Package 'blavaan'

October 31, 2018

Title Bayesian Latent Variable Analysis

Version 0.3-3

Description Fit a variety of Bayesian latent variable models, including confirmatory factor analysis, structural equation models, and latent growth curve models.

Depends R(>= 3.2.0), methods, lavaan(>= 0.6-3)

Imports stats, utils, graphics, MCMCpack, coda, mnormt, nonnest2(>= 0.5-2), loo(>= 2.0)

Suggests runjags(>= 2.0.4-2), rstan(>= 2.17-3), modeest, rjags, semTools, parallel, testthat(>= 2.0.0)

License GPL (>= 2)

NeedsCompilation no

Author Edgar Merkle [aut, cre] (<https://orcid.org/0000-0001-7158-0653>), Yves Rosseel [aut], Mauricio Garnier-Villarreal [ctb], Terrence D. Jorgensen [ctb], Huub Hoofs [ctb], Rens van de Schoot [ctb]

Maintainer Edgar Merkle <merklee@missouri.edu>

Repository CRAN

Date/Publication 2018-10-31 06:10:03 UTC

R topics documented:

---

bcfa *Fit Confirmatory Factor Analysis Models*

Description

Fit a Confirmatory Factor Analysis (CFA) model.

Usage

```
bcfa(..., cp = "srs",
     dp = NULL, n.chains = 3, burnin, sample,
     adapt, mcmcfile = FALSE, mcmcextra = list(), inits = "simple",
     convergence = "manual", target = "jags", save.lvs = FALSE,
     jags.ic = FALSE, seed = NULL, bcontrol = list())
```

Arguments

`...`	Default lavaan arguments. See `lavaan`.
`cp`	Handling of prior distributions on covariance parameters: possible values are `"srs"` or `"fa"`. Option `"srs"` is more flexible and better from a theoretical standpoint, but it is also slower.
`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.
`n.chains`	Number of desired MCMC chains.
`burnin`	Number of burnin iterations, NOT including the adaptive iterations.
`sample`	The total number of samples to take after burnin.
`adapt`	The number of adaptive iterations to use at the start of the simulation.
`mcmcfile`	If `TRUE`, the JAGS/Stan model will be written to file (in the lavExport directory). Can also supply a character string, which serves as the name of the directory to which files will be written.
`mcmcextra`	A list with potential names `syntax` and `monitor`. The `syntax` object is a text string containing extra code to insert in the JAGS/Stan model syntax, and the `monitor` object is a character vector containing extra JAGS/Stan parameters to sample.
`inits`	If it is a character string, the options are currently `"simple"` (default), `"Mplus"`, `"prior"`, and `"jags"`. In the first two cases, parameter values are set as though they will be estimated via ML (see `lavaan`). The starting parameter value for each chain is then perturbed from the original values through the addition of uniform noise. If `"prior"` is used, the starting parameter values are obtained based on the prior distributions (while also trying to ensure that the starting values will not crash the model estimation). If `"jags"`, no starting values are specified and JAGS will choose values on its own. If `start` is a fitted object of class `lavaan`, the estimated values of the corresponding parameters will be

extracted, then perturbed in the manner described above. If it is a model list, for example the output of the paramaterEstimates() function, the values of the est or start or ustart column (whichever is found first) will be extracted.

convergence — If "auto", parameters will be sampled until convergence is achieved (via autorun.jags). In this case, the arguments burnin and sample are passed to autorun.jags as startburnin and startsample, respectively. Otherwise, parameters are sampled as specified by the user (or by the run.jags defaults).

target — Desired MCMC package ("jags" is default, but "stan" also available).

save.lvs — Should sample latent variables (factor scores) be saved? Logical; defaults to FALSE

jags.ic — Should DIC be computed the JAGS way, in addition to the BUGS way? Logical; defaults to FALSE

seed — A vector of length n.chains containing random seeds for the MCMC run. If NULL, seeds will be chosen randomly.

bcontrol — A list containing additional parameters passed to run.jags (or autorun.jags) or stan. See the manpage of those functions for an overview of the additional parameters that can be set.

