

Package: plavaan (via r-universe)

October 1, 2026

Title Penalized Estimation for Latent Variable Models with 'lavaan'

Version 0.0.2.9000

Description Extends the popular 'lavaan' package by adding penalized estimation capabilities. It supports penalty on individual parameters as well as the difference between parameters.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Suggests knitr, rmarkdown, testthat (>= 3.1.7)

VignetteBuilder knitr

Config/testthat/edition 3

Imports lavaan (>= 0.6-15), methods, numDeriv

URL <https://marklhc.github.io/plavaan/>,
<https://github.com/marklhc/plavaan>

BugReports <https://github.com/marklhc/plavaan/issues>

Config/roxygen2/version 8.1.0

Repository <https://marklhc.r-universe.dev>

Date/Publication 2026-09-01 11:59:36 UTC

RemoteUrl <https://github.com/marklhc/plavaan>

RemoteRef HEAD

RemoteSha c8d8e02763e7ea19fda43cb27c25eb2273ec62e2

Contents

composite_pair_loss	2
effective_df	3
loss	5
penalized_est	6
penalized_est_multistart	9
print.plavaan_efdf	12

composite_pair_loss *Composite Pairwise Loss Function*

Description

Computes the total loss across all pairwise combinations of rows in a matrix.

Usage

```
composite_pair_loss(x, fun, trans = identity, rescale = "df", ...)
```

Arguments

x	A numeric vector, matrix, or data frame. If not a matrix, it will be coerced to one after applying the transformation function.
fun	A function to compute the loss for each pairwise difference. The package supports the alignment loss (<code>alf</code>) and the approximate L0 penalty (<code>l0a</code>), but users can provide custom functions as well.
trans	A transformation function to apply to x before computing pairwise differences. Default is <code>identity</code> (no transformation).
rescale	Either "df" (default) to rescale the total loss by $(nrow - 1) / ncombn(nrow, 2)$, where <code>nrow</code> is the number of rows, or a numeric value (likely between 0 and 1) to multiply the total loss by. With <code>rescale = "df"</code> and a 0/1-valued fun such as <code>l0a</code> , the returned value approximates the number of degrees of freedom of non-invariance in the block: 0 when all rows are equal (full invariance) and <code>nrow - 1</code> per column when all rows differ (fully free). It soft-counts how many of the <code>nrow - 1</code> pairwise-invariance constraints per column are violated.
...	Additional arguments passed to the loss function fun.

Details

The function works by:

1. Applying the transformation function `trans` to the input `x`
2. Converting the result to a matrix
3. Generating all possible pairwise combinations of row indices
4. Computing the difference between each pair of rows
5. Applying the loss function `fun` to each difference
6. Summing all the individual losses

`effective_df()` builds on this degrees-of-freedom interpretation (`rescale = "df"` with a 0/1-valued fun such as `l0a`) to report the effective number of parameters, and hence the effective model degrees of freedom, of penalized fits returned by `penalized_est()`.

Value

A numeric scalar representing the sum of losses across all pairwise combinations of rows.

Examples

```
# Example with a simple matrix
x <- matrix(runif(12), nrow = 4)
composite_pair_loss(x, fun = alf)

# Example with log transformation and L2 loss
composite_pair_loss(x, fun = function(x) x^2, trans = log)
```

effective_df	<i>Effective Number of Parameters and Degrees of Freedom for Penalized Fits</i>
--------------	---

Description

Computes the effective number of model parameters, and hence the effective model degrees of freedom, of a penalized lavaan fit using the soft-count property of the l0a penalty: a directly penalized parameter with estimate z is counted as $\text{l0a}(z, \text{eps}) = z^2 / (z^2 + \text{eps})$ effective parameters, which is close to 1 for large $|z|$ and close to 0 for small $|z|$. For a difference penalty, each column contributes one shared value (the invariance baseline) plus the l0a soft count of the pairwise differences in that column.

Usage

```
effective_df(
  x,
  pen_par_id = NULL,
  pen_diff_id = NULL,
  pen_fn = "l0a",
  eps = 0.01,
  ...
)
```

Arguments

- | | |
|-------------|---|
| x | A lavaan model object, typically the return value of penalized_est() . For such objects, the penalty specification is read from the object's 'penalized' attribute. |
| pen_par_id | Integer vector of directly penalized free-parameter IDs, in the same order as returned by <code>lavaan::coef()</code> . For a fit from penalized_est() , defaults to the value recorded in the object. |
| pen_diff_id | Named list of integer matrices of free-parameter IDs for difference penalties (same structure as in penalized_est()). For a fit from penalized_est() , defaults to the value recorded in the object. |

pen_fn	The penalty function: "l0a", "alf", or a function. For a fit from <code>penalized_est()</code> , defaults to the value recorded in the object.
eps	The epsilon used by the built-in penalties. For a fit from <code>penalized_est()</code> , defaults to the value recorded in the object.
...	Additional arguments passed to a user-supplied pen_fn.

