

Package ‘seminrExtras’

August 20, 2026

Type Package

Title Conduct Additional Modeling and Analysis for 'seminr'

Version 1.0.3

Date 2026-08-18

Description Supplementary tools for evaluating and validating partial least squares structural equation models estimated with 'seminr'. Provides methods for predictive model assessment, importance-performance analysis with necessary condition testing, overfitting diagnostics, measurement model verification, mediator contribution analysis, unobserved heterogeneity detection via latent class and prediction-oriented segmentation, and congruence coefficient testing. All functions accept estimated 'seminr' model objects and return results with print, summary, and plot methods.

Imports seminr (>= 2.4.0), stats, graphics, grDevices, utils, rpart

License GPL-3

Encoding UTF-8

Suggests testthat (>= 3.0.0), knitr, rmarkdown, MASS, paran, psych, learnr

Config/testthat/edition 3

URL <https://github.com/sem-in-r/seminrExtras>

BugReports <https://github.com/sem-in-r/seminrExtras/issues>

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Soumya Ray [aut, ths],
Nicholas Patrick Danks [aut, cre]

Maintainer Nicholas Patrick Danks <seminrgroup@gmail.com>

Repository CRAN

Date/Publication 2026-08-19 22:50:03 UTC

Contents

assess_cipma	2
assess_coa	5
assess_cta	7
assess_cvpat	9
assess_cvpat_compare	11
assess_fimix	14
assess_fimix_compare	16
assess_ipma	18
assess_nca	19
assess_nca_esse	21
assess_pcm	24
assess_pos	26
assess_pos_compare	28
competes	29
congruence_test	30
deviance_tree	32
group_rules	33
plot.cipma_analysis	34
plot.coa_analysis	34
plot.fimix_analysis	35
plot.fimix_comparison	35
plot.nca_analysis	36
plot.nca_esse	36
plot.pcm_analysis	37
pos_segments	38
predictive_deviance	38
unstable_params	39
Index	41

assess_cipma	<i>Combined Importance-Performance Map Analysis (cIPMA) for PLS-SEM</i>
--------------	---

Description

assess_cipma conducts an Importance-Performance Map Analysis (IPMA) and optionally combines it with Necessary Condition Analysis (NCA) to produce a combined IPMA (cIPMA). The analysis identifies constructs that are both important (high total effect) and necessary (without which the outcome cannot reach high levels), following the 7-step cIPMA procedure of Sarstedt et al. (2024).

Usage

```
assess_cipma(
  semnr_model,
  target,
  scale_min = 1,
  scale_max = 7,
  nca = TRUE,
  nca_ceilings = c("ce_fdh", "cr_fdh"),
  nca_test.rep = 0,
  nca_steps = 10,
  seed = 123
)
```

Arguments

<code>semnr_model</code>	An estimated SEMinR model from <code>estimate_pls()</code> .
<code>target</code>	Name of the target (endogenous) construct for the IPMA.
<code>scale_min</code>	Minimum of the measurement scale (default 1).
<code>scale_max</code>	Maximum of the measurement scale (default 7).
<code>nca</code>	Logical; if TRUE (default), run NCA to produce a cIPMA.
<code>nca_ceilings</code>	Character vector of ceiling techniques for NCA (default <code>c("ce_fdh", "cr_fdh")</code>).
<code>nca_test.rep</code>	Number of NCA permutation test repetitions (default 0). Set > 0 to obtain significance for the necessity classification.
<code>nca_steps</code>	Number of bottleneck table steps (default 10).
<code>seed</code>	Random seed for reproducibility (default 123).

Details

Importance is measured by the unstandardized total effect of each construct on the target. Unstandardized effects are obtained by scaling standardized path coefficients using the standard deviations of rescaled (0–100) construct scores.

Performance is the weighted average of rescaled indicator means, where each indicator is rescaled from the original measurement scale (`[scale_min, scale_max]`) to 0–100. Weights are the PLS outer weights, which must all be positive for valid IPMA rescaling.

Interaction constructs (e.g., "X*W") are automatically excluded from the IPMA because their performance is not meaningful on a 0–100 scale.

When `nca = TRUE`, NCA is run on construct scores for each predecessor–target pair, and constructs are classified into four categories based on crossing importance (at/above vs. below the mean, the standard IPMA crosshair) with necessity (NCA $d \geq 0.1$ and $p < 0.05$).

Value

An object of class `cipma_analysis` containing:

<code>importance_unstd</code>	Named numeric vector of unstandardized total effects
-------------------------------	--

importance_std	Named numeric vector of standardized total effects
performance	Named numeric vector of construct performances (0–100)
nca	An nca_analysis object (NULL if nca = FALSE)
classification	Data frame classifying each construct
target	Name of the target construct
constructs	Character vector of included constructs
scale_range	Numeric vector c(scale_min, scale_max)
negative_weight_constructs	Constructs with negative outer weights (if any)
excluded_interactions	Interaction constructs excluded from IPMA
pls_model	The original estimated seminr model

References

Ringle, C. M. & Sarstedt, M. (2016). Gain More Insight from Your PLS-SEM Results: The Importance-Performance Map Analysis. *Industrial Management & Data Systems*, 119(9), 1865-1886.

Sarstedt, M., Richter, N. F., Hauff, S. & Ringle, C. M. (2024). Combined Importance-Performance Map Analysis (cIPMA): A SmartPLS 4 Tutorial. *Journal of Marketing Analytics*, 12, 746-760.

Hauff, S. et al. (2024). Importance and Performance in PLS-SEM and NCA: Introducing the cIPMA. *Journal of Retailing and Consumer Services*, 78, 103723.

See Also

[assess_ipma](#) for IPMA without NCA, [assess_nca](#) for standalone NCA analysis

Examples

```
library(seminr)
library(seminrExtras)

mob_i_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Expectation", multi_items("CUEX", 1:3)),
  composite("Value",      multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",    multi_items("CUSL", 1:3))
)

mob_i_sm <- relationships(
  paths(from = "Image",      to = c("Expectation", "Satisfaction", "Loyalty")),
  paths(from = "Expectation", to = c("Value", "Satisfaction")),
  paths(from = "Value",      to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)
```

```

mobl_pls <- estimate_pls(data = mobil,
                        measurement_model = mobil_mm,
                        structural_model = mobil_sm)

cipma_result <- assess_cipma(mobil_pls,
                             target = "Loyalty",
                             scale_min = 1,
                             scale_max = 10)

print(cipma_result)
summary(cipma_result)
plot(cipma_result)

```

assess_coa

Composite Overfit Analysis (COA)

Description

assess_coa conducts a full Composite Overfit Analysis on a PLS-SEM model. It computes predictive deviance for a focal construct, identifies groups of deviant cases using a decision tree, and assesses parameter instability.

Usage

```

assess_coa(
  semlnr_model,
  focal_construct,
  deviance_bounds = c(0.025, 0.975),
  params = "path_coef",
  technique = predict_DA,
  noFolds = 10,
  reps = 1,
  cores = NULL,
  seed = 123,
  predict_model = NULL
)

```

Arguments

semlnr_model An estimated SEMlnR model from estimate_pls().

focal_construct Name of the endogenous construct to analyze (character).

deviance_bounds Two-element numeric vector of quantile bounds for identifying deviant cases (default c(0.025, 0.975)).

params	Character vector of model parameters to assess for instability. Valid values include "path_coef", "outer_weights", "outer_loadings", "rSquared" (default "path_coef").
technique	Prediction technique: predict_DA (default) or predict_EA.
noFolds	Number of folds for k-fold cross-validation (default 10).
reps	Number of CV repetitions (default 1).
cores	Number of cores for parallel CV (default NULL = sequential).
seed	Random seed for reproducibility (default 123).
predict_model	Optional pre-computed predict_pls_model object. If provided, skips the cross-validation step (saves time when predictions have already been computed).

