A description of a user-specified model. It takes the form of a lavaan-like model syntax. See below for additional details on how to specify a model syntax. |
| data | A data frame containing the (continuous) observed variables used in the model. Except for the group variable, all variables are treated as numeric. |
| group | Character. An optional variable name in the data frame defining the groups in a multiple-group analysis. |
| pen.shrink | Character. The type of penalty function used for shrinking a subset of the model parameters (see the eta argument for details on how to specify which model parameters shall be penalized). Possible values for pen.shrink are "lasso", "alasso" (i.e., adaptive lasso), "scad" (i.e., smoothly clipped absolute deviation), "mcp" (i.e., minimax concave penalty), "ridge", and "none" in case of no shrinkage penalization. |

`pen.diff` Character. The type of penalty function used for shrinking certain parameter differences across groups, and thus encouraging parameter equivalence across groups (see the `eta` argument for details on how to specify which model parameters shall be encouraged to be equivalent). Possible values for `pen.diff` are "lasso", "alasso" (i.e., adaptive lasso), "scad" (i.e., smoothly clipped absolute deviation), "mcp" (i.e., minimax concave penalty), "ridge", and "none" in case of no difference penalization. Note that the specification of `pen.diff` is only valid for multiple-group factor analyses when a `group` variable is defined. If a difference penalty is requested, the groups must have the same parameters.

`eta` A named list containing the starting value(s) of the tuning parameter(s) if the automatic procedure is requested (`strategy = "auto"`) or the fixed value(s) of the tuning parameter(s) to be used during optimization if `strategy = "fixed"`. The list has two components with names "shrink" and "diff", which refer to the tuning parameters to be used for shrinkage and group equivalence, respectively. The components of the list are, in turn, named vectors specifying the type of parameter matrices or vectors to be penalized. Common choices are "lambda" for the loading matrix and "tau" for the intercept vector of the observed variables. Other possible values are "phi" for the factor covariance matrix, "psi" for the covariance matrix of the unique factors, and "kappa" for the factor means. All non-fixed elements of the specified matrix/vector are penalized. When `strategy = "fixed"` and the tuning values in `eta` are equal to zero, specifying both list names as "none" results in ordinary maximum likelihood estimation (no penalization).

`strategy` Character. The strategy used for the selection of the tuning parameter(s). If `strategy = "auto"`, the optimal values of the tuning parameters are determined via an automatic tuning parameter procedure; if `strategy = "fixed"`, a penalized factor model with the values of the tuning parameters stored in the option `eta` is estimated.

`...` Additional options that can be defined using `name = "value"`. For a complete list, please refer to `penfaOptions`.

## Value

An object of class `penfa`, for which several methods are available. See the manual pages of `summary,penfa-method`, `show,penfa-method`, `coef,penfa-method`, and `fitted,penfa-method` for details.

## Data set vs Sample Moments

The `penfa` function currently takes as input a data set, as opposed to the sample moments (i.e., covariance matrices and mean vectors). Future implementations will allow `penfa` to additionally take as input sample covariance matrices and sample means. For now, if only sample moments are available, users can generate multivariate data from those sample moments, and apply the `penfa` function on the generated data.
All variables (except for the `group` variable in multiple-group analyses) are treated as continuous. Categorical items are not currently supported.

### Model syntax

The model syntax in the `model` argument describes the factor analysis model to be estimated, and specifies the relationships between the observed and latent variables (i.e., the common factors). To facilitate its formulation, the rules for the syntax specification broadly follow the ones in the `lavaan` package.

The model syntax is composed of one or multiple formula-like expressions describing specific parts of the model. The model syntax can be specified as a literal string enclosed by single quotes as in the example below.

```
model_syntax <- '
                # Common factors
                factor1 =~ x1 + x2 + x3 + x4 + x5 + x6
                factor2 =~ x1 + x2 + x3 + x4 + x5 + x6

                # Factor variances and covariances
                factor1 ~~ factor1
                factor1 ~~ factor2

                # Unique variances and covariances
                x1 ~~ x1
                x1 ~~ x2

                # Intercepts and factor means
                x1 ~ 1
                factor1 ~ 1
                '
```

Blank lines and comments can be used in between formulas, and formulas can be split over multiple lines. Multiple formulas can be placed on a single line if they are separated by a semicolon (;).

The current implementation allows for the following types of formula-like expressions in the model syntax:

1. Common factors: The `"=~"` operator can be used to define the continuous common factors (latent variables). The name of the factor (e.g., factor1) is on the left of the `"=~"` operator, whereas the terms on the right (e.g., `x1 + x2 + x3 + x4 + x5 + x6`), separated by `"+"` operators, are the indicators of the factor. The operator `"=~"` can be read as "is measured by".
2. Variances and covariances: The `"~~"` ("double tilde") operator specifies the (residual) variance of an observed or latent variable, or a set of covariances between one variable, and several other variables (either observed or latent). The distinction between variances and residual variances is made automatically. Covariances between unique factors are currently only allowed when `information = "fisher"`.
3. Intercepts and factor means: We can specify an intercept for an observed variable (`x1 ~ 1`) or a common factor (`factor1 ~ 1`). The variable name appears on the left of the `"~"` operator. On the right-hand side, there is the number "1", which stands for the intercept/mean. Including an intercept/mean formula in the model automatically implies `meanstructure = TRUE`. The distinction between observed variable intercepts and factor means is made automatically.