Details

For a fit returned by `penalized_est()`, any argument left at its default (NULL for `pen_par_id`/`pen_diff_id`, "l0a" for `pen_fn`, .01 for `eps`) is filled in from the object's 'penalized' attribute; explicitly supplied values take precedence. Note that a value equal to the default cannot be distinguished from an omitted one, so passing, e.g., `eps = .01` explicitly will still be overridden by the recorded value.

An ID that appears in both `pen_par_id` and a `pen_diff_id` matrix is a hard error, as the parameter would be double-counted.

The effective-df interpretation is calibrated for the l0a penalty, which takes the value 0 at the origin. With `pen_fn = "alf"`, $\text{alf}(0) = \text{eps}^{\theta} \cdot 0.25 \neq 0$, so the soft counts do not reach 0 and the effective df should be interpreted with care. A custom `pen_fn` is evaluated as if it had l0a-like 0/1 behavior; the result is approximate.

The nominal model df, `n_stats - length(coef(x))`, may be negative for penalized (often under-identified) models; the effective model df, `n_stats - npar_effective`, is the meaningful quantity. The number of sample statistics `n_stats` is counted per group from the model's parameter table.

Fit indices based on the effective df (chi-square, CFI, RMSEA, AIC, BIC) are obtained with `fitmeasures()` on a fit from `penalized_est()`, which freezes the parameters at the penalized estimates and refits. This fit evaluation is experimental and requires the fit to have been created with a non-"none" test (e.g. `test = "Chisq"`); with the default `test = "none"`, `fitmeasures()` is unavailable.

Value

A data frame of class `plavaan_efdf` with one row per penalty component ("direct penalty", and one row per named block of `pen_diff_id`) and a final "TOTAL" row; the row names are the component labels. Columns:

`npar` Number of (nominal) free parameters covered by the row. For the TOTAL row, `length(coef(x))`.

`npar_effective` Effective number of parameters, with penalized parameters soft-counted by their penalty values.

`df_saved` `npar - npar_effective`, the degrees of freedom saved by the penalty relative to the nominal model.

The data frame also carries an `info` attribute with `n_stats` (number of sample statistics), `df_model` (nominal model df, possibly negative), `df_model_effective`, `w`, `eps`, and `pen_fn`.

Examples

```
library(lavaan)
```

```
model <- "
  dem60 =~ y1 + y2 + y3 + y4
```

```

    dem65 =~ y5 + y6 + y7 + y8
    dem60 ~~ dem65
"
fit <- cfa(model, data = PoliticalDemocracy, std.lv = TRUE, do.fit = FALSE)
pt <- parTable(fit)
load_60 <- pt$free[pt$op == "=" & pt$lhs == "dem60"]
load_65 <- pt$free[pt$op == "=" & pt$lhs == "dem65"]

pen <- penalized_est(
  fit, w = 0.03,
  pen_diff_id = list(loadings = rbind(load_60, load_65)),
  eps = .01,
  test = "Chisq"
)
effective_df(pen)
fitmeasures(pen, c("chisq", "df", "cfi"))

```

loss

Loss functions

Description

For small ϵ s this provides a smooth, numerically stable approximation of $|x|^{1/2}$ (i.e. the square root of the absolute value). The function is vectorized over x .

Usage

```
alf(x, eps = 0.001)
```

```
l0a(x, eps = 0.01)
```

Arguments

x	Numeric vector. Input values to transform.
ϵ s	Positive numeric scalar (default .001 for <code>alf()</code> and .01 for <code>l0a()</code>). Small regularization constant to avoid non-differentiability and division-by-zero issues.

Details

The ALF, $(x^2 + \epsilon)^{1/4}$, is useful when a smooth surrogate for $\sqrt{|x|}$ is required (for optimization or regularization) while maintaining numerical stability near $x = 0$.

L0a, $x^2/(x^2 + \epsilon)$, is an approximation of the L0 penalty.

Value

Numeric vector of the same length as x .

