Package: vbpm (via r-universe)

October 5, 2026

Type Package

Title Variational Bayes Psychometric Models

Version 0.9.1

Description Variational Bayes estimation for a family of psychometric measurement models. Two models are provided. Variational Bayes factor analysis (vbfa) is a regularized partially confirmatory factor model spanning the confirmatory-exploratory continuum via spike-and-slab priors on the loadings (Chen, Guo, Zhang, and Pan, 2021 <doi:10.1037/met0000293>; Chen, 2023 <doi:10.3758/s13428-022-01884-7>; Jin and Chen, 2025 <doi:10.1080/10705511.2024.2432612>), with an optional dynamic (warm-started) regularization path, an orthogonal bifactor parameterization, and optional sparse residual (local dependence) estimation through a graphical spike-and-slab prior solved by QUIC (Jin, Chen, Yan, and Zhang, 2026 <doi:10.31234/osf.io/dehtv_v2>). Regularized MIMIC (vbmimic) extends this to multiple-indicators multiple-causes models, placing spike-and-slab priors on both the measurement and the structural part (Jin and Chen, 2025 <doi:10.1080/00273171.2025.2483253>). Companion tools compute SEM-like fit statistics, and sweep a factor-count window to report candidate fit, criterion, and between-candidate loading-correspondence measurements without selecting a count (Chen and Jin, 2026 <doi:10.48550/arXiv.2607.07159>). Data generators for either model family are also provided.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports Rcpp, MASS, stats

LinkingTo Rcpp, RcppArmadillo

Suggests numDeriv, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/Jinsong-Chen/vbpm

BugReports https://github.com/Jinsong-Chen/vbpm/issues

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Jinsong Chen [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0157-5469>), Yi Jin [aut] (ORCID:
<https://orcid.org/0000-0002-8604-5992>)

Maintainer Jinsong Chen <jinsong.chen@live.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-09-05 16:08:36 UTC

RemoteUrl https://github.com/cran/vbpm

RemoteRef HEAD

RemoteSha da9892a53bf026d69b47f6109bb5abad52dcd9da

Contents

coef.vbpm_fit . . . 2
fit_stats . . . 3
nlsy27 . . . 5
pefa . . . 6
print.pefa . . . 9
print.vbpm_fit . . . 11
sim_fa . . . 11
sim_lvm . . . 14
special_effects . . . 17
ssl . . . 19
vbfa . . . 20
vbmimic . . . 24
vbpm_fit . . . 27

Index **29**

`coef.vbpm_fit` *Extract coefficients from a fitted vbpm model*

Description

For a `vbfa()` fit, returns the loading matrix. For a `vbmimic()` fit, returns a list with the measurement loadings `A` and the structural coefficients `B`.

Usage

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

Arguments

object	A fit returned by vbfa() or vbmimic().
...	Ignored.

Value

A matrix (vbfa) or a list of two matrices (vbmimic).

Examples

```
sim <- sim_fa(N = 300, K = 3, ipf = 6, lam = .7, lac = .3, rseed = 1)
Q <- matrix(-1L, ncol(sim$dat), 3)
for (k in 1:3) { a <- which(rep(1:3, each = 6) == k)[1:2]
                 Q[a, ] <- 0L; Q[a, k] <- 1L }
fit <- vbfa(sim$dat, Q)
round(coef(fit), 2)
```

---

fit_stats *Fit statistics for vbpm models*

Description

Computes structural-equation-model fit statistics for a fitted vbfa() object under hard selection (an unspecified loading counts as active when its posterior inclusion probability is at least tau). Absolute indices (RMSEA, SRMR, CFI, TLI) and information criteria (AIC, BIC) are returned, together with the model's ELBO.

Usage

```
fit_stats(object, ...)

## S3 method for class 'vbfa'
fit_stats(
  object,
  Y = NULL,
  Q = object$Q,
  tau = 0.5,
  gamma = 0.5,
  orthogonal = NULL,
  rank_adjust = FALSE,
  rank_max_J = 100,
  ...
```

```
)

## S3 method for class 'vbmimic'
fit_stats(object, Y = NULL, Q = NULL, tau = 0.5, ...)

## S3 method for class 'vbpm_fit'
fit_stats(object, ...)
```

Arguments

object	A fitted vbpm_fit object.
...	Further arguments passed to methods. Unused by the methods supplied with the package.
Y	The N x J data matrix the model was fit to.
Q	The J x K loading design matrix used in the fit.
tau	Hard-selection threshold on the posterior inclusion probability (default 0.5).
gamma	Reserved (extended BIC tuning); currently unused in the returned vector.
orthogonal	Logical; whether the fit is an orthogonal (bifactor) model, in which case factor correlations are not free parameters. The default NULL reads the setting from the fit object itself, which is the recommended use. Supplying a value that contradicts the fit raises an error rather than silently miscounting parameters.
rank_adjust	Logical; if TRUE, compute the effective parameter count as the rank of the Jacobian of the model-implied covariance. This explicit opt-in requires the suggested numDeriv package. The default FALSE uses the nominal parameter count.
rank_max_J	Largest number of observed variables for which an explicit Jacobian-rank calculation is allowed (default 100). If rank_adjust = TRUE and J exceeds this value, the method errors rather than silently substituting the nominal count.

Value

An object of class vbpm_fit_stats (a named numeric vector) with t_nom, t, t_S (parameter counts), RMSEA, SRMR, CFI, TLI, AIC, BIC (hard selection), AIC_S, BIC_S (soft selection), ELBO, and objective. The last is the model's comparability score: equal to ELBO for a diagonal fit, and the terminal VECM objective under local dependence, where ELBO is NA. It is comparable only across fits sharing the objective_type attribute, which is also attached.

The vbmimic method returns the same shape with n_active_coef in place of a full soft count and typed NA elsewhere; see vbmimic().

References

Chen, J., & Jin, Y. (2026). Recovering latent structures after variational Bayesian variable selection: Fit assessment and factor-number selection in partially exploratory factor analysis. *arXiv preprint* arXiv:2607.07159.

Jin, Y., & Chen, J. (2025). Regularized variational approximation for partially confirmatory factor analysis. *Structural Equation Modeling: A Multidisciplinary Journal*, 32(3), 437-449. doi:10.1080/10705511.2024.2432612

See Also

vbfa(), pefa()

Examples

```
sim <- sim_fa(N = 300, K = 3, ipf = 6, lam = .7, lac = .3, rseed = 1)
Q <- matrix(-1L, ncol(sim$dat), 3)
for (k in 1:3) { a <- which(rep(1:3, each = 6) == k)[1:2]
                 Q[a, ] <- 0L; Q[a, k] <- 1L }
fit <- vbfa(sim$dat, Q)
round(fit_stats(fit), 3)


## bifactor: orthogonal is read from the fit object, so the same call works
Qb <- cbind(1L, Q)
fb <- vbfa(sim$dat, Qb, orthogonal = TRUE)
round(fit_stats(fb), 3)
```

---

nlsy27 *National Longitudinal Survey of Youth 1997 (27 items)*

Description

Responses of 3,458 individuals to 27 mixed-type items, with 1.12% missing data. Carried over from the LAWBL package (which vbpm supersedes) for empirical illustration of the estimators.