Value

An object of class `coa_analysis` containing:

pls_model	The original estimated model
focal_construct	Name of the analyzed construct
deviance_bounds	Quantile bounds used
predictive_deviance	Predictive deviance results (class <code>coa_deviance</code>)
deviance_tree	Deviance tree and deviant groups (class <code>coa_dtree</code>)
unstable	Parameter instability results (class <code>coa_unstable</code>)

See Also

[predictive_deviance\(\)](#), [deviance_tree\(\)](#), [unstable_params\(\)](#), [group_rules\(\)](#), [competes\(\)](#)

Examples

```
library(seminr)
library(seminrExtras)

mobi_mm <- constructs(
  composite("Image", multi_items("IMAG", 1:5)),
  composite("Value", multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty", multi_items("CUSL", 1:3))
)

mobi_sm <- relationships(
  paths(from = c("Image", "Value"), to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)

mobi_pls <- estimate_pls(
  data = mobi,
```

```

    measurement_model = mobi_mm,
    structural_model   = mobi_sm
  )

coa_result <- assess_coa(mobi_pls,
                        focal_construct = "Loyalty",
                        noFolds = 10,
                        cores = 1,
                        seed = 123)

print(coa_result)
summary(coa_result)
plot(coa_result, type = "pd")

```

assess_cta

Confirmatory Tetrad Analysis for PLS-SEM (CTA-PLS)

Description

`assess_cta` tests whether each construct's measurement model is consistent with a reflective (common factor) specification. Under a reflective model, all vanishing tetrads equal zero. If any tetrad is significantly non-zero (after multiple testing correction), the reflective specification is rejected.

Usage

```

assess_cta(
  seminr_model,
  constructs = NULL,
  nboot = 5000,
  seed = 123,
  alpha = 0.05,
  correction = "BH",
  borrow = TRUE
)

```

Arguments

<code>seminr_model</code>	A PLS-SEM model estimated by <code>seminr::estimate_pls()</code> .
<code>constructs</code>	Character vector of construct names to test. If <code>NULL</code> (default), all constructs in the model are tested.
<code>nboot</code>	Number of bootstrap subsamples (default 5000).
<code>seed</code>	Seed for reproducibility (default 123).
<code>alpha</code>	Significance level (default 0.05).
<code>correction</code>	Multiple testing correction method: "BH" (Benjamini-Hochberg, default), "bonferroni", or "none".

borrow Logical. If TRUE (default), constructs with 2–3 indicators borrow indicators from structurally connected constructs to reach the minimum of 4 needed for tetrad testing (Gudergan et al., 2008, Table 1). If FALSE, constructs with fewer than 4 indicators are skipped.

Details

Minimum indicator requirement: Without borrowing, constructs with fewer than 4 indicators are skipped. With borrow = TRUE (default), constructs with 2–3 indicators borrow from structurally adjacent constructs. A minimum of 2 own indicators is always required.

Borrowing rules (Gudergan et al., 2008, Table 1): For a reflective focal construct with 3 own indicators, 1 indicator is borrowed from an adjacent reflective construct; all tetrads vanish under H0. For 2 own indicators, 2 are borrowed from any adjacent construct; only the tau_1342 tetrad vanishes. Formative focal constructs cannot be tested via borrowing.

For HOC constructs, the lower-order constructs (LOCs) serve as "indicators"; at least 4 LOCs are required.

Interaction constructs (moderation terms with * in the name) are automatically excluded because their measurement specification is determined by the interaction method, not the data.

Multiple testing correction: The number of tetrads grows combinatorially with indicators. Benjamini-Hochberg (BH) correction is recommended as the default (Cefis et al., 2025). Bonferroni is more conservative. With only 4 indicators (2 tetrads), correction has minimal impact.

Sample size: CTA-PLS requires adequate sample sizes for reliable results. A warning is issued if $N < 200$.

Value

An S3 object of class `cta_analysis` containing:

construct_results Data frame summarising each construct: mode, number of indicators, tetrads tested, significant tetrads, and verdict.

tetrad_details Named list of per-construct data frames with individual tetrad estimates, bootstrap CIs, and adjusted p-values.

nboot Number of bootstrap subsamples used.

alpha Significance level used.

correction Multiple testing correction method used.

skipped Character vector of constructs that were skipped.

borrowing Named list of borrowing details for constructs that borrowed indicators (donor name, mode, and vanishing pattern).

References

- Gudergan, S. P., Ringle, C. M., Wende, S. & Will, A. (2008). Confirmatory Tetrad Analysis in PLS Path Modeling. *Journal of Business Research*, 61(12), 1238-1249.
- Cefis, M., Angelelli, M., Carpita, M. & Ciavolino, E. (2025). Confirmatory Tetrad Analysis in PLS-SEM: A Multiple Testing Correction Perspective. *Social Indicators Research* (advance online).

See Also

[congruence_test\(\)](#) for congruence coefficient testing

Examples

```
library(seminr)
library(seminrExtras)

# Specify measurement model
mobi_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Expectation", multi_items("CUEX", 1:3)),
  composite("Value",      multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",    multi_items("CUSL", 1:3))
)

# Specify structural model
mobi_sm <- relationships(
  paths(from = "Image",      to = c("Expectation", "Satisfaction", "Loyalty")),
  paths(from = "Expectation", to = c("Value", "Satisfaction")),
  paths(from = "Value",      to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)

# Estimate the model
pls_model <- estimate_pls(data = mobi, measurement_model = mobi_mm,
                          structural_model = mobi_sm)

# Run CTA-PLS with borrowing (low nboot for example speed)
cta <- assess_cta(pls_model, nboot = 50, seed = 123)
print(cta)
summary(cta)

# Without borrowing -- only constructs with >= 4 indicators are tested
cta_no_borrow <- assess_cta(pls_model, nboot = 50, borrow = FALSE)
```

assess_cvpat

Assess single model CVPAT against benchmarks

Description

`assess_cvpat` conducts a single model CVPAT assessment against item average and linear model prediction benchmarks. Use this to determine whether your PLS model has meaningful predictive power.

Usage

```

assess_cvpat(
  seminr_model,
  testtype = "two.sided",
  nboot = 2000,
  seed = 123,
  technique = predict_DA,
  noFolds = NULL,
  reps = NULL,
  cores = NULL
)

```

Arguments

<code>seminr_model</code>	The SEMinR model for CVPAT analysis
<code>testtype</code>	Either "two.sided" (default) or "greater".
<code>nboot</code>	The number of bootstrap subsamples to execute (defaults to 2000).
<code>seed</code>	The seed for reproducibility (defaults to 123).
<code>technique</code>	predict_EA or predict_DA (default).
<code>noFolds</code>	Number of folds for k-fold cross validation.
<code>reps</code>	Number of repetitions for cross validation.
<code>cores</code>	Number of cores for parallelization.

