Package 'PLSsemEngine'

August 9, 2026

Type Package

Title Transparent PLS-SEM Estimation for Composite-Based Reflective Models

Version 1.3.0

Date 2026-07-24

Description A transparent and modular implementation of Partial Least Squares Structural Equation Modeling (PLS-SEM) focused on reflective measurement models (Mode A). The package separates estimation, bootstrap inference, and predictive evaluation into independent components, emphasising algorithmic transparency, reproducibility, and researcher-controlled analysis. Methods are based on Tenenhaus, Esposito Vinzi, Chatelin & Lauro (2005) <doi:10.1016/j.csda.2004.03.005>, Hair, Risher, Sarstedt & Ringle (2019) <doi:10.1108/EBR-11-2018-0203>, and Henseler, Ringle & Sarstedt (2015) <doi:10.1007/s11747-014-0403-8>.

License MIT + file LICENSE

URL https://github.com/msoto-perez/PLSsemEngine

Encoding UTF-8

Depends R (>= 4.0.0)

Imports stats, graphics, grDevices

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Manuel Soto-Perez [aut, cre]

Maintainer Manuel Soto-Perez <msoto@up.edu.mx>

Repository CRAN

Date/Publication 2026-08-09 07:30:15 UTC

Contents

`bootstrap_paths` *Non-Parametric Bootstrap for Structural Paths*

Description

Executes a case-level percentile bootstrap to assess the statistical significance of structural path coefficients. Crucially, to capture the total sampling uncertainty, this function performs a full-model re-estimation for every bootstrap replica, recalculating both outer weights and inner paths.

Usage

```
bootstrap_paths(
  data,
  measurement_model,
  structural_model,
  nboot = 500,
  seed = 123,
  sign_correction = FALSE,
  inner_scheme = "factorial"
)
```

Arguments

`data`	A data frame containing the observed indicators.
`measurement_model`	A named list defining the reflective constructs.
`structural_model`	A list of formulas defining the structural paths.
`nboot`	Integer. Number of bootstrap resamples. Default is 500.
`seed`	Integer. Random seed for reproducibility. Default is 123.
`sign_correction`	Logical. If TRUE, applies deterministic sign alignment per resample.
`inner_scheme`	Character. Weighting scheme for the inner approximation.

Value

A data frame containing the estimated path coefficients for all bootstrap iterations.

Examples

```
set.seed(1)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100)
)
mm <- list(Quality = c("SQ1", "SQ2", "SQ3"), Satisfaction = c("CS1", "CS2", "CS3"))
sm <- list(Satisfaction ~ Quality)
boot <- bootstrap_paths(dat, mm, sm, nboot = 20)
head(boot)
```

`compute_cmb_vif` *Compute Full Collinearity VIF for Common Method Bias*

Description

Assesses potential common method bias using the full collinearity VIF approach proposed by Kock (2015). Values above 3.3 may indicate pathological collinearity. In line with the engine's design philosophy, no automatic classification or model modification is applied.

Usage

```
compute_cmb_vif(scores, digits = 2)
```

Arguments

`scores`	A matrix or data frame of estimated latent variable scores.
`digits`	Integer. Number of decimal places for the output. Default is 2.

Value

A data frame containing the VIF values for each construct.

Examples

```
set.seed(1)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100)
)
mm <- list(Quality = c("SQ1", "SQ2", "SQ3"), Satisfaction = c("CS1", "CS2", "CS3"))
sm <- list(Satisfaction ~ Quality)
engine <- pls_engine(dat, mm, sm)
compute_cmb_vif(engine$scores)
```

---

compute_htmt_metrics *Compute HTMT and HTMT2 Ratios for Discriminant Validity*

Description

Computes the Heterotrait-Monotrait ratio of correlations (HTMT) following Henseler et al. (2015), and the revised HTMT2 following Roemer et al. (2021). HTMT2 uses the geometric mean to relax the tau-equivalence assumption, making it suitable for congeneric models.

Usage

```
compute_htmt_metrics(data, measurement_model, digits = 2)
```

Arguments

`data`	A data frame containing the observed indicators.
`measurement_model`	A named list defining the reflective constructs.
`digits`	Integer. Number of decimal places for the output matrices. Default is 2.

Value

A list containing the HTMT and HTMT2 symmetric matrices.