Usage

```
nlsy27
```

Format

A list with three components:

dat A 3458 x 27 data frame of item responses (17 polytomous items, 10 continuous), containing NAs.

Q A 27 x 3 initial design matrix with two to three specified loadings per factor, in the -1/0/1 coding used by vbfa().

cati Indices of the 17 polytomous items.

Source

Bureau of Labor Statistics, U.S. Department of Labor. National Longitudinal Survey of Youth 1997 cohort. Previously distributed in the LAWBL package (Chen, 2022).

Examples

```
data(nlsy27)
dim(nlsy27$dat)
nlsy27$Q
```

`pefa` *Partially exploratory factor analysis over a factor-count window*

Description

Fits `vbfa()` for every integer `K` in a fixed consecutive window, holding the partially specified backbone `Q0` fixed and appending regularized exploratory columns. It reports candidate summaries and between-candidate loading measurements.

Usage

```
pefa(
  Q0,
  Y,
  Kmin,
  Kmax,
  bifactor = FALSE,
  general = 1,
  v0 = c(0.01, 0.005, 0.002, 0.001),
  max_it = 10000,
  convChk = FALSE,
  tolVal = 1e-04,
  tau = 0.5,
  rank_adjust = FALSE,
  rank_max_J = 100,
  verbose = TRUE
)
```

Arguments

`Q0`	A `J` by `K0` backbone with entries `1`, `0`, or `-1`. Zero columns (`K0 = 0`, fully exploratory) are allowed.
`Y`	A numeric `N` by `J` matrix of observed values, optionally with literal `NA`; `NaN` and infinite values are rejected. Every row and every item needs an observed value, and every item a positive finite SD.
`Kmin, Kmax`	Inclusive window, whole numbers with `1 <= Kmin <= Kmax <= .Machine$integer.max` and `Kmin >= ncol(Q0)`.
`bifactor`	Fit an orthogonal bifactor parameterization.
`general`	Scalar or length-`J` general design in `{-1, 0, 1}` with at least one `1`, used only when `bifactor = TRUE`.

`v0`, `max_it`, `convChk`, `tolVal`
: Optimizer controls passed to `vbfa()`: finite positive `v0`, positive whole `max_it`, `TRUE`/`FALSE` `convChk`, and finite `tolVal` `>= 0`.

`tau`
: Loading-inclusion threshold in `[0, 1]`, used by `fit_stats()`.

`rank_adjust`, `rank_max_J`
: Rank-adjustment controls for `fit_stats()`. `rank_adjust = TRUE` requires `J <= rank_max_J` and the suggested `numDeriv` package.

`verbose`
: Print one progress line per candidate.

Details

`pefa()` **chooses nothing**: no factor count, threshold, persistence horizon, collision policy, or stopping rule. Those belong to the analysis that consumes the object.

Ordinary candidates are oblique with diagonal residuals. In bifactor mode `K` counts group factors and `vbfa()` adds one general column, so a stored matrix has `K + 1` columns; every between-candidate comparison removes that column first and works on the `K` group columns.

One private comparator produces every between-candidate number, and both `transitions` and `persistence` are read from its stored results. The accessors do no matching of their own: `ssl()`, `print()`, `summary()`, and `plot()` only describe what is already stored, and none of them rematches columns or applies an analysis rule. There is no `persistence()` accessor: `x$persistence$phi`, `$rmsd`, and `$collision` are already the three stored matrices.

Value

An object of class `"pefa"`; see *The returned object* below.

The returned object

A list of class `"pefa"` with exactly these components, in order.

`sweep` One row per candidate, in column order. `K` is integer; `ELBO`, `AIC`, and `BIC` are required finite doubles; `RMSEA`, `SRMR`, `CFI`, and `TLI` are descriptive doubles; `t` is a double; `iter` is a nonnegative integer, or `NA_integer_` when a well-formed engine result reports none; `converged` is a nonmissing logical. `AIC` and `BIC` are hard-selection values at the resolved `tau`, and `t` is their effective parameter count – a finite, nonnegative, whole-valued count stored as a double – rank-adjusted when `rank_adjust = TRUE`. Each descriptive index is `NA_real_` exactly when its own definition does not apply, and is never silently substituted.

`transitions` One row per adjacent pair: the integers `K_from` and `K_to`, the doubles `ELBO_gain_pct` and `BIC_gain_pct`, then the seven structural facts – the doubles `phi_min`, `rmsd`, `rmsd_max`, `ari`, `pip_rmsd`, `unmatched_ssl` and the logical `collision`. Positive gains favor `K_to`. Each percentage divides by the largest positive gain on its own path, or is `NA_real_` when no positive gain exists. Raw criteria stay in `sweep`.

`persistence` Three upper-triangular `W` by `W` matrices over the fitted `K` values, named and ordered `phi`, `rmsd`, `collision`: `phi` holds `phi_min`, `rmsd` holds `rmsd_max`, and `collision` is the logical reuse mask. Diagonal and lower triangle are `NA`. Every cell is a direct endpoint comparison, never a chain through intervening candidates.

`loadings`, `pips` Named `K`-indexed lists of candidate matrices with items in rows and factors in columns: `J` by `K`, or `J` by `K + 1` with the general column first in bifactor mode. Loadings are finite and PIPs lie in `[0, 1]`.

`Q0`, `settings` The canonical `J` by `K0` integer backbone, and the resolved controls `bifactor`, `general`, `v0`, `max_it`, `convChk`, `tolVal`, `tau`, `rank_adjust`, `rank_max_J`, in that order. `general` is `NULL` outside bifactor mode. `settings` records fitting and fit-statistic controls only: no `verbose`, and no analysis rule.

The loading and PIP matrices and the criteria `ELBO`, `AIC`, `BIC`, and `t` are required. A fitting failure or a malformed required quantity aborts the call with an error naming the `K`; there is no status taxonomy and no partially populated object. A candidate that merely failed to converge is retained with `converged = FALSE`, and one aggregate warning names every affected `K`.

Loading correspondence

For source column `a` and target column `b`, let `s = 1` when `a'b >= 0` and `-1` otherwise, and let `d(a, b) = ||a - s b||^2`. Backbone columns pair by position. Each exploratory source independently selects the exploratory target minimizing `d`, which is Equation 17 of Chen (2023) made invariant to whole-column reflection. Ties break by larger finite absolute Tucker congruence, then by smaller target index.

The assignment is independent, not one-to-one, so two sources may select the same target. That reuse is a **collision**: it is recorded rather than repaired, the pair keeps every otherwise defined measurement, and the target left unselected still counts toward `unmatched_ssl`. There is no salience screen: every column participates, and a comparison always contains exactly `K_from` scored pairs.

A zero-norm column is no exception. Its aligned distance is defined, so it takes part in the assignment and in `rmsd` and `rmsd_max`, but its Tucker congruence is `0/0` and therefore undefined. An undefined congruence is reported as `NA_real_` and makes that comparison's `phi_min` `NA_real_`; it is never dropped by an `na.rm`.

Because `phi_min` is a minimum over *every* column, an overextracted candidate carrying a near-empty column will report a low `phi_min` and often `collision = TRUE`. Read that with `ssl()`: it usually means the candidate contains a column with nothing in it, not that the retained structure moved.

