

Package ‘plssem’

September 26, 2026

Type Package

Title Complex Partial Least Squares Structural Equation Modeling

Version 0.1.5

Maintainer Kjell Solem Slupphaug <slupphaugkjell@gmail.com>

Description Estimate complex Structural Equation Models (SEMs) by fitting Partial Least Squares Structural Equation Modeling (PLS-SEM) and Partial Least Squares consistent Structural Equation Modeling (PLSc-SEM) specifications that handle categorical data, non-linear relations, and multilevel structures. The implementation follows Lohmöller (1989) for the classic PLS-SEM algorithm, Dijkstra and Henseler (2015) for consistent PLSc-SEM, Dijkstra et al., (2014) for nonlinear PLSc-SEM, and Schuberth, Henseler, Dijkstra (2018) for ordinal PLS-SEM and PLSc-SEM. Additional extensions are under development. The MC-OrdPLSc algorithm, used to handle ordinal interaction models is detailed in Slupphaug et al., (2026).

References:

Lohmöller, J.-B. (1989, ISBN:9783790803002).

``Latent Variable Path Modeling with Partial Least Squares."

Dijkstra, T. K., & Henseler, J. (2015).

<doi:10.1016/j.jmva.2015.06.002>.

``Consistent partial least squares path modeling."

Dijkstra, T. K., & Schermelleh-Engel, K. (2014).

<doi:10.1016/j.csda.2014.07.008>.

``Consistent partial least squares for nonlinear structural equation models."

Schuberth, F., Henseler, J., & Dijkstra, T. K. (2018).

<doi:10.1007/s11135-018-0767-9>.

``Partial least squares path modeling using ordinal categorical indicators."

Slupphaug, K. Mehmetoglu, M. & Mittner, M. (2026).

<doi:10.31234/osf.io/fwzj6_v1>.

``Consistent Estimates from Biased Estimators: Monte-

Carlo Consistent Partial Least Squares for Latent Interaction Models with Ordinal Indicators."

License GPL-3

Encoding UTF-8

LazyData true

Imports methods, stats, modsem ($\geq 1.0.21$), lme4, lavaan, stringr,
Rfast, collapse, mvnfast, reformulas, future, future.apply,
progressr, FNN, MASS, pbivnorm, Rcpp ($\geq 1.1.2$)

LinkingTo Rcpp, RcppArmadillo

Depends R ($\geq 4.1.0$)

URL <https://github.com/kss2k/plssem>, <https://kss2k.github.io/plssem/>

Suggests knitr, rmarkdown, ggplot2, mice, mvtnorm, pkgload

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Kjell Solem Slupphaug [aut, cre] (ORCID:
<https://orcid.org/0009-0005-8324-2834>)

Repository CRAN

Date/Publication 2026-09-26 11:50:02 UTC

Contents

boot	3
coef,PlsModel-method	4
fit_measures	4
is_admissible	5
is_mcpls	5
mcpls_loglik	6
oneIntOrdered	6
parameter_estimates	7
parameter_estimates,PlsModel-method	7
pls	8
pls_boot	12
pls_chisq	13
pls_chisq_df	14
pls_construct_scores	14
pls_implied_construct_corr	15
pls_implied_indicator_corr	15
pls_implied_joint_corr	16
pls_inspect	17
pls_predict	18
pls_rmsea	19
pls_srmr	20
predict,PlsModel-method	20
print.PlsSemPredict	21
print.SummaryPlsSem	21
randomIntercepts	22
randomInterceptsOrdered	22
randomSlopes	23

boot 3

randomSlopesOrdered	23
show,PlsModel-method	24
summary,PlsModel-method	24
titanic	25
TPB_Ordered	26
unstandardized_estimates	26
vcov,PlsModel-method	28

Index 29

<i>boot</i>	<i>Retrieve bootstrap coefficient matrix</i>
-------------	--

Description

Retrieve bootstrap coefficient matrix

Usage

```
boot(object)

## S4 method for signature 'PlsModel'
boot(object)

## S4 method for signature 'PlsModel'
pls_boot(object)
```

Arguments

object A fitted model object.

Value

A *PlsSemMatrix* of bootstrap replicate parameter vectors (rows = replicates, cols = parameters).

Examples

```
library(modsem)
library(plssem)

m <- "
  X =~ x1 + x2 + x3
  Z =~ z1 + z2 + z3
  Y =~ y1 + y2 + y3
  Y ~ X + Z + X:Z
"

fit <- pls(m, oneInt, bootstrap = TRUE, boot.R = 50)
boot(fit)
```

coef, PlsModel-method	<i>Extract coefficients from a PlsModel model</i>
-----------------------	---

Description

Extract coefficients from a PlsModel model

Usage

```
## S4 method for signature 'PlsModel'
coef(object, use.labels = TRUE, ...)

## S4 method for signature 'PlsModel'
coefficients(object, ...)
```

Arguments

object	A PlsModel object.
use.labels	Logical; Should parameter labels be used as names?
...	Currently unused.