Details

The bcfa function is a wrapper for the more general blavaan function, using the following default lavaan 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.th = TRUE, auto.delta = TRUE, and auto.cov.y = TRUE.

Value

An object of class lavaan, for which several methods are available, including a summary method.

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

Edgar C. Merkle & Yves Rosseel (2018). blavaan: Bayesian Structural Equation Models via Parameter Expansion. Journal of Statistical Software, 85(4), 1-30. URL http://www.jstatsoft.org/v85/i04/.

See Also

blavaan

Examples

```
## Not run:
# The Holzinger and Swineford (1939) example
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '
```

```
fit <- bcfa(HS.model, data=HolzingerSwineford1939,
            bcontrol=list(method="rjparallel"))
summary(fit)

## End(Not run)
```

---

bgrowth *Fit Growth Curve Models*

Description

Fit a Growth Curve model.

Usage

```
bgrowth(..., cp = "srs", dp = NULL, n.chains = 3,
burnin, sample, adapt, mcmcfile = FALSE, mcmcextra = list(),
inits = "simple", convergence = "manual", target = "jags",
save.lvs = FALSE, jags.ic = FALSE, seed = NULL, bcontrol = list())
```

Arguments

`...`	Default lavaan arguments. See lavaan.
`cp`	Handling of prior distributions on covariance parameters: possible values are `"srs"` or `"fa"`. Option `"srs"` is more flexible and better from a theoretical standpoint, but it is also slower.
`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.
`n.chains`	Number of desired MCMC chains.
`burnin`	Number of burnin iterations, NOT including the adaptive iterations.
`sample`	The total number of samples to take after burnin.
`adapt`	The number of adaptive iterations to use at the start of the simulation.
`mcmcfile`	If `TRUE`, the JAGS/Stan model will be written to file (in the lavExport directory). Can also supply a character string, which serves as the name of the directory to which files will be written.
`mcmcextra`	A list with potential names `syntax` and `monitor`. The `syntax` object is a text string containing extra code to insert in the JAGS/Stan model syntax, and the `monitor` object is a character vector containing extra JAGS/Stan parameters to sample.
`inits`	If it is a character string, the options are currently `"simple"` (default), `"Mplus"`, `"prior"`, and `"jags"`. In the first two cases, parameter values are set as though they will be estimated via ML (see lavaan). The starting parameter value for each chain is then perturbed from the original values through the addition of uniform noise. If `"prior"` is used, the starting parameter values are obtained based on the prior distributions (while also trying to ensure that the starting

values will not crash the model estimation). If "jags", no starting values are specified and JAGS will choose values on its own. If start is a fitted object of class lavaan, the estimated values of the corresponding parameters will be extracted, then perturbed in the manner described above. If it is a model list, for example the output of the paramaterEstimates() function, the values of the est or start or ustart column (whichever is found first) will be extracted.

convergence — If "auto", parameters will be sampled until convergence is achieved (via autorun.jags). In this case, the arguments burnin and sample are passed to autorun.jags as startburnin and startsample, respectively. Otherwise, parameters are sampled as specified by the user (or by the run.jags defaults).

target — Desired MCMC package ("jags" is default, but "stan" also available).

save.lvs — Should sample latent variables (factor scores) be saved? Logical; defaults to FALSE

jags.ic — Should DIC be computed the JAGS way, in addition to the BUGS way? Logical; defaults to FALSE

seed — A vector of length n.chains containing random seeds for the MCMC run. If NULL, seeds will be chosen randomly.

bcontrol — A list containing additional parameters passed to run.jags (or autorun.jags) or stan. See the manpage of those functions for an overview of the additional parameters that can be set.

Details

The bgrowth function is a wrapper for the more general blavaan function, using the following default lavaan 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.th = TRUE, auto.delta = TRUE, and auto.cov.y = TRUE.

Value

An object of class blavaan, for which several methods are available, including a summary method.