Details

Two benchmarks are used:

- **Linear Model (LM)**: Multiple regression predicting each item
- **Indicator Average (IA)**: Naive benchmark using training set means

If PLS loss < LM loss (significantly), the structural model adds predictive value. If PLS loss < IA loss (significantly), the model has basic predictive relevance.

Value

A list containing two matrices: CVPAT_compare_LM (comparison with linear model) and CVPAT_compare_IA (comparison with indicator average).

References

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.

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.

See Also

[assess_cvpat_compare\(\)](#) for comparing two PLS models

Examples

```
# Load libraries
library(semplr)
library(semplrExtras)

# Create measurement model ----
corp_rep_mm_ext <- constructs(
  composite("QUAL", multi_items("qual_", 1:8), weights = mode_B),
  composite("PERF", multi_items("perf_", 1:5), weights = mode_B),
  composite("CSOR", multi_items("csor_", 1:5), weights = mode_B),
  composite("ATTR", multi_items("attr_", 1:3), weights = mode_B),
  composite("COMP", multi_items("comp_", 1:3)),
  composite("LIKE", multi_items("like_", 1:3))
)

# Create structural model ----
corp_rep_sm_ext <- relationships(
  paths(from = c("QUAL", "PERF", "CSOR", "ATTR"), to = c("COMP", "LIKE"))
)

# Estimate the model ----
corp_rep_pls_model_ext <- estimate_pls(
  data = corp_rep_data,
  measurement_model = corp_rep_mm_ext,
  structural_model = corp_rep_sm_ext,
  missing = mean_replacement,
  missing_value = "-99")

# Assess the base model ----
assess_cvpat(semplr_model = corp_rep_pls_model_ext,
  testtype = "two.sided",
  nboot = 20,
  seed = 123,
  technique = predict_DA,
  noFolds = 5,
  reps = 1,
  cores = 1)
```

assess_cvpat_compare *Compare CVPAT loss between two PLS models*

Description

`assess_cvpat_compare` conducts a CVPAT significance test of loss between two models. Use this to determine which of two competing models has superior predictive performance.

Usage

```

assess_cvpat_compare(
  established_model,
  alternative_model,
  testtype = "two.sided",
  nboot = 2000,
  seed = 123,
  technique = predict_DA,
  noFolds = NULL,
  reps = NULL,
  cores = NULL
)

```

Arguments

<code>established_model</code>	The base seminr model for CVPAT comparison.
<code>alternative_model</code>	The alternate seminr model for CVPAT comparison.
<code>testtype</code>	Either "two.sided" (default) or "greater".
<code>nboot</code>	The number of bootstrap subsamples to execute (defaults to 2000).
<code>seed</code>	The seed for reproducibility (defaults to 123).
<code>technique</code>	predict_EA or predict_DA (default).
<code>noFolds</code>	Number of folds for k-fold cross validation.
<code>reps</code>	Number of repetitions for cross validation.
<code>cores</code>	Number of cores for parallelization.

Value

A matrix of the estimated loss and results of significance testing.

References

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.

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.

See Also

[assess_cvpat\(\)](#) for single model assessment against benchmarks

Examples

```
# Load libraries
library(seminr)
library(seminrExtras)

# Create measurement model ----
corp_rep_mm_ext <- constructs(
  composite("QUAL", multi_items("qual_", 1:8), weights = mode_B),
  composite("PERF", multi_items("perf_", 1:5), weights = mode_B),
  composite("CSOR", multi_items("csor_", 1:5), weights = mode_B),
  composite("ATTR", multi_items("attr_", 1:3), weights = mode_B),
  composite("COMP", multi_items("comp_", 1:3)),
  composite("LIKE", multi_items("like_", 1:3))
)

alt_mm <- constructs(
  composite("QUAL", multi_items("qual_", 1:8), weights = mode_B),
  composite("PERF", multi_items("perf_", 1:5), weights = mode_B),
  composite("CSOR", multi_items("csor_", 1:5), weights = mode_B),
  composite("COMP", multi_items("comp_", 1:3)),
  composite("LIKE", multi_items("like_", 1:3))
)

# Create structural model ----

corp_rep_sm_ext <- relationships(
  paths(from = c("QUAL", "PERF", "CSOR", "ATTR"), to = c("COMP", "LIKE"))
)
alt_sm <- relationships(
  paths(from = c("QUAL", "PERF", "CSOR"), to = c("COMP", "LIKE"))
)

# Estimate the model ----
established_model <- estimate_pls(
  data = corp_rep_data,
  measurement_model = corp_rep_mm_ext,
  structural_model = corp_rep_sm_ext,
  missing = mean_replacement,
  missing_value = "-99")

alternative_model <- estimate_pls(
  data = corp_rep_data,
  measurement_model = alt_mm,
  structural_model = alt_sm,
  missing = mean_replacement,
  missing_value = "-99")

# Function to compare the Loss of two models
assess_cvpat_compare(established_model,
  alternative_model ,
  testtype = "two.sided",
  nboot = 20,
```

```
seed = 123,
technique = predict_DA,
noFolds = 5,
reps = 1,
cores = 1)
```

assess_fimix

FIMIX-PLS Analysis for PLS-SEM

Description

assess_fimix runs the FIMIX-PLS (Finite Mixture PLS) procedure to identify unobserved heterogeneity in PLS-SEM structural models. The EM algorithm probabilistically assigns observations to K latent segments, each with segment-specific structural path coefficients.

Usage

```
assess_fimix(
  seminr_model,
  K = 2,
  nstart = 10,
  max_iter = 5000,
  stop_criterion = 1e-06,
  seed = 123
)
```

Arguments

seminr_model	An estimated SEMinR model from estimate_pls().
K	Integer; the number of segments (default 2). Must be ≥ 2 .
nstart	Integer; number of random EM starts (default 10).
max_iter	Integer; maximum EM iterations per start (default 5000).
stop_criterion	Numeric; convergence threshold on log-likelihood change (default 1e-6).
seed	Random seed for reproducibility (default 123).

Details

FIMIX-PLS operates on construct scores from the estimated PLS model. It fits a mixture of regressions to each structural equation simultaneously, estimating segment-specific intercepts, path coefficients, and residual variances. Multiple random starts (`nstart`) are used to avoid local optima; the solution with the highest log-likelihood is retained.

Value

An object of class `fimix_analysis` containing:

<code>K</code>	Number of segments
<code>segment_proportions</code>	Named numeric vector of mixing proportions
<code>posterior</code>	N x K matrix of posterior segment probabilities
<code>segment_assignment</code>	Integer vector of hard assignments (argmax)
<code>segment_paths</code>	List of K path coefficient matrices
<code>segment_intercepts</code>	List of K intercept vectors
<code>segment_variances</code>	J x K matrix of residual variances
<code>log_likelihood</code>	Final log-likelihood
<code>n_parameters</code>	Number of free parameters
<code>info_criteria</code>	Named vector of information criteria
<code>converged</code>	Logical; whether EM converged
<code>iterations</code>	Number of EM iterations
<code>n_starts_completed</code>	Number of random starts completed
<code>pls_model</code>	The original estimated seminr model
<code>n_obs</code>	Sample size

References

Hahn, C., Johnson, M. D., Herrmann, A., & Huber, F. (2002). Capturing Customer Heterogeneity using a Finite Mixture PLS Approach. *Schmalenbach Business Review*, 54, 243-269.

