Package 'INLAvaan'

May 7, 2026

Type Package

Title Approximate Bayesian Latent Variable Analysis

Version 0.2.4

Description Implements approximate Bayesian inference for Structural Equation Models (SEM) using a custom adaptation of the Integrated Nested Laplace Approximation (Rue et al., 2009) <doi:10.1111/j.1467-9868.2008.00700.x> as described in Jamil and Rue (2026a) <doi:10.48550/arXiv.2603.25690>. Provides a computationally efficient alternative to Markov Chain Monte Carlo (MCMC) for Bayesian estimation, allowing users to fit latent variable models using the 'lavaan' syntax. See also the companion paper on implementation and workflows, Jamil and Rue (2026b) <doi:10.48550/arXiv.2604.00671>.

License GPL (>= 3)

URL https://inlavaan.haziqj.ml/, https://github.com/haziqj/INLAvaan

BugReports https://github.com/haziqj/INLAvaan/issues

Depends R (>= 3.5)

Imports cli, graphics, lavaan, methods, parallel, stats, utils

Suggests blavaan, ggplot2, knitr, lme4, numDeriv, qrng, quarto, sn, testthat (>= 3.0.0), ucminf

VignetteBuilder quarto

Config/Needs/website rmarkdown

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Haziq Jamil [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3298-1010>),
Håvard Rue [ctb] (ORCID: <https://orcid.org/0000-0002-0222-1881>, Statistical and computational methodology),
Alvin Bong [ctb] (Initial site build)

Maintainer Haziq Jamil <haziq.jamil@gmail.com>

Repository CRAN

Date/Publication 2026-04-03 02:20:22 UTC

Contents

`acfa` *Fit an Approximate Bayesian Confirmatory Factor Analysis Model*

Description

Fit an Approximate Bayesian Confirmatory Factor Analysis Model

Usage

```
acfa(
  model,
  data,
  dp = priors_for(),
  test = "standard",
  vb_correction = TRUE,
  marginal_method = c("skewnorm", "asymgaus", "marggaus", "sampling"),
  marginal_correction = c("shortcut", "shortcut_fd", "hessian", "none"),
  nsamp = 1000,
  samp_copula = TRUE,
  sn_fit_ngrid = 21,
  sn_fit_logthresh = -6,
  sn_fit_temp = 1,
  sn_fit_sample = TRUE,
  control = list(),
  verbose = TRUE,
  debug = FALSE,
  add_priors = TRUE,
  optim_method = c("nlminb", "ucminf", "optim"),
  numerical_grad = FALSE,
  cores = NULL,
  ...
)
```

Arguments

`model`	A description of the user-specified model. Typically, the model is described using the lavaan model syntax. See `model.syntax` for more information. Alternatively, a parameter table (eg. the output of the `lavParTable()` function) is also accepted.
`data`	An optional data frame containing the observed variables used in the model. If some variables are declared as ordered factors, lavaan will treat them as ordinal variables.
`dp`	Default prior distributions on different types of parameters, typically the result of a call to `dpriors()`. See the `dpriors()` help file for more information.
`test`	Character indicating whether to compute posterior fit indices. Defaults to "standard". Change to "none" to skip these computations.
`vb_correction`	Logical indicating whether to apply a variational Bayes correction for the posterior mean vector of estimates. Defaults to `TRUE`.
`marginal_method`	The method for approximating the marginal posterior distributions. Options include `"skewnorm"` (skew-normal), `"asymgaus"` (two-piece asymmetric Gaussian), `"marggaus"` (marginalising the Laplace approximation), and `"sampling"` (sampling from the joint Laplace approximation).
`marginal_correction`	Which type of correction to use when fitting the skew-normal or two-piece

Gaussian marginals. "hessian" computes the full "shortcut" (default) computes only diagonals via central differences (full z-trace plus Schur complement correction), "shortcut_fd" is the same formula using forward differences (roughly half the cost, less accurate), "hessian" computes the full Hessian-based correction (slow), and "none" (or FALSE) applies no correction.

nsamp — The number of samples to draw for all sampling-based approaches (including posterior sampling for model fit indices).

samp_copula — Logical. When TRUE (default), posterior samples are drawn using the copula method with the fitted marginals (e.g. skew-normal or asymmetric Gaussian), with NORTA correlation adjustment. When FALSE, samples are drawn from the Gaussian (Laplace) approximation. Only re

sn_fit_ngrid — Number of grid points to lay out per dimension when fitting the skew-normal marginals. A finer grid gives a better fit at the cost of more joint-log-posterior evaluations. Defaults to 21.

sn_fit_logthresh — The log-threshold for fitting the skew-normal. Points with log-posterior drop below this threshold (relative to the maximum) will be excluded from the fit. Defaults to -6.

sn_fit_temp — Temperature parameter for fitting the skew-normal. Defaults to 1 (weights are the density values themselves). If NA, the temperature is included as an additional optimisation parameter.

sn_fit_sample — Logical. When TRUE (default), a parametric skew-normal is fitted to the posterior samples for covariance and defined parameters. When FALSE, these are summarised using kernel density estimation instead.

control — A list of control parameters for the optimiser.

verbose — Logical indicating whether to print progress messages.

debug — Logical indicating whether to return debug information.

add_priors — Logical indicating whether to include prior densities in the posterior computation.

optim_method — The optimisation method to use for finding the posterior mode. Options include "nlminb" (default), "ucminf", and "optim" (BFGS).

numerical_grad — Logical indicating whether to use numerical gradients for the optimisation. Defaults to FALSE to use analytical gradients.

cores — Integer or NULL. Number of cores for parallel marginal fitting. When NULL (default), serial execution is used unless the number of free parameters exceeds 120, in which case parallelisation is enabled automatically using all available physical cores. Set to 1L to force serial execution. If cores > 1, marginal fits are distributed across cores using parallel::mclapply() (fork-based; no parallelism on Windows).

... — Additional arguments to be passed to the lavaan::lavaan model fitting function.

Details

The acfa() function is a wrapper for the more general inlavaan() function, using the following default arguments:

- int.ov.free = TRUE
- int.lv.free = FALSE
- auto.fix.first = TRUE (unless std.lv = TRUE)
- auto.fix.single = TRUE
- auto.var = TRUE
- auto.cov.lv.x = TRUE
- auto.efa = TRUE
- auto.th = TRUE
- auto.delta = TRUE
- auto.cov.y = TRUE

For further information regarding these arguments, please refer to the lavaan::lavOptions() documentation.

Value

An S4 object of class INLAvaan which is a subclass of the lavaan::lavaan class.

See Also

Typically, users will interact with the specific latent variable model functions instead, including acfa(), asem(), and agrowth().