Value

A named PlsSemVector of parameter estimates.

fit_measures	<i>Fit Measures</i>
--------------	---------------------

Description

Computes global fit measures (e.g., chi-square, SRMR, RMSEA) for a fitted model.

Usage

```
fit_measures(object, saturated = FALSE, mc.reps = 1e+06, ...)

## S4 method for signature 'PlsModel'
fit_measures(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

object	A fitted [PlsModel] object.
saturated	Logical; if 'TRUE', compute the saturated fit.
mc.reps	Integer; number of Monte Carlo resamples used for MC-PLSc fit.
...	Reserved for future extensions.

Value

A named list with fit statistics.

is_admissible	<i>Check whether a fitted model has admissible parameter estimates</i>
---------------	--

Description

Check whether a fitted model has admissible parameter estimates

Usage

```
is_admissible(object)
```

```
## S4 method for signature 'PlsModel'  
is_admissible(object)
```

Arguments

object A fitted PlsModel object.

Value

A single logical value.

is_mcpls	<i>Check whether an object uses the MC-OrdPLSc estimator</i>
----------	--

Description

Check whether an object uses the MC-OrdPLSc estimator

Usage

```
is_mcpls(object)
```

```
## S4 method for signature 'PlsModel'  
is_mcpls(object)
```

Arguments

object A fitted model object.

Value

TRUE or FALSE.

mcpls_loglik	<i>Loglikelihood of MC-PLS parameters</i>
--------------	---

Description

Compute Loglikelihood statistics for an MC-PLS model.

Usage

```
mcpls_loglik(object, boot.R = 500, verbose = interactive(), ...)

## S4 method for signature 'PlsModel'
mcpls_loglik(object, boot.R = 500, verbose = interactive(), ...)
```

Arguments

- object A fitted [PlsModel] object, using MC-PLS.
- boot.R Integer; Number of bootstrap replicates.
- verbose Logical; Should a progressbar be displayed?
- ... Reserved for future extensions.

Value

List with loglikelihood value, expected and observed auxiliary parameters, and variance covariance matrix of the expected auxiliary parameters.

oneIntOrdered	<i>oneIntOrdered</i>
---------------	----------------------

Description

A simulated dataset.

Examples

```
m <- '
  X =~ x1 + x2 + x3
  Z =~ z1 + z2 + z3
  Y =~ y1 + y2 + y3

  Y ~ X + Z + X:Z
'

fit <- pls(m, oneIntOrdered)
summary(fit)
```

parameter_estimates	<i>Generic accessor for model parameter estimates</i>
---------------------	---

Description

Generic accessor for model parameter estimates

Usage

```
parameter_estimates(object, ...)
```

Arguments

object	A fitted model object.
...	Additional arguments passed to methods.

Value

A parameter table describing the fitted model.

parameter_estimates, PlsModel-method
<i>Parameter estimates for PlsModel objects</i>

Description

Parameter estimates for PlsModel objects

Usage

```
## S4 method for signature 'PlsModel'
parameter_estimates(object, colon.pi = TRUE, label.renamed.prod = FALSE, ...)
```

Arguments

object	A PlsModel object.
colon.pi	Logical; replace product-indicator labels with colon notation (X:Z).
label.renamed.prod	Logical; retain renamed product labels when colon expansion occurs.
...	Currently unused.

Value

A PlsSemParTable data frame.

Description

`pls()` estimates Partial Least Squares Structural Equation Models (PLS-SEM) and their consistent (PLSc) variants. The function accepts lavaan-style syntax, handles ordered indicators through polychoric correlations and probit factor scores, and supports multilevel specifications expressed with lme4-style random effects terms inside the structural model.

Usage

```
pls(
  syntax,
  data,
  standardize = TRUE,
  consistent = TRUE,
  bootstrap = FALSE,
  ordered = NULL,
  missing = c("listwise", "mean", "kNN"),
  knn.k = 5,
  mcpls = NULL,
  mc.fast.lmer = mcpls,
  probit = NULL,
  tolerance = 1e-05,
  max.iter.0_5 = 500L,
  boot.ncores = 1L,
  boot.ncpus = NULL,
  boot.parallel = c("no", "multicore", "multisession", "snow"),
  boot.R = 500L,
  boot.iseed = NULL,
  sample = NULL,
  mc.min.iter = 50L,
  mc.max.iter = 1000L,
  mc.reps = 20000L,
  mc.fixed.seed = FALSE,
  mc.polyak.juditsky = TRUE,
  mc.pj.extrapolate = TRUE,
  mc.tol = if (mc.polyak.juditsky) 1e-04 else 0.001,
  mc.delta.se = TRUE,
  mc.delta.jacobian.k = max(floor(boot.R/100L), 1),
  mc.fn.args = list(),
  mc.rescov = c("auto", "reduced", "full"),
  mc.diag.secant = FALSE,
  mc.small.sample = FALSE,
  mc.small.sample.max.k = 100L,
  mc.small.sample.point.estimate = c("median", "mean"),
```