Identification

`pefa()` imposes no upper bound on `K` relative to `J`. The Ledermann bound `K <= (2J + 1 - sqrt(8J + 1)) / 2` orients the choice of `Kmax`, but it is derived for an unrestricted factor model and does not constrain a regularized partially confirmatory candidate, so applying it as a package check would be wrong.

Beyond that bound expect candidates that do not converge and fit indices that become `NA`: `RMSEA` and `TLI` are undefined at non-positive degrees of freedom. Both are reported evidence about the window rather than failures. `rank_adjust = TRUE` is the sharper diagnostic, since it counts the rank of the Jacobian of the unique covariance elements and therefore tests local identification directly; that count cannot exceed `J(J + 1)/2`, so its degrees of freedom saturate at zero instead of going negative.

Removed in 0.9.0

`select_K_elbow()`, the `pefa()` arguments `cuts`, `sustain`, and `stability_eps`, and the `$selected_K` and `$boundary` components existed in 0.8.3 and are gone. That is an intentional pre-1.0 breaking

change rather than a deprecation cycle: the package owns no count rule, so it can own neither a selector nor a stability cutoff. The replacement is ordinary analysis code over the stored path, as in the last chunk of the Examples, and such code has to say which variant it is, because look-ahead and boundary-fallback variants disagree on the same path. `NEWS.md` lists the removals in full.

References

Chen, J., & Jin, Y. (2026). Recovering latent structures after variational Bayesian variable selection: Fit assessment and factor-number selection in partially exploratory factor analysis. *arXiv preprint* arXiv:2607.07159.

Chen, J. (2023). Fully and partially exploratory factor analysis with bi-level Bayesian regularization. *Behavior Research Methods*, 55(4), 2125–2142. doi:10.3758/s13428022018847

See Also

`vbfa()`, `fit_stats()`, `ssl()`

Examples

```
sim <- sim_fa(N = 300, K = 3, ipf = 4, lam = .7, lac = .3, rseed = 1)
Q0 <- matrix(-1L, ncol(sim$dat), 2)
groups <- rep(1:3, each = 4)
for (k in 1:2) Q0[which(groups == k)[1:2], k] <- 1L

fit <- pefa(Q0, sim$dat, 2, 4, verbose = FALSE)
fit$sweep
fit$transitions[, c("K_from", "K_to", "ELBO_gain_pct", "phi_min")]
fit$persistence$phi
ssl(fit)

## A count rule is the analysis's, never the package's. This one stops at
## the first ELBO gain under 20% and looks no further ahead; a
## full-look-ahead variant can answer differently on the same path.
g <- fit$transitions$ELBO_gain_pct
fit$transitions$K_from[which(g < 20)[1]]
```

`print.pefa` *Methods for a PEFA factor-count sweep*

Description

Descriptive readers for the object returned by `pefa()`. They report what was fitted and measured and nothing else: no method selects, recommends, or marks a factor count, and none draws a threshold, because the package owns no count rule. Count rules, persistence thresholds, and collision policies belong to the analysis that consumes this object.

Usage

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

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

## S3 method for class 'summary.pefa'
print(x, digits = 3, ...)

## S3 method for class 'pefa'
plot(
  x,
  type = c("objective", "gain", "fit"),
  criterion = c("ELBO", "AIC", "BIC"),
  pct = FALSE,
  ...
)
```

Arguments

`...`	Further arguments; for `plot()`, graphical arguments passed to the first base plot in the selected display.
`object`, `x`	A result returned by `pefa()`; for `print.summary.pefa()`, the object returned by `summary()`.
`digits`	Number of decimals used when printing measured quantities.
`type`	Display: criterion trajectories (`"objective"`), one gain trajectory (`"gain"`), or descriptive absolute and incremental fit trajectories (`"fit"`).
`criterion`	Criterion path. `"ELBO"`, `"AIC"`, or `"BIC"` for `type = "objective"`; `"ELBO"` or `"BIC"` for `type = "gain"`, which has no AIC path. Ignored when `type = "fit"`.
`pct`	Gain view only. `FALSE` (the default) computes the raw oriented gains `ELBO_to - ELBO_from` or `BIC_from - BIC_to` from `$sweep`; `TRUE` reads the stored `ELBO_gain_pct` or `BIC_gain_pct` column of `$transitions`, which is `NA` for a path with no positive gain.

Value

`print()` and `plot()` return their input invisibly. `summary()` returns a list of class `"summary.pefa"` with exactly the components `window` (named integer `Kmin`, `Kmax`, `K0`), `sweep`, `transitions`, `persistence`, `ssl`, `settings`, and `nonconverged_K` (an integer vector of the K values whose candidate did not converge, `integer(0)` when all converged).

See Also

`pefa()`, `ssl()`

print.vbpm_fit *Compact display of a fitted vbpm model*

Description

Prints a short summary – model type, dimensions, convergence, timing, and selection results – instead of dumping the full list. Access the parts as usual (`fit$Lam` etc.; see vbpm_fit); `str(fit)` shows everything.

Usage

```
## S3 method for class 'vbpm_fit'
print(x, tau = 0.5, ...)
```

Arguments

`x`	A fit returned by `vbfa()` or `vbmimic()`.
`tau`	Threshold on the posterior inclusion probability used only for the "active loadings" line (default `0.5`, matching `fit_stats()`).
`...`	Ignored (present for compatibility with the `print()` generic).

Value

`x`, invisibly (the R convention for print methods, so `print(fit)` can be used in pipes without losing the object).

Examples

```
sim <- sim_fa(N = 300, K = 3, ipf = 6, lam = .7, lac = .3, rseed = 1)
Q <- matrix(-1L, ncol(sim$dat), 3)
for (k in 1:3) { a <- which(rep(1:3, each = 6) == k)[1:2]
                 Q[a, ] <- 0L; Q[a, k] <- 1L }
fit <- vbfa(sim$dat, Q)
fit               # dispatches here: compact summary, not a list dump
fit$Lam[1:3, ]    # the object is still an ordinary list underneath
```

sim_fa *Simulating data with Latent Variable Modeling*

Description

sim_fa can simulate data based on factor analysis or item response models with different response formats (continuous or categorical), loading patterns and residual covariance (local dependence) structures.

sim_fa() and sim_lvm() are complementary rather than interchangeable: sim_fa() is the *structure-side* generator, driven by the loading pattern itself – items per factor (ipf), alternating-sign cross-loadings (alt_sign), or minor factors (minor, definition-based per Auerswald & Moshagen, 2019) – and is the generator for vbfa(). sim_lvm() is the *model-side* generator, adding observed/latent predictors (P/b, K1/ph1/b1) and mixed response formats (ilvl), and is the generator for vbmimic(). Both generators also accept a population loading matrix directly (mla) with identical semantics.

Usage

