Package 'leunbachR'

June 2, 2026

Title Leunbach Test Equating

Version 0.1.0

URL https://github.com/pgmj/leunbachR,
https://pgmj.github.io/leunbachR/

BugReports https://github.com/pgmj/leunbachR/issues

Description Implements the Leunbach test equating method, following the 'DIGRAM' software written by Svend Kreiner. Both direct and indirect equating are available, with parametric bootstrap standard errors and diagnostic statistics including the Goodman-Kruskal gamma test and orbit analysis for person fit. See Adroher et al. (2019) <doi:10.1186/s12874-019-0768-y> for details of the method.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, mirai, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website rmarkdown, dplyr, tidyr, ggplot2, knitr, mirt, equate, SNSequate, kequate, brglm2

Config/testthat/edition 3

NeedsCompilation no

Author Magnus Johansson [aut, cre] (ORCID: <https://orcid.org/0000-0003-1669-592X>)

Maintainer Magnus Johansson <pgmj@pm.me>

Repository CRAN

Date/Publication 2026-06-02 08:20:02 UTC

Contents

`analyze_orbits` *Analyze Orbits for Leunbach Model*

Description

Performs orbit analysis for the Leunbach model. For each total score (orbit), computes the expected distribution of (Test1, Test2) score pairs and cumulative probabilities for assessing person fit.

Usage

```
analyze_orbits(fit, alpha = 0.05, verbose = FALSE)
```

Arguments

`fit`	A leunbach_ipf object from leunbach_ipf()
`alpha`	Significance level for critical values (default 0.05)
`verbose`	Print detailed output

Details

For each total score S = X + Y, the orbit consists of all valid (X, Y) pairs. The expected probability of each cell within an orbit is: P(X = x | S = s) = (gamma_x * delta_y) / sigma_s

Cumulative probabilities are computed in both directions:

- Left-to-right: P(X <= x | S = s) - tests if Test1 score is unusually low
- Right-to-left: P(X >= x | S = s) - tests if Test1 score is unusually high

Value

A list of class "leunbach_orbits" containing:

- orbits: Matrix of expected percentages within each total score
- left_right: Cumulative probabilities P(X <= x | S = s)
- right_left: Cumulative probabilities P(X >= x | S = s)
- crit_left: Critical values for left tail by total score
- crit_right: Critical values for right tail by total score
- crit_values: Combined critical values (two-tailed)
- expected_critical: Expected number of persons with significant differences
- variance_expected: Variance of expected critical count
- n_significant: Number of observed persons with significant differences
- orbit_df: Degrees of freedom for each orbit

Examples

```
set.seed(123)
n <- 400
theta <- rnorm(n)
test1 <- pmin(pmax(round(3 + 1.5 * theta + rnorm(n, sd = 0.8)), 0), 6)
test2 <- pmin(pmax(round(2.5 + 1.3 * theta + rnorm(n, sd = 0.7)), 0), 5)
fit <- leunbach_ipf(data.frame(test1, test2),
                    max_score1 = 6, max_score2 = 5)
orb <- analyze_orbits(fit)
print(orb)
```

---

diagnose_equating *Diagnostic function for equating*

Description

Diagnoses potential issues with person parameter estimation by displaying theta values and expected scores for each source score.

Usage

```
diagnose_equating(
  fit,
  direction = c("1to2", "2to1"),
  method = c("optimize", "newton")
)
```

Arguments

fit	A leunbach_ipf object
direction	Direction of equating: "1to2" or "2to1"
method	Optimization method: "optimize" or "newton"

Value

Invisibly returns NULL; called for its side effect of printing a diagnostic table.

