Package 'missalpha'

August 29, 2025

Type Package

Title Find Range of Cronbach Alpha with a Dataset Including Missing Data

Version 0.1.0

Maintainer Biying Zhou <biying.zhou@psu.edu>

Description Provides functions to calculate the minimum and maximum possible values of Cronbach's alpha when item-level missing data are present. Cronbach's alpha (Cronbach, 1951 <doi:10.1007/BF02310555>) is one of the most widely used measures of internal consistency in the social, behavioral, and medical sciences (Bland & Altman, 1997 <doi:10.1136/bmj.314.7080.572>; Tavakol & Dennick, 2011 <doi:10.5116/ijme.4dfb.8dfd>). However, conventional implementations assume complete data, and listwise deletion is often applied when missingness occurs, which can lead to biased or overly optimistic reliability estimates (Enders, 2003 <doi:10.1037/1082-989X.8.3.322>). This package implements computational strategies including enumeration, Monte Carlo sampling, and optimization algorithms (e.g., Genetic Algorithm, Differential Evolution, Sequential Least Squares Programming) to obtain sharp lower and upper bounds of Cronbach's alpha under arbitrary missing data patterns. The approach is motivated by Manski's partial identification framework and pessimistic bounding ideas from optimization literature.

License MIT + file LICENSE

Depends R (>= 3.1.0),

Imports GA,
DEoptim,
nloptr,
stats,

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

Contents

compute_alpha_max 2
compute_alpha_min 3

cronbachs_alpha 5
cronbach_alpha_enum 7
cronbach_alpha_rough 7
display_all 8
examine_alpha_bound 9
generate_scores_mat_bernoulli 11
qp_solver 12
qp_solver_DEoptim 13
qp_solver_GA 14
qp_solver_nloptr 15
sample 16

Index **17**

`compute_alpha_max` *Compute Maximum Possible Alpha Value*

Description

This function calculates the maximum possible value of Cronbach's alpha by using a binary search algorithm with optimization methods. The function iteratively narrows the bounds of alpha until the desired tolerance level is reached.

Usage

```
compute_alpha_max(
  n_person,
  sigma_x_info,
  sigma_y_info,
  score_max = 1,
  alpha_lb = 0,
  alpha_ub = 1,
  tol = 0.001,
  num_try = 1,
  method = "GA",
  ...
)
```

Arguments

`n_person`	An integer specifying the number of individuals (rows) in the score matrix.
`sigma_x_info`	A list containing the quadratic function information for sigma_x. It should include: • A matrix representing the quadratic coefficients for sigma_x. • A vector representing the linear coefficients for sigma_x. • A scalar constant term for sigma_x.
`sigma_y_info`	A list containing the quadratic function information for sigma_y. It should include: • A matrix representing the quadratic coefficients for sigma_y. • A vector representing the linear coefficients for sigma_y.

- A scalar constant term for sigma_y.

score_max	An integer specifying the largest possible score for any test item. Default is 1.
alpha_lb	A numeric value specifying the lower bound of alpha, usually set to 0.0.
alpha_ub	A numeric value specifying the upper bound of alpha, usually set to 1.0.
tol	A numeric value representing the desired accuracy for narrowing down the bounds between alpha_lb and alpha_ub. Default is 1e-3.
num_try	An integer specifying the number of times to run the optimization algorithm in each iteration. Default is 1.
method	A character string specifying the optimization method to be used. Options are 'GA' (Genetic Algorithm), 'DEoptim' (Differential Evolution), and 'nloptr' (Sequential Least Squares Programming). Default is 'GA'.
...	Additional parameters passed to the optimization algorithm.

Details

This function finds the maximum possible Cronbach's alpha by using an iterative binary search algorithm. It evaluates the feasibility of each midpoint value of alpha by solving the corresponding optimization problem with the chosen method.

The optimization methods can be specified via the method parameter, and additional control parameters for the optimization methods can be passed through the ... argument. The function adjusts the upper and lower bounds of alpha until the tolerance criterion is met.

Value

A numeric value representing the maximum possible Cronbach's alpha.

See Also

compute_alpha_min, examine_alpha_bound

---

compute_alpha_min *Compute Minimum Possible Alpha Value*

Description

This function calculates the minimum possible value of Cronbach's alpha by using a binary search algorithm with optimization methods. The function iteratively narrows the bounds of alpha until the desired tolerance level is reached.