```
sim_fa(
  N = 1000,
  mla = NULL,
  K = 3,
  ipf = 8,
  cpf = 2,
  lam = 0.7,
  lac = 0.3,
  alt_sign = TRUE,
  phi = 0.5,
  ph12 = -1,
  gamma = NULL,
  ecr = 0,
  ome_out = FALSE,
  cati = NULL,
  noc = c(4),
  misp = 0,
  rseed = 333,
  necw = K,
  necb = K,
  add_ind = c(),
  add_la = 0,
  add_phi = 0,
  add1_ind = c(),
  add1_la = 0,
  add1_phi = 0,
  zero_it = 0,
  minor = NULL,
  digits = 4
)
```

Arguments

N	Sample size.

mla	Population loading matrix (J x K). If supplied, the pattern arguments (ipf, cpf, lam, lac, alt_sign, add_*, minor, zero_it) are ignored and K, J are taken from it – the same convention as sim_lvm().
K	Number of factors.
ipf	Items per factor.
cpf	Cross-loadings per factor.
lam	Number of formal iterations for posterior sampling.
lac	Number of iterations to update the sampling information.
alt_sign	Logical; alternate the sign of the cross-loadings across the cross-loading items (default TRUE).
phi	Homogeneous correlations between any two factors.
ph12	Correlation between factor 1 and 2 (if it's different from phi). Shared with sim_lvm(), which uses ph1 for a different purpose (correlation among latent predictors); see sim_lvm() for that distinction.
gamma	Optional second-order loadings for direct **higher-order** generation: a scalar or length-K vector in $(0, 1)$. When supplied, each group factor η_k is driven by a general factor ξ through $\eta_k = \gamma_k \xi + \zeta_k$, and the data are generated from the equivalent Schmid-Leiman orthogonal bifactor matrix: general loadings $\lambda_j \gamma_k$ and group loadings $\lambda_j \sqrt{1 - \gamma_k^2}$ (a testlet model is the same model). The returned MLA is this $J \times (K + 1)$ bifactor matrix (general factor first), PHI is the identity, and the first-order matrix and gamma are returned as MLA1 and gamma. Incompatible with mla, cpf > 0, add_ind, add1_ind, minor, and zero_it; phi/ph12 are ignored (the SL solution is orthogonal by construction).
ecr	Residual correlation (local dependence).
ome_out	Output factor score or not.
cati	The set of categorical (polytomous) items in sequence number (i.e., 1 to J); NULL for no and -1 for all (default is NULL).
noc	Number of categories for categorical items
misp	Proportion of missingness.
rseed	An integer for the random seed.
necw	Number of within-factor local dependence.
necb	Number of between-factor local dependence.
add_ind	(Additional) minor factor with cross-loadings.
add_la	Value of cross-loadings on (Additional) minor factor.
add_phi	Correlations between (Additional) minor factor and other factors.
add1_ind	A second (additional) minor factor with cross-loadings, given as item indices; c() for none.
add1_la	Value of the cross-loadings on the second minor factor.
add1_phi	Correlations between the second minor factor and other factors.
zero_it	Surplus items with zero loading.
minor	Optional character vector requesting orthogonal minor factors loading on every indicator, each "weak" or "moderate" (Auerswald & Moshagen, 2019); NULL for none.
digits	Number of significant digits to print when printing numeric values.

Value

An object of class `list` containing the data, loading, and factorial correlation matrix.

Examples

```
# for continuous data with cross-loadings and local dependence effect .3
out <- sim_fa(N=1000,K=3,ipf=6,lam = .7, lac=.3,ecr=.3)
summary(out$dat)
out$MLA
out$ofd_ind

# for categorical data with cross-loadings .4 and 10% missingness
out <- sim_fa(N=1000,K=3,ipf=6,lam = .7, lac=.4,cati=-1,noc=4,misp=.1)
summary(out$dat)
out$MLA
out$ofd_ind

## matrix-driven: supply the population loading matrix directly
mla <- matrix(0, 12, 2)
mla[1:6, 1] <- .7; mla[7:12, 2] <- c(.8, .7, .6, .7, .8, .6)
s <- sim_fa(N = 300, mla = mla, phi = .3, rseed = 4)
s$MLA                      # the generating matrix, as supplied

## higher-order (testlet) data, generated directly: 3 group factors driven
## by one general factor with second-order loadings .8, .7, .6
s <- sim_fa(N = 500, K = 3, ipf = 6, lam = .7, gamma = c(.8, .7, .6))
round(s$MLA, 2)            # J x 4 Schmid-Leiman bifactor matrix (general first)
s$gamma                    # the second-order loadings
```

---

sim_lvm *Simulate data from a latent variable model*

Description

Generates factor-analytic or item-response data with optional observed and latent predictors (MIMIC / structural models), cross-loadings, local dependence, categorical or mixed response formats, and missingness.

Usage

```
sim_lvm(
  N = 1000,
  mla = NULL,
  K = 3,
  J = 18,
  cpf = 0,
  lam = 0.7,
  lac = 0.3,
```

```
  phi = 0.3,
  ph12 = -1,
  phx = NULL,
  ecr = 0,
  P = 0,
  b = 0.3,
  K1 = 0,
  ph1 = 0.2,
  b1 = 0.3,
  phd = 0,
  ilvl = NULL,
  cati = NULL,
  noc = c(4),
  misp = 0,
  ome_out = FALSE,
  necw = K,
  necb = K,
  add_ind = c(),
  add_la = 0.5,
  add_phi = 0,
  zero_it = 0,
  rseed = 333,
  digits = 4
)
```

Arguments

N	Sample size.
mla	Population loading matrix. If supplied, K, J and cpf are taken from it.
K	Number of factors (used when mla = NULL). Must divide J.
J	Number of items (used when mla = NULL).
cpf	Number of cross-loadings per factor (used when mla = NULL).
lam	Value of the primary loadings.
lac	Value of the cross-loadings.
phi	Homogeneous correlation between any two factors.
ph12	Correlation between factors 1 and 2, when it differs from phi; -1 (default) to use phi.
phx	Correlation among the observed covariates. Defaults to phi, which is LAWBL's behaviour; set phx = 0 for uncorrelated covariates.
ecr	Residual correlation (local dependence).
P	Number of observed predictors (MIMIC model); 0 for none.
b	Coefficients of the observed predictors: a scalar, or a (K - K1) x P matrix.
K1	Number of latent predictors (MIMIC / structural model).
ph1	Correlation among the latent predictors.

b1	Coefficients of the latent predictors: a scalar, or a (K - K1) x K1 matrix.
phd	Correlation among the structural disturbances (the residuals of the regression of factors on predictors). 0 (default) gives independent disturbances, which is LAWBL's behaviour; a positive value reproduces designs with a freely correlated factor residual covariance. Ignored when P = 0 and K1 = 0.
ilvl	Levels of every item, length J + P; values below 2 mean continuous. NULL to use cati/noc instead.
cati	Polytomous items by index; NULL for none, -1 for all. Used only when ilvl = NULL.
noc	Number of categories for the polytomous items.
misp	Proportion of missingness, applied to the J items only (never to the covariate block).
ome_out	Return the factor scores.
necw	Number of within-factor local dependencies.
necb	Number of between-factor local dependencies.
add_ind	Indices of items loading on an additional minor factor.
add_la	Loading value on the additional minor factor.
add_phi	Correlation between the additional minor factor and the others.
zero_it	Number of surplus items with zero loadings.
rseed	Random seed.
digits	Significant digits used while the function is running.

Details

sim_lvm() and sim_fa() are complementary rather than interchangeable. Both accept a population loading matrix directly (mla, identical semantics in either function); where they differ is what drives the pattern-based path when mla = NULL, and what else each one models:

sim_lvm() is the *model-side* generator. Use it when the design involves **predictors** – observed covariates (P, b) or latent predictors (K1, ph1, b1) – or **mixed response formats** per item (ilvl). This is the generator for vbmimic().

sim_fa() is the *structure-side* generator. Use it when the question is about the loading pattern itself: items per factor (ipf), alternating-sign cross-loadings (alt_sign), or minor factors (minor, definition-based per Auerswald & Moshagen, 2019). This is the generator for vbfa().

The two generators share ph12 for the factor 1-2 correlation. ph1 exists only in sim_lvm(), where it means something different: the correlation **among latent predictors** (K1), not the factor 1-2 correlation.

Value

A list with the data (dat; items first, then any covariates), the loading matrix (MLA), factor correlations (PHI), residual covariance (PSX), the indices of the local-dependence pairs (ofd_ind), and, when predictors are present, the coefficient matrices (mb, mb1) and covariate correlation (PHX).

References

Chen, J. (2022). *LAWBL: Latent (variable) analysis with Bayesian learning* (R package version 1.5.0).

Jin, Y., & Chen, J. (2025). Regularized variational Bayesian approximations for variable selection in extended multiple-indicators multiple-causes models. *Multivariate Behavioral Research*. doi:10.1080/00273171.2025.2483253

See Also

`sim_fa()` for the structure-side generator, `vbmimic()` and `vbfa()` for the estimators.

Examples