Usually, only a single variable name appears on the left side of an operator. However, if multiple variable names are specified, separated by the "+" operator, the formula is repeated for each element on the left side. For instance, the formula

```
x1 + x2 + x3 + x4 ~ 1
```

specifies an intercept for variables `x1`, `x2`, `x3` and `x4`.

On the right-hand side of these formula-like expressions, each element can be modified (using the `"*"` operator) by a numeric constant or the special function start(). This provides the user with a mechanism to fix parameters and provide alternative starting values, respectively. All `"*"` expressions are referred to as modifiers, and are explained in detail in the sections below.

Each parameter in a model is automatically given a name consisting of three parts, that are coerced to a single character vector. The first part is the name of the variable on the left-hand side of the formula where the parameter is implied. The middle part is based on the special "operator" used in the formula (e.g., `"=~"`, `"~"` or `"~~"`). The third part is the name of the variable on the right-hand side of the formula where the parameter is implied, or "1" if it is an intercept. The three parts are pasted together in a single string. For example, the name of the factor loading of `x2` on `factor1` is the string `"factor1~x2"`. The name of the parameter corresponding to the factor covariance between `factor1` and `factor2` is the string `"factor1~~factor2"`.

**Fixing parameters:**

It is often desirable to fix a model parameter that is otherwise (by default) estimated. Any parameter in a model can be fixed by using a modifier resulting in a numerical constant. For instance:

- Fixing factor loadings for scale setting or identification restrictions:

```
factor1 ~ 0.8*x1 + x2 + x3 +   0*x4 + x5 + x6
factor2 ~   0*x1 + x2 + x3 + 0.8*x4 + x5 + x6
```

- Specifying an orthogonal (zero) covariance between two factors:

```
factor1 ~~ 0*factor2
```

Notice that multiplying a certain parameter by `NA` forces it to be estimated.

**Starting values:**

User-defined starting values can be provided through the special function start(), containing a numeric constant. For instance, the formula below provides a starting value equal to 0.8 to the loading of `x2` on `factor1`.

```
factor1 ~ x1 + start(0.8)*x2 + x3 + x4 + x5 + x6
```

**Multiple groups:**

In a multiple group factor analysis, the modifiers containing a single element should be replaced by a vector of the same length as the number of groups. If a single element is provided, it is used for all groups. In the example below with two groups, the factor loadings of `x1` on `factor1` are fixed to 0.8 in both groups, whereas the factor loadings of `x4` are fixed to 0.75 and 0.85 in the first and second group, respectively.

```
multigroup_syntax <- '
 factor1 ~  0.8*x1 + x2 + x3 +               x4 + x5 + x6
 factor2 ~      x1 + x2 + x3 + c(0.75, 0.85)*x4 + x5 + x6 '
```

## Algorithm

Penalized factor analysis allows to produce parsimonious models using largely an automated procedure. The use of sparsity-inducing penalty functions leads to optimally sparse factor structures supported by the data. The resulting models are less prone to instability in the estimation process and are easier to interpret and generalize than their unpenalized counterparts. Multiple-group penalized factor analysis can be used to automatically ascertain the differences and similarities of parameter estimates across groups.

In `penfa`, estimation is achieved via a penalized likelihood-based framework that builds upon differentiable approximations of non-differentiable penalties, a theoretically founded definition of degrees of freedom, and an algorithm with automatic multiple tuning parameter selection (see section below for details).

The `penfa` function uses a `trust-region` algorithm approach. This strategy constructs a model function whose behavior near the current point and within a trust-region (usually a ball) is similar to that of the actual objective function. The algorithm exploits second-order analytical derivative information. This can come in the form of the penalized Hessian matrix (if `information = "hessian"`) or the penalized Fisher information matrix (if `information = "fisher"`). Models with a mean structure can be only estimated with the penalized Fisher information matrix, which exhibits similar performances to the penalized Hessian at a reduced computational cost.

## Tuning parameter selection

The selection of the tuning parameters is a crucial issue in penalized estimation strategies, as the tuning parameters are responsible for the optimal balance between goodness of fit and sparsity.

**Automatic procedure:**

The penalized framework discussed above is easily integrated with automatic multiple tuning parameter selection (if `strategy = "auto"`). The tuning parameters are chosen to minimize an approximate AIC. See below for additional details on how to introduce more sparsity, if desired. The automatic procedure is fast, efficient, and scales well with the number of tuning parameters. It also eliminates the need for time-consuming and computationally intensive grid-searches.

**Note:** Only lasso, adaptive lasso and ridge penalties can be used with the automatic procedure.

The automatic procedure returns the optimal value of the tuning parameter. Notice, however, that the parameter estimates from this model will slightly differ from the ones one would obtain by setting `strategy = "fixed"` and `eta` equal to that optimal tuning value. This is due to the different starting values employed in the two scenarios. In the automatic procedure, the starting values of the final model come from the ones of the previous model in the optimization loop; in the fixed-tuning context, the starting values come from the default ones in `penfa`.

**Grid-search:**

If `strategy = "fixed"`, `penfa` estimates a penalized factor model with the value of the tuning parameter stored in `eta`. This is useful if users wish to make multiple calls to the `penfa` function using a range of values for the tuning parameter. Then, the optimal penalized model can be picked on the basis of information criteria, which are easily computed by calling the `AIC` and `BIC` functions. It is often convenient to use the (Generalized) Bayesian Information Criterion as a selector, due to its recurrent use in sparse settings.

These information criteria use the theoretical definition of the effective degrees of freedom (*edf*) as their bias terms. This is because the use of differentiable penalty approximations make the

objective function twice-continuously differentiable. The total `edf` are as the sum of the effective degree of freedom for each model parameter, which in turn ranges from 0 to 1 and quantifies the extend to which each parameter has been penalized.