Sarstedt, M., Becker, J.-M., Ringle, C. M. & Schwaiger, M. (2011). Uncovering and Treating Unobserved Heterogeneity with FIMIX-PLS: Which Model Selection Criterion Provides an Appropriate Number of Segments? *Schmalenbach Business Review*, 63(1), 34-62.

See Also

[assess_fimix_compare](#) for comparing multiple K values

Examples

```
library(seminr)
library(seminrExtras)

mobi_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Expectation", multi_items("CUEX", 1:3)),
  composite("Value",      multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
```

```

  composite("Loyalty",      multi_items("CUSL", 1:3))
)

mobi_sm <- relationships(
  paths(from = "Image",      to = c("Expectation", "Satisfaction", "Loyalty")),
  paths(from = "Expectation", to = c("Value", "Satisfaction")),
  paths(from = "Value",      to = "Satisfaction"),
  paths(from = "Satisfaction",to = "Loyalty")
)

mobi_pls <- estimate_pls(data = mobi,
                        measurement_model = mobi_mm,
                        structural_model  = mobi_sm)

fimix_result <- assess_fimix(mobi_pls, K = 2, nstart = 5, seed = 123)
print(fimix_result)
summary(fimix_result)

```

assess_fimix_compare *Compare FIMIX-PLS Solutions Across Multiple K Values*

Description

assess_fimix_compare runs [assess_fimix](#) for each value of K in K_range and produces a comparison table of information criteria to guide segment number selection.

Usage

```

assess_fimix_compare(
  seminr_model,
  K_range = 2:5,
  nstart = 10,
  max_iter = 5000,
  stop_criterion = 1e-06,
  seed = 123
)

```

Arguments

seminr_model	An estimated SEMinR model from estimate_pls().
K_range	Integer vector of K values to evaluate (default 2:5).
nstart	Integer; number of random EM starts (default 10).
max_iter	Integer; maximum EM iterations per start (default 5000).
stop_criterion	Numeric; convergence threshold on log-likelihood change (default 1e-6).
seed	Random seed for reproducibility (default 123).

Details

The recommended approach is to inspect AIC3 and CAIC jointly: when both agree on K, accuracy reaches approximately 84\ AIC4 is the best-performing single criterion. Entropy (EN) indicates classification quality but should not be used alone to select K.

Value

An object of class `fimix_comparison` containing:

<code>solutions</code>	Named list of <code>fimix_analysis</code> objects (one per K)
<code>fit_table</code>	Data frame with K as rows and criteria as columns
<code>K_range</code>	The evaluated K values
<code>pls_model</code>	The original model

References

Sarstedt, M., Becker, J.-M., Ringle, C. M. & Schwaiger, M. (2011). Uncovering and Treating Unobserved Heterogeneity with FIMIX-PLS: Which Model Selection Criterion Provides an Appropriate Number of Segments? *Schmalenbach Business Review*, 63(1), 34-62.

See Also

[assess_fimix](#) for single-K FIMIX analysis

Examples

```
library(seminr)
library(seminrExtras)

mobi_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Expectation", multi_items("CUEX", 1:3)),
  composite("Value",      multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",    multi_items("CUSL", 1:3))
)

mobi_sm <- relationships(
  paths(from = "Image",      to = c("Expectation", "Satisfaction", "Loyalty")),
  paths(from = "Expectation", to = c("Value", "Satisfaction")),
  paths(from = "Value",      to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)

mobi_pls <- estimate_pls(data = mobi,
                        measurement_model = mobi_mm,
                        structural_model  = mobi_sm)

fimix_comp <- assess_fimix_compare(mobi_pls, K_range = 2:4,
```

```

                                nstart = 5, seed = 123)
print(fimix_comp)
plot(fimix_comp)

```

assess_ipma

*Importance-Performance Map Analysis (IPMA) for PLS-SEM***Description**

assess_ipma conducts an Importance-Performance Map Analysis (IPMA) for a given target construct. It computes each predecessor's importance (unstandardized total effect on the target) and performance (rescaled 0–100 mean construct score), identifying constructs with high importance but low performance as priority areas for improvement.

Usage

```
assess_ipma(seminr_model, target, scale_min = 1, scale_max = 7, seed = 123)
```

Arguments

seminr_model	An estimated SEminR model from <code>estimate_pls()</code> .
target	Name of the target (endogenous) construct for the IPMA.
scale_min	Minimum of the measurement scale (default 1).
scale_max	Maximum of the measurement scale (default 7).
seed	Random seed for reproducibility (default 123).

Details

This is a convenience wrapper around [assess_cipma](#) with `nca = FALSE`. Use [assess_cipma](#) instead if you also want Necessary Condition Analysis (NCA) to produce a combined IPMA.

Value

An object of class `cipma_analysis` (see [assess_cipma](#) for details). The `nca` element will be `NULL`.

References

Ringle, C. M. & Sarstedt, M. (2016). Gain More Insight from Your PLS-SEM Results: The Importance-Performance Map Analysis. *Industrial Management & Data Systems*, 119(9), 1865–1886.

See Also

[assess_cipma](#) for cIPMA (IPMA + NCA)

Examples

```
library(seminr)
library(seminrExtras)

mobi_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Expectation", multi_items("CUEX", 1:3)),
  composite("Value",      multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",    multi_items("CUSL", 1:3))
)

mobi_sm <- relationships(
  paths(from = "Image",      to = c("Expectation", "Satisfaction", "Loyalty")),
  paths(from = "Expectation", to = c("Value", "Satisfaction")),
  paths(from = "Value",      to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)

mobi_pls <- estimate_pls(data = mobi,
  measurement_model = mobi_mm,
  structural_model  = mobi_sm)

ipma_result <- assess_ipma(mobi_pls,
  target = "Loyalty",
  scale_min = 1,
  scale_max = 10)

print(ipma_result)
plot(ipma_result)
```

assess_nca

Necessary Condition Analysis (NCA) for PLS-SEM Models

Description

`assess_nca` conducts a Necessary Condition Analysis on construct scores from an estimated PLS-SEM model. It identifies whether predictors are necessary conditions for achieving a given level of the target construct.

Usage

```
assess_nca(
  seminr_model,
  target,
  predictors = NULL,
  ceilings = c("ce_fdh", "cr_fdh"),
```

```

    test.rep = 1000,
    steps = 10,
    seed = 123,
    ...
  )

```

Arguments

<code>seminr_model</code>	An estimated SEMinR model from <code>estimate_pls()</code> .
<code>target</code>	Name of the endogenous (outcome) construct (character).
<code>predictors</code>	Optional character vector of predictor construct names. If NULL (default), auto-detected from the structural model as direct predictors of target.
<code>ceilings</code>	Character vector of ceiling techniques to use (default <code>c("ce_fdh", "cr_fdh")</code>).
<code>test.rep</code>	Number of permutation test repetitions for significance testing (default 1000). Set to 0 to skip significance testing.
<code>steps</code>	Number of steps in the bottleneck table (default 10).
<code>seed</code>	Random seed for reproducibility (default 123).
<code>...</code>	Additional arguments (currently unused).

Details

NCA complements PLS-SEM's sufficiency-based regression by testing whether a certain level of a predictor is *necessary* (but not sufficient) for a certain level of the outcome. A predictor is considered a necessary condition when it has an effect size (d) ≥ 0.1 and a significant p-value ($p < 0.05$).

