

Package ‘plssem’

April 25, 2026

Type Package

Title Complex Partial Least Squares Structural Equation Modeling

Version 0.1.1

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

RoxygenNote 7.3.3

Imports stats, modsem (>= 1.0.17), lme4, lavaan, stringr, purrr, matrixStats, Rfast, collapse, mvnfast, reformulas, parallel, FNN

Depends R (>= 4.1.0)

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

Suggests knitr, rmarkdown, mice, mvtnorm

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-04-25 16:50:02 UTC

Contents

boot	2
isMC_PLS	3
oneIntOrdered	4
parameter_estimates	4
parameter_estimates.plssem	5
pls	5
pls_construct_scores	10
pls_predict	11
print.plssem	12
print.PlsSemPredict	12
print.SummaryPlsSem	13
randomIntercepts	13
randomInterceptsOrdered	14
randomSlopes	14
randomSlopesOrdered	14
summary.plssem	15
titanic	15
TPB_Ordered	16
Index	17

boot

Get bootstrapped coefficients from PLS model

Description

Get bootstrapped coefficients from PLS model

Usage

```
boot(object)
```

Arguments

```
object          A fitted model object.
```

Value

Matrix with bootstrapped coefficients.

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)
```

isMC_PLS

Check if object is a MC-PLS model

Description

Check if object is a MC-PLS model

Usage

```
isMC_PLS(object)
```

Arguments

```
object          A fitted model object.
```

Value

TRUE/FALSE.

oneIntOrdered

oneIntOrdered

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*Generic accessor for model parameter estimates*

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.plssem
```

Parameter estimates for plssem objects

Description

Parameter estimates for plssem objects

Usage

```
## S3 method for class 'plssem'
parameter_estimates(object, colon.pi = TRUE, label.renamed.prod = FALSE, ...)
```

Arguments

object	An object of class plssem.
colon.pi	Logical; whether to replace labels for interaction terms with colon notation.
label.renamed.prod	Logical; whether renamed product labels should be retained when colon expansion occurs.
...	Additional arguments (not used).

Value

A parameter table (data frame) describing the fitted model.

```
pls
```

Fit Partial Least Squares Structural Equation Models

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,
probit = NULL,
tolerance = 1e-05,
max.iter.0_5 = 100L,
boot.ncpus = 1L,
boot.parallel = c("no", "multicore", "snow"),
boot.R = 50L,
boot.iseed = NULL,
sample = NULL,
mc.min.iter = 5L,
mc.max.iter = 250L,
mc.reps = 20000L,
mc.tol = 0.001,
mc.fixed.seed = FALSE,
mc.polyak.juditsky = FALSE,
mc.fn.args = list(),
verbose = interactive(),
boot.optimize = TRUE,
mc.boot.control = list(min.iter = mc.min.iter, max.iter = mc.max.iter, mc.reps =
  floor(0.5 * mc.reps), tol = 2 * mc.tol, polyak.juditsky = FALSE, verbose = FALSE,
  fixed.seed = TRUE, reuse.p.start = TRUE),
reliabilities = NULL,
...
)

```

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.

<code>knn.k</code>	Integer specifying the number of neighbors (k) used when <code>missing = "kNN"</code> .
<code>mcpls</code>	Should a Monte-Carlo consistency correction be applied?
<code>probit</code>	Logical; overrides the automatic choice of probit factor scores that is based on whether ordered indicators are present.
<code>tolerance</code>	Numeric; Convergence criteria/tolerance.
<code>max.iter.0_5</code>	Maximum number of PLS iterations performed when estimating the measurement and structural models.
<code>boot.ncpus</code>	Integer: number of processes to be used in parallel operation. By default this is the number of cores (as detected by <code>parallel::detectCores()</code>) minus one.
<code>boot.parallel</code>	The type of parallel operation to be used (if any). If missing, the default is "no".
<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 (non-parallel) and parallel settings. Internally, <code>RNGkind()</code> is set to "L'Ecuyer-CMRG" if <code>parallel = "multicore"</code> . If <code>parallel = "snow"</code> (under windows), <code>parallel::clusterSetRNGStream()</code> is called which automatically switches to "L'Ecuyer-CMRG". When <code>iseed</code> is not NULL, <code>.Random.seed</code> (if it exists) in the global environment is left untouched.
<code>sample</code>	DEPRECTATED. 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.tol</code>	Tolerance in MC-PLS algorithm.
<code>mc.fixed.seed</code>	Should a fixed seed be used in the MC-PLS algorithm?
<code>mc.polyak.juditsky</code>	Should the polyak.juditsky running average method be applied in the MC-PLS algorithm?
<code>mc.fn.args</code>	Additional arguments to MC-PLS algorithm, mainly for controlling the step size.
<code>verbose</code>	Should verbose output be printed?
<code>boot.optimize</code>	Logical; if TRUE and <code>bootstrap = TRUE</code> , applies the settings in <code>mc.boot.control</code> inside each bootstrap replicate (MC-PLS only).
<code>mc.boot.control</code>	List of control parameters passed to the MC-PLS algorithm inside each bootstrap replicate when <code>boot.optimize = TRUE</code> . This can be used to speed up bootstrapping by, for example, increasing the tolerance or reducing <code>mc.reps</code> . The element <code>reuse.p.start</code> controls whether to reuse the original <code>p.start</code> for the bootstrap replicates.

`reliabilities` Optional named numeric vector of user-supplied reliabilities used for the PLS consistency correction. Values are interpreted as construct reliabilities (i.e., squared construct quality, Q^2) for the named constructs. These override the internally computed construct qualities.

`...` Internal arguments. For advanced users only.

Value

An object of class `plssem` containing the estimated parameters, fit measures, factor scores, and any bootstrap results. Methods such as `summary()`, `print()`, and `coef()` can be applied to inspect the fit.

See Also

[`summary.plssem()`], [`print.plssem()`]

Examples