Examples

```
set.seed(1)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100)
)
mm <- list(Quality = c("SQ1", "SQ2", "SQ3"), Satisfaction = c("CS1", "CS2", "CS3"))
compute_htmt_metrics(dat, mm)
```

`compute_model_fit` *Compute Global Model Fit Indices*

Description

Calculates global fit indices including the Standardised Root Mean Square Residual (SRMR), exact squared Euclidean distance (d_ULS), and exact geodesic distance (d_G). Calculations are based on the saturated model-implied correlation matrix following Henseler et al. (2014).

Usage

```
compute_model_fit(engine, data, measurement_model, digits = 3)
```

Arguments

`engine`	An object containing the core PLS estimation results.
`data`	A data frame containing the observed indicators.
`measurement_model`	A named list defining the reflective constructs.
`digits`	Integer. Number of decimal places for the output. Default is 3.

Value

A data frame containing the computed fit metrics.

Examples

```
set.seed(1)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100)
)
mm <- list(Quality = c("SQ1", "SQ2", "SQ3"), Satisfaction = c("CS1", "CS2", "CS3"))
sm <- list(Satisfaction ~ Quality)
engine <- pls_engine(dat, mm, sm)
compute_model_fit(engine, dat, mm)
```

cvpat *Cross-Validated Predictive Ability Test (CVPAT)*

Description

Implements the Cross-Validated Predictive Ability Test (CVPAT) proposed by Liengaard et al. (2021) and extended by Sharma et al. (2023). Unlike PLSpredict, which provides descriptive predictive metrics (RMSE, Q2_predict), CVPAT provides an inferential test of whether the PLS model predicts significantly better than the naive linear model (LM) benchmark.

The test computes, for each fold and each endogenous indicator, the mean squared loss difference (MSE_LM - MSE_PLS). A one-sample t-test evaluates whether the average loss difference across folds is significantly greater than zero (H0: mean loss difference = 0). A positive and significant result indicates that the PLS model predicts significantly better than the benchmark.

Usage

```
cvpat(plspredict_result)
```

Arguments

`plspredict_result`
: The list returned by plspredict(), which must include the fold_losses element (available when using PLSsemEngine >= v1.3.0).

Value

A data frame with one row per endogenous indicator, reporting: mean loss difference, standard deviation, t-statistic, degrees of freedom, p-value, and a significance flag. An attribute "interpretation" is attached with a brief methodological note.

References

Liengaard, B. D., Sharma, P. N., Hult, G. T. M., Jensen, M. B., Sarstedt, M., Hair, J. F., & Ringle, C. M. (2021). Prediction: coveted, yet forsaken? Introducing a cross-validated predictive ability test in partial least squares path modeling. Decision Sciences, 52(2), 362-392.

Sharma, P. N., Liengaard, B. D., Hair, J. F., Sarstedt, M., & Ringle, C. M. (2023). Predictive model assessment and selection in composite-based modeling using PLS-SEM: extensions and guidelines for using CVPAT. European Journal of Marketing, 57(6), 1662-1677.

Examples

```
set.seed(1)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100)
)
mm <- list(Quality = c("SQ1", "SQ2", "SQ3"), Satisfaction = c("CS1", "CS2", "CS3"))
sm <- list(Satisfaction ~ Quality)
```

```
pred <- plspredict(dat, mm, sm, k = 5)
cvpat(pred)
```

`export_lavaan_syntax` *Export to Lavaan Syntax (CB-SEM / CFA Integration)*

Description

Actively encourages cross-methodological validation by natively translating the PLSsemEngine model specification into lavaan-compatible syntax. This allows researchers to seamlessly evaluate their composite models against common factor models (CFA).

Usage

```
export_lavaan_syntax(measurement_model, structural_model = NULL)
```

Arguments

`measurement_model`
: A named list defining the reflective blocks.

`structural_model`
: An optional list of formulas defining the paths.

Value

Invisibly returns the generated lavaan syntax string while printing it to the console.

Examples

```
mm <- list(
  Quality = c("SQ1", "SQ2", "SQ3"),
  Satisfaction = c("CS1", "CS2", "CS3"),
  Loyalty = c("CL1", "CL2", "CL3")
)
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
export_lavaan_syntax(mm, sm)
```

`export_scores` *Export Latent Variable Scores*

Description

Extracts and exports the estimated latent variable scores to a CSV file. A sequential 'Case' identifier is appended to improve data traceability and facilitate subsequent external analyses.

Usage

```
export_scores(model, file = NULL)
```

Arguments

`model`	An object of class 'pls_model' generated by `pls_sem`.
`file`	Character. The file path and name for the exported CSV. Must be supplied explicitly by the user (e.g. `file.path(tempdir(), "scores.csv")`); there is no default location.