Usage

```
compute_alpha_min(
  n_person,
  sigma_x_info,
  sigma_y_info,
  score_max = 1,
  alpha_lb = 0,
  alpha_ub = 1,
  tol = 0.001,
```

```
  num_try = 1,
  method = "GA",
  ...
)
```

Arguments

n_person	An integer specifying the number of individuals (rows) in the score matrix.
sigma_x_info	A list containing the quadratic function information for sigma_x. It should include: • A matrix representing the quadratic coefficients for sigma_x. • A vector representing the linear coefficients for sigma_x. • A scalar constant term for sigma_x.
sigma_y_info	A list containing the quadratic function information for sigma_y. It should include: • A matrix representing the quadratic coefficients for sigma_y. • A vector representing the linear coefficients for sigma_y. • A scalar constant term for sigma_y.
score_max	An integer specifying the largest possible score for any test item. Default is 1.
alpha_lb	A numeric value specifying the lower bound of alpha, usually set to 0.0.
alpha_ub	A numeric value specifying the upper bound of alpha, usually set to 1.0.
tol	A numeric value representing the desired accuracy for narrowing down the bounds between alpha_lb and alpha_ub. Default is 1e-3.
num_try	An integer specifying the number of times to run the optimization algorithm in each iteration. Default is 1.
method	A character string specifying the optimization method to be used. Options are 'GA' (Genetic Algorithm), 'DEoptim' (Differential Evolution), and 'nloptr' (Sequential Least Squares Programming). Default is 'GA'.
...	Additional parameters passed to the optimization algorithm.

Details

This function finds the minimum possible Cronbach's alpha by using an iterative binary search algorithm. It evaluates the feasibility of each midpoint value of alpha by solving the corresponding optimization problem with the chosen method.

The optimization methods can be specified via the method parameter, and additional control parameters for the optimization methods can be passed through the ... argument. The function adjusts the upper and lower bounds of alpha until the tolerance criterion is met.

Value

A numeric value representing the minimum possible Cronbach's alpha.

See Also

compute_alpha_max, examine_alpha_bound

`cronbachs_alpha` *Compute Lower and Upper Bound of Cronbach's Alpha*

Description

This function computes the lower and upper bound of Cronbach's alpha using various methods such as enumeration, random sampling, or optimization algorithms. The function also supports rough approximations and allows integer-only or floating-point scores during sampling.

Usage

```
cronbachs_alpha(
  scores_mat,
  score_max,
  tol = 0.001,
  num_random = 1000,
  enum_all = FALSE,
  rough = FALSE,
  num_opt = 1,
  int_only = TRUE,
  method = "GA",
  ...
)
```

Arguments

`scores_mat`	A matrix where rows represent persons and columns represent tests (or items), providing the performance of a person on a test. NA should be used for missing values.
`score_max`	An integer indicating the largest possible score of the test.
`tol`	A numeric value representing the desired accuracy in computing the lower and upper bound of Cronbach's alpha.
`num_random`	An integer specifying the number of random samples used in estimating the lower and upper bound. Default is 1000.
`enum_all`	A logical value indicating whether to enumerate all possible scores for Cronbach's alpha. Default is FALSE.
`rough`	A logical value indicating whether to compute a rough approximation of Cronbach's alpha bounds. Default is FALSE.
`num_opt`	An integer specifying the number of times to run the optimization algorithm. Default is 1.
`int_only`	A logical value indicating whether the random sampling should be restricted to integer-only scores. Default is TRUE.
`method`	A character string specifying the optimization method to be used ('GA', 'DEoptim', 'nloptr'). Default is 'GA'.
`...`	Additional parameters passed to the optimization algorithm.

Value

A list containing:

`alpha_min_opt`	The smallest possible Cronbach's alpha computed by the optimization algorithm (if used).
`alpha_max_opt`	The largest possible Cronbach's alpha computed by the optimization algorithm (if used).
`alpha_min_enum`	The smallest possible Cronbach's alpha obtained by enumerating all possible scores (if used).
`alpha_max_enum`	The largest possible Cronbach's alpha obtained by enumerating all possible scores (if used).
`alpha_min_rough`	The smallest possible Cronbach's alpha obtained by rough approximation (if used).
`alpha_max_rough`	The largest possible Cronbach's alpha obtained by rough approximation (if used).
`method`	The optimization method used.
`runtime`	The total computation time in seconds.