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

Edgar C. Merkle & Yves Rosseel (2018). blavaan: Bayesian Structural Equation Models via Parameter Expansion. Journal of Statistical Software, 85(4), 1-30. URL http://www.jstatsoft.org/v85/i04/.

See Also

blavaan

Examples

```
## Not run:
## linear growth model with a time-varying covariate
model.syntax <- '
```

```
  # intercept and slope with fixed coefficients
    i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
    s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4

  # regressions
    i ~ x1 + x2
    s ~ x1 + x2

  # time-varying covariates
    t1 ~ c1
    t2 ~ c2
    t3 ~ c3
    t4 ~ c4
'

fit <- bgrowth(model.syntax, data=Demo.growth)
summary(fit)

## End(Not run)
```

`blavaan` *Fit a Bayesian Latent Variable Model*

Description

Fit a Bayesian latent variable model.

Usage

```
blavaan(..., cp = "srs",
    dp = NULL, n.chains = 3, burnin, sample,
    adapt, mcmcfile = FALSE, mcmcextra = list(), inits = "simple",
    convergence = "manual", target = "jags", save.lvs = FALSE,
    jags.ic = FALSE, seed = NULL, bcontrol = list())
```

Arguments

`...`	Default lavaan arguments. See `lavaan`.
`cp`	Handling of prior distributions on covariance parameters: possible values are `"srs"` or `"fa"`. Option `"srs"` is more flexible and better from a theoretical standpoint, but it is also slower.
`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.
`n.chains`	Number of desired MCMC chains.
`burnin`	Number of burnin iterations, NOT including the adaptive iterations.
`sample`	The total number of samples to take after burnin.
`adapt`	The number of adaptive iterations to use at the start of the simulation.

`mcmcfile` If `TRUE`, the JAGS/Stan model and data will be written to files (in the lavExport directory). Can also supply a character string, which serves as the name of the directory to which files will be written.

`mcmcextra` A list with potential names `syntax` and `monitor`. The `syntax` object is a text string containing extra code to insert in the JAGS/Stan model syntax, and the `monitor` object is a character vector containing extra JAGS/Stan parameters to sample.

`inits` If it is a character string, the options are currently `"simple"` (default), `"Mplus"`, `"prior"`, or `"jags"`. In the first two cases, parameter values are set as though they will be estimated via ML (see `lavaan`). The starting parameter value for each chain is then perturbed from the original values through the addition of random uniform noise. If `"prior"` is used, the starting parameter values are obtained based on the prior distributions (while also trying to ensure that the starting values will not crash the model estimation). If `"jags"`, no starting values are specified and JAGS will choose values on its own. If `start` is a fitted object of class `lavaan`, the estimated values of the corresponding parameters will be extracted, then perturbed in the manner described above. If it is a model list, for example the output of the `paramaterEstimates()` function, the values of the `est` or `start` or `ustart` column (whichever is found first) will be extracted.

`convergence` If `"auto"`, parameters will be sampled until convergence is achieved (via `autorun.jags`). In this case, the arguments `burnin` and `sample` are passed to `autorun.jags` as `startburnin` and `startsample`, respectively. Otherwise, parameters are sampled as specified by the user (or by the `run.jags` defaults).

`target` Desired MCMC package (`"jags"` is default, but `"stan"` also available).

`save.lvs` Should sample latent variables (factor scores) be saved? Logical; defaults to FALSE

`jags.ic` Should DIC be computed the JAGS way, in addition to the BUGS way? Logical; defaults to FALSE

`seed` A vector of length `n.chains` containing random seeds for the MCMC run. If `NULL`, seeds will be chosen randomly.

`bcontrol` A list containing additional parameters passed to `run.jags` (or `autorun.jags`) or `stan`. See the manpage of those functions for an overview of the additional parameters that can be set.

Value

An object that inherits from class `lavaan`, for which several methods are available, including a `summary` method.

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

Edgar C. Merkle & Yves Rosseel (2018). blavaan: Bayesian Structural Equation Models via Parameter Expansion. Journal of Statistical Software, 85(4), 1-30. URL http://www.jstatsoft.org/v85/i04/.