## Penalization

The `penfa` function penalizes every element in the parameter matrix/vector specified in the `eta` argument. For instance, if `eta = list("shrink" = c("lambda" = 0.01), "diff" = c("none" = 0))` all factor loadings are penalized through a shrinkage penalty.

### Choosing the penalty function:

It may be beneficial to try out different penalties, and see which one works best for the problem at hand. It is also useful to keep the following in mind:

- **Shrinkage**: lasso, alasso, scad, and mcp are able to shrink parameters to zero, contrarily to the ridge penalty whose purpose is just regularizing the estimation process.
- **Unbiasedness**: alasso, scad, and mcp enjoy the so-called "oracle" property. On the contrary, the lasso is biased since it applies the same penalization to all parameters.
- **Automatic procedure:** only lasso, alasso, and ridge are supported by the automatic procedure. This means that these penalties are a convenient choice with all the analyses requiring multiple penalty terms (e.g., multiple-group analyses), for which the automatic procedure is the only feasible alternative to otherwise computationally intensive multi-dimensional grid-searches.

Geminiani, Marra, and Moustaki (2021) performed numerical and empirical examples to evaluate and compare the performance of single- and multiple-group penalized factor models under different penalty functions. The alasso penalty generally produced the best trade-off between sparsity and goodness of fit. However, unlike other penalties, the alasso requires a set of adaptive weights. In some situations, the weights might not be available, or might be difficult to obtain. If this is the case, users are encouraged to resort to simpler penalties.

### More sparsity:

The penalized model automatically tries to generate the optimal trade-off between goodness of fit and model complexity (if `strategy = "auto"`). As a result of this delicate balance, it may not provide the sparsest factor solution. If users desire more sparsity, they can follow the guidelines below.

- Influence factor: increase the value of the influence factor stored in the option `gamma`. As a rule of thumb, in our experience, common values for obtaining sparse solutions usually range between 3.5 and 4.5.
- Penalties: some penalties rely on a second tuning parameter. It may be helpful to try out different values for it, and see which one performs best. For instance, increasing the value or the exponent of the alasso (by specifying, for instance, `a.alasso = 2`) leads to sparser solutions.

In case users fitted a penalized model with a fixed tuning parameter (`strategy = "fixed"`), they can manually and subjectively increase its value in the option `eta` to encourage more sparsity. When doing so, it is helpful to first do some trials and understand a reasonable range of values that the tuning parameter can take.

**Ordinary Maximum Likelihood:**

If `strategy = "fixed"`, `pen.shrink = "none"`, `pen.diff = "none"`, and `eta = list("shrink" = c("none" = 0), "diff" = c("none" = 0))`, no penalization is applied, and the model is estimated through ordinary maximum likelihood.

## Convergence & Admissibility

The function `penfa` internally assesses the convergence of the fitted model, and the admissibility of the final solution.

**Convergence:**

The convergence checks assess whether the penalized gradient vector is close to zero and the penalized Hessian/Fisher information matrix is positive definite. In case of convergence issues, `penfa` warns the users with explanatory messages.
**Note:** Due to the presence of possibly multiple penalty terms, our experiments highlighted that the penalized gradient need not be strictly close to zero to obtain meaningful results. It is enough that its elements do not exceed a pre-specified threshold, whose value can be changed through the `optim.dx.tol` option.

**Admissibility:**

The admissibility checks are carried out to determine whether the final solution is *admissible*. Specifically, the `penfa` function sequentially checks whether:

1. The final model includes any negative unique variances (Heywood cases);
2. The final model includes any negative factor variances;
3. The estimated common factor covariance matrix is positive definite;
4. The estimated unique factor covariance matrix is positive definite;
5. The estimated factor loading matrix is of full column rank;
6. The estimated factor loading matrix does not contain any null rows.

In case of multiple-group analyses, the function checks the admissibility of the parameter matrices of each group. If any of the above conditions are not satisfied, the `penfa` function warns the user with explanatory messages on the reasons why.

## Warnings & Errors

Occasionally the `penfa` function may print out warnings or produce errors. If the errors concern convergence issues, it may be helpful to go through the following steps:

1. Identification: please make sure that at least the minimum identification restrictions are satisfied. This implies fixing the scale and the origin of every factor in each group. In addition, other constraints - which usually come in the form of zero-restricted loadings - are necessary due to rotational freedom.
2. Starting values: the choice of the starting values is of paramount importance when it comes to convergence. The starting values internally used by `penfa` correspond to the ones used by the `lavaan` package for `confirmatory factor analysis`. If users have some prior knowledge or intuition about possible values for some of the parameters, it might be beneficial to include this information by providing the starting values for those parameters in the syntax specification

(see below for additional details). For instance, depending on the case, specifying the starting values of the primary loadings equal to 1 (`start(1)*x1 + ...`) often results in more stable optimization processes, especially when dealing with complicated models that require the estimation of many parameters, as in multiple-group penalized factor analysis.

3. Sample size: the penalized models fitted by `penfa` have a larger number of parameters than confirmatory factor analytic applications. This complexity should be accompanied by a reasonable sample size. If the sample size is too small for the complexity of the model, convergence issues will arise. In case of small sample sizes, it might in principle be more reliable to select the tuning parameter through a grid-search with the GBIC instead of using the automatic procedure.
4. Automatic procedure: if the starting values of the tuning parameters prevent the automatic procedure from finding the optimal estimates of the tuning parameters, the procedure is repeated with different starting values. If this fails, an error is printed out.
5. Adaptive weights: when using the alasso penalty, it is suggested to manually provide a vector of adaptive weights, especially for complex models. The adaptive weights often come in the form of (unpenalized) maximum likelihood estimates. If no vector of weights is provided, the `penfa` function internally estimates an unpenalized MLE model whose parameter estimates will serve as weights. If the unpenalized model does not converge, the `penfa` function internally estimates a ridge-regularized factor model and uses the resulting estimates as weights. If even this estimation fails, an error is printed out.