Examples

```
# The famous Holzinger and Swineford (1939) example
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
utils::data("HolzingerSwineford1939", package = "lavaan")

# Fit a CFA model with standardised latent variables
fit <- acfa(HS.model, data = HolzingerSwineford1939, std.lv = TRUE, nsamp = 100)
summary(fit)
```

---

agrowth *Fit an Approximate Bayesian Growth Curve Model*

Description

Fit an Approximate Bayesian Growth Curve Model

Usage

```
agrowth(
  model,
  data,
  dp = priors_for(),
  test = "standard",
  vb_correction = TRUE,
  marginal_method = c("skewnorm", "asymgaus", "marggaus", "sampling"),
  marginal_correction = c("shortcut", "shortcut_fd", "hessian", "none"),
  nsamp = 1000,
  samp_copula = TRUE,
  sn_fit_ngrid = 21,
  sn_fit_logthresh = -6,
  sn_fit_temp = 1,
  sn_fit_sample = TRUE,
  control = list(),
  verbose = TRUE,
  debug = FALSE,
  add_priors = TRUE,
  optim_method = c("nlminb", "ucminf", "optim"),
  numerical_grad = FALSE,
  cores = NULL,
  ...
)
```

Arguments

`model`	A description of the user-specified model. Typically, the model is described using the lavaan model syntax. See `model.syntax` for more information. Alternatively, a parameter table (eg. the output of the `lavParTable()` function) is also accepted.
`data`	An optional data frame containing the observed variables used in the model. If some variables are declared as ordered factors, lavaan will treat them as ordinal variables.
`dp`	Default prior distributions on different types of parameters, typically the result of a call to `dpriors()`. See the `dpriors()` help file for more information.
`test`	Character indicating whether to compute posterior fit indices. Defaults to "standard". Change to "none" to skip these computations.
`vb_correction`	Logical indicating whether to apply a variational Bayes correction for the posterior mean vector of estimates. Defaults to `TRUE`.
`marginal_method`	The method for approximating the marginal posterior distributions. Options include `"skewnorm"` (skew-normal), `"asymgaus"` (two-piece asymmetric Gaussian), `"marggaus"` (marginalising the Laplace approximation), and `"sampling"` (sampling from the joint Laplace approximation).
`marginal_correction`	Which type of correction to use when fitting the skew-normal or two-piece

Gaussian marginals. "hessian" computes the full "shortcut" (default) computes only diagonals via central differences (full z-trace plus Schur complement correction), "shortcut_fd" is the same formula using forward differences (roughly half the cost, less accurate), "hessian" computes the full Hessian-based correction (slow), and "none" (or FALSE) applies no correction.

nsamp — The number of samples to draw for all sampling-based approaches (including posterior sampling for model fit indices).

samp_copula — Logical. When TRUE (default), posterior samples are drawn using the copula method with the fitted marginals (e.g. skew-normal or asymmetric Gaussian), with NORTA correlation adjustment. When FALSE, samples are drawn from the Gaussian (Laplace) approximation. Only re

sn_fit_ngrid — Number of grid points to lay out per dimension when fitting the skew-normal marginals. A finer grid gives a better fit at the cost of more joint-log-posterior evaluations. Defaults to 21.

sn_fit_logthresh — The log-threshold for fitting the skew-normal. Points with log-posterior drop below this threshold (relative to the maximum) will be excluded from the fit. Defaults to -6.

sn_fit_temp — Temperature parameter for fitting the skew-normal. Defaults to 1 (weights are the density values themselves). If NA, the temperature is included as an additional optimisation parameter.

sn_fit_sample — Logical. When TRUE (default), a parametric skew-normal is fitted to the posterior samples for covariance and defined parameters. When FALSE, these are summarised using kernel density estimation instead.

control — A list of control parameters for the optimiser.

verbose — Logical indicating whether to print progress messages.

debug — Logical indicating whether to return debug information.

add_priors — Logical indicating whether to include prior densities in the posterior computation.

optim_method — The optimisation method to use for finding the posterior mode. Options include "nlminb" (default), "ucminf", and "optim" (BFGS).

numerical_grad — Logical indicating whether to use numerical gradients for the optimisation. Defaults to FALSE to use analytical gradients.

cores — Integer or NULL. Number of cores for parallel marginal fitting. When NULL (default), serial execution is used unless the number of free parameters exceeds 120, in which case parallelisation is enabled automatically using all available physical cores. Set to 1L to force serial execution. If cores > 1, marginal fits are distributed across cores using parallel::mclapply() (fork-based; no parallelism on Windows).

... — Additional arguments to be passed to the lavaan::lavaan model fitting function.

Details

The asem() function is a wrapper for the more general inlavaan() function, using the following default arguments:

- meanstructure = TRUE
- int.ov.free = FALSE
- int.lv.free = TRUE
- auto.fix.first = TRUE (unless std.lv = TRUE)
- auto.fix.single = TRUE
- auto.var = TRUE
- auto.cov.lv.x = TRUE
- auto.efa = TRUE
- auto.th = TRUE
- auto.delta = TRUE
- auto.cov.y = TRUE

Value

An S4 object of class INLAvaan which is a subclass of the lavaan::lavaan class.

See Also

Typically, users will interact with the specific latent variable model functions instead, including acfa(), asem(), and agrowth().

Examples

```
# Linear growth model with a time-varying covariate
mod <- "
  # Intercept and slope with fixed coefficients
    i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
    s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4

  # (Latent) regressions
    i ~ x1 + x2
    s ~ x1 + x2

  # Time-varying covariates
    t1 ~ c1
    t2 ~ c2
    t3 ~ c3
    t4 ~ c4
"
utils::data("Demo.growth", package = "lavaan")
str(Demo.growth)

fit <- agrowth(mod, data = Demo.growth, nsamp = 100)
summary(fit)
```

asem *Fit an Approximate Bayesian Structural Equation Model*

Description

Fit an Approximate Bayesian Structural Equation Model

Usage

```
asem(
  model,
  data,
  dp = priors_for(),
  test = "standard",
  vb_correction = TRUE,
  marginal_method = c("skewnorm", "asymgaus", "marggaus", "sampling"),
  marginal_correction = c("shortcut", "shortcut_fd", "hessian", "none"),
  nsamp = 1000,
  samp_copula = TRUE,
  sn_fit_ngrid = 21,
  sn_fit_logthresh = -6,
  sn_fit_temp = 1,
  sn_fit_sample = TRUE,
  control = list(),
  verbose = TRUE,
  debug = FALSE,
  add_priors = TRUE,
  optim_method = c("nlminb", "ucminf", "optim"),
  numerical_grad = FALSE,
  cores = NULL,
  ...
)
```

Arguments

model	A description of the user-specified model. Typically, the model is described using the lavaan model syntax. See model.syntax for more information. Alternatively, a parameter table (eg. the output of the lavParTable() function) is also accepted.
data	An optional data frame containing the observed variables used in the model. If some variables are declared as ordered factors, lavaan will treat them as ordinal variables.
dp	Default prior distributions on different types of parameters, typically the result of a call to dpriors(). See the dpriors() help file for more information.
test	Character indicating whether to compute posterior fit indices. Defaults to "standard". Change to "none" to skip these computations.

vb_correction Logical indicating whether to apply a variational Bayes correction for the posterior mean vector of estimates. Defaults to TRUE.

marginal_method
The method for approximating the marginal posterior distributions. Options include "skewnorm" (skew-normal), "asymgaus" (two-piece asymmetric Gaussian), "marggaus" (marginalising the Laplace approximation), and "sampling" (sampling from the joint Laplace approximation).

marginal_correction
Which type of correction to use when fitting the skew-normal or two-piece Gaussian marginals. "hessian" computes the full "shortcut" (default) computes only diagonals via central differences (full z-trace plus Schur complement correction), "shortcut_fd" is the same formula using forward differences (roughly half the cost, less accurate), "hessian" computes the full Hessian-based correction (slow), and "none" (or FALSE) applies no correction.

nsamp The number of samples to draw for all sampling-based approaches (including posterior sampling for model fit indices).

samp_copula Logical. When TRUE (default), posterior samples are drawn using the copula method with the fitted marginals (e.g. skew-normal or asymmetric Gaussian), with NORTA correlation adjustment. When FALSE, samples are drawn from the Gaussian (Laplace) approximation. Only re

sn_fit_ngrid Number of grid points to lay out per dimension when fitting the skew-normal marginals. A finer grid gives a better fit at the cost of more joint-log-posterior evaluations. Defaults to 21.

sn_fit_logthresh
The log-threshold for fitting the skew-normal. Points with log-posterior drop below this threshold (relative to the maximum) will be excluded from the fit. Defaults to -6.

sn_fit_temp Temperature parameter for fitting the skew-normal. Defaults to 1 (weights are the density values themselves). If NA, the temperature is included as an additional optimisation parameter.

sn_fit_sample Logical. When TRUE (default), a parametric skew-normal is fitted to the posterior samples for covariance and defined parameters. When FALSE, these are summarised using kernel density estimation instead.

control A list of control parameters for the optimiser.

verbose Logical indicating whether to print progress messages.

debug Logical indicating whether to return debug information.

add_priors Logical indicating whether to include prior densities in the posterior computation.

optim_method The optimisation method to use for finding the posterior mode. Options include "nlminb" (default), "ucminf", and "optim" (BFGS).

numerical_grad Logical indicating whether to use numerical gradients for the optimisation. Defaults to FALSE to use analytical gradients.

cores Integer or NULL. Number of cores for parallel marginal fitting. When NULL (default), serial execution is used unless the number of free parameters exceeds 120, in which case parallelisation is enabled automatically using all available

physical cores. Set to 1L to force serial execution. If cores > 1, marginal fits are distributed across cores using parallel::mclapply() (fork-based; no parallelism on Windows).