```

verbose = interactive(),
boot.optimize = TRUE,
boot.drop.inadmissible = FALSE,
mc.boot.control = list(min.iter = mc.min.iter, max.iter = mc.max.iter, mc.reps =
  floor(0.5 * mc.reps), tol = mc.tol, polyak.juditsky = mc.polyak.juditsky,
  pj.extrapolate = FALSE, verbose = FALSE, fixed.seed = TRUE, reuse.p.start = TRUE),
reliabilities = NULL,
default.path.estimator = c("ols", "gls"),
...
)

```

Arguments

syntax	Character string with lavaan-style model syntax describing both measurement (=~) and structural (~) relations. Random effects are specified with (term cluster) statements.
data	A data.frame or coercible object containing the manifest indicators referenced in syntax. Ordered factors are automatically detected, but can also be supplied explicitly through ordered.
standardize	Logical; if TRUE, indicators are standardized before estimation so that factor scores have comparable scales.
consistent	Logical; TRUE requests PLSc corrections, whereas FALSE fits the traditional PLS model.
bootstrap	Logical; if TRUE, nonparametric bootstrap standard errors are computed with boot.R resamples.
ordered	Optional character vector naming manifest indicators that should be treated as ordered when computing polychoric correlations.
missing	Character string specifying how to handle missing indicator data. "listwise" removes rows with missing values (listwise deletion). "mean" imputes missing indicator values using simple univariate imputation: the mean for continuous variables, the median for ordered variables with more than two categories, and the mode for binary ordered variables (two categories) or nominal variables. "kNN" (or "knn") imputes missing indicator values using k-nearest neighbors imputation (kNN). When missing = "kNN", rows with all indicators missing are removed prior to imputation, and rows with missing cluster values are removed for multilevel models.
knn.k	Integer specifying the number of neighbors (k) used when missing = "kNN".
mcpls	Should the model be estimated using the Monte-Carlo Consistent Partial Least Squares (MC-PLSc) algorithm?
mc.fast.lmer	Should a faster (biased) GLS based estimator of the Mixed-Effects model be used in conjunction with the MC-PLS algorithm?
probit	Logical; overrides the automatic choice of probit factor scores that is based on whether ordered indicators are present.
tolerance	Numeric; Convergence criteria/tolerance.
max.iter.0_5	Maximum number of PLS iterations performed when estimating the measurement and structural models.

<code>boot.ncores</code>	Integer: number of workers to be used for parallel bootstrapping. Parallel bootstrapping is enabled when <code>boot.ncores > 1</code> .
<code>boot.ncpus</code>	Deprecated alias for <code>boot.ncores</code> .
<code>boot.parallel</code>	The type of parallel operation to be used (if any). The default is "no". "multisession" runs the bootstrap using multiple background R sessions (works on all platforms), while "multicore" uses forked processes (not available on Windows). "snow" is kept for backwards compatibility and is treated as an alias for "multisession". Internally this is implemented using the future package
<code>boot.R</code>	Integer giving the number of bootstrap resamples drawn when <code>bootstrap = TRUE</code> .
<code>boot.iseed</code>	An integer to set the bootstrap seed. Or NULL if no reproducible results are needed. This works for both serial and parallel settings. When <code>boot.iseed</code> is not NULL, <code>.Random.seed</code> (if it exists) in the global environment is left untouched.
<code>sample</code>	DEPRECATED. Integer giving the number of bootstrap resamples drawn when <code>bootstrap = TRUE</code> .
<code>mc.min.iter</code>	Minimum number of iterations in MC-PLS algorithm.
<code>mc.max.iter</code>	Maximum number of iterations in MC-PLS algorithm.
<code>mc.reps</code>	Monte-Carlo sample size in MC-PLS algorithm.
<code>mc.fixed.seed</code>	Should a fixed seed be used in the MC-PLS algorithm? Setting a fixed seed will likely yield less accurate estimates, but can substantially improve the stability and computational efficiency of the algorithm.
<code>mc.polyak.juditsky</code>	Should the polyak.juditsky running average method be applied in the MC-PLS algorithm?
<code>mc.pj.extrapolate</code>	Logical; if TRUE (the default), the Polyak-Juditsky convergence point is estimated via NLS exponential extrapolation (with Aitken δ^2 as a fallback). If FALSE a warm start is performed instead, and the plain Polyak-Juditsky average is used. Only relevant when <code>mc.polyak.juditsky = TRUE</code> .
<code>mc.tol</code>	Tolerance in MC-PLS algorithm.
<code>mc.delta.se</code>	Should delta-method standard errors be computed for MC-PLS estimates?
<code>mc.delta.jacobian.k</code>	Integer number of Monte-Carlo Jacobians to average when computing delta-method standard errors. Defaults to one per 100 bootstrap resamples, with a minimum of 1.
<code>mc.fn.args</code>	Additional arguments to MC-PLS algorithm, mainly for controlling the step size.
<code>mc.rescov</code>	How residual covariances are treated in MC-PLS. One of "auto" (the default), "reduced", or "full". In "reduced" mode residual covariances are not treated as free parameters; they are identified from the rest of the structural model and the disturbances are simulated independently. In "full" mode the endogenous residual covariances ($\eta \sim \eta$ and $\xi \sim \eta$) are free parameters, explicitly simulated. "auto" uses "full" when the structural model is estimated by GLS (i.e. the model contains residual covariances) and "reduced" otherwise.

<code>mc.diag.secant</code>	Logical; if TRUE, the MC-PLS root-finding algorithm uses a per-coordinate diagonal-secant step (estimating each coordinate's local slope from consecutive iterates), instead of the standard Robbins-Monro step. This might be useful if the binding function is non-monotone.
<code>mc.small.sample</code>	Logical; if TRUE, average MC-PLS estimating equations over simulated samples of the observed sample size. If FALSE (default), fit one sample using all simulated observations. This assumes that the auxiliary estimator (i.e., traditional PLS) has negligible finite sample bias. I.e., that the bias of the estimator is not affected by the sample size.
<code>mc.small.sample.max.k</code>	Maximum number of simulated samples to average when <code>mc.small.sample</code> = TRUE. Defaults to 100. The number of samples is also limited by <code>mc.reps</code> , rounded down to a multiple of the observed sample size, with at least one sample.
<code>mc.small.sample.point.estimate</code>	Which point estimate of the simulated auxiliary parameters the root equation matches to the observed ones, when <code>mc.small.sample</code> = TRUE? "mean" solves $E[\theta^* \theta] = \hat{\theta}^*$. "median" (the default) solves $median[\theta^* \theta] = \hat{\theta}^*$. The median commutes with the (monotone) binding function where the mean does not, so median-matching targets a median-unbiased estimator. This removes the finite-sample bias, which can be introduced by the curvature size models, and when the indicators are uninformative (small loadings, few categories, strongly asymmetric thresholds). It makes the estimating function somewhat noisier for a given number of simulated samples. Ignored when <code>mc.small.sample</code> = FALSE.
<code>verbose</code>	Should verbose output be printed?
<code>boot.optimize</code>	Logical; if TRUE and <code>bootstrap</code> = TRUE, applies the settings in <code>mc.boot.control</code> inside each bootstrap replicate (MC-PLS only). In general it will lead to slightly larger and less accurate standard errors.
<code>boot.drop.inadmissible</code>	Logical; if TRUE and <code>bootstrap</code> = TRUE, bootstrap replicates that yield an inadmissible solution (e.g. a Heywood case) are discarded before computing standard errors, just like replicates where the estimation procedure fails outright. Defaults to FALSE, which keeps inadmissible replicates. In either case the number of inadmissible replicates is reported. Note that dropping inadmissible solutions conditions the bootstrap distribution on well-behaved resamples and may bias the standard errors downward.
<code>mc.boot.control</code>	List of control parameters passed to the MC-PLS algorithm inside each bootstrap replicate when <code>boot.optimize</code> = TRUE.
<code>reliabilities</code>	Optional named numeric vector of user-supplied reliabilities used for the PLS consistency correction.
<code>default.path.estimator</code>	Character string selecting the estimator used for the structural (path) model when the model does not require Generalized Least Squares (GLS). The default

"ols" uses Ordinary Least Squares whenever possible, falling back to GLS automatically when the model contains residual covariances. Setting `default.path.estimator = "gls"` forces GLS estimation of the structural model even when OLS would otherwise be used.

... Internal arguments. For advanced users only.

Value

A Plssem object containing the estimated parameters, fit measures, factor scores, and any bootstrap results. Methods such as `summary()`, `coef()`, and `parameter_estimates()` can be applied to inspect the fit.

See Also

[summary](#), [show](#)

Examples