The function extracts construct scores from the `seminr` model, auto-detects direct predictors of the target from the structural model, and computes ceiling envelopes using the specified techniques.

Value

An object of class `nca_analysis` containing:

<code>nca_raw</code>	Reserved (always NULL)
<code>effect_sizes</code>	Matrix of effect sizes (d) per predictor and ceiling
<code>significance</code>	Matrix of p-values per predictor and ceiling (NULL if <code>test.rep = 0</code>)
<code>bottleneck</code>	List of bottleneck tables, one per ceiling technique
<code>pls_model</code>	The original estimated <code>seminr</code> model
<code>target</code>	Name of the target construct
<code>predictors</code>	Character vector of predictor names used
<code>ceilings</code>	Character vector of ceiling techniques used

References

- Dul, J. (2016). Necessary Condition Analysis (NCA): Logic and Methodology of 'Necessary but Not Sufficient' Causality. *Organizational Research Methods*, 19(1), 10-52.
- Richter, N. F., Schubring, S., Hauff, S., Ringle, C. M., & Sarstedt, M. (2020). When predictors of outcomes are necessary: guidelines for the combined use of PLS-SEM and NCA. *Industrial Management & Data Systems*, 120(12), 2243-2267.

Examples

```
library(seminr)
library(seminrExtras)

mobi_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Value",      multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",     multi_items("CUSL", 1:3))
)

mobi_sm <- relationships(
  paths(from = c("Image", "Value"), to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)

mobi_pls <- estimate_pls(data = mobi,
  measurement_model = mobi_mm,
  structural_model  = mobi_sm)

nca_result <- assess_nca(mobi_pls,
  target = "Satisfaction",
  test.rep = 100)

print(nca_result)
summary(nca_result)
plot(nca_result, type = "effects")
```

assess_nca_esse

NCA with Effect Size Sensitivity Extension (NCA-ESSE)

Description

assess_nca_esse applies the NCA-ESSE method (Becker et al., 2026) to assess how sensitive NCA effect sizes are to extreme response patterns. It systematically varies the ECDF ceiling threshold and compares empirical effect size changes against a theoretical benchmark (joint uniform distribution) to determine whether a necessary condition is robust.

Usage

```

assess_nca_esse(
  semnr_model,
  target,
  predictors = NULL,
  thresholds = seq(0, 0.05, by = 0.005),
  ceiling = "ce_fdh",
  test.rep = 0,
  steps = 10,
  seed = 123,
  ...
)

```

Arguments

<code>semnr_model</code>	An estimated SEMinR model from <code>estimate_pls()</code> .
<code>target</code>	Name of the endogenous (outcome) construct.
<code>predictors</code>	Optional character vector of predictor construct names. If NULL (default), auto-detected from the structural model.
<code>thresholds</code>	Numeric vector of ECDF thresholds to evaluate (default <code>seq(0, 0.05, by = 0.005)</code>). Values must be in [0, 1].
<code>ceiling</code>	Ceiling technique (default "ce_fdh"). The analytical benchmark is derived for CE-FDH; other techniques will trigger a warning.
<code>test.rep</code>	Number of permutation test repetitions at each threshold (default 0). Set > 0 to obtain p-values, but note this multiplies computation time by the number of thresholds.
<code>steps</code>	Number of steps in the bottleneck table (default 10).
<code>seed</code>	Random seed for reproducibility (default 123).
<code>...</code>	Additional arguments (currently unused).

Details

The method proceeds in three steps:

1. Run standard NCA (threshold 0\ conditions
2. Apply NCA-ESSE: vary the ECDF threshold from 0\ (default 5\ comparing against the uniform benchmark
3. (Optional) Select a threshold where the empirical effect size meaningfully exceeds the benchmark for further evaluation

At each threshold t , observations whose joint ECDF_NCA value ($P(X \leq x, Y \geq y)$) is at most t are treated as extreme and excluded before computing the CE-FDH ceiling. The uniform benchmark $d = t(1 - \ln(t))$ gives the expected effect size if no necessity exists.

Value

An object of class `nca_esse` containing:

<code>effect_sizes</code>	Matrix of empirical effect sizes (thresholds x predictors)
<code>benchmark</code>	Matrix of theoretical benchmark effect sizes
<code>delta</code>	Matrix of empirical minus benchmark differences
<code>significance</code>	Matrix of p-values (NULL if <code>test.rep = 0</code>)
<code>pls_model</code>	The original estimated <code>semnr</code> model
<code>target</code>	Name of the target construct
<code>predictors</code>	Character vector of predictor names
<code>thresholds</code>	Numeric vector of ECDF thresholds used
<code>ceiling</code>	Ceiling technique used
<code>n_obs</code>	Number of observations

References

Becker, J.-M., Richter, N. F., Ringle, C. M., & Sarstedt, M. (2026). Must-have, or maybe not? A sensitivity-based extension to necessary condition analysis. *Journal of Business Research*, 206, 115920.

See Also

[assess_nca](#) for standard NCA analysis

Examples

```
library(semnr)
library(semnrExtras)

mob_i_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Value",      multi_items("PERV", 1:2)),
  composite("Satisfaction", multi_items("CUSA", 1:3))
)

mob_i_sm <- relationships(
  paths(from = c("Image", "Value"), to = "Satisfaction")
)

mob_i_pls <- estimate_pls(data = mob_i,
  measurement_model = mob_i_mm,
  structural_model  = mob_i_sm)

esse_result <- assess_nca_esse(mob_i_pls,
  target = "Satisfaction",
  seed = 123)

print(esse_result)
```

```
plot(esse_result, type = "sensitivity")
```

assess_pcm

Predictive Contribution of the Mediator (PCM)

Description

Evaluates the predictive contribution of mediating constructs in a PLS-SEM model. For each mediation path (antecedent -> mediator -> target), computes the PCM metric by comparing cross-validated predictions from the Direct Antecedents (DA) and Earliest Antecedents (EA) approaches on an isolated partial mediation sub-model.

Usage

```
assess_pcm(seminr_model, target = NULL, noFolds = 10, reps = 10)
```

Arguments

<code>seminr_model</code>	A PLS model estimated by <code>seminr::estimate_pls()</code> .
<code>target</code>	Character string specifying the outcome construct. If <code>NULL</code> , auto-detects when a single final endogenous construct exists.
<code>noFolds</code>	Integer, number of folds for cross-validation (default 10).
<code>reps</code>	Integer, number of cross-validation repetitions (default 10).

Details

The PCM metric (Danks, 2021) quantifies the predictive improvement due to the mediator. It is computed by isolating each mediation path into a partial mediation sub-model (X -> M -> Y with direct path X -> Y), then comparing out-of-sample predictions from two approaches:

- **DA (Direct Antecedents)**: Predicts Y using its direct structural predictors (M and X).
- **EA (Earliest Antecedents)**: Predicts Y using only the earliest antecedent (X), propagated through the structural model.

$$PCM = \frac{METRIC_{EA} - METRIC_{DA}}{METRIC_{EA}}$$

Rules of thumb for PCM interpretation:

- **0 to 0.05**: Weak predictive contribution
- **0.05 to 0.10**: Moderate predictive contribution
- **Greater than 0.10**: Strong predictive contribution
- **Negative**: Mediator damages predictive accuracy

Value

An object of class `pcm_analysis` containing:

target The outcome construct name.

mediation_paths List of identified mediation triples (antecedent, mediator, target).

pcm_results List of per-path results, each with RMSE/MAE for DA and EA approaches plus PCM values per indicator.

noFolds Number of cross-validation folds used.

reps Number of cross-validation repetitions used.