See Also

`compute_alpha_min`, `compute_alpha_max`, `cronbach_alpha_enum`, `cronbach_alpha_rough`, `generate_scores_mat`, `qp_solver`

Examples

```
## Not run:
# Example 1: Run `cronbachs_alpha` with a sample matrix
scores_mat <- matrix(c(
  NaN, 1, 0, 0, 0, 0, 0, 0, NaN, 0, 0, 0,
  2, 0, 0, 1, NaN, 0, 0, 0, 0, 0, 0, 0,
  1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1
), nrow = 10, ncol = 4, byrow = TRUE)

result <- cronbachs_alpha(scores_mat, score_max = 4, enum_all = FALSE)
print(result$alpha_min_opt)
print(result$alpha_max_opt)

# Example 2: Generate a Bernoulli matrix and compute Cronbach's alpha
score_max <- 2
scores_mat_bernoulli <- generate_scores_mat_bernoulli(50, 10, 20, score_max)
result <- cronbachs_alpha(scores_mat_bernoulli, score_max, enum_all = FALSE)
print(result$alpha_min_opt)
print(result$alpha_max_opt)

# Example 3: Using a predefined dataset from missalpha
scores_df <- missalpha::sample
scores_mat <- as.matrix(scores_df)
result <- cronbachs_alpha(scores_mat, score_max = 4, enum_all = FALSE)
print(result$alpha_min_opt)
print(result$alpha_max_opt)

## End(Not run)
```

cronbach_alpha_enum *Compute Exact Bounds of Cronbach's Alpha via Enumeration*

Description

This function computes the minimum and maximum possible values of Cronbach's alpha by enumerating all possible values for unknown entries in the score matrix.

Usage

```
cronbach_alpha_enum(scores_mat, score_max)
```

Arguments

scores_mat	A matrix where rows represent individuals and columns represent test items. It contains the performance of individuals on different test items, with NA for missing values.
score_max	An integer specifying the largest possible score for any test item.

Details

This function works by enumerating all possible combinations of values for missing entries (represented by NA) in the scores_mat. It systematically explores all combinations of missing values from 0 to score_max using the expand.grid function. For each combination, it calculates Cronbach's alpha using the compute_cronbach_alpha function and keeps track of the minimum and maximum alpha values encountered.

The enumeration ensures that the function finds the exact minimum and maximum possible values of Cronbach's alpha given the possible missing score combinations. However, due to the exhaustive nature of enumeration, this function may become computationally expensive for large datasets or a high number of missing values.

Value

A numeric vector of length 2, where the first element is the minimum Cronbach's alpha and the second element is the maximum Cronbach's alpha.

See Also

cronbach_alpha_rough

cronbach_alpha_rough *Compute Rough Approximation of Cronbach's Alpha Bounds*

Description

This function computes a rough approximation of the lower and upper bounds of Cronbach's alpha by performing random sampling or integer sampling for missing values in the score matrix.

Usage

```
cronbach_alpha_rough(scores_mat, score_max, num_try = 1000, int_only = FALSE)
```

Arguments

scores_mat	A matrix where rows represent persons and columns represent tests (or items), providing the performance of a person on a test. NA should be used for missing values.
score_max	An integer indicating the largest possible score of the test.
num_try	An integer specifying the number of random samples to generate in order to estimate the lower and upper bounds. Default is 1000.
int_only	A logical value indicating whether to sample only integer values for missing scores. If FALSE, floating-point values between 0 and score_max are sampled. Default is FALSE.

Details

This function performs random sampling to estimate the bounds of Cronbach's alpha for a given test score matrix with missing values. It first calculates the alpha assuming all missing values are either 0 or score_max. Then, it iteratively samples either integer values or continuous values (depending on the value of int_only) for the missing scores and recalculates the Cronbach's alpha. The minimum and maximum alphas observed over all iterations are returned as the estimated bounds.

Value

A numeric vector of length 2, where the first value is the estimated minimum Cronbach's alpha and the second value is the estimated maximum Cronbach's alpha.