Examples

```
set.seed(123)
n <- 400
theta <- rnorm(n)
test1 <- pmin(pmax(round(3 + 1.5 * theta + rnorm(n, sd = 0.8)), 0), 6)
test2 <- pmin(pmax(round(2.5 + 1.3 * theta + rnorm(n, sd = 0.7)), 0), 5)
fit <- leunbach_ipf(data.frame(test1, test2),
                    max_score1 = 6, max_score2 = 5)
diagnose_equating(fit, direction = "1to2")
```

---

get_equating_table *Extract equating table with bootstrap confidence intervals*

Description

Extracts equating results from a bootstrap object as a clean data frame with log theta values, confidence intervals and standard errors.

Usage

```
get_equating_table(boot, direction = c("1to2", "2to1"))
```

Arguments

boot	A leunbach_bootstrap object
direction	"1to2" or "2to1"

Value

A data frame with equating results and CIs

Examples

```
set.seed(123)
n <- 300
theta <- rnorm(n)
test1 <- pmin(pmax(round(3 + 1.5 * theta + rnorm(n, sd = 0.8)), 0), 6)
test2 <- pmin(pmax(round(2.5 + 1.3 * theta + rnorm(n, sd = 0.7)), 0), 5)
fit <- leunbach_ipf(data.frame(test1, test2),
                    max_score1 = 6, max_score2 = 5)
boot <- leunbach_bootstrap(fit, nsim = 25, parallel = FALSE, seed = 1)
get_equating_table(boot, direction = "1to2")
```

---

get_indirect_equating_table

Get indirect equating table

Description

Get indirect equating table

Usage

```
get_indirect_equating_table(boot)
```

Arguments

boot	A leunbach_indirect_bootstrap object

Value

A data frame with equating results and CIs

Examples

```
set.seed(123)
n <- 300
theta1 <- rnorm(n)
a <- pmin(pmax(round(3 + 1.5 * theta1 + rnorm(n, sd = 0.8)), 0), 6)
b1 <- pmin(pmax(round(2.5 + 1.3 * theta1 + rnorm(n, sd = 0.7)), 0), 5)
theta2 <- rnorm(n)
b2 <- pmin(pmax(round(2.5 + 1.3 * theta2 + rnorm(n, sd = 0.7)), 0), 5)
cc <- pmin(pmax(round(3 + 1.4 * theta2 + rnorm(n, sd = 0.8)), 0), 6)
fit_ab <- leunbach_ipf(data.frame(a, b1), max_score1 = 6, max_score2 = 5)
fit_bc <- leunbach_ipf(data.frame(b2, cc), max_score1 = 5, max_score2 = 6)
```

```
boot <- leunbach_indirect_bootstrap(fit_ab, fit_bc, nsim = 25,
                                    parallel = FALSE, seed = 1)
get_indirect_equating_table(boot)
```

`get_orbit` *Get orbit distribution for a specific total score*

Description

Get orbit distribution for a specific total score

Usage

```
get_orbit(orbits, total_score)
```

Arguments

`orbits`	A leunbach_orbits object
`total_score`	The total score to examine

Value

A data frame with the orbit distribution

Examples

```
set.seed(123)
n <- 400
theta <- rnorm(n)
test1 <- pmin(pmax(round(3 + 1.5 * theta + rnorm(n, sd = 0.8)), 0), 6)
test2 <- pmin(pmax(round(2.5 + 1.3 * theta + rnorm(n, sd = 0.7)), 0), 5)
fit <- leunbach_ipf(data.frame(test1, test2),
                    max_score1 = 6, max_score2 = 5)
orb <- analyze_orbits(fit)
get_orbit(orb, total_score = 5)
```

`leunbach_bootstrap` *Parametric Bootstrap for Leunbach Model with Standard Error of Equating*

Description

Performs parametric bootstrapping to assess significance of tests and compute standard errors of equating (SEE) with confidence intervals. Supports parallel processing using the mirai package.