References

Danks, N. P. (2021). The Piggy in the Middle: The Role of Mediators in PLS-SEM Prediction. *The DATA BASE for Advances in Information Systems*, 52(SI), 24-42.

See Also

[predict_pls](#) for the underlying prediction methods.

Examples

```
library(seminr)
data(mobi)

# Simple mediation: Image -> Satisfaction -> Loyalty
mobi_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",     multi_items("CUSL", 1:3))
)
mobi_sm <- relationships(
  paths(from = "Image", to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty"),
  paths(from = "Image", to = "Loyalty")
)
model <- estimate_pls(mobi, mobi_mm, mobi_sm)

# Compute PCM
pcm <- assess_pcm(model, target = "Loyalty", noFolds = 10, reps = 10)
pcm
summary(pcm)
```

assess_pos

*PLS-POS: Prediction-Oriented Segmentation***Description**

Performs prediction-oriented segmentation (PLS-POS) to uncover unobserved heterogeneity in PLS-SEM models. PLS-POS uses a deterministic hill-climbing approach to maximize the sum of R-squared values across all endogenous constructs across K segments (Becker et al., 2013).

Usage

```
assess_pos(
  seminr_model,
  K = 2,
  nstart = 10,
  max_iter = 100,
  search_depth = NULL,
  min_segment_size = NULL,
  seed = 123
)
```

Arguments

seminr_model	An estimated SEMinR model from <code>estimate_pls()</code> .
K	Integer ≥ 2 . Number of segments to extract.
nstart	Number of random starting partitions (default 10). The best solution (highest objective) is returned.
max_iter	Maximum hill-climbing iterations per start (default 100).
search_depth	Maximum number of candidates to evaluate per iteration before accepting the first improvement (default = N, the sample size). Use a smaller value for faster but potentially less optimal results.
min_segment_size	Minimum observations per segment. Default is $\max(10, \text{max_predictors} + 2)$ where <code>max_predictors</code> is the largest number of predictors for any endogenous construct.
seed	Random seed for reproducibility.

Details

Unlike FIMIX-PLS, PLS-POS makes no distributional assumptions and can detect heterogeneity in both structural and (formative) measurement models.

Value

An S3 object of class "pos_analysis" with components:

K Number of segments

segment_assignment Integer vector of segment assignments (length N)

segment_sizes Named integer vector of segment sizes

segment_models List of K re-estimated semir model objects

segment_rsquared Matrix of R-squared values (endogenous x K)

segment_paths List of K path coefficient matrices

objective Sum of R-squared (objective criterion)

converged Whether the best start converged

iterations Number of iterations in the best start

nstart Number of random starts attempted

all_objectives Objective values for all starts

References

Becker, J.-M., Rai, A., Ringle, C. M., & Voelckner, F. (2013). Discovering Unobserved Heterogeneity in Structural Equation Models to Avert Validity Threats. *MIS Quarterly*, 37(3), 665-694.

See Also

[assess_fimix\(\)](#) for probabilistic (EM-based) segmentation

Examples

```
library(semir)

mobi_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Expectation", multi_items("CUEX", 1:3)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",    multi_items("CUSL", 1:3))
)
mobi_sm <- relationships(
  paths(from = "Image",      to = c("Expectation", "Satisfaction", "Loyalty")),
  paths(from = "Expectation", to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)
pls_model <- estimate_pls(data = mobi, measurement_model = mobi_mm,
  structural_model = mobi_sm)

pos_result <- assess_pos(pls_model, K = 2, nstart = 2, max_iter = 10,
  seed = 123)

print(pos_result)
summary(pos_result)
```

assess_pos_compare	<i>Compare PLS-POS Solutions Across K Values</i>
--------------------	--

Description

Runs [assess_pos](#) for each K in K_range and returns a comparison table of objective criterion (sum of R-squared) values. This helps identify the optimal number of segments.

Usage

```
assess_pos_compare(
  seminr_model,
  K_range = 2:5,
  nstart = 10,
  max_iter = 100,
  search_depth = NULL,
  min_segment_size = NULL,
  seed = 123
)
```

Arguments

seminr_model	An estimated SEMinR model from <code>estimate_pls()</code> .
K_range	Integer vector of K values to compare (default 2:5).
nstart	Number of random starting partitions (default 10). The best solution (highest objective) is returned.
max_iter	Maximum hill-climbing iterations per start (default 100).
search_depth	Maximum number of candidates to evaluate per iteration before accepting the first improvement (default = N, the sample size). Use a smaller value for faster but potentially less optimal results.
min_segment_size	Minimum observations per segment. Default is $\max(10, \text{max_predictors} + 2)$ where max_predictors is the largest number of predictors for any endogenous construct.
seed	Random seed for reproducibility.

Value

An S3 object of class "pos_comparison" with components:

solutions Named list of pos_analysis objects

fit_table Data frame comparing K values

K_range The K values compared

See Also

[assess_fimix_compare\(\)](#) for comparing FIMIX solutions across K

Examples

```
library(seminr)

mob_i_mm <- constructs(
  composite("Image",      multi_items("IMAG", 1:5)),
  composite("Expectation", multi_items("CUEX", 1:3)),
  composite("Satisfaction", multi_items("CUSA", 1:3)),
  composite("Loyalty",    multi_items("CUSL", 1:3))
)
mob_i_sm <- relationships(
  paths(from = "Image",      to = c("Expectation", "Satisfaction", "Loyalty")),
  paths(from = "Expectation", to = "Satisfaction"),
  paths(from = "Satisfaction", to = "Loyalty")
)
pls_model <- estimate_pls(data = mobi, measurement_model = mob_i_mm,
  structural_model = mob_i_sm)

pos_comp <- assess_pos_compare(pls_model, K_range = 2:3,
  nstart = 2, max_iter = 10, seed = 123)

print(pos_comp)
plot(pos_comp)
```

competes

Report Competing Split Criteria

Description

Shows all competing split variables at tree nodes, ranked by improvement. Useful for understanding which constructs nearly split instead of the chosen variable.

Usage

```
competes(node_id, dtree = NULL)
```

Arguments

node_id	A coa_dtree object (to get competing splits for all group roots), or an integer node ID from the rpart tree.
dtree	A coa_dtree object (from coa_result\$deviance_tree). Required only when node_id is an integer.

Details

When called with a single `coa_dtree` object, returns competing splits at all deviant group root nodes. When called with a node ID and a `coa_dtree` object, returns competing splits at that specific node.

Value

A data frame (or named list of data frames) with columns `criterion`, `sign`, `value`, and `improve`.

See Also

`group_rules()`, `assess_coa()`

congruence_test	<i>Bootstrap congruence coefficient test</i>
-----------------	--

Description

`congruence_test` conducts a bootstrapped significance test of congruence coefficients between all pairs of constructs in a PLS-SEM model.