See Also

bcfa, bsem, bgrowth

Examples

```
## Not run:
# The Holzinger and Swineford (1939) example
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '

fit <- blavaan(HS.model, data=HolzingerSwineford1939,
               auto.var=TRUE, auto.fix.first=TRUE,
               auto.cov.lv.x=TRUE,
               bcontrol=list(method="rjparallel"))
summary(fit)
coef(fit)

## End(Not run)
```

---

blavCompare *Bayesian model comparisons.*

Description

Bayesian model comparisons, including WAIC, LOO, and Bayes factor approximation.

Usage

```
blavCompare(object1, object2, ...)
```

Arguments

object1	An object of class blavaan.
object2	A second object of class blavaan.
...	Other arguments (unused for now).

Details

This function approximates the log-Bayes factor of two candidate models using the Laplace approximation to each model's marginal log-likelihood.

Value

The log-Bayes factor approximation, along with each model's approximate marginal log-likelihood.

References

Raftery, A. E. (1993). Bayesian model selection in structural equation models. In K. A. Bollen & J. S. Long (Eds.), Testing structural equation models (pp. 163-180). Beverly Hills, CA: Sage.

Examples

```
## Not run:
hsm1 <- ' visual  =~ x1 + x2 + x3 + x4
          textual =~ x4 + x5 + x6
          speed   =~ x7 + x8 + x9 '

fit1 <- bcfa(hsm1, data=HolzingerSwineford1939)

hsm2 <- ' visual  =~ x1 + x2 + x3
          textual =~ x4 + x5 + x6 + x7
          speed   =~ x7 + x8 + x9 '

fit2 <- bcfa(hsm2, data=HolzingerSwineford1939)

blavCompare(fit1, fit2)

## End(Not run)
```

`blavFitIndices` *SEM Fit Indices for Bayesian SEM*

Description

This function provides a posterior distribution of some χ^2-based fit indices to assess the global fit of a latent variable model.

Usage

```
blavFitIndices(object, pD = c("loo","waic","dic"),
               rescale = c("devM","ppmc","mcmc"),
               fit.measures = "all", baseline.model = NULL)

## S4 method for signature 'blavFitIndices'
## S4 method for signature 'blavFitIndices'
summary(object, central.tendency = c("mean","median","mode"),
        hpd = TRUE, prob = .90)
```

Arguments

`object`	An object of class `blavaan`.
`pD`	`character` indicating from which information criterion returned by `fitMeasures(object)` to use the estimated number of parameters. The default is from the leave-one-out information criterion (LOO-IC), which is most highly recommended by Vehtari et al. (2017).

`rescale` `character` indicating the method used to calculate fit indices. If `rescale = "devM"` (default), the Bayesian analog of the χ^2 statistic (the deviance evaluated at the posterior mean of the model parameters) is approximated by rescaling the deviance at each iteration by subtracting the estimated number of parameters. If `rescale = "PPMC"`, the deviance at each iteration is rescaled by subtracting the deviance of data simulated from the posterior predictive distribution (as in posterior predictive model checking; see Hoofs et al., 2017). If `rescale = "MCMC"` (not implemented yet), the fit measures are simply calculated at each iteration of the Markov chain(s), based on the model-implied moments at that iteration.

`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. If `rescale = "devM"` or `"PPMC"`, the currently available indices are `"BRMSEA"`, `"BGammaHat"`, `"adjBGammaHat"`, `"BMc"`, `"BCFI"`, `"BTLI"`, or `"BNFI"`. If `rescale = "MCMC"`, the user may request any indices returned by `fitMeasures` for objects of class `lavaan`.

`baseline.model` If not `NULL`, an object of class `blavaan`, representing a user-specified baseline model. If a `baseline.model` is provided, incremental fit indices (BCFI, BTLI, or BNFI) can be requested in `fit.measures`.

`central.tendency`
`character` indicating which statistics should be used to characterize the location of the posterior distribution. By default, all 3 statistics are returned. The posterior mean is labeled `EAP` for *expected a posteriori* estimate, and the mode is labeled `MAP` for *modal a posteriori* estimate.