Usage

```
leunbach_bootstrap(
  fit,
  nsim = 1000,
  conf_level = 0.95,
  see_type = c("rounded", "expected"),
  method = c("optimize", "newton"),
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

`fit`	A leunbach_ipf object from leunbach_ipf()
`nsim`	Number of bootstrap samples (default: 1000)
`conf_level`	Confidence level for intervals (default: 0.95)
`see_type`	Type of SEE calculation: "rounded" uses rounded (integer) scores, "expected" uses continuous expected scores
`method`	Optimization method for person parameter estimation: "optimize" (default) uses stats::optimize() with Brent's method, "newton" uses custom Newton-Raphson with bisection fallback
`parallel`	Use parallel processing if mirai package is available (default: TRUE)
`n_cores`	Number of cores to use for parallel processing. Default NULL uses all available cores minus one.
`verbose`	Print progress messages
`seed`	Random seed for reproducibility (optional)

Value

A list of class "leunbach_bootstrap" containing bootstrap results and standard errors of equating

Examples

```
# Simulate paired test score data
set.seed(123)
n <- 300
theta <- rnorm(n)
test1 <- pmin(pmax(round(3 + 1.5 * theta + rnorm(n, sd = 0.8)), 0), 6)
test2 <- pmin(pmax(round(2.5 + 1.3 * theta + rnorm(n, sd = 0.7)), 0), 5)
fit <- leunbach_ipf(data.frame(test1, test2),
                    max_score1 = 6, max_score2 = 5)

# Sequential bootstrap with few iterations for the example
boot <- leunbach_bootstrap(fit, nsim = 25, parallel = FALSE, seed = 1)
print(boot)
```

`leunbach_equate` *Equate Scores Between Tests using Leunbach Model*

Description

Creates equating tables to convert scores from one test to another using the estimated score parameters from the Leunbach model.

Usage

```
leunbach_equate(
  fit,
  direction = c("1to2", "2to1"),
  method = c("optimize", "newton"),
  verbose = FALSE
)
```

Arguments

`fit`	A leunbach_ipf object from leunbach_ipf()
`direction`	Direction of equating: "1to2" (Test1 to Test2) or "2to1" (Test2 to Test1)
`method`	Optimization method for person parameter estimation: "optimize" (default) uses stats::optimize() with Brent's method, "newton" uses custom Newton-Raphson with bisection fallback
`verbose`	Print detailed output

Value

A list of class "leunbach_equating" containing:

- equating_table: Data frame with original scores, theta, expected equated scores, and rounded scores

- direction: Direction of equating
- method: Optimization method used
- fit: Original leunbach_ipf object

Examples

```
# Simulate paired test score data
set.seed(123)
n <- 400
theta <- rnorm(n)
test1 <- pmin(pmax(round(3 + 1.5 * theta + rnorm(n, sd = 0.8)), 0), 6)
test2 <- pmin(pmax(round(2.5 + 1.3 * theta + rnorm(n, sd = 0.7)), 0), 5)
fit <- leunbach_ipf(data.frame(test1, test2),
                    max_score1 = 6, max_score2 = 5)

# Equate Test 1 scores onto the Test 2 metric
eq <- leunbach_equate(fit, direction = "1to2")
print(eq)
```

---

leunbach_indirect_bootstrap

Bootstrap for Indirect Equating

Description

Performs parametric bootstrapping for indirect equating to assess standard errors of the indirect equated scores.

Usage

```
leunbach_indirect_bootstrap(
  fit_ab,
  fit_bc,
  direction_ab = c("1to2", "2to1"),
  direction_bc = c("1to2", "2to1"),
  nsim = 1000,
  conf_level = 0.95,
  see_type = c("rounded", "expected"),
  method = c("optimize", "newton"),
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

fit_ab	A leunbach_ipf object for the A-B equating
fit_bc	A leunbach_ipf object for the B-C equating
direction_ab	Direction for A-B equating: "1to2" or "2to1"
direction_bc	Direction for B-C equating: "1to2" or "2to1"
nsim	Number of bootstrap samples (default: 1000)
conf_level	Confidence level for intervals (default: 0.95)
see_type	Type of SEE calculation: "rounded" or "expected"
method	Optimization method: "optimize" (default) or "newton"
parallel	Use parallel processing if mirai is available (default: TRUE)
n_cores	Number of cores for parallel processing
verbose	Print progress messages
seed	Random seed for reproducibility