Examples

```
alf(0)
alf(c(-4, -1, 0, 1, 4))
alf(0.5, eps = 1e-6)
l0a(0)
l0a(c(0, 1e-3, 0.1, 1))
l0a(c(-2, 0, 2), eps = 1e-4)
```

penalized_est

Penalized Parameter Estimation for Longitudinal CFA Models

Description

Performs penalized estimation on a lavaan model object by optimizing a penalized objective function. The function extracts the objective function from a lavaan model, applies a penalty function to specified parameters or pairwise differences of parameters, and returns an updated model with the optimized parameter estimates.

Usage

```
penalized_est(
  x,
  w,
  pen_par_id = NULL,
  pen_diff_id = NULL,
  pen_fn = "l0a",
  pen_gr = NULL,
  se = "none",
  test = "none",
  opt_control = list(),
  start = NULL,
  eps = 0.01,
  telescoping_control = list(eps_1 = 1, eps_end = 1e-05, eps_steps = 20, warm_start =
    FALSE),
  ...
)
```

Arguments

x	A fitted lavaan model object from which estimation components will be extracted.
w	Numeric scalar. Penalty weight (multiplier) applied to the penalty terms.
pen_par_id	Integer vector of parameter IDs to apply the penalty function directly to, in the same order as returned by <code>lavaan::coef()</code> and by <code>lavaan::parTable()</code> , with only the free elements.

pen_diff_id	A named list of integer matrices of free-parameter IDs (same order as <code>lavaan::coef()</code> / the free column of <code>lavaan::parTable()</code>). Each matrix has one row per group or time point and one column per matched parameter; the penalty is the sum of pairwise row differences within each column, rescaled by $(nrow - 1) / ncombn(nrow, 2)$. Structural NA entries mark a parameter absent in that row and are excluded from the differences.
pen_fn	A character string ("l0a" or "alf") or a function that computes the penalty. Default is "l0a".
pen_gr	A function that computes the gradient of the penalty function. If pen_fn is "l0a" or "alf", this is automatically set.
se	Character string specifying the type of standard errors to compute. Options are "none" (default; no standard errors) or "robust.huber.white" (robust sandwich estimator using numerical Hessian and first-order information, which is the same as used in the "mlr" estimator).
test	Character string specifying the model test used by the fit evaluation on the returned object (<code>fitmeasures()</code> and the chi-square test in <code>summary()</code>), via an internal "frozen" refit. Fit evaluation for penalized fits is experimental , so it is disabled by default: "none" (default) means no model test is run, <code>fitmeasures()</code> is unavailable, and <code>summary()</code> shows no chi-square test. Set to "Chisq" (ML/PML estimators) or "SatorraBentler" (WLSMV/MLM/MLR) to enable fit measures and the chi-square test; an experimental notice is then shown when <code>fitmeasures()</code> or <code>summary()</code> is called.
opt_control	A list of control parameters passed to <code>stats::nlminb()</code> . Default includes <code>eval.max = 2e4</code> , <code>iter.max = 1e4</code> , and <code>abs.tol = 1e-20</code> .
start	Numeric vector of starting values for the optimizer, or NULL (default) to use lavaan's default starting values. If supplied, its length must match the number of free parameters in the model.
eps	A positive numeric scalar used by the built-in penalties, or "telescoping" to fit a sequence of decreasing epsilon values. Default is .01. This argument does not alter custom pen_fn or pen_gr functions.
telescoping_control	A named list controlling telescoping, with <code>eps_1</code> (default 1), <code>eps_end</code> (default 1e-5), <code>eps_steps</code> (default 20), and <code>warm_start</code> (default FALSE). When <code>warm_start</code> is FALSE, every epsilon stage uses the original starting values; when TRUE, each stage after the first uses the preceding stage's estimates.
...	Additional arguments passed to a user-supplied pen_fn / pen_gr. Custom penalty functions must accept Built-in penalties ("l0a", "alf") ignore it.

Details

The function uses `nlminb()` to minimize a penalized objective function that combines the standard lavaan objective function with a penalty term. Only the parameter estimates and the log-likelihood should be interpreted. The returned object was not "fitted" (`do.fit = FALSE`) to avoid users interpreting the standard errors, which are generally not valid with penalized estimation. The nominal model degrees of freedom can also be misleading, as the penalized model is often under-identified; `effective_df()` reports the effective number of parameters and the effective model degrees of

freedom. When `test` is not "none", `fitmeasures()` / `summary()` on the returned object additionally report fit indices at the effective df (frozen refit at the penalized estimates); this fit evaluation is experimental and disabled by default. If the optimization does not converge (convergence code != 0), a warning is issued.