Ultimately, if none of the above succeeds, users shall consider re-specifying the model, either by simplifying the hypothesized factor structure or considering a subset of the observed variables. Increasing the number of restrictions (for instance, by specifying some additional fixed loadings) might be advantageous. Also, as a general practice, when conducting a multiple-group analysis, make sure beforehand that the groups share similar factor structures: if the groups have different factor configurations, the final results will be distorted.

It is always important to assess whether the distributional assumptions of the normal linear factor model hold. The `penfa` function fits penalized factor models to continuous observed variables; this excludes categorical items or items with a few number of categories that would instead require tailored approaches that specifically take into account the qualitative nature of the data.

## Standard Errors

The standard errors are derived from the inverse of the penalized Fisher information matrix (if `information = "fisher"`) or penalized Hessian (if `information = "hessian"`), which relies on the Bayesian result for the covariance matrix of the estimated parameters. The implemented framework allows to have a standard error for every model parameter. However, users should take extra caution when using the standard errors associated with the penalized parameters that were shrunken to zero.

## Vignettes and Tutorials

To learn more about `penfa`, start with the vignettes and tutorials at `browseVignettes(package = "penfa")` and https://egeminiani.github.io/penfa/articles/.

## Author(s)

Elena Geminiani <geminianielena@gmail.com>.

## References

Geminiani, E., Marra, G., & Moustaki, I. (2021). "Single- and Multiple-Group Penalized Factor Analysis: A Trust-Region Algorithm Approach with Integrated Automatic Multiple Tuning Parameter Selection." Psychometrika, 86(1), 65-95. doi: 10.1007/s11336021097518

Geminiani E. (2020), "A penalized likelihood-based framework for single and multiple-group factor analysis models" (Doctoral dissertation, University of Bologna). Available at http://amsdottorato.unibo.it/9355/.

## See Also

penfa-class

## Examples

```
data(ccdata)

### Single-group analysis (no mean-structure, unit factor variances)
syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto",
                    gamma = 4)



### Multiple-group analysis (mean structure, marker-variable approach, starting values)
syntax_mg = '
help  =~ 1*h1 +          h2 +          h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
voice =~ 0*h1 + start(0)*h2 + start(0)*h3 + h4 + h5 + h6 + h7 + 1*v1 + v2 + v3 + v4 + v5
h1 + v1 ~ 0*1 '

# Compute weights for alasso from unpenalized model
mle_fitMG <- penfa(model = syntax_mg,
                   data  = ccdata,
                   group = "country",
                   int.ov.free = TRUE,
                   int.lv.free = TRUE,
                   pen.shrink = "none",
                   pen.diff = "none",
```

```
                    eta = list(shrink = c("lambda" = 0), diff = c("none" = 0)),
                    strategy = "fixed")
mle_weightsMG <- coef(mle_fitMG)

# Fit model
alasso_fitMG <- penfa(## factor model
                      model = syntax_mg,
                      data = ccdata,
                      group = "country",
                      int.ov.free = TRUE,
                      int.lv.free = TRUE,
                      ## penalization
                      pen.shrink = "alasso",
                      pen.diff = "alasso",
                      eta = list(shrink = c("lambda" = 0.01),
                      diff = c("lambda" = 0.1, "tau" = 0.01)),
                      ## automatic procedure
                      strategy = "auto",
                      gamma = 4,
                      ## alasso
                      weights = mle_weightsMG)

### For additional examples, see the vignettes and tutorials at
### browseVignettes(package = "penfa") and https://egeminiani.github.io/penfa/articles/
```

---

`penfa-class` *S4 Class for describing a* `penfa` *model*

---

## Description

The `penfa` class represents a (fitted) penalized factor analysis model. It contains a description of the model as specified by the user, a summary of the data, an internal matrix representation, the fitting results, and the penalized quantities.

## Objects from the Class

Objects can be created via the `penfa` function.

## Slots

`version`**:** The `penfa` package version used to create this object.

`call`**:** The function call as returned by `match.call()`.

`timing`**:** The elapsed time (user + system) for various parts of the program as a list, including the total time.

`Options`**:** Named list of options that were provided by the user or filled-in automatically. See `penfaOptions` for additional details.

`ParTable`: Named list describing the model parameters. Can be coerced to a data.frame. This is also called "parameter table". It includes information on the fixed, free and penalized parameters, their indices, the active penalization strategies ("none", "shrink", "diff", or "shrink + diff"), the starting values, the estimated parameters and the associated standard errors.

`pta`: Named list containing parameter table attributes, like observed and latent variable names, their indices, and the number of groups.

`Data`: Object of internal class `"penfaData"`; contains information about the data set. See the `penfaData` class for additional details.

`SampleStats`: Object of internal class `"penfaSampleStats"`; contains the sample statistics. See the `penfaSampleStats` class for additional details.

`Model`: Object of internal class `"penfaModel"`: the internal (matrix) representation of the model. See the `penfaModel` class for additional details.