Value

Invisibly returns the data frame of scores while writing the CSV file to disk.

Examples

```
set.seed(123)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100),
  CL1 = rnorm(100), CL2 = rnorm(100), CL3 = rnorm(100)
)
mm <- list(
  Quality = c("SQ1", "SQ2", "SQ3"),
  Satisfaction = c("CS1", "CS2", "CS3"),
  Loyalty = c("CL1", "CL2", "CL3")
)
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
model <- pls_sem(dat, mm, sm, nboot = 20)
out_file <- file.path(tempdir(), "latent_scores.csv")
export_scores(model, file = out_file)
unlink(out_file)
```

get_boundary_offset *Calculate Exact Boundary Intersections for Rectangles*

Description

A geometric helper function that calculates the precise offset required for arrow positioning, ensuring that structural paths touch the boundaries of the construct rectangles rather than overlapping them.

Usage

```
get_boundary_offset(angle, bw, bh)
```

Arguments

angle	Numeric. The angle of the path in radians.
bw	Numeric. The half-width of the rectangle.
bh	Numeric. The half-height of the rectangle.

Value

Numeric. The geometric offset distance.

Examples

```
get_boundary_offset(angle = pi / 4, bw = 1.1, bh = 0.3)
```

get_indirect_effects *Calculate Indirect Effects*

Description

Computes indirect effects as the mathematical product of the respective structural path coefficients. Consistent with the engine's design philosophy, no automated mediation classifications (e.g., "full" or "partial") are provided, ensuring the theoretical interpretation remains entirely in the hands of the researcher.

Usage

```
get_indirect_effects(model, digits = 3)
```

Arguments

model	An object of class 'pls_model'.
digits	Integer. Number of decimal places for the output. Default is 3.

Value

A data frame containing the calculated indirect effects, or NULL if no structural mediators exist.

Examples

```
set.seed(123)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100),
  CL1 = rnorm(100), CL2 = rnorm(100), CL3 = rnorm(100)
)
mm <- list(
  Quality = c("SQ1", "SQ2", "SQ3"),
  Satisfaction = c("CS1", "CS2", "CS3"),
  Loyalty = c("CL1", "CL2", "CL3")
)
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
model <- pls_sem(dat, mm, sm, nboot = 20)
get_indirect_effects(model)
```

---

get_references *Retrieve Standard Methodological References*

Description

Provides a quick-reference table outlining standard evaluation heuristics and literature citations for PLS-SEM. This tool is intended to support researcher-led interpretation rather than enforce mechanical rule application.

Usage

```
get_references()
```

Value

A data frame mapping analytical components to standard thresholds and academic references.

Examples

```
get_references()
```

`htmt_item_diagnostics` *HTMT-Guided Item Diagnostics*

Description

When discriminant validity is compromised at the construct level (HTMT > threshold), this function provides a granular, item-level diagnostic. It calculates cross-construct correlations to help researchers isolate specific problematic indicators that may be inflating the trait ratios.

Usage

```
htmt_item_diagnostics(model, threshold = 0.85, digits = 2)
```

Arguments

`model`	An object of class 'pls_model'.
`threshold`	Numeric. The HTMT threshold used to flag discriminant validity issues. Default is 0.85.
`digits`	Integer. Number of decimal places for the output. Default is 2.

Value

A data frame detailing problematic construct pairs and their item-level cross-correlations, or NULL if no issues are detected.

Examples

```
set.seed(123)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100),
  CL1 = rnorm(100), CL2 = rnorm(100), CL3 = rnorm(100)
)
mm <- list(
  Quality = c("SQ1", "SQ2", "SQ3"),
  Satisfaction = c("CS1", "CS2", "CS3"),
  Loyalty = c("CL1", "CL2", "CL3")
)
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
model <- pls_sem(dat, mm, sm, nboot = 20)
htmt_item_diagnostics(model)
```

`interpret_model` *Optional Methodological Assessment*

Description

Provides a narrative diagnostic layer based on established PLS-SEM literature (e.g., Hair et al., 2017; Henseler et al., 2015). Crucially, this function contextualises the raw metrics without imposing mechanical or automated decision-making, keeping the final analytical judgement with the researcher.

Usage

```
interpret_model(model)
```

Arguments

`model`	An object of class 'pls_model'.

Value

Invisibly prints a structured console report flagging potential issues.