With `eps = "telescoping"`, the model is fit along a log-spaced sequence from `telescoping_control$eps_1` to `telescoping_control$eps_end`. By default, each stage uses the original starting values; set `telescoping_control$warm_start = TRUE` to initialize later stages from the preceding solution. The sequence stops when the largest absolute change between consecutive parameter vectors is at most $5e-4$. The returned object has a "telescoping" attribute with stage diagnostics.

Value

A lavaan model object updated with the penalized parameter estimates. The object has S4 class `plavaan` (a subclass of `lavaan`) and a `penalized` attribute recording the penalty specification, which enables `effective_df()` and, when `test` is not "none", `fitmeasures()` and `summary()` with effective degrees of freedom. With `eps = "telescoping"`, it also includes a "telescoping" data frame with per-stage epsilon values, parameter changes, objective values, and convergence indicators.

Warning

The returned object is not fitted using standard ML. Standard errors reported by `summary()` or `parameterEstimates()` will be missing unless `se = "robust.huber.white"` was specified. Even then, they are based on an experimental sandwich approximation and should be interpreted with caution.

Fit evaluation (`fitmeasures()` and the chi-square test in `summary()`) is also **experimental** and disabled by default (`test = "none"`). Enable it with `test = "Chisq"` (or "SatorraBentler"); interpret any resulting fit indices with caution, as they are based on a frozen refit at the penalized estimates with the effective degrees of freedom.

See Also

[lavaan](#), [nlminb](#)

Examples

```
library(lavaan)

# Define a longitudinal factor model with PoliticalDemocracy data
model <- "
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
  dem60 ~~ dem65
  dem60 ~~ 1 * dem60
  dem65 ~~ NA * dem65
  dem60 ~ 0
  dem65 ~ NA * 1
  y1 ~~ y5
  y2 ~~ y6
  y3 ~~ y7
```

```

    y4 ~~ y8
"

# Fit the model without constraints first to get parameter table
fit_un <- cfa(model, data = PoliticalDemocracy, std.lv = TRUE,
             meanstructure = TRUE, do.fit = FALSE)

# Get parameter IDs
pt <- parTable(fit_un)
# Loadings
load_60 <- pt$free[pt$op == "~" & pt$lhs == "dem60"]
load_65 <- pt$free[pt$op == "~" & pt$lhs == "dem65"]
# Intercepts
int_60 <- pt$free[pt$op == "~1" & pt$lhs %in% c("y1", "y2", "y3", "y4")]
int_65 <- pt$free[pt$op == "~1" & pt$lhs %in% c("y5", "y6", "y7", "y8")]

# Apply penalized estimation to penalize differences in loadings and intercepts
pen_fit <- penalized_est(
  x = fit_un,
  w = 0.03,
  pen_diff_id = list(
    loadings = rbind(load_60, load_65),
    intercepts = rbind(int_60, int_65)
  ),
  pen_fn = "l0a"
)

# Compare parameter estimates
summary(pen_fit)

# Effective number of parameters and degrees of freedom
effective_df(pen_fit)

```

penalized_est_multistart

Multistart Penalized Estimation

Description

Repeatedly calls `penalized_est()` with different starting vectors and returns the solution with the lowest penalized objective value. This is recommended when using non-convex penalties (`l0a`, `alf`), which can lead to local optima.

Usage

```

penalized_est_multistart(
  x,
  w,
  pen_par_id = NULL,

```

```

pen_diff_id = NULL,
pen_fn = "l0a",
pen_gr = NULL,
se = "none",
opt_control = list(),
eps = 0.01,
telescoping_control = list(eps_1 = 1, eps_end = 1e-05, eps_steps = 20, warm_start =
  FALSE),
n_starts = 10,
starts = NULL,
keep_all = FALSE,
verbose = FALSE,
...
)