`Optim`: List. Information about the optimization process. This includes the estimated parameters (`x`), the number of estimated parameters (`npar`), the number of trust-region iterations (`iterations`), the value of the penalized objective function (`fx.pen`), the value of the unpenalized objective function (`fx.unpen`), the penalized log-likelihood (`logl.pen`; this is equal to `fx.pen` multiplied by (-1)), the unpenalized log-likelihood (`logl.unpen`; this is equal to `fx.unpen` multiplied by (-1)), the penalized gradient (`dx.pen`), the penalized Hessian/Fisher information matrix (`hessian.pen`), the list of control arguments for the trust-region algorithm (`control`), and how many times the objective function became non-positive definite during the estimation process (`npd`). If penfa was called with the option `verbose = TRUE`, the following additional arguments coming from the trust-region function `trust` are reported in the `Optim` slot: `argpath`, `argtry`, `type`, `accept`, `radii`, `rho`, `fx.val`, `fx.valtry`, `change`, `stepnorm`. See the manual page of `trust` from the `trust` package for an overview of these quantities.

`Penalize`: Object of internal class `"penfaPenalty"`; contains information about the penalization. See the `penfaPenalty` for additional details.

`Implied`: List. Model-implied moments (covariance matrix and mean vector).

`Vcov`: List. Information about the covariance matrix (vcov) of the model parameters. This slot includes the following quantities: the type of penalized information matrix used in the model (either Hessian or Fisher; `information`), the vcov matrix of parameters (`vcov`), whether the convergence checks on the penalized gradient and the penalized information matrix were satisfied (`solution`), whether the employed information matrix was positive-definite (`pdef`), whether the estimated factor solution was admissible (`admissibility`), the standard errors computed according to the Bayesian result from the information matrix reported in `information` (`se`), and the 95% confidence intervals (`ci`).

`Inference`: List. Information on effective degrees of the model and information criteria for model selection. This slot reports the following quantities: effective degree of freedom for each parameter (`edf.single`), total edf (`edf`), influence matrix (`influence.mat`), generalized information criteria (`IC`), such as AIC and BIC.

`external`: List. Empty slot.

## Methods

The following methods are available for an object of class `penfa`:

**show** `signature(object = "penfa")`: Prints a short summary of the estimation process, including the optimization method, the specified penalty functions, the convergence status, the number of iterations, the tuning selection strategy, and the effective degrees of freedom. See the manual page of `show,penfa-method` for details.

**summary** `signature(object = "penfa", header = TRUE, estimates = TRUE, ci = TRUE, level = 0.95,nd = 3L, cutoff = 0.05, extra = TRUE)`: Prints a summary of the model parameter estimates, and the optimization process. See the manual page of `summary,penfa-method` for details.

**coef** `signature(object = "penfa", type = "free", labels = TRUE)`: Returns the estimates of the parameters in the model as a named numeric vector. See the manual page of `coef,penfa-method` for details.

**fitted** `signature(object = "penfa", labels = TRUE)`: Returns a list of the model-implied moments (per group). See the manual page of `fitted,penfa-method` for details.

## References

Geminiani, E., Marra, G., & Moustaki, I. (2021). "Single- and Multiple-Group Penalized Factor Analysis: A Trust-Region Algorithm Approach with Integrated Automatic Multiple Tuning Parameter Selection." Psychometrika, 86(1), 65-95. doi: 10.1007/s11336021097518

## See Also

`penfa`, `penfaParEstim`

---

# penfaData-class *S4 Class for describing the input data*

---

## Description

The `penfaData` class gives information on the data set provided in input for analysis. This class is an adaptation of the `lavData` class from the lavaan package.

## Slots

`ngroups` Integer. The number of groups.

`group` Character. The observed variables defining the groups.

`group.label` Character. The group labels, that is, the values of the `group` variable, if any.

`std.ov` Logical indicating whether the observed variables should be standardized.

`nobs` List of the effective number of observations in each group.

`norig` List of the original number of observations in each group.

`ov.names` List of the observed variable names in each group.

`ov` List of details at the observed variable level.

`case.idx` List of the case (i.e., observation) indices in each group.

`X` List. Local copy of the input data set split into groups.

## See Also

penfa

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto")

alasso_fit@Data
str(alasso_fit@Data)
```

---

# penfaModel-class *S4 Class for internal representation of a factor model*

---

## Description

The penfaModel class gives the internal matrix representation of a factor analysis model. Note that this representation summarizes the characteristics of the model itself (e.g., number of items, number of factors, parameter indices, etc), without information on the penalization process (see penfaPenalty for that aspect). This class is an adaptation of the lavModel class from the lavaan package.

## Slots

GLIST List. The model matrices and vectors: "lambda" for the factor loading matrix, "psi" for the covariance matrix of the unique factors, "phi" for the covariance matrix of the common factors, "tau" for the intercept vector, and "kappa" for the vector of factor means. In case of a multiple-group analysis, the elements of each group are presented sequentially.

dimNames List. Dimension names (row names and column names) of every model matrix and vector.

isSymmetric Logical vector declaring whether each model matrix/vector is symmetric.

mmSize Integer vector specifying the size (unique elements only) of each model matrix/vector.

meanstructure Logical. It declares whether the model includes a meanstructure.

`ngroups` Integer. The number of groups.

`nmat` Integer vector specifying the number of model matrices/vectors for each group.

`nvar` Integer vector specifying the number of observed variables in each group.

`num.idx` List of the indices of the observed variables in each group.

`nx.free` Integer. The number of parameters of the factor model. This count does not include the fixed parameters, but it does include the parameters that will be penalized (if any) during optimization. (see `penfaPenalty` for additional details in this respect).

`nx.user` Integer. The total count of the parameters that are being estimated and the ones that have been fixed.

`m.free.idx` List. For each model matrix, the indices of the elements to be estimated (i.e., non-fixed). The counter starts at 1 for every model matrix.

`x.free.idx` List. For each model matrix, the indices of the elements to be estimated (i.e., non-fixed). The counter continues from the previous model matrix.

`m.user.idx` List. Much like `m.free.idx`, but it also contains the indices of the parameters that have been fixed by the user.

`x.user.idx` List. Much like `x.free.idx`, but it also contains the indices of the parameters that have been fixed by the user.

`x.free.var.idx` Vector of integers denoting the indices corresponding to the unique variances.

## See Also

`penfa`

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto")

alasso_fit@Model
```