```
## MIMIC data: 18 items on 3 factors, 9 covariates in disjoint blocks of 3.
## Supply `b` as a matrix for a sparse structural design; a scalar `b` makes
## every covariate predict every factor, which quickly leaves no disturbance
## variance (the function stops if it does).
B <- matrix(0, 3, 9)
for (k in 1:3) B[k, ((k - 1) * 3 + 1):(k * 3)] <- .3

s <- sim_lvm(N = 500, K = 3, J = 18, P = 9, b = B, phx = 0, rseed = 1)
dim(s$dat)      # 500 x 27 -- items first, covariates last
s$mb            # the generating structural coefficients

## correlated disturbances (unrestricted factor residual covariance)
s2 <- sim_lvm(N = 500, K = 3, J = 18, P = 9, b = B, phx = 0, phd = .3,
              rseed = 1)
round(s2$PHI, 2)

## matrix-driven: supply the population loading matrix directly
mla <- matrix(0, 12, 2)
mla[1:6, 1] <- .7; mla[7:12, 2] <- c(.8, .7, .6, .7, .8, .6)
s3 <- sim_lvm(N = 300, mla = mla, phi = .3, rseed = 4)
s3$MLA                    # the generating matrix, as supplied
```

---

special_effects *Special-effect summaries of a bifactor fit*

Description

Post-processes an orthogonal bifactor `vbfa()` fit into the three summaries used throughout the `bifactor` vignette: (i) the **Schmid-Leiman (SL) proportionality check** – the within-group coefficient of variation (CV) of the group-to-general loading ratio, per group factor and item-weighted overall; (ii) the **(approximate) higher-order parameterization** – second-order loadings $\hat{\gamma}_k = 1/\sqrt{1+\bar{r}_k^2}$ from each group's mean ratio $\bar{r}_k$, and first-order loadings $\hat{\lambda}_j = |b_j^{gen}|/\hat{\gamma}_{k(j)}$; and (iii) **testlet/special effect sizes** by Eq. 16 of Zhang and Chen (2024): the average squared loading on each group factor, counting specified loadings always and unspecified ones only when their posterior inclusion probability exceeds `tau`.

Usage

```
special_effects(object, tau = 0.5)

## S3 method for class 'special_effects'
print(x, digits = 3, ...)
```

Arguments

object	A fitted vbfa() object with a bifactor structure (see Details).
tau	Hard-selection threshold on the posterior inclusion probability (default 0.5), used both for group-membership inference and for the Eq.-16 effect sizes.
x	A special_effects object.
digits	Number of decimals in the printed table.
...	Unused.

Details

All output is **descriptive**. Near-zero CVs signal SL proportionality – the fitted bifactor is then (approximately) a higher-order model, and by the same token a testlet model, so the higher-order parameters are a faithful re-expression. Values near .1 are conventionally read as approximately higher-order, but the cutoff is suggestive, not a test; the higher-order parameters and effect sizes are reported regardless, and with clearly non-constant ratios they are a deliberately lossy summary of a genuinely richer bifactor. The ratio inherits the sampling softness of the general/group split, so CVs are noisy below (roughly) a few thousand observations – see the bifactor vignette for demonstrations, caveats, and the covariance geometry behind all three summaries.

Which fits are accepted. A fit with bifactor = TRUE is used as-is (general column first, group columns after). A legacy hand-built fit is also accepted when it has orthogonal = TRUE and exactly one all-specified column in Q, which is taken as the general factor. Group membership of each item is the group column where it is anchored (Q == 1) or, for unspecified rows, its single active group column (inclusion probability above tau). Items with no active group loading are excluded from the ratio summaries and reported; items active on several group columns are assigned to the anchored one (else the largest absolute loading) and reported. Ratios use absolute loadings; items whose general loading is fixed at zero (bifactor-(S-1)-style designs) carry no ratio and are reported.

Value

An object of class special_effects: a list with

groups	A data frame with one row per group factor: n_items (assigned items), mean_ratio, sd_ratio, cv, gamma (second-order loading), and D (Eq.-16 effect size).
mean_cv	The item-weighted mean CV across group factors.
items	A data frame with one row per item: assigned group (NA if unassigned), b_gen, b_grp, ratio, and lambda (the first-order loading implied by the higher-order parameterization).
gamma	Named vector of second-order loadings (same as in groups).
unassigned, crossloaded, no_general	Integer vectors of item indices flagged by the membership rules above.

`tau`, `source`, `call`
: Bookkeeping (`source` is `"bifactor"` for native fits, `"legacy"` for hand-built ones).

References

Schmid, J., & Leiman, J. M. (1957). The development of hierarchical factor solutions. *Psychometrika*, 22(1), 53-61.

Yung, Y.-F., Thissen, D., & McLeod, L. D. (1999). On the relationship between the higher-order factor model and the hierarchical factor model. *Psychometrika*, 64(2), 113-128.

Zhang, Y., & Chen, J. (2024). Accommodating and extending various models for special effects within the generalized partially confirmatory factor analysis framework. *Applied Psychological Measurement*, 48(4-5), 208-229. doi:10.1177/01466216241261704

See Also

`vbfa()`, and the `bifactor` vignette for the full workflow.

Examples

```
## higher-order data: the ratios are near-constant and gamma is recovered
sim <- sim_fa(N = 500, K = 3, ipf = 6, lam = .8, gamma = c(.8, .7, .6),
              rseed = 11)
Q <- matrix(-1L, 18, 3)
for (k in 1:3) { a <- which(rep(1:3, each = 6) == k)[1:2]
                 Q[a, ] <- 0L; Q[a, k] <- 1L }
fit <- vbfa(sim$dat, Q, bifactor = TRUE, v0 = .001)
special_effects(fit)
```

`ssl` *Sums of squared loadings for a PEFA sweep*

Description

Descriptive column sizes for every candidate in a `pefa()` sweep: `colSums(Lam^2)` on the stored loading matrix of each candidate. They are not Gram eigenvalues, they take part in no correspondence or selection rule, and no cutoff is applied to them. `summary()` calls this function, so there is one definition.

Usage