Examples

```
set.seed(123)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100),
  CL1 = rnorm(100), CL2 = rnorm(100), CL3 = rnorm(100)
)
mm <- list(
  Quality = c("SQ1", "SQ2", "SQ3"),
  Satisfaction = c("CS1", "CS2", "CS3"),
  Loyalty = c("CL1", "CL2", "CL3")
)
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
model <- pls_sem(dat, mm, sm, nboot = 20)
interpret_model(model)
```

`plot_model_results` *Plot Model Results (Beta & R-squared)*

Description

Overlays the estimated path coefficients and R-squared values onto the structural model plot. In response to usability requests, R-squared values can be toggled on or off to provide maximum formatting flexibility.

Usage

```
plot_model_results(
  model,
  layout = NULL,
  show_r2 = TRUE,
  box_width = 1.1,
  box_height = 0.3,
  cex_node = 0.9,
  cex_beta = 0.8,
  cex_r2 = 0.75,
  arr_lwd = 1.2,
  box_col = "white",
  save_plot = FALSE,
  file_name = NULL,
  width = 2500,
  height = 1500,
  res = 300
)
```

Arguments

`model`	An object of class 'pls_model'.
`layout`	Optional data frame with node coordinates.
`show_r2`	Logical. If TRUE, displays R-squared values for endogenous constructs. Default is TRUE.
`box_width`	Numeric. Half-width of the nodes.
`box_height`	Numeric. Half-height of the nodes.
`cex_node`	Numeric. Text expansion for node labels.
`cex_beta`	Numeric. Text expansion for path coefficients.
`cex_r2`	Numeric. Text expansion for R-squared values.
`arr_lwd`	Numeric. Line width for arrows.
`box_col`	Character. Background colour for nodes.
`save_plot`	Logical. Exports plot if TRUE.

file_name	Character. Target file path for export. Required (no default location) when save_plot = TRUE, e.g. file.path(tempdir(), "plot.png").
width	Integer. Image width.
height	Integer. Image height.
res	Integer. Image resolution.

Value

Invisibly generates a plot on the active graphic device.

Examples

```
set.seed(123)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100),
  CL1 = rnorm(100), CL2 = rnorm(100), CL3 = rnorm(100)
)
mm <- list(
  Quality = c("SQ1", "SQ2", "SQ3"),
  Satisfaction = c("CS1", "CS2", "CS3"),
  Loyalty = c("CL1", "CL2", "CL3")
)
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
model <- pls_sem(dat, mm, sm, nboot = 20)
plot_model_results(model)
```

---

plot_structural_model *Plot Structural Model Base Geometry*

Description

Generates a pure base R plot of the structural model layout. It relies exclusively on native graphical parameters to avoid third-party dependencies, ensuring long-term reproducibility and stability.

Usage

```
plot_structural_model(
  structural_model,
  layout = NULL,
  box_width = 1.1,
  box_height = 0.3,
  cex_node = 0.9,
  arr_lwd = 1.2,
```

```
  box_col = "white",
  save_plot = FALSE,
  file_name = NULL,
  width = 2500,
  height = 1500,
  res = 300
)
```

Arguments

structural_model	A list of formulas defining the paths.
layout	Optional data frame with 'name', 'x', and 'y' coordinates for nodes.
box_width	Numeric. Half-width of the construct nodes. Default is 1.1.
box_height	Numeric. Half-height of the construct nodes. Default is 0.3.
cex_node	Numeric. Text expansion factor for node labels. Default is 0.9.
arr_lwd	Numeric. Line width for structural arrows. Default is 1.2.
box_col	Character. Background colour for nodes. Default is "white".
save_plot	Logical. If TRUE, exports the plot as a high-resolution PNG.
file_name	Character. Target file path for export. Required (no default location) when `save_plot = TRUE`, e.g. `file.path(tempdir(), "plot.png")`.
width	Integer. Width of the exported image in pixels.
height	Integer. Height of the exported image in pixels.
res	Integer. Resolution of the exported image in PPI.

Value

Invisibly generates a plot on the active graphic device.

Examples

```
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
plot_structural_model(sm)
```

plspredict *PLSpredict: Out-of-Sample Predictive Evaluation*

Description

Implements the PLSpredict algorithm using k-fold cross-validation. The function estimates the model exclusively on the training set and explicitly projects the outer weights onto the test set. Predictive performance (RMSE) is then compared against a naive linear model (LM) benchmark. Fold-level losses are retained to enable inferential testing via cvpat().