penfaOptions penfa *Options*

## Description

The default options internally used by the penfa function. These options can be changed by passing "name = value" arguments to the penfa function call, where they are being added to the "..." argument.

## Usage

```
penfaOptions(
  opt = list(meanstructure = FALSE, int.ov.free = FALSE, int.lv.free = FALSE,
   orthogonal = FALSE, std.lv = FALSE, auto.fix.first = FALSE, auto.fix.single = FALSE,
   std.ov = FALSE, information = "fisher", control = list(), optim.dx.tol = 100, a.scad
   = 3.7, a.mcp = 3, a.alasso = 1, weights = NULL, cbar = 1e-08, gamma = 4, user.start =
    FALSE, start.val = c(), verbose = TRUE, warn = TRUE, debug = FALSE)
)
```

## Arguments

| | |
|---|---|
| opt | List of default options. See below for details. |

## Details

The following section details the full list of options currently accepted by the penfa function.

Model features:

meanstructure: Logical. If TRUE, a meanstructure is requested. It should be used in conjunction with int.ov.free and int.lv.free or intercept-like formulas in the model syntax. Default to FALSE.

int.ov.free: Logical. If FALSE, the intercepts of the observed variables are fixed to zero. Default to FALSE.

int.lv.free: Logical. If FALSE, the intercepts of the common factors are fixed to zero. Default to FALSE.

orthogonal: Logical. If TRUE, all covariances among the common factors are set to zero. Default to FALSE.

std.lv: Logical. If TRUE, the factor variances are fixed to 1.0. Default to FALSE.

auto.fix.first: Logical. If TRUE, the factor loading of the first indicator is set to 1.0 for every factor. Default to FALSE.

auto.fix.single: Logical. If TRUE, the residual variance (if included) of an observed indicator is set to zero if it is the only indicator of a common factor. Default to FALSE.

Data options:

`std.ov`**:** Logical. If `TRUE`, all observed variables are standardized before entering the analysis. Default to `FALSE`.

Estimation and optimization:

`information`**:** Character. If `"fisher"`, the penalized expected Fisher information matrix is used as second-order derivatives in the trust-region algorithm and for computing the standard errors of the model parameters. If `"hessian"`, the penalized Hessian matrix is used. Default to `"fisher"`.

`control`**:** A list containing control parameters passed to the trust-region optimizer. See the manual page of `trust` from the `trust` package for an overview of its control parameters. Default values for these parameters are `rinit=1L`, `rmax=100L`, `iterlim=1000L`, `fterm = sqrt(.Machine$double.eps)`, `mterm = sqrt(.Machine$double.eps)`.

`optim.dx.tol` Numeric. The tolerance value used when checking the size of the elements of the gradient of the objective function. Default equal to 100.

Penalization:

`a.scad` Numeric. The shape parameter for the scad penalty. Default to 3.7, as recommended by Fan & Li (2001).

`a.mcp` Numeric. The shape parameter of the mcp penalty. Default to 3.

`a.alasso` Numeric. The exponent in the adaptive weights for the alasso penalty. Default to 1.

`weights` Numeric. Only valid when either `pen.shrink` or `pen.diff` is equal to "alasso". An optional vector of values provided by the user representing a consistent estimate for each model parameter. The vector is then internally used for computing the adaptive weights. If unspecified, the maximum likelihood estimates (MLE) from the unpenalized model are used.

`cbar` Numeric. Numerical constant used in the local approximation of the penalty functions. Default to 1e-08.

Automatic procedure:

`gamma` Numeric. The value of the influence factor used in the automatic tuning parameter procedure. Default to 4.

`user.start` Logical whether the user has provided a vector of starting values for the model parameter estimates.

`start.val` Numeric. An optional vector of parameter estimates to be used as starting values for the model parameters. This option is also internally used by the automatic procedure.

Verbosity options:

`verbose`**:** Logical. If `TRUE`, some information on the estimation process (e.g., convergence and admissibility checks, effective degrees of freedom) are printed out. Default to `TRUE`.

`warn`**:** Logical. If `TRUE`, some warnings are printed out during the iterations. Default to `TRUE`.

`debug`**:** Logical. If `TRUE`, debugging information is printed out. Default to `FALSE`.

## Value

A list of default options internally used by the `penfa` function.

## penfaOut *Print estimated parameter matrices*

### Description

A utility that extracts the estimated parameter matrices and vectors of the penalized factor model for each group and rounds them to the specified number of decimal digits.

### Usage

```
penfaOut(
  object,
  which = c("lambda", "psi", "phi", "tau", "kappa"),
  ...,
  nd = 3L
)
```

### Arguments

| | |
|---|---|
| object | An object of class penfa, that is, a fitted penalized factor model. |
| which | Character denoting the name of the estimated matrix or vector to display. Possible values are "lambda", "psi", "phi", "tau", and "kappa". Multiple elements can be specified. By default, all estimated matrices are shown. |
| ... | Additional options. |
| nd | The number of decimal digits to be used. |

### Value

List of the estimated parameter matrices and vectors for each group.