```
ssl(x)
```

Arguments

`x`
: A result returned by `pefa()`.

Details

A low `phi_min` in `$transitions` or `$persistence$phi` is a weakest-link reading over every source column, including one the fit has driven to nearly zero. Such a column matches the smallest-norm target and carries an essentially arbitrary congruence. `ssl()` is what separates "this candidate contains a column with nothing in it" from "the retained structure moved".

Value

A list with one element per candidate, named as `x$loadings` is (the candidate `K` as a character string). Each element is the double vector `colSums(L^2)` of that candidate's loading matrix and keeps the stored loading-column names when the matrix has them. In bifactor mode each vector covers all `K + 1` stored columns with the general factor first, so the group-block scale used by every between-candidate comparison is `ssl(x)[[k]][-1]`.

See Also

`pefa()`

---

vbfa *Variational Bayes partially confirmatory factor analysis*

Description

Fits a partially confirmatory factor-analytic measurement model by regularized mean-field variational Bayes. Loadings on *specified* factors are anchored by the design matrix `Q`; loadings on *unspecified* factors are selected from the data through continuous spike-and-slab priors. Optional features are a warm-started dynamic regularization path, an orthogonal bifactor parameterization, and sparse residual (local-dependence) estimation.

Usage

```
vbfa(
  Y,
  Q,
  ld = FALSE,
  Qe = NULL,
  orthogonal = FALSE,
  bifactor = FALSE,
  general = 1,
  v0 = c(0.01, 0.005, 0.002, 0.001),
  max_it = 5000,
  convChk = FALSE,
  tolVal = 1e-04,
  ld_control = list()
)
```

Arguments

Y	Numeric N x J data matrix (observations by items). Standardized internally. Missing continuous responses are handled by deterministic Gaussian conditional-moment augmentation inside the variational loop.
Q	Integer J x K design matrix for the loadings: 1 = specified (anchored) loading, 0 = fixed zero, -1 = unspecified (estimated by spike-and-slab). K is the number of factors.
ld	Logical switch for local-dependence (sparse residual) estimation. FALSE (default) fits diagonal residuals and ignores Qe (with a warning if one is supplied). TRUE estimates a sparse residual precision by a graphical spike-and-slab prior solved by QUIC, with the search space given by Qe.
Qe	Residual design matrix (J x J), read only when ld = TRUE. Entries 1 (freely estimated residual dependence), -1 (uncertain; selected by spike-and-slab), 0 (shrunk to zero). Symmetric; its diagonal is ignored. The default NULL means fully exploratory local dependence, i.e. Qe = matrix(-1, J, J).
orthogonal	Logical. If TRUE, the factor correlation is fixed at the identity (a plain orthogonal factor model; also the engine under a hand-built bifactor design). Default FALSE (oblique; factor correlation estimated). For bifactor models prefer bifactor = TRUE, which implies orthogonality and keeps K counting group factors.
bifactor	Logical. If TRUE, fit an orthogonal bifactor model with Q supplying only the **group** design (J x K for K group factors): the general column is added internally according to general, so the user-facing K counts group factors and is directly comparable to an oblique K-factor model. Implies orthogonal factors – bifactor = TRUE overrides orthogonal = FALSE (with a message). Supplying a Q that already contains an all-specified column is an error: with bifactor = TRUE the general column must come from general, not from Q.
general	Design of the general column when bifactor = TRUE: a scalar or length-J vector with the usual codes (1 specified, 0 fixed zero, -1 unspecified). The default 1 gives the standard bifactor (every item loads on the general factor). A vector with 0 entries yields bifactor-(S-1)-style designs in which some items carry no general loading; at least one entry must be 1.
v0	Loading-spike variance. A decreasing **vector** gives a warm-started dynamic regularization path (each stage runs to convergence and warm-starts the next); a **scalar** gives a single fixed spike. The default path ends at 0.001; a scalar v0 = 0.001 reproduces the fixed-spike estimator exactly. For a **gentler path that aids convergence**, pass a finer, longer, log-spaced descent starting from a larger v0, e.g. v0 = exp(seq(log(0.05), log(0.001), length.out = 10)). Each stage is then a small perturbation of the previous one, so its warm start is a good initialization and it converges quickly and stably. Prefer this when a fit fails to converge within max_it, when the loading solution looks unstable, or for near-singular / strongly regularized models: it navigates the non-convex spike-and-slab landscape better than a single hard fit, and is usually preferable to (and often faster than) simply raising max_it. Keep the endpoint at your target; log spacing suits the multiplicative spike.
max_it	Maximum variational iterations **per stage**. Default 5000.

convChk — Logical; report per-iteration relative error via message() (so it can be silenced with suppressMessages()). Default FALSE (quiet).

tolVal — Convergence tolerance on the maximum relative error.

ld_control — A named list of local-dependence controls, read only when ld = TRUE: xi0 (spike-penalty path, units of N; default seq(0.1, 1, length.out = 5)), xi1 (slab penalty, units of N; default 0.01), diag_penalty (1 penalizes the precision diagonal with xi1, 0 leaves it unpenalized), quic_eps, quic_max_it, and the Beta prior parameters a1, b1 on the local-dependence proportion.

Value

An object of class c("vbfa", "vbpm_fit") – a named list (see vbpm_fit for what the class does and does not change) with elements:

Lam J x K posterior mean loadings.

pi Posterior inclusion probabilities of the unspecified loadings (specified entries are reported as coded in Q).

eta N x K posterior mean factor scores.

Phi Factor correlation matrix; the identity when orthogonal = TRUE.

PsiInv Residual precisions; NULL when ld = TRUE.

ELBO Evidence lower bound; NA when ld = TRUE.

rho Posterior inclusion rate of the unspecified loadings.

Lam_var Posterior variances of the loadings.

eta_cov Posterior covariance of the factor scores.

Phi_inv_mean Posterior mean of the factor precision, E[Phi^-1].

iter, flag, time Total iterations, 1 if the final stage converged within max_it (else 0), and elapsed time.

Q, Qe, orthogonal, ld The design matrices and settings the model was fit with (Qe is NULL when ld = FALSE); read by fit_stats() and print.vbpm_fit() so they never need repeating.

bifactor, n_general, general Bifactor bookkeeping: whether the fit declared a bifactor structure, how many general columns it has, and the general design (NULL unless bifactor = TRUE).

Psi, W, q_star, xi_star, tau Local-dependence results (NULL when ld = FALSE): residual precision, its inverse, residual-edge inclusion probabilities, edge shrinkage, and the LD proportion.

path The v0 schedule (and xi0 under LD) with per-stage iteration counts.

preprocess Centering and scaling used, n_missing, the response type, and missing_mask – the mask is stored only when the data actually contain missing values, and is NULL otherwise.

sample_cov The sample correlation matrix the fit statistics use.

Every quantity appears exactly once: as of 0.7.0 the fit no longer carries $coefficients, $design, $posterior, or $settings, which duplicated the elements above.

Notes

The diagonal-residual estimator is **deterministic** (fixed initialization, no random numbers), so results are reproducible without a seed; there is no `rseed` argument. (The local-dependence branch's QUIC solver shuffles its active set with the RNG, but that is a convex solve, so the shuffle order does not change the solution.)

Missing continuous responses are supported (see `Y`). Support for categorical or mixed responses is not yet implemented.

Under local dependence `ELBO` is `NA` by design, and this is not a placeholder: the residual precision and the mixing proportion are point updates rather than variational factors, so no single joint mean-field bound over all unknowns is defined. The quantities that *are* defined are returned under their own names — `objective` (the terminal VECM objective, with `objective_type = "vecm"`) and `ELBO_conditional` (the conditional bound given the terminal precision and mixing proportion). Use `objective` for model comparison, and only across fits sharing one `objective_type`.

References

Chen, J., Guo, Z., Zhang, L., & Pan, J. (2021). A partially confirmatory approach to scale development with the Bayesian Lasso. *Psychological Methods*, 26(2), 210-235. doi:10.1037/met0000293

Chen, J. (2021). A generalized partially confirmatory factor analysis framework with mixed Bayesian Lasso methods. *Multivariate Behavioral Research*, 57(6), 879-894. doi:10.1080/00273171.2021.1925520

Jin, Y., & Chen, J. (2025). Regularized variational approximation for partially confirmatory factor analysis. *Structural Equation Modeling: A Multidisciplinary Journal*, 32(3), 437-449. doi:10.1080/10705511.2024.2432612

Jin, Y., Chen, J., Yan, Z., & Zhang, Y. (2026). Sparse residual estimation in partially confirmatory factor analysis. *PsyArXiv preprint*. doi:10.31234/osf.io/dehtv_v2 — the warm-started `v0` path and the `ld = TRUE` sparse-residual (local dependence) branch.

Rockova, V., & George, E. I. (2018). The spike-and-slab LASSO. *Journal of the American Statistical Association*, 113(521), 431-444. doi:10.1080/01621459.2016.1260469

Examples