```

Arguments

x	A fitted lavaan model object from which estimation components will be extracted.
w	Numeric scalar. Penalty weight (multiplier) applied to the penalty terms.
pen_par_id	Integer vector of parameter IDs to apply the penalty function directly to, in the same order as returned by <code>lavaan::coef()</code> and by <code>lavaan::parTable()</code> , with only the free elements.
pen_diff_id	A named list of integer matrices of free-parameter IDs (same order as <code>lavaan::coef()</code> / the free column of <code>lavaan::parTable()</code>). Each matrix has one row per group or time point and one column per matched parameter; the penalty is the sum of pairwise row differences within each column, rescaled by $(nrow - 1) / ncombn(nrow, 2)$. Structural NA entries mark a parameter absent in that row and are excluded from the differences.
pen_fn	A character string ("l0a" or "alf") or a function that computes the penalty. Default is "l0a".
pen_gr	A function that computes the gradient of the penalty function. If <code>pen_fn</code> is "l0a" or "alf", this is automatically set.
se	Character string specifying the type of standard errors to compute. Options are "none" (default; no standard errors) or "robust.huber.white" (robust sandwich estimator using numerical Hessian and first-order information, which is the same as used in the "mlr" estimator).
opt_control	A list of control parameters passed to <code>stats::nlminb()</code> . Default includes <code>eval.max = 2e4</code> , <code>iter.max = 1e4</code> , and <code>abs.tol = 1e-20</code> .
eps	A positive numeric scalar used by the built-in penalties, or "telescoping" to fit a sequence of decreasing epsilon values. Default is .01. This argument does not alter custom <code>pen_fn</code> or <code>pen_gr</code> functions.
telescoping_control	A named list controlling telescoping, with <code>eps_1</code> (default 1), <code>eps_end</code> (default 1e-5), <code>eps_steps</code> (default 20), and <code>warm_start</code> (default FALSE). When <code>warm_start</code> is FALSE, every epsilon stage uses the original starting values; when TRUE, each stage after the first uses the preceding stage's estimates.

<code>n_starts</code>	Integer. Number of random starting vectors to try. The first start is always lavaan's default (unperturbed). Later random starts perturb only parameters participating in <code>pen_par_id</code> or <code>pen_diff_id</code> . Default is 10.
<code>starts</code>	Matrix or list of numeric vectors, each with length equal to the number of free parameters. If supplied, random generation is bypassed entirely and <code>n_starts</code> is ignored. A message is printed noting this.
<code>keep_all</code>	Logical. If TRUE, attach all fitted lavaan objects as an attribute for inspection of alternative local solutions. Default is FALSE.
<code>verbose</code>	Logical. If TRUE, print progress messages during optimization. Default is FALSE.
<code>...</code>	Additional arguments passed to a user-supplied <code>pen_fn</code> / <code>pen_gr</code> . Custom penalty functions must accept <code>...</code> . Built-in penalties ("l0a", "alf") ignore it.

Details

Non-convex penalties like `l0a` and `alf` create rugged optimization surfaces where the optimizer can settle in different local solutions depending on starting values. Multistart optimization mitigates this risk by trying several starts and selecting the best.

Random starts perturb only the free parameters participating in `pen_par_id` or `pen_diff_id`; all nuisance parameters retain lavaan's base values. This targets the non-convex penalized surface while preserving a valid covariance structure from the supplied model start. Eligible loadings and intercepts are jittered around their base values, eligible variances are drawn within bounds based on observed variance, and eligible regression/covariance parameters receive a random correlation-scale perturbation.

Execution is sequential. For parallel execution, call `penalized_est()` with `start = ...` directly and use `future.apply`, `parallel`, or similar.

Users should set their own seed with `set.seed()` for reproducibility. This function does not call `set.seed()` internally.

Value

A lavaan model object (same shape as `penalized_est()`'s return), with additional attributes:

`multistart` A data frame with one row per start, containing columns `start_id`, `objective` (final penalized objective value), and `converged` (logical). Rows are sorted by ascending objective.

`all_fits` If `keep_all = TRUE`, a named list of all fitted lavaan objects, one per starting vector.

See Also

[penalized_est](#)

<code>print.plavaan_efdf</code>	<i>Print Method for Effective Degrees of Freedom Tables</i>
---------------------------------	---

Description

Prints the component table from `effective_df()` followed by summary lines with the number of sample statistics, the nominal and effective model degrees of freedom, and the penalty settings.

Usage

```
## S3 method for class 'plavaan_efdf'  
print(x, ...)
```

Arguments

<code>x</code>	An object of class <code>plavaan_efdf</code> , typically the return value of <code>effective_df()</code> .
<code>...</code>	Passed to <code>print.data.frame()</code> .

Index

`alf (loss)`, [5](#)

`composite_pair_loss`, [2](#)

`effective_df`, [3](#)

`effective_df()`, [2](#), [7](#), [8](#), [12](#)

`l0a (loss)`, [5](#)

`lavaan`, [8](#)

`lavaan::parTable()`, [6](#), [7](#), [10](#)

`loss`, [5](#)

`nlminb`, [8](#)

`penalized_est`, [6](#), [11](#)

`penalized_est()`, [2–4](#), [9](#), [11](#)

`penalized_est_multistart`, [9](#)

`print.plavaan_efdf`, [12](#)

`stats::nlminb()`, [7](#), [10](#)