```
library(plssem)
library(modsem)

tpb <- '
  ATT =~ att1 + att2 + att3 + att4 + att5
  SN =~ sn1 + sn2
  PBC =~ pbc1 + pbc2 + pbc3
  INT =~ int1 + int2 + int3
  BEH =~ b1 + b2
  INT ~ ATT + SN + PBC
  BEH ~ INT + PBC
'

fit <- pls(tpb, TPB, bootstrap = TRUE)
summary(fit)
```

pls_boot	<i>Retrieve bootstrap coefficient matrix</i>
----------	--

Description

Retrieve bootstrap coefficient matrix

Usage

```
pls_boot(object)
```

Arguments

object A fitted model object.

Value

A PlsSemMatrix of bootstrap replicate parameter vectors (rows = replicates, cols = parameters).

Examples

```
library(modsem)
library(plssem)

m <- "
  X =~ x1 + x2 + x3
  Z =~ z1 + z2 + z3
  Y =~ y1 + y2 + y3
  Y ~ X + Z + X:Z
"

fit <- pls(m, oneInt, bootstrap = TRUE, boot.R = 50)
pls_boot(fit)
```

pls_chisq

Chi-Square

Description

Compute Chi-Square value for a PLS model.

Usage

```
pls_chisq(object, saturated = FALSE, mc.reps = 1e+06, ...)