```
## partially confirmatory: two anchors per factor, the rest exploratory
sim <- sim_fa(N = 200, K = 3, ipf = 6, lam = .7, lac = .3, rseed = 1)
Q <- matrix(-1L, ncol(sim$dat), 3)
for (k in 1:3) { a <- which(rep(1:3, each = 6) == k)[1:2]; Q[a, ] <- 0; Q[a, k] <- 1 }
fit <- vbfa(sim$dat, Q)
fit$flag                    # 1 if converged
round(fit$Lam, 2)           # loadings
round(fit$pi, 2)            # posterior inclusion probabilities


## orthogonal bifactor, preferred form: Q holds only the GROUP design and
## K counts group factors (comparable to an oblique K-factor model)
fb <- vbfa(sim$dat, Q, bifactor = TRUE, v0 = .001)  # fixed spike: fast demo
fb                          # prints "1 general + 3 group factors"
fb$Phi                      # identity by construction
## equivalent long form: vbfa(sim$dat, cbind(1L, Q), orthogonal = TRUE)
```

```
## local dependence: simulate correlated residuals, then set ld = TRUE
simLD <- sim_fa(N = 300, K = 3, ipf = 6, lam = .7, lac = .3, ecr = .3,
                rseed = 2)
fLD <- vbfa(simLD$dat, Q, ld = TRUE, max_it = 300,
            tolVal = 1e-3)
## largest recovered residual edges vs the planted ones
Poff <- abs(fLD$Psi); Poff[lower.tri(Poff, diag = TRUE)] <- 0
which(Poff >= sort(Poff, decreasing = TRUE)[3], arr.ind = TRUE)
simLD$ofd_ind

## empirical illustration: NLSY 1997, 27 items treated as continuous.
## A 500-row subset keeps the example quick; the vignette fits all 3,458.
data(nlsy27)
Yn <- as.matrix(nlsy27$dat)[1:500, ]
sum(is.na(Yn))            # incomplete cells, imputed in-loop
fn <- vbfa(Yn, nlsy27$Q)
round(fn$Lam, 2)
```

---

vbmimic *Regularized variational Bayes for extended MIMIC models*

Description

Fits a partially confirmatory multiple-indicators multiple-causes (MIMIC) model by a variational Bayes EM algorithm, with continuous spike-and-slab regularization on **both** the measurement and the structural part. Items load on factors through a measurement design matrix `Q_A`, while covariates predict the factors through a structural design matrix `Q_B`; entries of either that are left unspecified are selected from the data.

Usage

```
vbmimic(
  Y,
  X,
  Q_A,
  Q_B,
  v0 = 0.001,
  standardize = FALSE,
  max_it = 1000,
  convChk = FALSE,
  tolVal = 1e-05
)
```

Arguments

`Y`	Numeric `N x J` data matrix (observations by items). Missing values in `Y` are supported: because the residual covariance is diagonal, a missing response is

replaced in-loop by its conditional expectation `eta_i A'`, with the conditional variance carried into the residual sum of squares. Items or rows with no observed response are rejected.

`X` Numeric `N x P` matrix of observed covariates (predictors of the factors), with the same number of rows as `Y`. Missing covariates are **not** supported: `X` is conditioned on rather than modelled, so imputing it would require a distributional assumption the MIMIC model does not make.

`Q_A` Integer `J x K` measurement design matrix: `1` = specified (anchored) loading, `0` = fixed zero, `-1` = unspecified (selected by spike-and-slab). `K` is the number of factors.

`Q_B` Integer `K x P` structural design matrix using the same coding, for the regression of the `K` factors on the `P` covariates.

`v0` Spike variance. A scalar gives a single fixed spike; a decreasing vector gives a warm-started regularization path, each stage initialized at the previous stage's solution (Rockova & George, 2018). The same schedule is applied to the measurement and structural parts. `v0 = 0.001` (the default) is the fixed spike of the published estimator.

`standardize` Logical; standardize `Y` and `X` before fitting. The default `FALSE` is the behaviour of the published estimator. Note that `vbfa()` always standardizes internally; set `standardize = TRUE` for the analogous behaviour here.

`max_it` Maximum number of variational iterations *per stage*.

`convChk` Logical; print per-iteration convergence information via `message()`. Default `FALSE` (quiet).

`tolVal` Convergence tolerance on the maximum absolute change in the weighted residual vector.

Details

This is the MIMIC member of the `vbpm` family and is the companion of `vbfa()`, which fits the measurement part alone. The two share the `-1/0/1` design-matrix convention and the spike-and-slab formulation.

Value

An object of class `c("vbmimic", "vbpm_fit")` – a named list (see vbpm_fit) with components:

A `J x K` posterior mean measurement loadings.

B `K x P` posterior mean structural coefficients.

pi_A, pi_B Posterior inclusion probabilities for the unspecified entries of `Q_A` and `Q_B` (specified entries are held at 1, fixed zeros at 0).

Q_A, Q_B The design matrices the model was fit with.

eta `N x K` posterior mean factor scores.

Phi `K x K` factor correlation matrix implied by `Sig`.

Sig, U Factor covariance matrix and its inverse.

V Length-`J` residual precisions.

rho, theta Inclusion rates for the measurement and structural parts.

A_var, B_var Posterior variances of the loadings and structural coefficients.

iter, flag, time Total iterations, `1` if the final stage converged within `max_it` (else `0`), and elapsed time.

ELBO `NA_real_`; see Note.

standardize The `standardize` setting the model was fit with.

path The `v0` schedule and per-stage iteration counts.

preprocess `n_missing`, the response type, and `missing_mask` – stored only when the data contain missing values, `NULL` otherwise.

Every quantity appears exactly once: as of 0.7.0 the fit no longer carries `$coefficients`, `$design`, `$posterior`, or `$settings`, which duplicated the elements above.

Note

The evidence lower bound is **not** currently returned. The published algorithm converges on the weighted residual vector rather than on the bound, and the bound's implementable closed form has not been assembled. `fit_stats()` does accept a `vbmimic` fit, but returns a deliberately limited result: `n_active_coef` (soft-selected measurement and structural coefficients) plus typed `NA` for every covariance-based index and parameter count, with an explanatory `note` attribute. `pefa()` does not accept `vbmimic` fits at all: its sweep reports criterion gains across a factor-count window, which requires one comparable objective for every candidate in that window.

References

Jin, Y., & Chen, J. (2025). Regularized variational Bayesian approximations for variable selection in extended multiple-indicators multiple-causes models. *Multivariate Behavioral Research*. doi:10.1080/00273171.2025.2483253

Rockova, V., & George, E. I. (2018). The spike-and-slab LASSO. *Journal of the American Statistical Association*, 113(521), 431-444.

See Also

`vbfa()` for the measurement model alone, `sim_lvm()` to simulate MIMIC data.

Examples