Usage

```
plspredict(
  data,
  measurement_model,
  structural_model,
  k = 5,
  seed = 123,
  sign_correction = FALSE,
  inner_scheme = "factorial"
)
```

Arguments

data	A data frame containing the observed indicators.
measurement_model	A named list defining the reflective constructs.
structural_model	A list of formulas defining the structural paths.
k	Integer. Number of folds for cross-validation. Default is 5.
seed	Integer. Random seed for fold generation to ensure reproducibility. Default is 123.
sign_correction	Logical. Applies deterministic sign alignment to training models.
inner_scheme	Character. Weighting scheme for the inner approximation.

Value

A list containing a predictive evaluation table (RMSE and Q2_predict), fold-level squared losses per indicator (fold_losses), and conditional warnings.

Examples

```
set.seed(1)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
```

```
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100)
)
mm <- list(Quality = c("SQ1", "SQ2", "SQ3"), Satisfaction = c("CS1", "CS2", "CS3"))
sm <- list(Satisfaction ~ Quality)
pred <- plspredict(dat, mm, sm, k = 5)
pred$table
```

`pls_engine` *Core PLS-SEM Estimation Engine (Mode A)*

Description

Implements the standard Partial Least Squares (PLS) iterative algorithm restricted to reflective measurement models (Mode A). The engine estimates latent variable scores, indicator loadings, and structural path coefficients using pure matrix operations. By design, it avoids automatic model re-specification or hidden interpretative heuristics.

Usage

```
pls_engine(
  data,
  measurement_model,
  structural_model,
  max_iter = 300,
  tol = 1e-06,
  sign_correction = FALSE,
  inner_scheme = "factorial"
)
```

Arguments

`data`	A data frame or matrix containing the observed indicators.
`measurement_model`	A named list defining the reflective constructs and their corresponding indicators.
`structural_model`	A list of formulas defining the structural relationships between latent constructs.
`max_iter`	Integer. Maximum number of iterations for the PLS algorithm. Default is 300.
`tol`	Numeric. Tolerance threshold for algorithmic convergence. Default is 1e-6.
`sign_correction`	Logical. If TRUE, applies deterministic sign alignment. Disabled by default to prevent bootstrap truncation.
`inner_scheme`	Character. Weighting scheme for the inner approximation ("factorial" or "centroid"). Default is "factorial".

Value

A list containing latent scores, outer weights, indicator loadings, structural paths, R-squared values, and metadata.

Examples

```
set.seed(1)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100)
)
mm <- list(Quality = c("SQ1", "SQ2", "SQ3"), Satisfaction = c("CS1", "CS2", "CS3"))
sm <- list(Satisfaction ~ Quality)
engine <- pls_engine(dat, mm, sm)
engine$paths
```

---

pls_sem *Estimate a PLS-SEM Model (Mode A)*

Description

This is the main high-level wrapper of the PLSsemEngine. It coordinates the estimation, assessment, and predictive evaluation stages, returning all results in a structured, publication-ready S3 object. It deliberately restricts estimation to reflective blocks (Mode A) and avoids hidden model modifications.

Usage

```
pls_sem(
  data,
  measurement_model,
  structural_model,
  k = 5,
  nboot = 500,
  digits = 2,
  sign_correction = FALSE,
  inner_scheme = "factorial"
)
```

Arguments

`data`	A data frame containing the observed indicators.
`measurement_model`	A named list defining the reflective constructs and their items.
`structural_model`	A list of formulas defining the inner paths.

k	Integer. Number of folds for PLSpredict cross-validation. Default is 5.
nboot	Integer. Number of bootstrap resamples for inference. Default is 500.
digits	Integer. Number of decimal places for output tables. Default is 2.
sign_correction	Logical. If TRUE, applies deterministic sign alignment.
inner_scheme	Character. Weighting scheme for the inner model ("factorial" or "centroid"). Default is "factorial".

Value

A list of class 'pls_model' containing structured tables for measurement, structural, and predictive assessments.

Examples

```
set.seed(123)
dat <- data.frame(
  SQ1 = rnorm(100), SQ2 = rnorm(100), SQ3 = rnorm(100),
  CS1 = rnorm(100), CS2 = rnorm(100), CS3 = rnorm(100),
  CL1 = rnorm(100), CL2 = rnorm(100), CL3 = rnorm(100)
)
mm <- list(
  Quality = c("SQ1", "SQ2", "SQ3"),
  Satisfaction = c("CS1", "CS2", "CS3"),
  Loyalty = c("CL1", "CL2", "CL3")
)
sm <- list(
  Satisfaction ~ Quality,
  Loyalty ~ Satisfaction + Quality
)
model <- pls_sem(dat, mm, sm, nboot = 20)
model$structural_model
```

Index