See Also

cronbachs_alpha

---

display_all *Display All Possible Parameter Combinations for Cronbach's Alpha*

Description

This function computes and displays all possible combinations of Cronbach's alpha bounds for various parameter settings, including combinations of optimization methods, rough approximation, random sampling, and enumeration.

Usage

```
display_all(
  scores_mat,
  score_max,
  tol = 0.001,
  num_random = 1000,
  num_opt = 1,
  methods = c("GA", "DEoptim", "nloptr"),
```

```
  enum_all = FALSE,
  rough = TRUE
)
```

Arguments

scores_mat	A matrix where rows represent persons and columns represent tests (or items), providing the performance of a person on a test. NA should be used for missing values.
score_max	An integer indicating the largest possible score of the test.
tol	A numeric value representing the desired accuracy in computing the lower and upper bound of Cronbach's alpha.
num_random	An integer specifying the number of random samples used in estimating the lower and upper bound. Default is 1000.
num_opt	An integer specifying the number of times to run the optimization algorithm. Default is 1.
methods	A character vector specifying the optimization methods to be used (e.g., 'GA', 'DEoptim', 'nloptr'). Default is c('GA').
enum_all	Logical, whether to include enumeration in the parameter combinations. Default is FALSE.
rough	Logical, whether to include rough approximation in the parameter combinations. Default is TRUE.

Value

A list where each element is a result of the cronbachs_alpha function for a unique parameter combination, including computation time.

---

examine_alpha_bound *Check Feasibility of Alpha Bound for Optimization Problem*

Description

This function checks whether a given value of alpha is a feasible solution to a min/max optimization problem using quadratic functions for sigma_x and sigma_y. The function supports different optimization methods and iteratively attempts to solve the problem.

Usage

```
examine_alpha_bound(
  alpha,
  n_person,
  sigma_x_info,
  sigma_y_info,
  alpha_type,
  score_max = 1,
  num_try = 1,
  method = "GA",
  ...
)
```

Arguments

alpha	A numeric value representing the alpha value to check.
n_person	An integer representing the number of individuals or rows in the data.
sigma_x_info	A list containing the quadratic function information for sigma_x, including: • A matrix representing the quadratic coefficients for sigma_x. • A vector representing the linear coefficients for sigma_x. • A scalar constant term for sigma_x.
sigma_y_info	A list containing the quadratic function information for sigma_y, including: • A matrix representing the quadratic coefficients for sigma_y. • A vector representing the linear coefficients for sigma_y. • A scalar constant term for sigma_y.
alpha_type	A character string indicating whether the problem is to minimize or maximize alpha. It must be either 'min' or 'max'.
score_max	An integer representing the largest possible score for any test item. Default is 1.
num_try	An integer specifying the number of times to run the optimization algorithm. Default is 1.
method	A character string specifying the optimization method to use. Options are 'GA' (Genetic Algorithm), 'DEoptim' (Differential Evolution), and 'nloptr' (Sequential Least Squares Programming). Default is 'GA'.
...	Additional parameters passed to the optimization algorithm.

Details

The function combines quadratic information from sigma_x_info and sigma_y_info to form a new optimization problem. The optimization checks whether the value of alpha is feasible for either a minimization or maximization problem, depending on the value of alpha_type.

The function supports multiple optimization methods, including Genetic Algorithm (GA), Differential Evolution (DEoptim), and Sequential Least Squares Programming (nloptr). Additional control parameters can be passed through the ... argument to fine-tune the optimization process.

For each iteration, the function calls qp_solver with the combined quadratic function and checks whether the objective function's value is feasible (i.e., less than or equal to 0).

Value

A list with the following elements:

result	A logical value indicating whether the alpha is feasible (TRUE) or not (FALSE).
x_value	A numeric vector representing the optimal values of the decision variables, or NULL if not feasible.

See Also

qp_solver, compute_alpha_min, compute_alpha_max

`generate_scores_mat_bernoulli` *Generate Bernoulli Distributed Scores Matrix with Missing Values*

Description

This function generates a matrix of scores for a set of people and items, where the scores are generated using a Bernoulli distribution with person-specific probabilities. It also allows for some scores to be missing (represented by `NA`).