```
## 3 factors, 18 items, 9 covariates in disjoint blocks of 3
B <- matrix(0, 3, 9)
for (k in 1:3) B[k, ((k - 1) * 3 + 1):(k * 3)] <- .3
sim <- sim_lvm(N = 300, K = 3, J = 18, P = 9, b = B, phx = 0, rseed = 1)
Y <- sim$dat[, 1:18]
X <- sim$dat[, 19:27]

## two anchored items per factor; the structural part left exploratory.
## Anchoring at least one part is advisable: with both Q_A and Q_B fully
## exploratory the solution can be rotationally ambiguous.
Q_A <- matrix(-1L, 18, 3)
```

```
for (k in 1:3) {
  a <- which(rep(1:3, each = 6) == k)[1:2]
  Q_A[a, ] <- 0L; Q_A[a, k] <- 1L
}
Q_B <- matrix(-1L, 3, 9)

fit <- vbmimic(Y, X, Q_A, Q_B)
fit$flag           # 1 if converged
round(fit$B, 2)    # structural coefficients
```

---

vbpm_fit *The vbpm_fit class: what it is and why it exists*

Description

Every estimator in vbpm returns its results with an S3 class attached: vbfa() returns c("vbfa", "vbpm_fit") and vbmimic() returns c("vbmimic", "vbpm_fit"). This page explains what that means in practice, because "S3 class" sounds like more machinery than it is.

What an S3 class actually is

An S3 class is nothing more than a **label attached to an ordinary R object**. A vbpm_fit is still a plain named list – the class attribute changes none of its contents and none of the ways you already use it:

- fitLam, fiteta, fit$ELBO, ... work exactly as before;
- names(fit) lists every component;
- str(fit) shows the full structure;
- unclass(fit) strips the label and gives back the bare list, should you ever want it.

The label does exactly two things. First, generic functions such as print() can now **dispatch**: typing fit at the console no longer floods the screen with every matrix in the object, but shows a compact summary (model type, dimensions, convergence, ELBO, active loadings). Second, functions receiving a fit can **recognize what it is**: this is how fit_stats() knows a bifactor fit from an oblique one without being told (the fit carries its own orthogonal and ld settings), and how it sends a vbmimic() fit to its own method. That method, fit_stats.vbmimic(), returns a populated but deliberately limited result – NA for every SEM-like index, plus a single n_active_coef count of soft-selected measurement and structural coefficients – because the MIMIC objective and joint covariance derivation those indices need are not available yet. It is the family fallback fit_stats.vbpm_fit() that stops instead of computing something meaningless, and it applies to classes with no method of their own.

Why the family shares a parent class

The specific class ("vbfa", "vbmimic") comes first, the family class "vbpm_fit" second. Methods written for vbpm_fit therefore apply to every model in the family unless a model overrides them – one print method serves all estimators, and future family members (e.g. further variational models) inherit sensible behaviour on day one.

Why two classes rather than one class plus a model field

A natural-looking alternative is a single `"vbpm_fit"` class with a field saying which model it is. The field exists – every fit carries `$model` (`"vbfa"` or `"vbmimic"`), and it is the right thing to *read* when a script wants to branch on model type. But it cannot replace the class vector, because the class vector is what R's method dispatch runs on. With the two-element class, model-specific behaviour is added by writing a method (`fit_stats.vbmimic()`), the family default by writing `fit_stats.vbpm_fit()`, and a future model (say `vblvm`) joins by returning `c("vblvm", "vbpm_fit")` and adding only the methods where it differs – including from outside this package. With a single class, every generic would have to contain a hand-written `switch(object$model, ...)` that this package alone can extend, and `inherits(fit, "vbfa")` would stop working. This is the same convention base R uses: a `glm` object has class `c("glm", "lm")`, not class `"lm"` with a family field. So: dispatch on the classes, read `$model` for display and bookkeeping.

What is deliberately NOT hidden

Some R packages wrap results in opaque objects whose internals are discouraged territory. `vbpm` does the opposite: the list components are documented in each estimator's help page (`?vbfa`, `?vbmimic`) and are part of the public API. The class adds convenience on top; it takes nothing away.

See Also

`vbfa()`, `vbmimic()`, `print.vbpm_fit()`

Index

∗ **datasets**
nlsy27, 5

coef.vbpm_fit, 2

fit_stats, 3
fit_stats(), *7*, *9*, *11*, *22*, *26*, *27*

message(), *22*, *25*

nlsy27, 5

pefa, 6
pefa(), *5*, *9*, *10*, *19*, *20*, *26*
plot.pefa *(*print.pefa*)*, 9
print(), *11*, *27*
print.pefa, 9
print.special_effects
*(*special_effects*)*, 17
print.summary.pefa *(*print.pefa*)*, 9
print.vbpm_fit, 11
print.vbpm_fit(), *22*, *28*

sim_fa, 11
sim_fa(), *16*, *17*
sim_lvm, 14
sim_lvm(), *12*, *13*, *26*
special_effects, 17
ssl, 19
ssl(), *7–10*
summary.pefa *(*print.pefa*)*, 9
suppressMessages(), *22*

vbfa, 20
vbfa(), *2*, *3*, *5–7*, *9*, *11*, *12*, *16–19*, *25–28*
vbmimic, 24
vbmimic(), *2–4*, *11*, *12*, *16*, *17*, *27*, *28*
vbpm_fit, *11*, *22*, *25*, 27
vbpm_fit-class *(*vbpm_fit*)*, 27