### See Also

penfa

## penfaParEstim *Print parameter estimates in table format*

### Description

The parameter estimates of the penalized factor analysis model in each group.

## Usage

```
penfaParEstim(
  object,
  se = TRUE,
  ci = TRUE,
  level = 0.95,
  remove.nonfree = FALSE,
  output = "data.frame",
  header = FALSE
)
```

## Arguments

| | |
|---|---|
| object | An object of class penfa. |
| se | Logical. If TRUE, it includes a column with the standard errors. |
| ci | Logical. If TRUE, the confidence intervals are added to the output. |
| level | The confidence level, default is 0.95. |
| remove.nonfree | Logical. If TRUE, it filters the output and removes all rows with fixed (that is, neither free, nor penalized) parameters. |
| output | Character. If "data.frame", the parameter table is displayed as a standard formatted data.frame. If "text", the parameter table is displayed with subsections (as used by the summary function). |
| header | Logical, only used if output = "text". If TRUE, it prints a header on top of the parameter list with details on the group levels and the information matrix used during optimization by the trust-region algorithm. |

## Value

A dataframe of class penfa.data.frame with the parameter estimates of a penfa model for each group.

## See Also

penfa

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
```

```
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto")

penfaParEstim(alasso_fit)
```

---

penfaPenalty-class *S4 Class for describing the penalization process*

---

## Description

The penfaPenalty class provides information on the penalization process, such as the user-specified penalty functions, the optimal values of the tuning parameters, and the penalty matrices at convergence.

## Slots

`strategy` Character. The strategy used for the selection of the tuning parameter(s). If `strategy = "auto"`, the optimal values of the tuning parameters are determined via the automatic tuning parameter procedure; if `strategy = "fixed"`, a penalized factor model with the values of the tuning parameters stored in the option `eta` is estimated.

`penalty` List. A list of the user-specified penalty functions for sparsity ("shrink") and parameter equivalence ("diff").

`tuning` List. A named list containing the optimal values of the tuning parameter(s) if `strategy = "auto"` or the user-specified fixed values of the tuning parameter(s) if `strategy = "fixed"`. The list has two components with names "shrink" and "diff", and refers to the tuning parameters used for shrinkage and group equivalence, respectively. The components of the list are, in turn, the named vectors specifying the type of parameter matrices or vectors that were penalized.

`pmat` List. A named list containing the names of the parameter matrices and vectors that were penalized for sparsity ("shrink") and/or group equivalence ("diff").

`pen.idx` List. A named list with the indices of the parameters that were penalized for sparsity ("shrink") and/or group equivalence ("diff").

`Sh.info` List. A list of the penalization terms, vectors and matrices evaluated at the optimal values of the tuning parameters. In particular, its argument `S.h` returns the estimated penalty matrix. If the factor model is penalized only through a shrinkage penalty (i.e., `pen.shrink` is not `'none'`), and there is no penalization on the differences (i.e., `pen.diff = 'none'`), then `S.h` is a diagonal matrix whose elements precisely quantify the extent to which each model parameter has been penalized.

`extra` List. A list possibly containing additional information on the penalization process, such as the hyperparameter values for some penalty functions (e.g., for the alasso, the value of the exponent and the adaptive weights.)

`automatic` List. If `strategy = "auto"`, it contains information on the automatic multiple tuning parameter procedure, such as the optimal values of the tuning parameters, the convergence status, the specified value of the influence factor, the number of necessary iterations, and the tolerance level.

## See Also

penfa

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto")

alasso_fit@Penalize

str(alasso_fit@Penalize)
```

---

# penfaPredict *Compute the factor scores from a fitted* penfa *model*

---

## Description

The penfaPredict function estimates the factor scores from a fitted penalized factor model. The factor scores are the estimated values ("predictions") of the common factors.

## Usage

```
penfaPredict(
  object,
  newdata = NULL,
  method = "regression",
  label = TRUE,
  append.data = FALSE,
  assemble = FALSE
)
```

## Arguments

| | |
|---|---|
| object | An object of class penfa. |

| | |
|---|---|
| newdata | An optional data frame containing the same variables as the ones appearing in the original data frame used for fitting the model in object. |
| method | Character indicating the method for computing the factor scores. Possible options are "regression" and "bartlett". For the normal linear continuous case, the regression method is equivalent to the Empirical Bayes Method (EBM), whereas Bartlett's strategy is equivalent to maximum likelihood's method. |
| label | Logical. If TRUE, the columns are labeled. |
| append.data | Logical. If TRUE, the original data set (or the data set provided in the newdata argument) is appended to the factor scores. |
| assemble | Logical. If TRUE, the factor scores from each group are assembled in a single data frame of the same dimensions as the original data set and with a group column defining the groups. |

## Value

A matrix with the factor scores from a fitted penfa model.

## References

Geminiani E. (2020), "A penalized likelihood-based framework for single and multiple-group factor analysis models" (Doctoral dissertation, University of Bologna). Available at http://amsdottorato.unibo.it/9355/.