## S4 method for signature 'PlsModel'
pls_chisq(object, saturated = FALSE, mc.reps = 1e+06, ...)

## S4 method for signature 'PlsModel'
pls_chisq_df(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

object	A fitted [PlsModel] object.
saturated	Logical; if 'TRUE', compute the saturated fit.
mc.reps	Integer; number of Monte Carlo resamples used for MC-PLSc fit.
...	Reserved for future extensions.

Value

Chi-Square value

pls_chisq_df	<i>Chi-Square Degrees of Freedom</i>
--------------	--------------------------------------

Description

Compute Chi-Square degrees of freedom for a PLS model.

Usage

```
pls_chisq_df(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

object	A fitted [PlsModel] object.
saturated	Logical; if 'TRUE', compute the saturated fit.
mc.reps	Integer; number of Monte Carlo resamples used for MC-PLSc fit.
...	Reserved for future extensions.

Value

Chi-Square degrees of freedom

pls_construct_scores	<i>Construct latent variable scores</i>
----------------------	---

Description

Convenience wrapper around [pls_predict()] returning only the predicted latent scores matrix.

Usage

```
pls_construct_scores(object, ...)
```

Arguments

object	A fitted Plssem model.
...	Passed to [pls_predict()].

Value

A PlsSemMatrix of predicted latent scores.

`pls_implied_construct_corr`*Implied Construct Correlation Matrix*

Description

Returns the implied construct correlation matrix for a fitted model.

Usage

```
pls_implied_construct_corr(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

```
## S4 method for signature 'PlsModel'
```

```
pls_implied_construct_corr(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

<code>object</code>	A fitted [PlsModel] object.
<code>saturated</code>	Logical; if 'TRUE', return the saturated implied matrix.
<code>mc.reps</code>	Integer; number of Monte Carlo resamples used for MC-PLSc.
<code>...</code>	Reserved for future extensions.

Details

For higher-order models, this is computed for the combined model returned by [combinedModel()].

Value

A [PlsSemMatrix].

`pls_implied_indicator_corr`*Implied Indicator Correlation Matrix*

Description

Returns the implied indicator correlation matrix for a fitted model.

Usage

```
pls_implied_indicator_corr(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

```
## S4 method for signature 'PlsModel'
```

```
pls_implied_indicator_corr(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

object	A fitted [PlsModel] object.
saturated	Logical; if 'TRUE', return the saturated implied matrix.
mc.reps	Integer; number of Monte Carlo resamples used for MC-PLSc.
...	Reserved for future extensions.

Details

For higher-order models, this is computed for the combined model returned by [combinedModel()].

Value

A numeric matrix.

pls_implied_joint_corr

Implied Joint Correlation Matrix

Description

Returns the joint implied correlation matrix of the observed and latent variables for a fitted model.

Usage

```
pls_implied_joint_corr(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

```
## S4 method for signature 'PlsModel'
```

```
pls_implied_joint_corr(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

object	A fitted [PlsModel] object.
saturated	Logical; if 'TRUE', return the saturated implied matrix.
mc.reps	Integer; number of Monte Carlo resamples used for MC-PLSc.
...	Reserved for future extensions.

Details

For higher-order models, this is computed for the combined model returned by [combinedModel()].

Value

A [PlsSemMatrix].

pls_inspect

*Inspect a fitted PLS-SEM model***Description**

Extract important information from a fitted `PlsModel`. The interface is modelled after `lavaan::lavInspect()`: a single `what` argument selects which piece of information to return.

Usage

```
pls_inspect(object, what = "estimates", ...)
```

```
## S4 method for signature 'PlsModel'
pls_inspect(object, what = "estimates", ...)
```

Arguments

object	A fitted <code>PlsModel</code> object.
what	A single string selecting what to extract (case-insensitive); defaults to "estimates". Several values accept aliases, given in parentheses. One of: "estimates" (aliases "est", "x", "matrices") A list of the estimated model matrices in lavaan-style representation (lambda, wmat, theta, psi, C, gamma). "lambda", "wmat", "theta", "psi", "C", "gamma" The corresponding single matrix from the "estimates" list. "coef" (alias "coefficients") The model coefficients. "par" (alias "partable") The parameter table. "fit" Fit measures. "mcpls.history" (alias "history") The history of the MC-PLS estimates. "chisq" The model chi-square statistic. "chisq.df" (alias "df") The chi-square degrees of freedom. "srmr" The standardized root mean square residual. "rmsea" The root mean square error of approximation. "se" The standard errors of the estimates. "vcov" The variance-covariance matrix of the estimates. "boot" The bootstrap results. "info" A list with the number of observations (nobs), the number of latent variables (nlv), the number of observed variables (nov), and the estimation modes ("A"/"B") for each latent variable. "status" A list with the number of iterations, whether the algorithm converged, and whether the solution is admissible. "qualities" The construct qualities (Q^2). "reliabilities" (alias "rel") The construct reliabilities. "cov.lv" The model-implied covariance matrix of the latent variables.

