Package 'RSEML'

September 10, 2026

Type Package

Title Case-Based Least Squares Estimation of Nonlinear Structural Equation Models

Version 0.1.0

Date 2026-08-28

Description Estimates structural equation models by case-based least squares: the latent scores of every observation are treated as free variables of a constrained optimization problem, so that arbitrary nonlinear model equations, bounds and constraints on latent variables and inequality constraints on parameters become possible. Model equations are specified as plain text (e.g. ``y == a*exp(b*eta)''). Gradients are obtained by automatic differentiation via 'RTMB', and the constrained problem is solved with 'nloptr' (SLSQP or augmented Lagrangian). Missing data are handled case-wise. The methodology is described in Oldenburg (2024) <doi:10.19139/soic-2310-5070-1868> and Oldenburg (2025) <doi:10.19139/soic-2310-5070-2324>.

License GPL (>= 3)

Encoding UTF-8

Imports RTMB, nloptr, stats

Suggests knitr, rmarkdown

NeedsCompilation no

Author Reinhard Oldenburg [aut, cre]

Maintainer Reinhard Oldenburg <reinhard.oldenburg@math.uni-augsburg.de>

Repository CRAN

Date/Publication 2026-09-10 09:20:12 UTC

Contents

RSEML-package — *Case-Based Least Squares Estimation of Nonlinear Structural Equation Models*

Description

RSEML estimates structural equation models by case-based least squares. In contrast to covariance-based SEM, the latent score of every observation is treated as a free variable of a constrained optimization problem. This makes three things possible that are difficult in classical SEM software: (i) arbitrary *nonlinear* model equations, e.g. products of latent variables or `y == a*exp(b*eta)`; (ii) *bounds and constraints on latent variables* (e.g. restriction to the unit interval, so that products of latent variables can be read as fuzzy conjunctions); and (iii) inequality constraints on parameters.

Gradients of the objective and of all constraints are obtained by automatic differentiation with **RTMB**; the constrained problem is solved with **nloptr** (SLSQP or augmented Lagrangian with an L-BFGS core). Missing data are handled case-wise per equation.

The main function is `fsem`. `sim_bollen` simulates example data.

Author(s)

Reinhard Oldenburg

References

Oldenburg, R. (2024). Least square estimation of non-linear structural models. *Statistics, Optimization & Information Computing*, 12(2), 281–297. doi:10.19139/soic231050701868

Oldenburg, R. (2025). Geometric weighted least squares estimation. *Statistics, Optimization & Information Computing*, 13(2), 611–615. doi:10.19139/soic231050702324

fsem — *Case-Based Least Squares Estimation of Nonlinear Structural Equation Models*

Description

Estimates a structural equation model by case-based least squares: the latent score of every observation is a free variable of a constrained optimization problem. Model equations may be arbitrary nonlinear expressions in observed variables, latent variables and parameters; latent variables can be bounded or constrained. Gradients are obtained by automatic differentiation (package **RTMB**); the constrained problem is solved with **nloptr**.

Usage

```
fsem(data, observed, latent, equations, errnames = NULL,
     emean0 = TRUE, lmean0 = FALSE,
     constraints = character(), latent_bounds = NULL,
     weights = 10, with_w1 = FALSE, with_w2 = FALSE,
     le_damp = FALSE, ee_damp = FALSE,
     eee_damp = FALSE, eeee_damp = FALSE,
     damp_e_expo = 1, damp_l_expo = 1,
     damp_ee_expo = 1, damp_eee_expo = 1,
     ee_dist = 1L,
     init = list(), lat_init = 0, lat_jitter = 0.1,
     warm_start = TRUE,
     algorithm = c("auto", "slsqp", "auglag"),
     max_iter = 1000L, tol = 1e-8, verbose = 0L)
```

Arguments

`data`	numeric matrix or data frame with one row per case. If column names are present they are matched against `observed`; otherwise the columns must be in the order of `observed`. Missing values (`NA`) are allowed and are handled case-wise per equation.
`observed`	character vector of observed variable names.
`latent`	character vector of latent variable names.
`equations`	list or vector of model equations, each given as a string `"lhs == rhs"` (or a two-sided formula). The right hand side may be any arithmetic expression in observed variables, latent variables and free parameters, e.g. `"y == a*exp(b*eta) + c"`. Every symbol that is neither an observed nor a latent variable is treated as a free parameter.
`errnames`	optional character vector of error-term labels (used for naming the residual matrix); defaults to `e1, e2, ...`.
`emean0`	logical; if `TRUE` the residual of every equation is constrained to have mean zero.
`lmean0`	logical; if `TRUE` every latent variable is constrained to have mean zero.
`constraints`	character vector of constraints such as `"b1 > 0"`, `"eta <= 1"` or `"a + b == 1"`. Constraints of the simple form *symbol op number* are converted to box bounds (cheap); all others are passed to the optimizer as general nonlinear constraints. A constraint that evaluates to a vector of length n (because it involves a latent variable) is applied per case.
`latent_bounds`	optional numeric vector `c(lower, upper)` applied as box bounds to *all* latent scores, e.g. `c(0, 1)`.
`weights`	equation weights: a positive scalar (used for all equations), a vector with one entry per equation, or a value `<= 0` to use the data-driven default $10/(0.001 + sd(data))$.
`with_w1`	logical; if `TRUE`, after the first solution the model is re-estimated with weights $1/sd$ of the residuals (warm-started from the first solution).