... Additional arguments to be passed to the lavaan::lavaan model fitting function.

Details

The asem() function is a wrapper for the more general inlavaan() function, using the following default arguments:

- int.ov.free = TRUE
- int.lv.free = FALSE
- auto.fix.first = TRUE (unless std.lv = TRUE)
- auto.fix.single = TRUE
- auto.var = TRUE
- auto.cov.lv.x = TRUE
- auto.efa = TRUE
- auto.th = TRUE
- auto.delta = TRUE
- auto.cov.y = TRUE

For further information regarding these arguments, please refer to the lavaan::lavOptions() documentation.

Value

An S4 object of class INLAvaan which is a subclass of the lavaan::lavaan class.

See Also

Typically, users will interact with the specific latent variable model functions instead, including acfa(), asem(), and agrowth().

Examples

```
# The industrialization and Political Democracy Example from Bollen (1989), page
# 332
model <- "
  # Latent variable definitions
     ind60 =~ x1 + x2 + x3
     dem60 =~ y1 + a*y2 + b*y3 + c*y4
     dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # (Latent) regressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60

  # Residual correlations
```

```
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
"
utils::data("PoliticalDemocracy", package = "lavaan")

fit <- asem(model, PoliticalDemocracy, test = "none")
summary(fit)
```

`as_fun_string` *Convert function to single string*

Description

Convert function to single string

Usage

```
as_fun_string(f)
```

Arguments

`f`	Function to convert.

Value

A single character vector representing the function.

Examples

```
f <- function(x) { x^2 + 1 }
as_fun_string(f)
```

`bfit_indices` *Bayesian Fit Indices*

Description

Compute posterior distributions of Bayesian fit indices for an INLAvaan model, analogous to `blavaan::blavFitIndices()`.

Usage

```
bfit_indices(
  object,
  baseline.model = NULL,
  rescale = c("devM", "MCMC"),
  nsamp = NULL,
  samp_copula = TRUE
)

## S3 method for class 'bfit_indices'
summary(object, ...)

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

Arguments

`object`	An object of class INLAvaan.
`baseline.model`	An optional INLAvaan object representing the baseline (null) model. Required for incremental fit indices (BCFI, BTLI, BNFI).
`rescale`	Character string controlling how the Bayesian chi-square is rescaled. `"devM"` (default) subtracts pD from the deviance at each sample. `"MCMC"` uses the classical chi-square and classical df at each sample.
`nsamp`	Number of posterior samples to draw. Defaults to the value used when fitting the model.
`samp_copula`	Logical. When `TRUE` (default), posterior samples are drawn using the copula method with the fitted marginals. When `FALSE`, samples are drawn from the Gaussian (Laplace) approximation.
`...`	Additional arguments passed to methods.
`x`	An object of class `bfit_indices` (for `print`).

Value

An S3 object of class `"bfit_indices"` containing:

`indices` Named list of numeric vectors (one per posterior sample) for each computed fit index.

`details` List with `chisq` (per-sample deviance), `df`, `pD`, `rescale`, and `nsamp`.

Use `summary()` to obtain a table of posterior summaries (Mean, SD, quantiles, Mode) for each index.

See Also

`lavaan::fitMeasures()`, `blavaan::blavFitIndices()`, `fitmeasures()`, `compare()`

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            verbose = FALSE)

# Absolute fit indices
bf <- bfit_indices(fit)
bf
summary(bf)
```

`compare` *Compare Bayesian Models Fitted with INLAvaan*

Description

Compare two or more Bayesian SEM fitted with INLAvaan, reporting model-fit statistics and (optionally) fit indices side by side.

Usage

```
compare(x, y, ..., fit.measures = NULL)

## S4 method for signature 'INLAvaan'
compare(x, y, ..., fit.measures = NULL)
```

Arguments

`x`	An INLAvaan (or `inlavaan_internal`) object used as the **baseline** (null) model. It is included in the comparison table and passed to fitMeasures() for incremental indices.
`y, ...`	One or more INLAvaan (or `inlavaan_internal`) objects to compare against the baseline.
`fit.measures`	Character vector of additional fit-measure names to include (e.g. `"BRMSEA"`, `"BCFI"`). Use `"all"` to include every measure returned by fitMeasures(). The default (`NULL`) shows only the core comparison statistics.

Details

The first argument `x` serves as the **baseline** (null) model. All models (including the baseline) appear in the comparison table. The baseline is also passed to fitMeasures() when incremental fit indices (BCFI, BTLI, BNFI) are requested via `fit.measures`.

The default table always includes:

- **npar**: Number of free parameters.
- **Marg.Loglik**: Approximated marginal log-likelihood.
- **logBF**: Natural-log Bayes factor relative to the best model.
- **DIC / pD**: Deviance Information Criterion and effective number of parameters (when `test != "none"` was used during fitting).

Set `fit.measures` to a character vector of measure names (anything returned by fitMeasures()) to append extra columns. Use `fit.measures = "all"` to include every available measure.

Value

A data frame of class `compare.inlavaan_internal` containing model fit statistics, sorted by descending marginal log-likelihood.

References

https://lavaan.ugent.be/tutorial/groups.html

See Also

`fitmeasures()`, `bfit_indices()`

Examples

```
# Model comparison on multigroup analysis (measurement invariance)
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")

# Configural invariance
fit1 <- acfa(HS.model, data = HolzingerSwineford1939, group = "school")

# Weak invariance
fit2 <- acfa(
  HS.model,
  data = HolzingerSwineford1939,
  group = "school",
  group.equal = "loadings"
)

# Strong invariance
```

```
fit3 <- acfa(
  HS.model,
  data = HolzingerSwineford1939,
  group = "school",
  group.equal = c("intercepts", "loadings")
)

# Compare models (fit1 = configural = baseline, always first argument)
compare(fit1, fit2, fit3)

# With extra fit measures
compare(fit1, fit2, fit.measures = c("BRMSEA", "BMc"))

# With incremental indices (baseline = fit1, passed to fitMeasures())
compare(fit1, fit2, fit3, fit.measures = c("BCFI", "BTLI"))
```

---

dbeta_box *Density of a Beta distribution on a bounded interval*

Description

Density of a Beta distribution on a bounded interval

Usage

```
dbeta_box(x, shape1, shape2, a, b, log = FALSE)
```

Arguments

x	A numeric vector of quantiles.
shape1, shape2	non-negative parameters of the Beta distribution.
a	The lower bound of the interval.
b	The upper bound of the interval.
log	Logical; if TRUE, probabilities p are given as log(p).

Value

A numeric vector of density values.