Usage

```
congruence_test(  
  seminr_model,  
  nboot = 2000,  
  seed = 123,  
  alpha = 0.05,  
  threshold = 1,  
  reliability = c("rhoA", "rhoC", "cronbach", "one")  
)
```

Arguments

<code>seminr_model</code>	The SEMinR model for congruence analysis
<code>nboot</code>	The number of bootstrap subsamples to execute (defaults to 2000).
<code>seed</code>	The seed for reproducibility (defaults to 123).
<code>alpha</code>	The required level of alpha for statistical testing (defaults to 0.05). Used to compute confidence intervals.
<code>threshold</code>	The threshold with which to compare significance testing. H0: $rc < threshold$ (defaults to 1).
<code>reliability</code>	Which reliability estimate to place on the diagonal of the construct-correlation matrix: "rhoA" (default, matches SmartPLS), "rhoC" (composite reliability, the behaviour of <code>seminrExtras</code> $\leq$ 1.0.2), "cronbach" (Cronbach's alpha), or "one" (unity, as permitted by Franke et al. (2021) when reliabilities are unknown). Not

to be confused with the alpha argument above, which sets the confidence level. The option is named "cronbach" rather than "alpha" precisely to keep the two apart.

Franke et al. (2021, Eq. 2) specify "the reliabilities" without fixing an estimator, so all four are in specification. They agree wherever a construct has a well-defined internal consistency and diverge where it does not:

- **Single-item constructs** get 1 under "rhoA", "rhoC" and "cronbach" alike. In a model built entirely from single indicators the argument has no effect at all.
- **Mode B (formative) constructs** are where they part company. "rhoA" returns exactly 1, because internal consistency is undefined for a composite and nothing is estimated. "rhoC" and "cronbach" both still compute a value from the indicators. That is arguably the less honest choice — each presumes a measurement model the construct does not have — but it does keep one rule for every multi-item construct.

Note that this only ever affects pairs that **involve** a Mode B construct. Each column of the matrix carries exactly one reliability – its own construct's – so a coefficient between two reflective constructs is invariant to whatever sits on any other construct's diagonal.

"cronbach" is offered chiefly for comparison with covariance-based SEM, where rho_A is not available.

Details

The congruence coefficient (rc) measures how similarly two constructs relate to other constructs in the model. Values close to 1 indicate high congruence. The test evaluates $H_0: rc < \text{threshold}$ (default = 1).

Value

A list containing a matrix of congruence coefficients and significance test results for all construct pairs.

Model types

Interaction constructs are excluded from the construct set — entirely, not merely from the pair list, since Eq. 2 sums over the whole set. An interaction term's measurement is fixed by the product method rather than by theory, so its position in a nomological network is not interpretable. A message names any construct dropped. This matches `assess_cta()`, `assess_pos()`, `assess_pcm()` and `assess_cipma()`.

Higher-order models are not supported and are refused with a warning. Two-stage estimation replaces the lower-order constructs with a single higher-order composite, and it has not been established what belongs on that composite's diagonal, nor whether a congruence coefficient between a higher-order and a first-order construct is interpretable.

References

Franke, G. R., Sarstedt, M., & Danks, N. P. (2021). Assessing measure congruence in nomological networks. *Journal of Business Research*, 130, 318-334.

See Also

[assess_cvpat\(\)](#) and [assess_cvpat_compare\(\)](#) for predictive validity testing

Examples

```
# Load libraries
library(seminr)
library(seminrExtras)

# Create measurement model ----
corp_rep_mm <- constructs(
  composite("QUAL", multi_items("qual_", 1:8), weights = mode_B),
  composite("PERF", multi_items("perf_", 1:5), weights = mode_B),
  composite("CSOR", multi_items("csor_", 1:5), weights = mode_B),
  composite("ATTR", multi_items("attr_", 1:3), weights = mode_B),
  composite("COMP", multi_items("comp_", 1:3)),
  composite("LIKE", multi_items("like_", 1:3))
)

# Create structural model ----
corp_rep_sm <- relationships(
  paths(from = c("QUAL", "PERF", "CSOR", "ATTR"), to = c("COMP", "LIKE"))
)

# Estimate the model ----
corp_rep_pls_model <- estimate_pls(
  data = corp_rep_data,
  measurement_model = corp_rep_mm,
  structural_model = corp_rep_sm,
  missing = mean_replacement,
  missing_value = "-99")

# Assess the base model ----
congruence_test(seminr_model = corp_rep_pls_model,
  nboot = 20,
  seed = 123,
  alpha = 0.05,
  threshold = 1)
```

deviance_tree

Build Deviance Tree

Description

Grows a decision tree on predictive deviance scores to identify groups of cases with extreme (deviant) PD values.

Usage

```
deviance_tree(pd_result, deviance_bounds = c(0.025, 0.975))
```

Arguments

`pd_result` A `coa_deviance` object from `predictive_deviance()`.
`deviance_bounds` Two-element numeric vector of quantile bounds (default `c(0.025, 0.975)`).

Value

An object of class `coa_dtree` containing the `rpart` tree, deviant groups, group roots, unique deviants, and sorted PD values.

See Also

[assess_coa\(\)](#), [predictive_deviance\(\)](#), [group_rules\(\)](#), [competes\(\)](#)

<code>group_rules</code>	<i>Extract Split Rules for Deviant Groups</i>
--------------------------	---

Description

Returns the decision tree split criteria that define deviant group(s). When called with a single `coa_analysis` or `coa_dtree` object, returns rules for all deviant groups. When called with a group name and a `coa_analysis` object, returns rules for that specific group.

Usage

```
group_rules(group_name, coa_result = NULL)
```

Arguments

`group_name` A `coa_analysis` object (to get rules for all groups), a `coa_dtree` object, or a single uppercase letter identifying a specific group (e.g., "A").
`coa_result` A `coa_analysis` object from [assess_coa\(\)](#). Required only when `group_name` is a character group letter.

Value

A data frame (or named list of data frames) with columns `construct`, `gte`, and `lt` describing the construct score ranges that define each group.

See Also

[assess_coa\(\)](#), [competes\(\)](#)

plot.cipma_analysis *Plot cIPMA Results*

Description

Produces the Importance-Performance Map with optional NCA overlay. When type = "cipma" (default), necessary conditions are highlighted with filled red circles; other constructs use open blue circles. When type = "ipma", all constructs use the same symbol.

Usage

```
## S3 method for class 'cipma_analysis'
plot(
  x,
  type = c("cipma", "ipma"),
  importance_metric = c("unstandardized", "standardized"),
  ...
)
```

Arguments

x	A cipma_analysis object from assess_cipma().
type	One of "cipma" (default, with NCA overlay) or "ipma" (standard IPMA without NCA distinction).
importance_metric	One of "unstandardized" (default) or "standardized" to choose which total effect is plotted on the x-axis.
...	Additional arguments passed to plot().

plot.coa_analysis *Plot COA Results*

Description

Plot COA Results

Usage

```
## S3 method for class 'coa_analysis'
plot(x, type = c("pd", "groups", "tree"), ...)
```

Arguments

x	A coa_analysis object from assess_coa().
type	One of "pd" (predictive deviance scatter), "groups" (average construct scores per group), or "tree" (rpart tree diagram).
...	Additional arguments passed to the underlying plot function.

plot.fimix_analysis *Plot FIMIX-PLS Results*

Description

Plot FIMIX-PLS Results

Usage

```
## S3 method for class 'fimix_analysis'
plot(x, type = c("segments", "paths"), ...)
```