`with_w2` logical; if `TRUE` a further stage is run in which the weights are free variables with $w_j \geq 0.001$ and $\sum_j \log(w_j^2 \sigma_j^2) \geq 0$. The stage is accepted only if the optimizer converges and the objective is smaller than $1.05M$, where M is the number of objective blocks.

`le_damp`, `ee_damp`, `eee_damp`, `eeee_damp`
logical damping switches that add penalties on latent-error covariances, and on pairwise, triple and quadruple error-error (co)moments, respectively.

`damp_e_expo`, `damp_l_expo`, `damp_ee_expo`, `damp_eee_expo`
exponents of the sample size in the corresponding damping weights.

`ee_dist` index stride used in the triple and quadruple damping sums.

`init` named list of start values: scalars for parameters, scalars or length-n vectors for latent variables.

`lat_init` default start value for latent scores.

`lat_jitter` standard deviation of Gaussian jitter added to the latent start values. An all-constant start is a stationary point of the objective, so a small jitter (default 0.1) is recommended; set to 0 to disable (use `set.seed` for reproducibility).

`warm_start` logical; if `TRUE` (default) and any damping option is active, the undamped problem is solved first and the damped optimization is warm-started from its solution. Damped objectives are hard to optimize from a cold start, so this should normally be left enabled.

`algorithm` `"slsqp"` (sequential quadratic programming, reliable for small and medium problems), `"auglag"` (augmented Lagrangian with an L-BFGS core, scales to large problems) or `"auto"` (default; chooses by problem size).

`max_iter` maximum number of optimizer iterations.

`tol` relative convergence tolerance passed to **nloptr**.

`verbose` 0, 1 or 2 for increasing diagnostic output.

Details

For equations $f_j(\mathrm{obs}, \mathrm{lat}, \theta) = 0$ with residuals r_{ij} of case i in equation j, the objective is the weighted sum of residual variances

$$F = \sum_j w_j^2 \, \widehat{Var}(r_{\cdot j})$$

subject to the chosen mean constraints, bounds and user constraints, plus optional damping penalties that push error-error and latent-error covariances towards zero. All means and variances are computed over the cases that are complete for the respective equation, so missing data reduce the effective sample size per equation but do not remove whole cases.

Because the latent scores are explicit variables, the model equations may be nonlinear in the latent variables (products, powers, `exp`, `log`, `sqrt`, trigonometric functions, ...) and constraints such as `latent_bounds = c(0, 1)` give the latent variables a direct interpretation (e.g. as fuzzy truth values).

Note that `emean0 = TRUE` and `lmean0 = TRUE` are *joint* constraints: an equation such as `"x1 == eta"` without an intercept parameter then forces `mean(x1) == mean(eta) == 0`, which contradicts the

data and makes the problem infeasible (the optimizer stops immediately, often with **nloptr** status -4). Include intercept parameters (`"x1 == eta + t1"`) or drop one of the two mean constraints.

`RMSEA` is the square root of the mean unexplained residual variance per equation; `RMSEAad` additionally adjusts for the number of latent scores and free parameters. These indices are analogues of, but not identical to, the covariance-based RMSEA of classical SEM; see Oldenburg (2024) for details.

Value

An object of class `"fsem"`, a list with components

`coefficients`	named vector of parameter estimates.
`standardized`	standardized coefficients (where defined, i.e. for parameters that multiply a parameter-free regressor).
`scores`	$n \times L$ matrix of estimated latent scores.
`residuals`	$n \times m$ matrix of residuals (`NA` for cases that are incomplete in an equation).
`EVars`, `ECor`, `ECov`	residual variances, correlations and covariances (pairwise complete).
`LLCor`	correlation matrix of the latent scores.
`LECor`	correlations between latent scores and residuals.
`RMSEA`, `RMSEAad`, `Fmin`	fit measures, see Details.
`equations`	fitted equations with estimates substituted.
`weights`	equation weights used in the final stage.
`convergence`, `status`	convergence flag and **nloptr** status.
`time`, `algorithm`, `call`	timing, algorithm and matched call.

`print`, `summary`, `coef` and `residuals` methods are available.

Equation syntax

Each model equation is a string of the form

```
"<expression> == <expression>"
```

(`"="` or a two-sided formula `lhs ~ rhs` are also accepted). Both sides are parsed with `str2lang` and must be valid R arithmetic expressions built from

- *observed variables* (names listed in `observed`; they stand for the corresponding data column),
- *latent variables* (names listed in `latent`; they stand for the case-wise vector of latent scores),
- *parameters*: every other symbol is treated as a free scalar parameter to be estimated (so misspelled variable names silently become parameters – check the parameter list of the result),
- numeric literals, parentheses, and the operators `+ - * / ^` (unary minus included).