## See Also

penfa

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto",
                    gamma = 4)

fscores <- penfaPredict(alasso_fit)
```

penfaSampleStats-class *S4 Class for describing the sample moments*

## Description

The penfaSampleStats class provides information on the sample moments of the factor analysis model. This class is an adaptation of the lavSampleStats class from the lavaan package.

## Slots

var List of the variances of the observed variables in every group.

cov List of the covariance matrices of the observed variables in every group.

mean List of the means of the observed variables in every group.

group.w List of group weights.

nobs List of the effective number of observations for every group.

ntotal Integer. Total number of observations across all groups.

ngroups Integer. Number of groups.

icov List of the inverse matrices of the covariance matrices of the observed variables in every group.

cov.log.det List of the logarithms of the determinants of the covariance matrices of the observed variables for every group.

## See Also

penfa

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto")

alasso_fit@SampleStats
```

## penmat *Extract estimated penalty matrix*

### Description

A utility that extracts the estimated penalty matrix from a fitted object of class penfa.

### Usage

```
penmat(x, type = "full", which = NULL)
```

### Arguments

| | |
|---|---|
| x | An object of class penfa, that is, a fitted penalized factor model. |
| type | Character denoting the type of penalization. Type equal to "full" returns the complete penalty matrix; type="shrink" returns the penalty matrix for shrinkage; type="diff" the penalty matrix for parameter equivalence. The matrix returned by type="full" is the sum of all the shrink and diff penalty submatrices. |
| which | Character prompting the extraction of the penalty matrix component corresponding to the specified model matrix. It is only valid when type="shrink" or type="diff". Possible values are "lambda", "psi", "phi", "tau", "kappa" and "none". Only the model matrices penalized during model fitting (i.e., in the penfa call) can appear in the which argument. |

### Value

A penalty matrix of class penfaPenMat. If multiple elements are specified in the which argument, a list of penalty matrices (one for each element, and each of class penfaPenMat) is returned.

### See Also

penfa

## show,penfa-method *Display a* penfa *object*

### Description

An S4 method printing a short summary of the estimation process, including the optimization method, the specified penalty functions, the convergence status, the number of iterations, the tuning selection strategy, and the effective degrees of freedom.

### Usage

```
## S4 method for signature 'penfa'
show(object)
```

### Arguments

| | |
|---|---|
| object | An object of class penfa, found as a result of a call to penfa. |

### Value

An object reporting a short summary of a fitted penfa model.

### See Also

penfa, penfa-class

---

## show,penfaData-method *Display details on the input data*

---

### Description

An S4 method showing information on the input data, including the number of observations. In case of a multiple-group analysis, the sample sizes for each group are displayed.

### Usage

```
## S4 method for signature 'penfaData'
show(object)
```

### Arguments

| | |
|---|---|
| object | An object of class penfaData, found in the Data slot from a penfa class object. |

### Value

An object reporting a short summary of the input data.

### See Also

penfaData-class

## show,penfaPenalty-method *Display details on the penalization*

### Description

An S4 method showing information on the penalization process, including the employed penalty functions and the model matrices they affect. Additionally, it reports the optimal values of the tuning parameters and the tuning parameter selection strategy. If the automatic procedure was used, the output would also show the value of the influence factor, and the number of two-steps iterations.

### Usage

```
## S4 method for signature 'penfaPenalty'
show(object)
```

### Arguments

| | |
|---|---|
| `object` | An object of class `penfaPenalty`, found in the `Penalize` slot from an object of `penfa` class. |

### Value

An object reporting a short summary of the penalization process for a fitted `penfa` model.

### See Also

`penfaPenalty-class`

## summary,penfa-method *Summary constructor for a* `penfa` *object*

### Description

An S4 method printing a summary of the model parameter estimates for an object of class `penfa`.

### Usage

```
## S4 method for signature 'penfa'
summary(
  object,
  header = TRUE,
  estimates = TRUE,
  ci = TRUE,
  level = 0.95,
  nd = 3L,
```

```
  cutoff = 0.05,
  extra = TRUE
)
```

## Arguments

| | |
|---|---|
| object | An object of class penfa, found as a result of a call to penfa. |
| header | Logical. If TRUE, the header section is printed. The header contains relevant information about the data, the fitted model, the optimization process, and the penalization strategy, including, for instance, the employed penalties, the estimated effective degrees of freedom (*edf*), the optimal values of the tuning parameter(s), the GBIC and many others. |
| estimates | Logical. If TRUE, a section with the parameter estimates is printed out. |
| ci | Logical. If TRUE, confidence intervals are added to the parameter estimates section. |
| level | Logical. It denotes the significance level used for the statistical tests. |
| nd | Integer. It determines the number of digits after the decimal point to be printed in the parameter estimates section. |
| cutoff | Numeric. Standard errors and confidence intervals for the penalized parameter estimates falling below the cutoff value are not displayed. Confidence intervals for the parameters that have been penalized and shrunken to zero must be treated with caution. |
| extra | Logical. If TRUE, additional information on the model are displayed. |

## Value

An object reporting a detailed summary of the estimated parameters for a penfa model.

## See Also

penfa, penfa-class

## Examples

```
data(ccdata)

syntax = 'help  =~   h1 + h2 + h3 + h4 + h5 + h6 + h7 + 0*v1 + v2 + v3 + v4 + v5
          voice =~ 0*h1 + h2 + h3 + h4 + h5 + h6 + h7 +   v1 + v2 + v3 + v4 + v5'

alasso_fit <- penfa(## factor model
                    model  = syntax,
                    data   = ccdata,
                    std.lv = TRUE,
                    ## penalization
                    pen.shrink = "alasso",
                    eta = list(shrink = c("lambda" = 0.01), diff = c("none" = 0)),
                    ## automatic procedure
                    strategy = "auto")
```

```
summary(alasso_fit)
```

# Index