Examples

```
# Beta(2,5) on (0,100)
x <- seq(0, 100, length.out = 100)
y <- dbeta_box(x, shape1 = 2, shape2 = 5, a = 0, b = 100)
plot(x, y, type = "l", main = "Beta(2,5) on (0,100)")

# Beta(1,1) i.e. uniform on (-1, 1)
```

```
x <- seq(-1, 1, length.out = 100)
y <- dbeta_box(x, shape1 = 1, shape2 = 1, a = -1, b = 1)
plot(x, y, type = "l", main = "Beta(1,1) on (-1,1)")
```

`diagnostics` *Convergence and Approximation Diagnostics for INLAvaan Models*

Description

Extract convergence and approximation-quality diagnostics from a fitted `INLAvaan` model.

Usage

```
diagnostics(object, ...)

## S4 method for signature 'INLAvaan'
diagnostics(object, type = c("global", "param"), ...)
```

Arguments

`object`	An object of class INLAvaan.
`...`	Currently unused.
`type`	Character. `"global"` (default) returns a named numeric vector of scalar diagnostics. `"param"` returns a data frame with one row per free parameter containing per-parameter diagnostics.

Details

Global diagnostics (`type = "global"`):

`npar` Number of free parameters.

`nsamp` Number of posterior samples drawn.

`converged` 1 if the optimiser converged, 0 otherwise.

`iterations` Number of optimiser iterations.

`grad_inf` L-infinity norm of the analytic gradient at the mode (max |grad|). Should be ~0 at convergence.

`grad_inf_rel` Relative L-infinity norm of the analytic gradient (max |grad| / (|par| + 1e-6)).

`grad_l2` L2 (Euclidean) norm of the analytic gradient at the mode.

`hess_cond` Condition number of the Hessian (precision matrix) computed from Σ_θ. Large values indicate near-singularity.

`vb_kld_global` Global KL divergence from the VB mean correction (NA if VB correction was not applied).

`vb_applied` 1 if VB correction was applied, 0 otherwise.

`kld_max` Maximum per-parameter KL divergence from the VB correction.

`kld_mean` Mean per-parameter KL divergence.

`nmad_max` Maximum normalised max-absolute-deviation across marginals (skew-normal method only; NA otherwise).

`nmad_mean` Mean NMAD across marginals.

Per-parameter diagnostics (`type = "param"`): A data frame with columns:

`param` Parameter name.

`grad` Analytic gradient of the negative log-posterior at the mode. Should be ~0 at convergence.

`grad_num` Numerical (finite-difference) gradient at the mode. Should agree with `grad`; large discrepancies indicate a bug in the analytic gradient.

`grad_diff` Difference `grad_num - grad`: should be ~0.

`grad_abs` Absolute analytic gradient.

`grad_rel` Relative analytic gradient |grad| / (|par| + 1e-6).

`kld` Per-parameter KL divergence from the VB correction.

`vb_shift` VB correction shift (in original scale).

`vb_shift_sigma` VB shift in units of posterior SD.

`nmad` Normalised max-absolute-deviation of the skew-normal fit (NA when not using the skewnorm method).

Value

For `type = "global"`, a named numeric vector (class `"diagnostics.INLAvaan"`). For `type = "param"`, a data frame (class `c("diagnostics.INLAvaan.param", "data.frame")`).

See Also

`timing()`, `fitmeasures()`, `plot()`

Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            test = "none", verbose = FALSE)

# Global convergence summary
diagnostics(fit)

# Per-parameter table
diagnostics(fit, type = "param")
```

dsnorm *The Skew Normal Distribution*

Description

Density for the skew normal distribution with location `xi`, scale `omega`, and shape `alpha`.

Usage

```
dsnorm(x, xi, omega, alpha, logC = 0, log = FALSE)
```

Arguments

`x`	Vector of quantiles.
`xi`	Location parameter.
`omega`	Scale parameter.
`alpha`	Shape parameter.
`logC`	Log-normalization constant.
`log`	Logical; if TRUE, returns the log density.

Value

A numeric vector of (log) density values.

References

https://en.wikipedia.org/wiki/Skew_normal_distribution

Examples

```
x <- seq(-2, 5, length.out = 100)
y <- dsnorm(x, xi = 0, omega = 1, alpha = 5)
plot(x, y, type = "l", main = "Skew Normal Density")
```

fitmeasures *Fit Measures for a Latent Variable Model estimated using INLA*

Description

Fit Measures for a Latent Variable Model estimated using INLA

Usage

```
## S4 method for signature 'INLAvaan'
fitMeasures(
  object,
  fit.measures = "all",
  baseline.model = NULL,
  h1.model = NULL,
 fm.args = list(standard.test = "default", scaled.test = "default", rmsea.ci.level =
  0.9, rmsea.close.h0 = 0.05, rmsea.notclose.h0 = 0.08, robust = TRUE, cat.check.pd =
    TRUE),
  output = "vector",
  ...
)

## S4 method for signature 'INLAvaan'
fitmeasures(
  object,
  fit.measures = "all",
  baseline.model = NULL,
  h1.model = NULL,
 fm.args = list(standard.test = "default", scaled.test = "default", rmsea.ci.level =
  0.9, rmsea.close.h0 = 0.05, rmsea.notclose.h0 = 0.08, robust = TRUE, cat.check.pd =
    TRUE),
  output = "vector",
  ...
)
```

Arguments

`object`	An object of class INLAvaan.
`fit.measures`	If `"all"`, all fit measures available will be returned. If only a single or a few fit measures are specified by name, only those are computed and returned.
`baseline.model`	An optional INLAvaan object representing the baseline (null) model. Required for incremental fit indices (BCFI, BTLI, BNFI). Must have been fitted with `test != "none"`.
`h1.model`	Ignored (included for compatibility with the lavaan generic).
`fm.args`	Ignored (included for compatibility with the lavaan generic).
`output`	Ignored (included for compatibility with the lavaan generic).
`...`	Additional arguments. Currently supports:
	`rescale` Character string controlling how the Bayesian chi-square is computed, following `blavaan::blavFitIndices()`. Options are `"devM"` (default) which uses the deviance rescaled by `pD` from DIC, or `"MCMC"` which uses the classical chi-square (`(N-1) * F_ML`) and classical degrees of freedom (`p - npar`) at each posterior sample.

Value

A named numeric vector of fit measures.

See Also

bfit_indices(), compare(), diagnostics()

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            verbose = FALSE)

# All available fit measures
fitMeasures(fit)

# Specific measures
fitMeasures(fit, c("npar", "DIC", "pD", "ppp"))
```

---

fit_skew_normal *Fit a skew normal distribution to log-density evaluations*

Description

Fit a skew normal distribution to log-density evaluations

Usage

```
fit_skew_normal(x, y, threshold_log_drop = -6, temp = NA)
```

Arguments

x	A numeric vector of points where the density is evaluated.
y	A numeric vector of log-density evaluations at points x.
threshold_log_drop	A negative numeric value indicating the log-density drop threshold below which points are ignored in the fitting. Default is -6.
temp	A numeric value for the temperature parameter k. If NA (default), it is included in the optimisation.

Details

This skew normal fitting function uses a weighted least squares approach to fit the log-density evaluations provided in `y` at points `x`. The weights are set to be the density evaluations raised to the power of the temperature parameter `k`. This has somewhat an interpretation of finding the skew normal fit that minimises the Kullback-Leibler divergence from the true density to it.

In R-INLA, the C code implementation from which this was translated from can be found here.

Value

A list with fitted parameters:

- `xi`: location parameter
- `omega`: scale parameter
- `alpha`: shape parameter
- `logC`: log-normalization constant
- `k`: temperature parameter
- `rsq`: R-squared of the fit

Note that `logC` and `k` are not used when fitting from a sample.