```
# Linear Model with Continuous Data

library(plssem)
library(modsem)

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, bootstrap = TRUE)
summary(fit)

# Linear Model with Ordered Data
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, bootstrap = TRUE)
summary(fit)

# Multilevel Random Slopes Model with Continuous Data
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, bootstrap = TRUE)
summary(fit)

# Multilevel Random Slopes Model with Ordered Data
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, bootstrap = TRUE)
summary(fit)

# Multilevel Random Intercepts Model with Continuous Data
syntax <- '
  f =~ y1 + y2 + y3
  f ~ x1 + x2 + x3 + w1 + w2 + (1 | cluster)
'

fit <- pls(syntax, data = randomIntercepts, bootstrap = TRUE)
summary(fit)

# Multilevel Random Intercepts Model with Ordered Data
syntax <- '
  f =~ y1 + y2 + y3
  f ~ x1 + x2 + x3 + w1 + w2 + (1 | cluster)
'

fit <- pls(syntax, data = randomInterceptsOrdered, bootstrap = TRUE)
summary(fit)

# Interaction Model with Continuous Data
m <- '
  X =~ x1 + x2 + x3
  Z =~ z1 + z2 + z3

```

```
Y =~ y1 + y2 + y3

Y ~ X + Z + X:Z
,

fit <- pls(m, modsem::oneInt, bootstrap = TRUE)
summary(fit)

# Interaction Model with Ordered Data
m <- '
X =~ x1 + x2 + x3
Z =~ z1 + z2 + z3
Y =~ y1 + y2 + y3

Y ~ X + Z + X:Z
,

fit <- pls(m, oneIntOrdered, bootstrap = TRUE)
summary(fit)
```

pls_construct_scores *Construct latent variable scores*

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 PLSemMatrix of predicted latent scores.

 pls_predict

Predict from a fitted PLS-SEM model

Description

Predict from a fitted PLS-SEM model

Usage

```
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 plssem model.
approach	Prediction approach. If approach = "earliest" (default), then only indicators of exogenous benchmark.vars are used for prediction. If approach = "direct", 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: "r2", "rmse", "mae", "q2_predict", "acc", "ord_mae".
benchmark.vars	What predictions should be benchmarked? If benchmark.vars = "endog" (default), prediction benchmarks are applied to indicators of endogenous benchmark.vars. If benchmark.vars = "exog", prediction benchmarks are applied to indicators of exogenous benchmark.vars. If benchmark.vars = "all", prediction benchmarks are applied to all of the indicators in the model.
...	Additional arguments passed to internal helpers.

Value

A PlsSemPredict object with matrices and benchmark results.

print.plssem *Print a plssem object*

Description

Print a plssem object

Usage

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

Arguments

x An object of class plssem.
... Additional arguments for compatibility with the generic.

Value

The input object, invisibly.

print.PlsSemPredict *Print a PlsSemPredict object*

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 *Print a SummaryPlsSem object*

Description

Print a SummaryPlsSem object

Usage

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

Arguments

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

Value

The input object, invisibly.

randomIntercepts *randomIntercepts*

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
randomInterceptsOrdered

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 *randomSlopes*

Description

A simulated dataset. `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 *randomSlopesOrdered*

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)
```

summary.plssem	<i>Summarize a fitted plssem model</i>
----------------	--

Description

Summarize a fitted plssem model

Usage

```
## S3 method for class 'plssem'
summary(object, fit = TRUE, ...)
```

Arguments

object	An object of class plssem.
fit	Should fit measures be calculated?
...	Additional arguments passed to or from methods.

Value

A SummaryPlsSem object containing formatted parameter estimates.

titanic	<i>Titanic Passenger Survival Data Set.</i>
---------	---

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

TPB_Ordered

Description

A simulated dataset.

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
'

fit <- pls(tpb, TPB_Ordered)
summary(fit)
```

Index

`boot`, [2](#)

`isMC_PLS`, [3](#)

`oneIntOrdered`, [4](#)

`parameter_estimates`, [4](#)

`parameter_estimates.plssem`, [5](#)

`pls`, [5](#)

`pls_construct_scores`, [10](#)

`pls_predict`, [11](#)

`print.plssem`, [12](#)

`print.PlsSemPredict`, [12](#)

`print.SummaryPlsSem`, [13](#)

`randomIntercepts`, [13](#)

`randomInterceptsOrdered`, [14](#)

`randomSlopes`, [14](#)

`randomSlopesOrdered`, [14](#)

`summary.plssem`, [15](#)

`titanic`, [15](#)

`TPB_Ordered`, [16](#)