Value

A list of class "leunbach_indirect_bootstrap" containing:

- indirect_eq: The observed indirect equating object
- Bootstrap results and standard errors
- Bootstrap p-values for LR and Gamma tests for both equatings

Examples

```
set.seed(123)
n <- 300
theta1 <- rnorm(n)
a <- pmin(pmax(round(3 + 1.5 * theta1 + rnorm(n, sd = 0.8)), 0), 6)
b1 <- pmin(pmax(round(2.5 + 1.3 * theta1 + rnorm(n, sd = 0.7)), 0), 5)
theta2 <- rnorm(n)
b2 <- pmin(pmax(round(2.5 + 1.3 * theta2 + rnorm(n, sd = 0.7)), 0), 5)
cc <- pmin(pmax(round(3 + 1.4 * theta2 + rnorm(n, sd = 0.8)), 0), 6)
fit_ab <- leunbach_ipf(data.frame(a, b1), max_score1 = 6, max_score2 = 5)
fit_bc <- leunbach_ipf(data.frame(b2, cc), max_score1 = 5, max_score2 = 6)

boot <- leunbach_indirect_bootstrap(fit_ab, fit_bc,
                                    direction_ab = "1to2",
                                    direction_bc = "1to2",
                                    nsim = 25, parallel = FALSE,
                                    seed = 1)
print(boot)
```

`leunbach_indirect_equate` *Indirect Equating via an Anchor Test*

Description

Performs indirect equating from Test A to Test C via an anchor Test B. This chains two direct equatings: A -> B and B -> C.

Usage

```
leunbach_indirect_equate(
  fit_ab,
  fit_bc,
  direction_ab = c("1to2", "2to1"),
  direction_bc = c("1to2", "2to1"),
  method = c("optimize", "newton"),
  verbose = FALSE
)
```

Arguments

`fit_ab`	A leunbach_ipf object for the A-B equating (Tests A and B)
`fit_bc`	A leunbach_ipf object for the B-C equating (Tests B and C)
`direction_ab`	Direction for A-B equating: "1to2" or "2to1"
`direction_bc`	Direction for B-C equating: "1to2" or "2to1"
`method`	Optimization method: "optimize" (default) or "newton"
`verbose`	Print detailed output

Details

Indirect equating works by chaining two direct equatings:

1. For a score x on Test A, find the expected score on Test B (typically non-integer)
2. Find expected Test C scores for the integer B scores below and above
3. Interpolate to get the expected Test C score for the non-integer B score
4. Round to get the equated integer score

Value

A list of class "leunbach_indirect" containing:

- equating_table: Data frame with source scores, expected equated scores, and rounded scores
- eq_ab: Direct equating object for A -> B
- eq_bc: Direct equating object for B -> C
- fit_ab: Original leunbach_ipf object for A-B
- fit_bc: Original leunbach_ipf object for B-C

Examples

```
# Simulate scores for three tests, with B serving as the anchor.
# Group 1 takes tests A and B; group 2 takes tests B and C.
set.seed(123)
n <- 400
theta1 <- rnorm(n)
a <- pmin(pmax(round(3 + 1.5 * theta1 + rnorm(n, sd = 0.8)), 0), 6)
b1 <- pmin(pmax(round(2.5 + 1.3 * theta1 + rnorm(n, sd = 0.7)), 0), 5)

theta2 <- rnorm(n)
b2 <- pmin(pmax(round(2.5 + 1.3 * theta2 + rnorm(n, sd = 0.7)), 0), 5)
cc <- pmin(pmax(round(3 + 1.4 * theta2 + rnorm(n, sd = 0.8)), 0), 6)

fit_ab <- leunbach_ipf(data.frame(a, b1), max_score1 = 6, max_score2 = 5)
fit_bc <- leunbach_ipf(data.frame(b2, cc), max_score1 = 5, max_score2 = 6)

# Indirect equating: A -> B -> C
indirect <- leunbach_indirect_equate(fit_ab, fit_bc,
                                     direction_ab = "1to2",
                                     direction_bc = "1to2")
print(indirect)
```