Examples

```
# Fit a SN curve to gamma log-density
logdens <- function(x) dgamma(x, shape = 3, rate = 1, log = TRUE)

x_grid <- seq(0.1, 8, length.out = 21)
y_log  <- sapply(x_grid, logdens)
y_log  <- y_log - max(y_log)  # normalise to have maximum at zero

res <- fit_skew_normal(x_grid, y_log, temp = 10)
unlist(res)

# Compare truth vs skew-normal approximation
x_fine <- seq(0.1, 8, length.out = 200)
y_true <- exp(logdens(x_fine))
y_sn   <- dsnorm(x_fine, xi = res$xi, omega = res$omega, alpha = res$alpha)

plot(x_fine, y_true, type = "n", xlab = "x", ylab = "Density", bty = "n")
polygon(c(x_fine, rev(x_fine)), c(y_true, rep(0, length(x_fine))),
        col = adjustcolor("#131516", 0.25), border = NA)
lines(x_fine, y_sn, col = "#00A6AA", lwd = 2)
legend("topright", legend = c("Truth", "SN Approx."),
       fill = c(adjustcolor("#131516", 0.25), NA), border = NA,
       col = c(NA, "#00A6AA"), lwd = c(NA, 2), lty = c(NA, 1),
       bty = "n")
```

`fit_skew_normal_samp` *Fit a skew normal distribution to a sample*

Description

Fit a skew normal distribution to a sample

Usage

```
fit_skew_normal_samp(x)
```

Arguments

`x`	A numeric vector of sample data.

Details

Uses maximum likelihood estimation to fit a skew normal distribution to the provided numeric vector `x`.

Value

A list with fitted parameters:

- `xi`: location parameter
- `omega`: scale parameter
- `alpha`: shape parameter
- `logC`: log-normalization constant
- `k`: temperature parameter
- `rsq`: R-squared of the fit

Note that `logC` and `k` are not used when fitting from a sample.

Examples

```
x <- rnorm(100, mean = 5, sd = 1)
unlist(fit_skew_normal_samp(x))
```

`get_inlavaan_internal` *Extract the Internal INLAvaan Object*

Description

Returns the `inlavaan_internal` list stored inside a fitted INLAvaan object, optionally extracting a single named element.

Usage

```
get_inlavaan_internal(object, what)
```

Arguments

`object`	An object of class INLAvaan.
`what`	Character. Name of the element to extract from the internal list. If missing, the entire list is returned. Common elements include `"coefficients"`, `"summary"`, `"Sigma_theta"`, `"vcov_x"`, `"theta_star"`, `"approx_data"`, `"pdf_data"`, `"partable"`, `"marginal_method"`, `"nsamp"`, `"mloglik"`, `"DIC"`, `"ppp"`, `"vb"`, `"opt"`, `"timing"`, `"visual_debug"`.

Value

The full `inlavaan_internal` list, or the named element when `what` is supplied.

See Also

INLAvaan, `diagnostics()`, `timing()`

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            test = "none", verbose = FALSE)

# Full internal object
int <- get_inlavaan_internal(fit)
names(int)

# Extract a specific element
get_inlavaan_internal(fit, "coefficients")
```

inlavaan *Fit an Approximate Bayesian Latent Variable Model*

Description

This function fits a Bayesian latent variable model by approximating the posterior distributions of the model parameters using various methods, including skew-normal, asymmetric Gaussian, marginal Gaussian, or sampling-based approaches. It leverages the lavaan package for model specification and estimation.

Usage

```
inlavaan(
  model,
  data,
  model.type = "sem",
  dp = priors_for(),
  test = "standard",
  vb_correction = TRUE,
  marginal_method = c("skewnorm", "asymgaus", "marggaus", "sampling"),
  marginal_correction = c("shortcut", "shortcut_fd", "hessian", "none"),
  nsamp = 1000,
  samp_copula = TRUE,
  sn_fit_ngrid = 21,
  sn_fit_logthresh = -6,
  sn_fit_temp = 1,
  sn_fit_sample = TRUE,
  control = list(),
  verbose = TRUE,
  debug = FALSE,
  add_priors = TRUE,
  optim_method = c("nlminb", "ucminf", "optim"),
  numerical_grad = FALSE,
  cores = NULL,
  ...
)
```

Arguments

model	A description of the user-specified model. Typically, the model is described using the lavaan model syntax. See model.syntax for more information. Alternatively, a parameter table (eg. the output of the lavParTable() function) is also accepted.
data	An optional data frame containing the observed variables used in the model. If some variables are declared as ordered factors, lavaan will treat them as ordinal variables.

`model.type` Set the model type: possible values are `"cfa"`, `"sem"` or `"growth"`. This may affect how starting values are computed, and may be used to alter the terminology used in the summary output, or the layout of path diagrams that are based on a fitted lavaan object.

`dp` Default prior distributions on different types of parameters, typically the result of a call to `dpriors()`. See the `dpriors()` help file for more information.

`test` Character indicating whether to compute posterior fit indices. Defaults to "standard". Change to "none" to skip these computations.

`vb_correction` Logical indicating whether to apply a variational Bayes correction for the posterior mean vector of estimates. Defaults to `TRUE`.

`marginal_method`
The method for approximating the marginal posterior distributions. Options include `"skewnorm"` (skew-normal), `"asymgaus"` (two-piece asymmetric Gaussian), `"marggaus"` (marginalising the Laplace approximation), and `"sampling"` (sampling from the joint Laplace approximation).

`marginal_correction`
Which type of correction to use when fitting the skew-normal or two-piece Gaussian marginals. `"hessian"` computes the full `"shortcut"` (default) computes only diagonals via central differences (full z-trace plus Schur complement correction), `"shortcut_fd"` is the same formula using forward differences (roughly half the cost, less accurate), `"hessian"` computes the full Hessian-based correction (slow), and `"none"` (or `FALSE`) applies no correction.

`nsamp` The number of samples to draw for all sampling-based approaches (including posterior sampling for model fit indices).

`samp_copula` Logical. When `TRUE` (default), posterior samples are drawn using the copula method with the fitted marginals (e.g. skew-normal or asymmetric Gaussian), with NORTA correlation adjustment. When `FALSE`, samples are drawn from the Gaussian (Laplace) approximation. Only re

`sn_fit_ngrid` Number of grid points to lay out per dimension when fitting the skew-normal marginals. A finer grid gives a better fit at the cost of more joint-log-posterior evaluations. Defaults to `21`.

`sn_fit_logthresh`
The log-threshold for fitting the skew-normal. Points with log-posterior drop below this threshold (relative to the maximum) will be excluded from the fit. Defaults to `-6`.

`sn_fit_temp` Temperature parameter for fitting the skew-normal. Defaults to `1` (weights are the density values themselves). If `NA`, the temperature is included as an additional optimisation parameter.

`sn_fit_sample` Logical. When `TRUE` (default), a parametric skew-normal is fitted to the posterior samples for covariance and defined parameters. When `FALSE`, these are summarised using kernel density estimation instead.

`control` A list of control parameters for the optimiser.

`verbose` Logical indicating whether to print progress messages.

`debug` Logical indicating whether to return debug information.

add_priors	Logical indicating whether to include prior densities in the posterior computation.
optim_method	The optimisation method to use for finding the posterior mode. Options include "nlminb" (default), "ucminf", and "optim" (BFGS).
numerical_grad	Logical indicating whether to use numerical gradients for the optimisation. Defaults to FALSE to use analytical gradients.
cores	Integer or NULL. Number of cores for parallel marginal fitting. When NULL (default), serial execution is used unless the number of free parameters exceeds 120, in which case parallelisation is enabled automatically using all available physical cores. Set to 1L to force serial execution. If cores > 1, marginal fits are distributed across cores using parallel::mclapply() (fork-based; no parallelism on Windows).
...	Additional arguments to be passed to the lavaan::lavaan model fitting function.

Value

An S4 object of class INLAvaan which is a subclass of the lavaan::lavaan class.

See Also

Typically, users will interact with the specific latent variable model functions instead, including acfa(), asem(), and agrowth().

Examples

```
# The Holzinger and Swineford (1939) example
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")

fit <- inlavaan(
  HS.model,
  data = HolzingerSwineford1939,
  auto.var = TRUE,
  auto.fix.first = TRUE,
  auto.cov.lv.x = TRUE
)
summary(fit)
```

INLAvaan-class *Class For Representing a (Fitted) Latent Variable Model*

Description

This is a class that extends the lavaan::lavaan class. Several S4 methods are available.

Usage

```
## S4 method for signature 'INLAvaan'
coef(object, ...)