```

"cov.ov" The model-implied covariance matrix of the observed variables.
"cov.all" The joint model-implied covariance matrix of the observed and la-
tent variables.
"r2.lv", "r2.ov", "r2.all" The model-implied  $R^2$  for the latent variables,
the observed variables, or both.
"data" The (standardized) data matrix used for estimation.
... Currently ignored.

```

Value

The requested information; the type depends on what (see above).

Examples

```

## Not run:
fit <- pls(model, data = data)
pls_inspect(fit, "info")
pls_inspect(fit, "cov.lv")

## End(Not run)

```

pls_predict	<i>Predict from a fitted PLS-SEM model</i>
-------------	--

Description

Predict from a fitted PLS-SEM model

Usage

```

pls_predict(object, ...)

## S4 method for signature 'PlsModel'
pls_predict(
  object,
  approach = c("earliest", "direct"),
  newdata = NULL,
  std.ord.exp = FALSE,
  benchmark = "R2",
  benchmark.vars = c("endog", "exog", "all"),
  ...
)

```

Arguments

object	A fitted <code>PlsModel</code> object.
...	Additional arguments passed to internal helpers.
approach	Prediction approach. If <code>approach = "earliest"</code> (default), then only indicators of exogenous <code>benchmark.vars</code> are used for prediction. If <code>approach = "direct"</code> , then all indicators are used.
newdata	Optional new data matrix/data frame.
std.ord.exp	Logical; standardize ordinal expectation scores.
benchmark	Benchmark type(s). Either length 1 (recycled) or one entry per indicator (optionally named). Supported: <code>"r2"</code> , <code>"rmse"</code> , <code>"mae"</code> , <code>"q2_predict"</code> , <code>"acc"</code> , <code>"ord_mae"</code> .
benchmark.vars	What predictions should be benchmarked? If <code>benchmark.vars = "endog"</code> (default), prediction benchmarks are applied to indicators of endogenous <code>benchmark.vars</code> . If <code>benchmark.vars = "exog"</code> , prediction benchmarks are applied to indicators of exogenous <code>benchmark.vars</code> . If <code>benchmark.vars = "all"</code> , prediction benchmarks are applied to all of the indicators in the model.

Value

A `PlsSemPredict` object with matrices and benchmark results.

<code>pls_rmsea</code>	<i>RMSEA</i>
------------------------	--------------

Description

Compute RMSEA for a PLS model.

Usage

```
pls_rmsea(object, saturated = FALSE, mc.reps = 1e+06, ...)

## S4 method for signature 'PlsModel'
pls_rmsea(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

object	A fitted [<code>PlsModel</code>] object.
saturated	Logical; if <code>'TRUE'</code> , compute the saturated fit.
mc.reps	Integer; number of Monte Carlo resamples used for MC-PLSc fit.
...	Reserved for future extensions.

Value

RMSEA value

pls_srmr	<i>SRMR</i>
----------	-------------

Description

Compute SRMR for a PLS model.

Usage

```
pls_srmr(object, saturated = FALSE, mc.reps = 1e+06, ...)

## S4 method for signature 'PlsModel'
pls_srmr(object, saturated = FALSE, mc.reps = 1e+06, ...)
```

Arguments

object	A fitted [PlsModel] object.
saturated	Logical; if 'TRUE', compute the saturated fit.
mc.reps	Integer; number of Monte Carlo resamples used for MC-PLSc fit.
...	Reserved for future extensions.

Value

SRMR value

predict,PlsModel-method	<i>Predict from a fitted PlsModel (alias for pls_predict)</i>
-------------------------	---

Description

Predict from a fitted PlsModel (alias for [pls_predict](#))

Usage

```
## S4 method for signature 'PlsModel'
predict(object, newdata = NULL, ...)
```

Arguments

object	A fitted PlsModel object.
newdata	Optional new data matrix/data frame.
...	Further arguments passed to pls_predict .

Value

A PlsSemPredict object.

print.PlsSemPredict	<i>Print a PlsSemPredict object</i>
---------------------	-------------------------------------

Description

Print a PlsSemPredict object

Usage

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

Arguments

x	A PlsSemPredict object.
...	Additional arguments for compatibility with the generic.

Value

The input object, invisibly.

print.SummaryPlsSem	<i>Print a SummaryPlsSem object</i>
---------------------	-------------------------------------

Description

Print a SummaryPlsSem object

Usage

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

Arguments

x	A SummaryPlsSem object as returned by summary() .
...	Additional arguments for compatibility with the generic.

Value

The input object, invisibly.

randomIntercepts	<i>randomIntercepts</i>
------------------	-------------------------

Description

A simulated dataset.