`leunbach_ipf` *Estimate Leunbach Score Parameters using Iterative Proportional Fitting*

Description

Estimates score parameters for Leunbach test equating based on the power series distribution framework. The total score distribution in Rasch models follows a power series distribution where score parameters are defined by generalized symmetric functions of the item parameters.

Usage

```
leunbach_ipf(
  data,
  max_score1 = NULL,
  max_score2 = NULL,
  max_iter = 1000,
  tol = 1e-10,
  verbose = FALSE
)
```

Arguments

`data`	A data. frame or matrix with two columns: Column 1: Sum scores from Test 1 Column 2: Sum scores from Test 2

max_score1	Maximum possible score for Test 1 (default: max observed)
max_score2	Maximum possible score for Test 2 (default: max observed)
max_iter	Maximum number of iterations for IPF
tol	Convergence tolerance
verbose	Print iteration progress

Details

The model is based on the power series distribution (PSD) for sum scores in Rasch models. For a test with k polytomous items, the probability of obtaining sum score r is:

$$P(R = r|\theta) = \frac{\gamma_r \theta^r}{\sum_s \gamma_s \theta^s}$$

where γ_r is the generalized symmetric function of order r of the item parameters, and $\theta = exp(\beta)$ is the person parameter.

The model is over-parameterized, so we apply identification constraints:

- γ_{min} = 1 (lowest observed score parameter fixed at 1)
- δ_{min} = 1 (lowest observed score parameter fixed at 1)
- The highest score parameters are constrained so that their product equals the product of all score parameters (Rasch-style constraint)

The sigma parameters for the total score (S = X + Y) are computed as:

$$\sigma_s = \sum_{x+y=s} \gamma_x \delta_y$$

Value

A list of class "leunbach_ipf" containing:

- gamma: Score parameters for Test1 (generalized symmetric functions)
- delta: Score parameters for Test2 (generalized symmetric functions)
- sigma: Score parameters for total score (Test1 + Test2)
- log_gamma: Log of gamma parameters
- log_delta: Log of delta parameters
- log_sigma: Log of sigma parameters
- fitted: Fitted frequency table
- observed: Observed frequency table
- iterations: Number of iterations to convergence
- converged: Logical indicating convergence
- test1_scores: Score values for Test1
- test2_scores: Score values for Test2
- total_scores: Score values for total score

- g_sq: Likelihood ratio test statistic
- chi_sq: Pearson chi-square statistic
- df: Degrees of freedom
- p_value: P-value for likelihood ratio test

References

Leunbach, G. (1976). A probabilistic measurement model for assessing whether two tests measure the same personal factor. Copenhagen: Danish Institute for Educational Research.

Adroher, N. D., Kreiner, S., Young, C., Mills, R., & Tennant, A. (2019). Test equating sleep scales: Applying the Leunbach's model. BMC Medical Research Methodology, 19(1), 141. doi:10.1186/s128740190768y

Kreiner, S. (2007). Validity and objectivity: Reflections on the role and nature of Rasch models. Nordic Psychology, 59(3), 268-298. doi:10.1027/19012276.59.3.268

Examples

```
# Simulate test score data
set.seed(123)
n <- 500
theta <- rnorm(n)
test1 <- pmin(pmax(round(3 + 1.5 * theta + rnorm(n, sd = 0.8)), 0), 6)
test2 <- pmin(pmax(round(2.5 + 1.3 * theta + rnorm(n, sd = 0.7)), 0), 5)
score_data <- data.frame(test1 = test1, test2 = test2)

# Estimate parameters (specifying max possible scores)
fit <- leunbach_ipf(score_data, max_score1 = 6, max_score2 = 5, verbose = TRUE)
print(fit)
summary(fit)
```