## S4 method for signature 'INLAvaan'
nobs(object, ...)

## S4 method for signature 'INLAvaan'
show(object)

## S4 method for signature 'INLAvaan'
summary(
  object,
  header = TRUE,
  fit.measures = TRUE,
  estimates = TRUE,
  standardized = FALSE,
  rsquare = FALSE,
  postmedian = FALSE,
  postmode = FALSE,
  nmad = TRUE,
  kld = FALSE,
  vb_shift = FALSE,
  priors = TRUE,
  nd = 3L,
  ...
)
```

Arguments

object	An object of class INLAvaan.
...	Additional arguments passed to methods.
header	Logical; if TRUE, print model fit information header.
fit.measures	Logical; if TRUE, print fit measures (DIC and PPP).
estimates	Logical; if TRUE, print parameter estimates table.
standardized	Logical; if TRUE, include standardized estimates.
rsquare	Logical; if TRUE, include R-square values.
postmedian	Logical; if TRUE, include posterior median in estimates.
postmode	Logical; if TRUE, include posterior mode in estimates.
nmad	Logical; if TRUE (default), include the NMAD column for skew-normal marginal fit quality.
kld	Logical; if FALSE (default), omit the per-parameter KLD column. Set to TRUE to show it.
vb_shift	Logical; if FALSE (default), omit the VB shift column (shift in units of posterior SD). Set to TRUE to show it.

priors	Logical; if TRUE, include prior information in estimates.
nd	Integer; number of decimal places to print for numeric values.

Slots

external A list containing an inlavaan_internal object.

See Also

lavaan::lavaan, inlavaan(), acfa(), asem(), agrowth()

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            test = "none", verbose = FALSE)

# Print basic info
fit

# Detailed summary
summary(fit)

# Extract coefficients
coef(fit)
```

---

is_same_function *Helper function to check if two functions are the same*

Description

Helper function to check if two functions are the same

Usage

```
is_same_function(f, g)
```

Arguments

f, g	Functions to compare.

Value

Logical.

Examples

```
f1 <- function(x) { x^2 + 1 }
f2 <- function(x) { x^2 + 1 }
is_same_function(f1, f2)  # TRUE
```

---

plot *Plot an INLAvaan Object*

Description

Generates diagnostic plots for a fitted INLAvaan model.

Usage

```
## S4 method for signature 'INLAvaan,ANY'
plot(
  x,
  y,
  type = c("marg_pdf", "sn_fit", "sn_fit_log"),
  params = "all",
  nrow = NULL,
  ncol = NULL,
  use_ggplot = TRUE,
  points = FALSE,
  ...
)
```

Arguments

x	An object of class INLAvaan.
y	Not used.
type	Character. One of "marg_pdf" (default; posterior marginal densities), "sn_fit" (skew-normal fit diagnostic on natural scale), or "sn_fit_log" (same on log scale).
params	Character vector of parameter names to plot, or "all" (default) to plot all free parameters.
nrow, ncol	Integer. Number of rows/columns for the facet grid when use_ggplot = TRUE. If NULL (default), layout is chosen automatically.
use_ggplot	Logical. When TRUE (default) and **ggplot2** is available, a **ggplot2** object is returned. Set to FALSE for a base-R plot.
points	Logical. When TRUE, individual grid points are overlaid on the curves. Defaults to FALSE.
...	Additional arguments (currently unused).

See Also

diagnostics(), summary()

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            test = "none", verbose = FALSE)

# Posterior marginal densities (default)
plot(fit)

# Skew-normal fit diagnostic for a single parameter
plot(fit, type = "sn_fit", params = "visual=~x1")
```

---

predict *Posterior Predictions for INLAvaan Models*

Description

Compute posterior predictions from a fitted INLAvaan model, including latent variable scores, predicted observed values, and imputed missing data.

Usage

```
## S4 method for signature 'INLAvaan'
predict(
  object,
  type = c("lv", "yhat", "ov", "ypred", "ydist", "ymis", "ovmis"),
  newdata = NULL,
  level = 1L,
  nsamp = 1000,
  ymis_only = FALSE,
  ...
)
```

Arguments

object	An object of class INLAvaan.
type	Character string specifying the type of prediction:
	"lv" (default) Posterior draws of latent variable scores $\eta \mid y, \theta$.

"yhat", "ov" Predicted means for observed variables $E(y|\eta, \theta) = \nu + \Lambda\eta$; no residual noise.

"ypred", "ydist" Predicted observed values including residual noise $y = \nu + \Lambda\eta + \varepsilon, \varepsilon \sim N(0, \Theta)$.

"ymis", "ovmis" Imputed values for missing observations, drawn from the conditional distribution $y_{mis}|y_{obs}, \theta$.

newdata — An optional data frame of new observations. If supplied, predictions are computed for newdata rather than the original training data. Not supported for type = "ymis".

level — Integer; for type = "lv" in multilevel models, specifies whether level 1 or level 2 latent variables are desired (default 1L).

nsamp — Integer; number of posterior samples to use for prediction. Defaults to 1000.

ymis_only — Logical; only applies when type = "ymis". When TRUE, returns only the imputed values as a named numeric vector per sample (names of the form "varname[rowindex]", matching the blavaan convention). When FALSE (default), returns the full data matrix with missing values filled in.

... — Currently unused.

Value

A matrix of posterior draws with rows corresponding to samples and columns to variables or latent factors, or a list of such matrices when ymis_only = FALSE.

See Also

sampling(), simulate(), summary()

Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            test = "none", verbose = FALSE)

# Posterior latent variable scores
lv_scores <- predict(fit)
head(lv_scores)

# Predicted observed variable means
yhat <- predict(fit, type = "yhat")
head(yhat)
```

`priors_for` *Specify priors for a SEM*

Description

Specify priors for a SEM, similar to how `blavaan::dpriors()` works.

Usage

```
priors_for(...)
```

Arguments

`...`	Named arguments specifying prior distributions for lavaan parameter types.

Details

This function provides a convenient way to specify prior distributions for different types of parameters in a structural equation model (SEM). It uses a registry of default priors for common lavaan parameter types (e.g., loadings, regressions, residuals, etc.) and allows users to override these defaults by passing named arguments.

The parameter names, and default settings, are:

- `nu` = `"normal(0,32)"`: Observed variable intercepts
- `alpha` = `"normal(0,10)"`: Latent variable intercepts
- `lambda` = `"normal(0,10)"`: Factor loadings
- `beta` = `"normal(0,10)"`: Regression coefficients
- `theta` = `"gamma(1,.5)[sd]"`: Residual precisions
- `psi` = `"gamma(1,.5)[sd]"`: Latent variable precisions
- `rho` = `"beta(1,1)"`: Correlations (both latent and observed)
- `tau` = `"normal(0,1.5)"`: Thresholds for ordinal variables

Note that the normal distributions are parameterised using standard deviations, and not variances. For example, `normal(0,10)` means a normal distribution with mean 0 and standard deviation 10 (not variance 10).

Value

A named character vector of prior specifications, where names correspond to lavaan parameter types (e.g., "lambda", "beta", "theta", etc.) and values are character strings specifying the prior distribution (e.g., `"normal(0,10)"`, `"gamma(1,0.5)[sd]"`, `"gamma(1,1)[prec]"`, etc.).