Supported functions are those with an automatic-differentiation method for **RTMB**'s `advector` class, in particular `exp`, `log`, `log1p`, `expm1`, `sqrt`, `abs`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `sinh`, `cosh`, `tanh`, `gamma`, `lgamma`, and the distribution functions provided by **RTMB** such as `dnorm`, `pnorm`, `plogis`. All operations are applied case-wise (vectorized over the n cases). Not allowed are control flow (`if`, loops), comparisons, indexing, assignments and functions without an `advector` method; smooth replacements should be used instead, e.g. `(x + sqrt(x^2 + 1e-4))/2` for the positive part of `x`. Examples of valid equations:

```
"y1 == b1*eta + c1"
"y  == a*exp(b*eta)"
"A1 == b0 + b1*GEO*LOC*MUL"
"p  == plogis(a + b*theta)"
```

Constraints in `constraints` follow the same expression syntax, combined with one relational operator ==, <=, >=, < or >.

Author(s)

Reinhard Oldenburg

References

Oldenburg, R. (2024). Least square estimation of non-linear structural models. *Statistics, Optimization & Information Computing*, 12(2), 281–297. doi:10.19139/soic231050701868

Oldenburg, R. (2025). Geometric weighted least squares estimation. *Statistics, Optimization & Information Computing*, 13(2), 611–615. doi:10.19139/soic231050702324

See Also

`sim_bollen`, `bollen_equations`

Examples

```
## a small nonlinear one-factor model:  x1 = eta,  x2 = c*eta,
## y = a*exp(b*eta), with the latent variable restricted to [0, 2.5]
set.seed(1)
n <- 60
eta <- abs(rnorm(n))
dat <- cbind(x1 = eta + rnorm(n, 0, 0.15),
             x2 = 0.7 * eta + rnorm(n, 0, 0.15),
             y  = 0.5 * exp(0.8 * eta) + rnorm(n, 0, 0.10))
fit <- fsem(dat,
            observed = c("x1", "x2", "y"),
            latent = "eta",
            equations = c("x1 == eta",
                          "x2 == c2*eta",
                          "y  == a*exp(b*eta)"),
            constraints = c("a > 0.01", "b > 0.01"),
            latent_bounds = c(0, 2.5),
            init = list(a = 0.3, b = 0.5, c2 = 0.5, eta = 0.5),
            lat_jitter = 0)
```

```
fit
cor(fit$scores[, "eta"], eta)


## Bollen-type model with three latent variables (takes a few seconds).
## The damping penalties on error-error and latent-error covariances
## with small sample-size exponents improve the estimates.
set.seed(2)
dat <- sim_bollen(50)
fit <- fsem(dat, observed = colnames(dat),
            latent = c("ind60", "dem60", "dem65"),
            equations = bollen_equations(),
            lmean0 = TRUE,
            constraints = c("b1 > 0", "b2 > 0", "b3 > 0"),
            le_damp = TRUE, ee_damp = TRUE,
            damp_e_expo = 0.1, damp_l_expo = 0.1)
summary(fit)
```

`sim_bollen` *Simulate Bollen-Type Example Data and the Matching Model*

Description

`sim_bollen` simulates data from a three-latent-variable model in the style of Bollen's political democracy model with eleven observed indicators; `bollen_equations` returns the matching model equations for `fsem`. The structural equation for `dem65` can be made nonlinear through the exponents `expos`:

$$DEM65 = b_2\, IND60^{e_1} + b_3\, DEM60^{e_2} + \epsilon.$$

Usage

```
sim_bollen(n,
           para = c(b1 = 1.2, b2 = 0.5, b3 = 0.8, c2 = 0.7, c3 = 0.9,
                    d2 = 0.3, d3 = 0.9, d4 = 1.7, d6 = 0.6, d7 = 0.4,
                    d8 = 1.3),
           sigma = c(0.1, 0.2, 0.3, 0.2, 0.1, 0.2, 0.3, 0.2, 0.1, 0.2,
                     0.3, 0.3, 0.2),
           expos = c(1, 1))

bollen_equations(expos = c(1, 1))
```

Arguments

`n`	number of cases to simulate.
`para`	named vector of true structural coefficients and loadings.
`sigma`	error standard deviations (11 indicators followed by the two structural disturbances).

`expos` exponents (e_1, e_2) of the structural equation for `dem65`; `c(1, 1)` gives the linear model.

Value

`sim_bollen` returns an $n \times 11$ matrix with columns `y1, ..., y8, x1, x2, x3`; the true latent scores are attached as attribute `"latent"`. `bollen_equations` returns a character vector of 13 model equations.

See Also

`fsem`

Examples

```
set.seed(1)
dat <- sim_bollen(40)
head(dat)
bollen_equations(expos = c(2, 1))[2]
```

Index