Examples

```
syntax <- '  
  f =~ y1 + y2 + y3  
  f ~ x1 + x2 + x3 + w1 + w2 + (1 | cluster)  
'  
  
fit <- pls(syntax, data = randomIntercepts)  
summary(fit)
```

randomInterceptsOrdered	<i>randomInterceptsOrdered</i>
-------------------------	--------------------------------

Description

A simulated dataset.

Examples

```
syntax <- '  
  f =~ y1 + y2 + y3  
  f ~ x1 + x2 + x3 + w1 + w2 + (1 | cluster)  
'  
  
fit <- pls(syntax, data = randomInterceptsOrdered)  
summary(fit)
```

randomSlopes	<i>randomSlopes</i>
--------------	---------------------

Description

A simulated dataset.

Examples

```
syntax <- "  
  X =~ x1 + x2 + x3  
  Z =~ z1 + z2 + z3  
  Y =~ y1 + y2 + y3  
  W =~ w1 + w2 + w3  
  Y ~ X + Z + (1 + X + Z | cluster)  
  W ~ X + Z + (1 + X + Z | cluster)  
"  
  
fit <- pls(syntax, data = randomSlopes)  
fit
```

randomSlopesOrdered	<i>randomSlopesOrdered</i>
---------------------	----------------------------

Description

A simulated dataset.

Examples

```
syntax <- "  
  X =~ x1 + x2 + x3  
  Z =~ z1 + z2 + z3  
  Y =~ y1 + y2 + y3  
  W =~ w1 + w2 + w3  
  Y ~ X + Z + (1 + X + Z | cluster)  
  W ~ X + Z + (1 + X + Z | cluster)  
"  
  
fit <- pls(syntax, data = randomSlopesOrdered)  
fit  
summary(fit)
```

show,PlsModel-method *Show a PlsModel object*

Description

Called automatically when an object is printed at the prompt. Displays the package version, iteration count, and the parameter table.

Usage

```
## S4 method for signature 'PlsModel'
show(object)
```

Arguments

object A PlsModel object.

Value

object, invisibly.

summary,PlsModel-method
 Summarize a fitted PlsModel model

Description

Summarize a fitted PlsModel model

Usage

```
## S4 method for signature 'PlsModel'
summary(object, fit = TRUE, unstandardized = FALSE, ci = FALSE, ...)
```

Arguments

object A PlsModel object.
fit Logical; Whether to compute fit measures.
unstandardized Logical; Should unstandardized estimates be included?
ci Logical; Should confidence intervals for parameter estimates be included?
... Arguments passes to [unstandardized_estimates](#).

Value

A SummaryPlsSem list with formatted results.

titanic*Titanic Passenger Survival Data Set.*

Description

This dataset has been re-packaged for convenience from <https://github.com/paulhendricks/titanic>

PassengerId Passenger ID

Survived Passenger Survival Indicator

Pclass Passenger Class

Name Name

Sex Sex

Age Age

SibSp Number of Siblings/Spouses Aboard

Parch Number of Parents/Children Aboard

Ticket Ticket Number

Fare Passenger Fare

Cabin Cabin

Embarked Port of Embarkation

Female Dummy variable for Sex="female"

Format

A data frame with 1309 rows and 12 variables:

Source

<https://www.kaggle.com/c/titanic/data>

Examples

```
fit <- pls("Survived ~ Age + Female + Age:Female",  
          data = titanic, ordered = "Survived")  
pls_predict(fit, benchmark = "acc")
```

TPB_Ordered	<i>TPB_Ordered</i>
Description	
A simulated dataset.	
Examples	
<pre>tpb <- ' # Outer Model (Based on Hagger et al., 2007) ATT =~ att1 + att2 + att3 + att4 + att5 SN =~ sn1 + sn2 PBC =~ pbc1 + pbc2 + pbc3 INT =~ int1 + int2 + int3 BEH =~ b1 + b2 # Inner Model (Based on Steinmetz et al., 2011) INT ~ ATT + SN + PBC BEH ~ INT + PBC , fit <- pls(tpb, TPB_Ordered) summary(fit)</pre>	
unstandardized_estimates	<i>Unstandardized Parameter Estimates</i>

Description

Transform parameter estimates from a fitted PLS-SEM model to observed- and latent-variable scales. Variables not selected through unstandardized remain on their standardized scales.