`hpd` `logical` indicating whether to calculate the highest posterior density (HPD) credible interval for each fit index.

`prob` The "confidence" level of the credible interval(s).

Value

An S4 object of class `blavFitIndices` consisting of 2 slots:

`@details` A `list` containing the choices made by the user (or defaults; e.g., which values of `pD` and `rescale` were set), as well as the posterior distribution of the χ^2 (deviance) statistic (rescaled, if `rescale = "devM"` or `"PPMC"`).

`@indices` A `list` containing the posterior distribution of each requested `fit.measure`.

The `summary()` method returns a `data.frame` containing one row for each requested `fit.measure`, and columns containing the specified measure(s) of `central.tendency`, the posterior *SD*, and (if requested) the HPD credible-interval limits.

Author(s)

Mauricio Garnier-Villareal (Marquette University; <mauricio.garniervillarreal@marquette.edu>)

Terrence D. Jorgensen (University of Amsterdam; <TJorgensen314@gmail.com>)

References

`rescale = "PPMC"` based on:

Hoofs, H., van de Schoot, R., Jansen, N. W., & Kant, I. (2017). Evaluating model fit in Bayesian confirmatory factor analysis with large samples: Simulation study introducing the BRMSEA. *Educational and Psychological Measurement*. doi:10.1177/0013164417709314

`rescale = "devM"` based on:

Garnier-Villarreal, M., & Jorgensen, T. D. (2018). *Adapting fit indices for Bayesian SEM: Comparison to maximum likelihood*. Unpublished manuscript (see https://osf.io/afkcw/).

Other references:

Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing, 27*(5), 1413–1432. doi:10.1007/s11222-016-9696-4

Examples

```
 ## Not run:
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '
## fit target model
fit1 <- bcfa(HS.model, data = HolzingerSwineford1939, cp = "fa",
             n.chains = 2, burnin = 1000, sample = 1000)

## fit null model to calculate CFI, TLI, and NFI
null.model <- c(paste0("x", 1:9, " ~~ x", 1:9), paste0("x", 1:9, " ~ 1"))
fit0 <- bcfa(null.model, data = HolzingerSwineford1939, cp = "fa",
             n.chains = 2, burnin = 1000, sample = 1000)

## calculate posterior distributions of fit indices

## The default method mimics fit indices derived from ML estimation
ML <- blavFitIndices(fit1, baseline.model = fit0)
ML
summary(ML)

## other options:

## - use Hoofs et al.'s (2017) PPMC-based method
## - use the estimated number of parameters from WAIC instead of LOO-IC
PPMC <- blavFitIndices(fit1, baseline.model = fit0,
                       pD = "waic", rescale = "PPMC")
## issues a warning about using rescale="PPMC" with N < 1000 (see Hoofs et al.)

## - specify only the desired measures of central tendency
## - specify a different "confidence" level for the credible intervals
summary(PPMC, central.tendency = c("mean","mode"), prob = .95)
```

```
## Access the posterior distributions for further investigation
head(distML <- data.frame(ML@indices))

## For example, diagnostic plots using the bayesplot package:

## distinguish chains
nChains <- blavInspect(fit1, "n.chains")
distML$Chain <- rep(1:nChains, each = nrow(distML) / nChains)

library(bayesplot)
mcmc_pairs(distML, pars = c("BRMSEA","BMc","BGammaHat","BCFI","BTLI"),
           diag_fun = "hist")
## Indices are highly correlated across iterations in both chains

## Compare to PPMC method
distPPMC <- data.frame(PPMC@indices)
distPPMC$Chain <- rep(1:nChains, each = nrow(distPPMC) / nChains)
mcmc_pairs(distPPMC, pars = c("BRMSEA","BMc","BGammaHat","BCFI","BTLI"),
           diag_fun = "dens")
## nonlinear relation between BRMSEA, related to the floor effect of BRMSEA
## that Hoofs et al. found for larger (12-indicator) models


## End(Not run)
```

`blavInspect` *Inspect or Extract Information from a fitted blavaan object*

Description

The `blavInspect()` and `blavTech()` functions can be used to inspect/extract information that is stored inside (or can be computed from) a fitted blavaan object. This is similar to lavaan's `lavInspect()` function.