Scale qualifiers

For variance parameters (`theta`, `psi`), the prior distribution can be placed on a transformed scale by appending a qualifier:

- `[sd]`: Prior is on the standard deviation σ. Example: `"gamma(1,0.5)[sd]"` places a Gamma(1, 0.5) prior on $\sigma = \sqrt{\text{variance}}$.
- `[prec]`: Prior is on the precision $\tau = 1/\sigma^2$. Example: `"gamma(1,1)[prec]"` places a Gamma(1, 1) prior on $\tau = 1/\text{variance}$. This is the parameterisation used by blavaan and corresponds to an Inverse-Gamma prior on the variance.

The necessary Jacobian adjustment is applied automatically in both cases.

See Also

`inlavaan()`, `acfa()`, `asem()`, `agrowth()`

Examples

```
priors_for(nu = "normal(0,10)", lambda = "normal(0,1)", rho = "beta(3,3)")

# Precision-scale prior for residual variances (blavaan-style)
priors_for(theta = "gamma(1,1)[prec]")
```

---

qsnorm_fast *Fast Approximation of Skew-Normal Quantile Function*

Description

A fast approximation of skew-normal quantiles using the high-performance approximation algorithm from the INLA GMRFLib C source, and originally by Thomas Luu (see details for reference).

Usage

```
qsnorm_fast(p, xi = 0, omega = 1, alpha = 0)
```

Arguments

`p`	Vector of probabilities.
`xi`	Location parameter (numeric vector).
`omega`	Scale parameter (numeric vector).
`alpha`	Shape parameter (numeric vector).

Details

This function implements a high-performance approximation for the skew-normal quantile function based on the algorithm described by Luu (2016). The method uses a domain decomposition strategy to achieve high accuracy ($< 10^{-7}$ relative error) without iterative numerical inversion.

The domain is split into two regions:

- **Tail Regions:** For extreme probabilities where $|u|$ is large, the quantile is approximated using the Lambert W-function, $W(z)$, solving $z = \Phi(q)$ via asymptotic expansion:

$$q \approx \sqrt{2W\left(\frac{1}{2\pi(1-p)^2}\right)}$$

- **Central Region:** For the main body of the distribution, the function uses a high-order Taylor expansion of the inverse error function around a carefully selected expansion point x_0:

$$\Phi^{-1}(p) \approx \sum_{k=0}^{5} c_k (z - x_0)^k$$

This approach is significantly faster than standard numerical inversion (e.g., `uniroot`) while maintaining sufficient precision for most statistical applications.

Value

Vector of quantiles.

References

Luu, T. (2016). *Fast and accurate parallel computation of quantile functions for random number generation* #' (Doctoral thesis). UCL (University College London). https://discovery.ucl.ac.uk/1482128/

Examples

```
qsnorm_fast(c(0.025, 0.5, 0.975))
qsnorm_fast(c(0.025, 0.5, 0.975), xi = 2, omega = 0.5, alpha = 1)
```

`sampling` *Draw Samples from the Generative Model*

Description

Sample model parameters, latent variables, or observed variables from the generative model underlying a fitted INLAvaan model. By default, parameters are drawn from the **posterior** distribution; set `prior = TRUE` to draw from the **prior** instead (useful for prior predictive checks).

Usage

```
sampling(object, ...)

## S4 method for signature 'INLAvaan'
sampling(
  object,
  type = c("lavaan", "theta", "latent", "observed", "implied", "all"),
  nsamp = 1000L,
  samp_copula = TRUE,
  prior = FALSE,
  silent = FALSE,
  ...
)
```

Arguments

object	An object of class INLAvaan (or inlavaan_internal).
...	Additional arguments (currently unused).
type	Character string specifying what to sample:
	"lavaan" (Default) The lavaan-side (constrained) model parameters. Returns an nsamp by npar matrix.
	"theta" The INLAvaan-side unconstrained parameters. Returns an nsamp by npar matrix.
	"latent" Latent variables from the model-implied distribution. Returns an nsamp by nlv matrix (one draw per posterior sample, not tied to any individual).
	"observed" Observed variables generated from the full model. Returns an nsamp by nobs_vars matrix.
	"implied" Model-implied moments. Returns a length-nsamp list, each element a list with cov (model-implied covariance matrix) and, when meanstructure = TRUE, mean (model-implied mean vector). For multi-group models each element is itself a list of groups.
	"all" A named list with elements lavaan, theta, latent, observed, and implied.
nsamp	Number of samples to draw.
samp_copula	Logical. When TRUE (default), posterior parameter samples use the copula method with the fitted marginals. When FALSE, samples are drawn from the joint Gaussian (Laplace) approximation. Ignored when prior = TRUE.
prior	Logical. When TRUE, parameters are drawn from the prior distribution and then propagated through the generative model. When FALSE (default), parameters come from the posterior.
silent	Logical. When TRUE, suppresses the informational message about rejected non-PD draws during prior rejection sampling. Default FALSE.

Details

Each row of the output corresponds to a **fresh parameter draw**: a new $\boldsymbol{\theta}^{(s)}$ is sampled and then propagated through the generative chain to produce one latent vector and one observed vector. This makes `sampling()` ideal for **prior and posterior predictive checks** (e.g., density overlays, test statistic distributions).

The generative chain is:

$$\boldsymbol{\theta}^{(s)} \sim \pi(\boldsymbol{\theta} \mid \mathbf{y})$$

$$\boldsymbol{\eta}^{(s)} \sim N((\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\alpha},\, \boldsymbol{\Phi})$$

$$\mathbf{y}^{*(s)} \sim N(\boldsymbol{\Lambda}\boldsymbol{\eta}^{(s)} + \boldsymbol{\nu},\, \boldsymbol{\Theta})$$

If you need **complete replicate datasets** (many observations from a single parameter draw) — for example, for simulation-based calibration (SBC) — use `simulate()` instead.

This is distinct from `predict()`, which computes individual-specific factor scores $\boldsymbol{\eta} \mid \mathbf{y}, \boldsymbol{\theta}$ conditional on observed data.

Value

A matrix or named list, depending on `type`.

See Also

`simulate()` for generating complete replicate datasets (e.g., for SBC); `predict()` for individual-specific factor scores; `bfit_indices()` for Bayesian fit indices.

Examples

```
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa("visual =~ x1 + x2 + x3", HolzingerSwineford1939)

# Posterior samples of lavaan-side parameters
samps <- sampling(fit, nsamp = 500)
head(samps)

# Compare copula vs Gaussian sampling
s_cop <- sampling(fit, nsamp = 500, samp_copula = TRUE)
s_gaus <- sampling(fit, nsamp = 500, samp_copula = FALSE)

# Prior predictive samples
y_prior <- sampling(fit, type = "observed", nsamp = 500, prior = TRUE)
```

simulate *Simulate Datasets from the Generative Model*

Description

Generate complete synthetic datasets from a fitted INLAvaan model. For each simulation, a single parameter vector is drawn (from the posterior or prior), and then `sample.nobs` observations are generated from the model-implied distribution at that parameter value.

Usage

```
## S4 method for signature 'INLAvaan'
simulate(
  object,
  nsim = 1L,
  seed = NULL,
  sample.nobs = NULL,
  prior = FALSE,
  samp_copula = TRUE,
  silent = FALSE,
  ...
)
```

Arguments

`object`	An object of class INLAvaan.
`nsim`	Number of replicate datasets to generate (default 1).
`seed`	Optional random seed (passed to `set.seed()`).
`sample.nobs`	Number of observations per dataset. Defaults to the sample size of the original data.
`prior`	Logical. When `TRUE`, parameters are drawn from the prior; when `FALSE` (default), from the posterior.
`samp_copula`	Logical. When `TRUE` (default) and `prior = FALSE`, posterior parameter draws use the copula method. Ignored when `prior = TRUE`.
`silent`	Logical. When `TRUE`, suppresses the informational message about rejected non-PD draws. Default `FALSE`.
`...`	Additional arguments (currently unused).