Arguments

x	A fimix_analysis object from assess_fimix().
type	One of "segments" (segment proportion bar plot) or "paths" (grouped bar plot of path coefficients across segments).
...	Additional arguments passed to plot().

plot.fimix_comparison *Plot FIMIX-PLS Comparison*

Description

Plot FIMIX-PLS Comparison

Usage

```
## S3 method for class 'fimix_comparison'
plot(
  x,
  type = c("criteria", "entropy"),
  criteria = c("AIC3", "AIC4", "BIC", "CAIC"),
  ...
)
```

Arguments

x	A fimix_comparison object from assess_fimix_compare().
type	One of "criteria" (line plot of IC vs K) or "entropy" (EN vs K).
criteria	Character vector of criteria to plot when type = "criteria" (default: c("AIC3", "AIC4", "BIC", "CAIC")).
...	Additional arguments passed to plot().

plot.nca_analysis	<i>Plot NCA Results</i>
-------------------	-------------------------

Description

Plot NCA Results

Usage

```
## S3 method for class 'nca_analysis'
plot(x, type = c("scatter", "effects"), ...)
```

Arguments

x	An nca_analysis object from assess_nca().
type	One of "scatter" (ceiling line scatter plots) or "effects" (bar plot of effect sizes).
...	Additional arguments passed to the underlying plot function.

plot.nca_esse	<i>Plot NCA-ESSE Results</i>
---------------	------------------------------

Description

Plot NCA-ESSE Results

Usage

```
## S3 method for class 'nca_esse'
plot(x, type = c("sensitivity", "difference"), ...)
```

Arguments

x	An nca_esse object from assess_nca_esse().
type	One of "sensitivity" (effect size vs threshold, Fig. 4 in Becker et al.) or "difference" (incremental empirical minus benchmark difference, Fig. 6 in Becker et al.).
...	Additional arguments passed to the underlying plot function.

plot.pcm_analysis *Plot PCM Results*

Description

Creates a grouped barplot of PCM values for each mediation path, with one bar per indicator and threshold lines for the weak/moderate/strong classification boundaries.

Usage

```
## S3 method for class 'pcm_analysis'
plot(
  x,
  metric = "RMSE",
  legend_pos = "topright",
  bar_col = "steelblue",
  neg_col = "tomato",
  cex.labels = 0.8,
  cex.legend = 0.75,
  ...
)
```

Arguments

x	An object of class pcm_analysis from assess_pcm .
metric	Character, either "RMSE" (default) or "MAE".
legend_pos	Character giving the legend position (e.g. "topright", "topleft", "bottomright"). Set to NULL or FALSE to suppress the legend entirely. Default "topright".
bar_col	Colour for positive-PCM bars (default "steelblue").
neg_col	Colour for negative-PCM bars (default "tomato").
cex.labels	Numeric expansion factor for the value labels printed on each bar. Default 0.8.
cex.legend	Numeric expansion factor for the legend text. Default 0.75.
...	Additional graphical parameters passed to barplot .

Examples

```
# Customise legend placement and bar colours
# plot(pcm, legend_pos = "topleft", bar_col = "dodgerblue")

# Suppress legend entirely
# plot(pcm, legend_pos = NULL)
```

pos_segments

Extract Segment-Specific PLS Models from PLS-POS Results

Description

Returns a list of fully re-estimated PLS models, one per segment, using the hard segment assignment from PLS-POS.

Usage

```
pos_segments(pos_result)
```

Arguments

pos_result A pos_analysis object from assess_pos().

Value

A named list of K semnr model objects.

See Also

[assess_pos\(\)](#) for running PLS-POS segmentation

predictive_deviance

Compute Predictive Deviance

Description

Computes per-case predictive deviance (PD) for a focal construct using k-fold cross-validated predictions from predict_pls().

Usage

```
predictive_deviance(
  semnr_model,
  focal_construct,
  technique = predict_DA,
  noFolds = 10,
  reps = 1,
  cores = NULL,
  seed = 123,
  predict_model = NULL
)
```

Arguments

seminr_model	An estimated SEMinR model.
focal_construct	Name of the construct to analyze.
technique	Prediction technique (default predict_DA).
noFolds	Number of CV folds (default 10).
reps	Number of CV repetitions (default 1).
cores	Number of cores for parallel CV (default NULL).
seed	Random seed (default 123).
predict_model	Optional pre-computed predict_pls_model object.

Details

PD = in-sample fitted score - out-of-sample predicted score. High PD indicates a case that is well-fitted in-sample but poorly predicted out-of-sample, suggesting overfit.

Value

An object of class `coa_deviance` containing PD values, metrics, and the data frame used for tree construction.

See Also

[assess_coa\(\)](#), [deviance_tree\(\)](#)

unstable_params	<i>Assess Parameter Instability</i>
-----------------	-------------------------------------

Description

Identifies unstable model parameters by removing each deviant group and re-estimating the model, then computing the difference in specified parameters.

Usage

```
unstable_params(seminr_model, deviant_groups, params = "path_coef")
```

Arguments

seminr_model	An estimated SEMinR model.
deviant_groups	A <code>coa_dtree</code> object from deviance_tree() , or a named list of integer vectors of case indices. When a <code>coa_dtree</code> is passed, the <code>\$deviant_groups</code> element is extracted automatically.
params	Character vector of model parameters to diff (default "path_coef").

Value

An object of class `coa_unstable`: a named list (one entry per group) each containing cases and `param_diffs`.

See Also

[assess_coa\(\)](#), [deviance_tree\(\)](#)

Index

assess_cipma, [2](#), [18](#)
assess_cipma(), [31](#)
assess_coa, [5](#)
assess_coa(), [30](#), [33](#), [39](#), [40](#)
assess_cta, [7](#)
assess_cta(), [31](#)
assess_cvpat, [9](#)
assess_cvpat(), [12](#), [32](#)
assess_cvpat_compare, [11](#)
assess_cvpat_compare(), [11](#), [32](#)
assess_fimix, [14](#), [16](#), [17](#)
assess_fimix(), [27](#)
assess_fimix_compare, [15](#), [16](#)
assess_fimix_compare(), [29](#)
assess_ipma, [4](#), [18](#)
assess_nca, [4](#), [19](#), [23](#)
assess_nca_esse, [21](#)
assess_pcm, [24](#), [37](#)
assess_pcm(), [31](#)
assess_pos, [26](#), [28](#)
assess_pos(), [31](#), [38](#)
assess_pos_compare, [28](#)

barplot, [37](#)

competes, [29](#)
competes(), [6](#), [33](#)
congruence_test, [30](#)
congruence_test(), [9](#)

deviance_tree, [32](#)
deviance_tree(), [6](#), [39](#), [40](#)

group_rules, [33](#)
group_rules(), [6](#), [30](#), [33](#)

plot.cipma_analysis, [34](#)
plot.coa_analysis, [34](#)
plot.fimix_analysis, [35](#)
plot.fimix_comparison, [35](#)
plot.nca_analysis, [36](#)

plot.nca_esse, [36](#)
plot.pcm_analysis, [37](#)
pos_segments, [38](#)
predict_pls, [25](#)
predictive_deviance, [38](#)
predictive_deviance(), [6](#), [33](#)

semnr::estimate_pls(), [7](#)

unstable_params, [39](#)
unstable_params(), [6](#)