---

plot.leunbach_ipf *Plot method for leunbach_ipf objects*

Description

Plot method for leunbach_ipf objects

Usage

```
## S3 method for class 'leunbach_ipf'
plot(x, type = c("parameters", "residuals", "observed", "sigma"), ...)
```

Arguments

x	A leunbach_ipf object.
type	One of "parameters", "residuals", "observed", or "sigma". Selects what to display.
...	Further arguments passed to or from other methods.

Value

Invisibly returns `x`.

---

plot.leunbach_orbits *Plot method for leunbach_orbits objects*

Description

Plot method for leunbach_orbits objects

Usage

```
## S3 method for class 'leunbach_orbits'
plot(x, type = c("orbits", "cumulative", "significant"), ...)
```

Arguments

`x`	A `leunbach_orbits` object.
`type`	One of `"orbits"`, `"cumulative"`, or `"significant"`. Selects what to display.
`...`	Further arguments passed to or from other methods.

Value

Invisibly returns `x`.

---

print.leunbach_bootstrap *Print method for leunbach_bootstrap objects*

Description

Print method for leunbach_bootstrap objects

Usage

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

Arguments

`x`	A `leunbach_bootstrap` object.
`...`	Further arguments passed to or from other methods.

Value

Invisibly returns `x`.

`print.leunbach_equating` *Print method for leunbach_equating objects*

Description

Print method for leunbach_equating objects

Usage

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

Arguments

`x`	A `leunbach_equating` object.
`...`	Further arguments passed to or from other methods.

Value

Invisibly returns `x`.

`print.leunbach_indirect` *Print method for leunbach_indirect objects*

Description

Print method for leunbach_indirect objects

Usage

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

Arguments

`x`	A `leunbach_indirect` object.
`...`	Further arguments passed to or from other methods.

Value

Invisibly returns `x`.

print.leunbach_indirect_bootstrap *Print method for leunbach_indirect_bootstrap objects*

Description

Print method for leunbach_indirect_bootstrap objects

Usage

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

Arguments

x	A leunbach_indirect_bootstrap object.
...	Further arguments passed to or from other methods.

Value

Invisibly returns x.

print.leunbach_ipf *Print method for leunbach_ipf objects*

Description

Print method for leunbach_ipf objects

Usage

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

Arguments

x	A leunbach_ipf object.
...	Further arguments passed to or from other methods.

Value

Invisibly returns x.

print.leunbach_orbits *Print method for leunbach_orbits objects*

Description

Print method for leunbach_orbits objects

Usage

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

Arguments

x	A leunbach_orbits object.
...	Further arguments passed to or from other methods.

Value

Invisibly returns x.

summary.leunbach_indirect_bootstrap *Summary method for leunbach_indirect_bootstrap objects*

Description

Summary method for leunbach_indirect_bootstrap objects

Usage

```
## S3 method for class 'leunbach_indirect_bootstrap'
summary(object, ...)
```

Arguments

object	A leunbach_indirect_bootstrap object.
...	Further arguments passed to or from other methods.

Value

Invisibly returns object.

summary.leunbach_ipf *Summary method for leunbach_ipf objects*

Description

Summary method for leunbach_ipf objects

Usage

```
## S3 method for class 'leunbach_ipf'
summary(object, ...)
```

Arguments

object	A leunbach_ipf object.
...	Further arguments passed to or from other methods.

Value

Invisibly returns a list of fit statistics.

summary.leunbach_orbits *Summary method for leunbach_orbits objects*

Description

Summary method for leunbach_orbits objects

Usage

```
## S3 method for class 'leunbach_orbits'
summary(object, ...)
```

Arguments

object	A leunbach_orbits object.
...	Further arguments passed to or from other methods.

Value

Invisibly returns object.

Index