Details

This function is designed for tasks that require **full replicate datasets** from a single parameter draw, such as simulation-based calibration (SBC) and posterior predictive p-values. It differs from `sampling()` which generates one observation per parameter draw (useful for prior/posterior predictive density overlays).

For each simulation $s = 1, \ldots, S$:

1. Draw $\boldsymbol{\theta}^{(s)}$ from the posterior (or prior).
2. Compute the model-implied covariance $\boldsymbol{\Sigma}(\boldsymbol{\theta}^{(s)})$. If it is not positive-definite, reject and re-draw.
3. Generate a dataset of `sample.nobs` rows from $N(\boldsymbol{\mu}(\boldsymbol{\theta}^{(s)}),\ \boldsymbol{\Sigma}(\boldsymbol{\theta}^{(s)}))$.

Parameter draws reuse the same internal machinery as `sampling()` (`sample_params_prior` / `sample_params_posterior`), so the prior specification is consistent.

Value

A list of length `nsim`. Each element is a data frame with `sample.nobs` rows and two attributes:

- `"truth"` — named numeric vector of lavaan-side (x-space, constrained) parameter values used to generate the dataset.
- `"truth_theta"` — named numeric vector of the corresponding unconstrained (theta-space) parameter values.

See Also

`sampling()` for single-observation draws from the predictive distribution (prior/posterior predictive checks).

Examples

```
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa("visual =~ x1 + x2 + x3", HolzingerSwineford1939)

# Simulate one replicate dataset from the posterior
sims <- simulate(fit, nsim = 1)
head(sims[[1]])                    # data frame
attr(sims[[1]], "truth")           # true lavaan-side (x-space) parameters
attr(sims[[1]], "truth_theta")     # corresponding unconstrained (theta-space) parameters

# Simulate from the prior (e.g., for SBC)
sims_prior <- simulate(fit, nsim = 5, prior = TRUE)
lapply(sims_prior, nrow)
```

---

standardisedsolution *Standardised solution of a latent variable model*

Description

Standardised solution of a latent variable model

Usage

```
standardisedsolution(
  object,
  type = "std.all",
  se = TRUE,
  ci = TRUE,
  level = 0.95,
  postmedian = FALSE,
  postmode = FALSE,
  cov.std = TRUE,
  remove.eq = TRUE,
  remove.ineq = TRUE,
  remove.def = FALSE,
  nsamp = 250,
  ...
)

standardisedSolution(
  object,
  type = "std.all",
  se = TRUE,
  ci = TRUE,
  level = 0.95,
  postmedian = FALSE,
  postmode = FALSE,
  cov.std = TRUE,
  remove.eq = TRUE,
  remove.ineq = TRUE,
  remove.def = FALSE,
  nsamp = 250,
  ...
)

standardizedsolution(
  object,
  type = "std.all",
  se = TRUE,
  ci = TRUE,
  level = 0.95,
  postmedian = FALSE,
  postmode = FALSE,
  cov.std = TRUE,
  remove.eq = TRUE,
  remove.ineq = TRUE,
  remove.def = FALSE,
  nsamp = 250,
  ...
)
```

```
standardizedSolution(
  object,
  type = "std.all",
  se = TRUE,
  ci = TRUE,
  level = 0.95,
  postmedian = FALSE,
  postmode = FALSE,
  cov.std = TRUE,
  remove.eq = TRUE,
  remove.ineq = TRUE,
  remove.def = FALSE,
  nsamp = 250,
  ...
)
```

Arguments

`object`	An object of class INLAvaan.
`type`	If `"std.lv"`, the standardized estimates are on the variances of the (continuous) latent variables only. If `"std.all"`, the standardized estimates are based on both the variances of both (continuous) observed and latent variables. If `"std.nox"`, the standardized estimates are based on both the variances of both (continuous) observed and latent variables, but not the variances of exogenous covariates.
`se`	Logical. If TRUE, standard errors for the standardized parameters will be computed, together with a z-statistic and a p-value.
`ci`	If `TRUE`, simple symmetric confidence intervals are added to the output
`level`	The confidence level required.
`postmedian`	Logical; if TRUE, include posterior median in estimates.
`postmode`	Logical; if TRUE, include posterior mode in estimates.
`cov.std`	Logical. If TRUE, the (residual) observed covariances are scaled by the square root of the 'Theta' diagonal elements, and the (residual) latent covariances are scaled by the square root of the 'Psi' diagonal elements. If FALSE, the (residual) observed covariances are scaled by the square root of the diagonal elements of the observed model-implied covariance matrix (Sigma), and the (residual) latent covariances are scaled by the square root of diagonal elements of the model-implied covariance matrix of the latent variables.
`remove.eq`	Logical. If TRUE, filter the output by removing all rows containing equality constraints, if any.
`remove.ineq`	Logical. If TRUE, filter the output by removing all rows containing inequality constraints, if any.
`remove.def`	Logical. If TRUE, filter the ouitput by removing all rows containing parameter definitions, if any.
`nsamp`	The number of samples to draw from the approximate posterior distribution for the calculation of standardised estimates.
`...`	Additional arguments sent to `lavaan::standardizedSolution()`.

Value

A data.frame containing standardised model parameters.

See Also

summary(), coef(), vcov()

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
"
utils::data("HolzingerSwineford1939", package = "lavaan")

# Fit a CFA model with standardised latent variables
fit <- acfa(
  HS.model,
  data = HolzingerSwineford1939,
  test = "none",
  nsamp = 10,
  vb_correction = FALSE,
  verbose = FALSE
)
standardisedsolution(fit, nsamp = 10, se = FALSE, ci = FALSE)
```

---

timing *Timing Information for INLAvaan Models*

Description

Extract wall-clock timings for individual computation stages of a fitted INLAvaan model.

Usage

```
timing(object, ...)

## S4 method for signature 'INLAvaan'
timing(object, what = "total", ...)
```

Arguments

object	An object of class INLAvaan.
...	Currently unused.
what	Character vector of timing segment names to return, or "all" to return every segment. Defaults to "total". Available segments (depending on model options): "init", "optim", "vb", "loglik", "marginals", "norta", "sampling", "covariances", "definedpars", "deltapars", "test", "total".

Value

A named numeric vector (class `c("timing.INLAvaan", "numeric")`) of elapsed times in seconds. Printing formats short durations as seconds, longer ones as minutes or hours.

See Also

`diagnostics()`, `summary()`

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            test = "none", verbose = FALSE)

# Total elapsed time
timing(fit)

# All stages
timing(fit, what = "all")

# Specific stages
timing(fit, what = c("optim", "marginals"))
```

---

vcov *Variance-Covariance Matrix for INLAvaan Models*

Description

Extract the posterior variance-covariance matrix of model parameters from a fitted `INLAvaan` model.

Usage

```
## S4 method for signature 'INLAvaan'
vcov(object, type = c("lavaan", "theta"), ...)
```

Arguments

`object`	An object of class INLAvaan.
`type`	Character. `"lavaan"` (default) returns the posterior covariance matrix of the model parameters computed from posterior samples (matching lavaan output). `"theta"` returns the Laplace approximation covariance in the internal parameterisation.
`...`	Currently unused.

Value

A square numeric matrix.

See Also

summary(), coef(), standardisedsolution()

Examples

```
HS.model <- "
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
"
utils::data("HolzingerSwineford1939", package = "lavaan")
fit <- acfa(HS.model, HolzingerSwineford1939, std.lv = TRUE, nsamp = 100,
            test = "none", verbose = FALSE)

# Default: posterior covariance of lavaan parameters
vcov(fit)

# Internal parameterisation (Laplace approximation)
vcov(fit, type = "theta")
```

Index