Usage

```
blavInspect(blavobject, what, ...)

blavTech(blavobject, what, ...)
```

Arguments

`blavobject`	An object of class blavaan.
`what`	Character. What needs to be inspected/extracted? See Details for Bayes-specific options, and see lavaan's `lavInspect()` for additional options. Note: the `what` argument is not case-sensitive (everything is converted to lower case.)
`...`	Default lavaan arguments supplied to `lavInspect()`; see lavaan.

Details

Below is a list of Bayesian-specific values for the `what` argument; additional values can be found in the `lavInspect()` documentation.

`"start"`: A list of starting values for each chain, unless `inits="jags"` is used during model estimation. Aliases: `"starting.values"`, `"inits"`.

`"psrf"`: Each parameter's Gelman-Rubin PSRF (potential scale reduction factor) for convergence assessment.

`"ac.10"`: Each parameter's estimated lag-10 autocorrelation.

`"neff"`: Each parameters effective sample size, taking into account autocorrelation.

`"mcmc"`: An object of class `mcmc` containing the individual parameter draws from the MCMC run. Aliases: `"draws"`, `"samples"`.

`"mcobj"`: The underlying run.jags or stan object that resulted from the MCMC run.

`"n.chains"`: The number of chains sampled.

`"cp"`: The approach used for estimating covariance parameters (`"srs"` or `"fa"`).

`"dp"`: Default prior distributions used for each type of model parameter.

`"postmode"`: Estimated posterior mode of each free parameter.

`"postmean"`: Estimated posterior mean of each free parameter.

`"postmedian"`: Estimated posterior median of each free parameter.

`"lvs"`: An object of class `mcmc` containing latent variable (factor score) draws.

`"lvmeans"`: A matrix of mean factor scores (rows are observations, columns are variables).

`"hpd"`: HPD interval of each free parameter. In this case, an additional argument `level` can be supplied to specify a number in (0,1) reflecting the percentage of the interval.

See Also

`lavInspect`, `bcfa`, `bsem`, `bgrowth`

Examples

```
## Not run:
# The Holzinger and Swineford (1939) example
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '

fit <- bcfa(HS.model, data=HolzingerSwineford1939,
            jagcontrol=list(method="rjparallel"))

# extract information
blavInspect(fit, "psrf")
blavInspect(fit, "hpd", level=.9)

## End(Not run)
```

blav_internal *blavaan internal functions*

Description

Internal functions related to Bayesian model estimation. Not to be called by the user.

bsem *Fit Structural Equation Models*

Description

Fit a Structural Equation Model (SEM).

Usage

```
bsem(..., cp = "srs",
     dp = NULL, n.chains = 3, burnin, sample,
     adapt, mcmcfile = FALSE, mcmcextra = list(), inits = "simple",
     convergence = "manual", target = "jags", save.lvs = FALSE,
     jags.ic = FALSE, seed = NULL, bcontrol = list())
```

Arguments

`...`	Default lavaan arguments. See `lavaan`.
`cp`	Handling of prior distributions on covariance parameters: possible values are `"srs"` or `"fa"`. Option `"srs"` is more flexible and better from a theoretical standpoint, but it is also slower.
`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.
`n.chains`	Number of desired MCMC chains.
`burnin`	Number of burnin iterations, NOT including the adaptive iterations.
`sample`	The total number of samples to take after burnin.
`adapt`	The number of adaptive iterations to use at the start of the simulation.
`mcmcfile`	If `TRUE`, the JAGS/Stan model will be written to file (in the lavExport directory). Can also supply a character string, which serves as the name of the directory to which files will be written.
`mcmcextra`	A list with potential names `syntax` and `monitor`. The `syntax` object is a text string containing extra code to insert in the JAGS/Stan model syntax, and the `monitor` object is a character vector containing extra JAGS/Stan parameters to sample.