Usage

```
unstandardized_estimates(  
  model,  
  unstandardized = "all",  
  se = c("delta", "none"),  
  scale.uncertainty = FALSE,  
  eps = 1e-04,  
  zero.tol = 1e-10,  
  rm.tmp.ov = TRUE,  
  clean.tmp.ind = TRUE,  
  clean.tmp.mimic = TRUE
```

```

)

## S4 method for signature 'PlsModel'
unstandardized_estimates(
  model,
  unstandardized = "all",
  se = c("delta", "none"),
  scale.uncertainty = FALSE,
  eps = 1e-04,
  zero.tol = 1e-10,
  rm.tmp.ov = TRUE,
  clean.tmp.ind = TRUE,
  clean.tmp.mimic = TRUE
)

```

Arguments

<code>model</code>	A fitted <code>PlsModel</code> object.
<code>unstandardized</code>	Character vector naming variables to unstandardize, or one of "all", "ov", or "lv".
<code>se</code>	Character string selecting delta-method standard errors ("delta") or no standard errors ("none").
<code>scale.uncertainty</code>	Should scale uncertainty be included? defaults to FALSE.
<code>eps</code>	Positive numeric finite-difference step used for the delta-method Jacobian.
<code>zero.tol</code>	Non-negative numeric tolerance below which standard errors are returned as missing.
<code>rm.tmp.ov</code>	Logical; whether rows involving temporary observed variables should be removed from the returned parameter table.
<code>clean.tmp.ind</code>	Logical; whether rows involving temporary indicators should be cleaned from the returned parameter table.
<code>clean.tmp.mimic</code>	Logical; whether rows involving temporary mimic indicators should be cleaned from the returned parameter table.

Value

A `PlsSemParTable` containing transformed estimates and (when requested) delta-method standard errors. The transformed covariance matrix is stored in the "vcov" attribute.

Examples

```

tpb <- '
# Outer Model (Based on Hagger et al., 2007)
ATT <~ att1 + att2 + att3 + att4 + att5
SN =~ sn1 + sn2
PBC =~ pbc1 + pbc2 + pbc3

```

```

    INT =~ int1 + int2 + int3
    BEH <~ b1 + b2

# Inner Model (Based on Steinmetz et al., 2011)
    INT ~ ATT + SN + PBC
    BEH ~ INT + PBC + INT:PBC
    ,

fit <- pls(tpb, modsem::TPB, bootstrap = TRUE, boot.R = 50)
unstandardized_estimates(fit)

```

vcov,PlsModel-method *Extract the variance-covariance matrix from a PlsModel model*

Description

Extract the variance-covariance matrix from a PlsModel model

Usage

```
## S4 method for signature 'PlsModel'
vcov(object, use.labels = TRUE, ...)
```

Arguments

object	A PlsModel object.
use.labels	Logical; Should parameter labels be used as names?
...	Currently unused.

Value

A PlsSemMatrix (bootstrap-based vcov, or NULL).

Index

boot, [3](#)
boot, PlsModel-method (boot), [3](#)

coef, PlsModel-method, [4](#)
coefficients, PlsModel-method
 (coef, PlsModel-method), [4](#)

fit_measures, [4](#)
fit_measures, PlsModel-method
 (fit_measures), [4](#)

is_admissible, [5](#)
is_admissible, PlsModel-method
 (is_admissible), [5](#)
is_mcpls, [5](#)
is_mcpls, PlsModel-method (is_mcpls), [5](#)

mcpls_loglik, [6](#)
mcpls_loglik, PlsModel-method
 (mcpls_loglik), [6](#)

oneIntOrdered, [6](#)

parameter_estimates, [7](#)
parameter_estimates, PlsModel-method, [7](#)
pls, [8](#)
pls_boot, [12](#)
pls_boot, PlsModel-method (boot), [3](#)
pls_chisq, [13](#)
pls_chisq, PlsModel-method (pls_chisq),
 [13](#)
pls_chisq_df, [14](#)
pls_chisq_df, PlsModel-method
 (pls_chisq), [13](#)
pls_construct_scores, [14](#)
pls_implied_construct_corr, [15](#)
pls_implied_construct_corr, PlsModel-method
 (pls_implied_construct_corr),
 [15](#)
pls_implied_indicator_corr, [15](#)

pls_implied_indicator_corr, PlsModel-method
 (pls_implied_indicator_corr),
 [15](#)
pls_implied_joint_corr, [16](#)
pls_implied_joint_corr, PlsModel-method
 (pls_implied_joint_corr), [16](#)
pls_inspect, [17](#)
pls_inspect, PlsModel-method
 (pls_inspect), [17](#)
pls_predict, [18](#), [20](#)
pls_predict, PlsModel-method
 (pls_predict), [18](#)
pls_rmsea, [19](#)
pls_rmsea, PlsModel-method (pls_rmsea),
 [19](#)
pls_srmr, [20](#)
pls_srmr, PlsModel-method (pls_srmr), [20](#)
predict, PlsModel-method, [20](#)
print.PlsSemPredict, [21](#)
print.SummaryPlsSem, [21](#)

randomIntercepts, [22](#)
randomInterceptsOrdered, [22](#)
randomSlopes, [23](#)
randomSlopesOrdered, [23](#)

show, [12](#)
show, PlsModel-method, [24](#)
summary, [12](#), [21](#)
summary, PlsModel-method, [24](#)

titanic, [25](#)
TPB_Ordered, [26](#)

unstandardized_estimates, [24](#), [26](#)
unstandardized_estimates, PlsModel-method
 (unstandardized_estimates), [26](#)

vcov, PlsModel-method, [28](#)