inits	If it is a character string, the options are currently "simple" (default), "Mplus", "prior", and "jags". In the first two cases, parameter values are set as though they will be estimated via ML (see lavaan). The starting parameter value for each chain is then perturbed from the original values through the addition of uniform noise. If "prior" is used, the starting parameter values are obtained based on the prior distributions (while also trying to ensure that the starting values will not crash the model estimation). If "jags", no starting values are specified and JAGS will choose values on its own. If start is a fitted object of class lavaan, the estimated values of the corresponding parameters will be extracted, then perturbed in the manner described above. If it is a model list, for example the output of the paramaterEstimates() function, the values of the est or start or ustart column (whichever is found first) will be extracted.
convergence	If "auto", parameters will be sampled until convergence is achieved (via autorun.jags). In this case, the arguments burnin and sample are passed to autorun.jags as startburnin and startsample, respectively. Otherwise, parameters are sampled as specified by the user (or by the run.jags defaults).
target	Desired MCMC package ("jags" is default, but "stan" also available).
save.lvs	Should sample latent variables (factor scores) be saved? Logical; defaults to FALSE
jags.ic	Should DIC be computed the JAGS way, in addition to the BUGS way? Logical; defaults to FALSE
seed	A vector of length n.chains containing random seeds for the MCMC run. If NULL, seeds will be chosen randomly.
bcontrol	A list containing additional parameters passed to run.jags (or autorun.jags) or stan. See the manpage of those functions for an overview of the additional parameters that can be set.

Details

The bsem function is a wrapper for the more general blavaan function, using the following default lavaan 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.th = TRUE, auto.delta = TRUE, and auto.cov.y = TRUE.

Value

An object of class lavaan, for which several methods are available, including a summary method.

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

Edgar C. Merkle & Yves Rosseel (2018). blavaan: Bayesian Structural Equation Models via Parameter Expansion. Journal of Statistical Software, 85(4), 1-30. URL http://www.jstatsoft.org/v85/i04/.

See Also

blavaan

Examples

```
## Not run:
## The industrialization and Political Democracy Example
## 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

  # regressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60

  # residual correlations
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
'

## unique priors for mv intercepts; parallel chains
fit <- bsem(model, data=PoliticalDemocracy,
            dp=dpriors(nu="dnorm(5,1e-2)"),
            bcontrol=list(method="rjparallel"))
summary(fit)

## End(Not run)
```

`dpriors` *Specify default prior distributions*

Description

Specify "default" prior distributions for classes of model parameters.

Usage

```
dpriors(..., target = "jags")
```

Arguments

`...`	Parameter names paired with desired priors (see example below).
`target`	Are the priors for jags (default) or stan?

Details

The prior distributions always use JAGS/Stan syntax and parameterizations. For example, the normal distribution in JAGS is parameterized via the precision, whereas the normal distribution in Stan is parameterized via the standard deviation.

User-specified prior distributions for specific parameters (using the `prior()` operator within the model syntax) always override prior distributions set using `dpriors()`.

The parameter names are:

- nu: Observed variable intercept parameters.
- alpha: Latent variable intercept parameters.
- lambda: Loading parameters.
- beta: Regression parameters.
- itheta: Observed variable precision parameters.
- ipsi: Latent variable precision parameters.
- rho: Correlation parameters (associated with covariance parameters).
- ibpsi: Inverse covariance matrix of blocks of latent variables (used for `target="jags"`).
- tau: Threshold parameters (ordinal data only).
- delta: Delta parameters (ordinal data only).

Value

A character vector containing the prior distribution for each type of parameter.

References

Edgar C. Merkle & Yves Rosseel (2018). blavaan: Bayesian Structural Equation Models via Parameter Expansion. Journal of Statistical Software, 85(4), 1-30. URL http://www.jstatsoft.org/v85/i04/.

See Also

`bcfa`, `bsem`, `bgrowth`

Examples

```
dpriors(nu = "dunif(0,10)", lambda = "dnorm(0,1e-2) T(0,)", itheta = "dexp(1)")
```

Index