Usage

```
generate_scores_mat_bernoulli(n_person, n_item, n_missing, score_max = 1)
```

Arguments

`n_person`	An integer representing the number of people (rows in the matrix).
`n_item`	An integer representing the number of items (columns in the matrix).
`n_missing`	An integer representing the number of missing scores (set to `NA` in the matrix).
`score_max`	An integer representing the largest possible score for any item. Default is 1.

Details

The function generates a score matrix where each person's score for each item is drawn from a Bernoulli distribution with a person-specific probability. A number of scores are set to `NA` to simulate missing values.

The probability of each person scoring on the items is determined by randomly generating a probability for each person using `runif`. The Bernoulli distribution is then used (via `rbinom`) to generate the scores, and `NA` values are assigned to randomly selected positions in the matrix based on `n_missing`.

Value

A matrix of size `n_person` by `n_item` containing generated scores (0 or `score_max`) with some values replaced by `NA` to simulate missing data.

Examples

```
# Generate a 10x5 score matrix with 10 missing values and maximum score of 1
scores_mat <- generate_scores_mat_bernoulli(10, 5, 10, score_max = 1)
print(scores_mat)
```

qp_solver *General Solver for Quadratic Programming Problems*

Description

This function provides a general interface to solve quadratic programming problems using different optimization methods. It supports Genetic Algorithm (GA), Differential Evolution (DEoptim), and Sequential Least Squares Programming (SLSQP).

Usage

```
qp_solver(n, A, b, c, x_max = 1, method = "GA", print_message = FALSE, ...)
```

Arguments

n	An integer representing the number of decision variables.
A	A matrix representing the quadratic coefficients.
b	A numeric vector representing the linear coefficients.
c	A numeric scalar representing the constant term in the objective function.
x_max	An integer representing the upper bound for the decision variables. Default is 1.
method	A character string specifying the optimization method to use. Can be 'GA', 'DEoptim', or 'nloptr'.
print_message	A logical value indicating whether to print optimization details. Default is FALSE.
...	Additional control parameters passed to the chosen optimization method.

Value

A list containing:

f_cd	The optimal objective function value.
x_value	The optimal values of the decision variables.

See Also

qp_solver_DEoptim, qp_solver_GA, qp_solver_nloptr

qp_solver_DEoptim *Solve Quadratic Programming Problem using DEoptim*

Description

This function solves a quadratic programming problem using the Differential Evolution optimization method from the DEoptim package.

Usage

```
qp_solver_DEoptim(
  n,
  A,
  b,
  c,
  x_max = 1,
  print_message = FALSE,
  NP = 100,
  itermax = 100,
  ...
)
```

Arguments

n	An integer representing the number of decision variables.
A	A matrix representing the quadratic coefficients.
b	A numeric vector representing the linear coefficients.
c	A numeric scalar representing the constant term in the objective function.
x_max	An integer representing the upper bound for the decision variables. Default is 1.
print_message	A logical value indicating whether to print optimization details. Default is FALSE.
NP	An integer specifying the population size for the DEoptim algorithm. Default is 50.
itermax	An integer specifying the maximum number of iterations. Default is 100.
...	Additional control parameters for DEoptim.

Value

A list containing:

f_cd	The optimal objective function value.
x_value	The optimal values of the decision variables.

See Also

qp_solver, qp_solver_GA, qp_solver_nloptr

qp_solver_GA *Solve Quadratic Programming Problem using GA*

Description

This function solves a quadratic programming problem using the Genetic Algorithm (GA) from the GA package.

Usage

```
qp_solver_GA(
  n,
  A,
  b,
  c,
  x_max = 1,
  print_message = FALSE,
  maxiter = 1000,
  popSize = 50,
  pmutation = 0.2,
  elitism = 5,
  monitor = FALSE,
  seed = 123,
  ...
)
```

Arguments

n	An integer representing the number of decision variables.
A	A matrix representing the quadratic coefficients.
b	A numeric vector representing the linear coefficients.
c	A numeric scalar representing the constant term in the objective function.
x_max	An integer representing the upper bound for the decision variables. Default is 1.
print_message	A logical value indicating whether to print optimization details. Default is FALSE.
maxiter	An integer specifying the maximum number of iterations. Default is 1000.
popSize	An integer specifying the population size for the GA. Default is 50.
pmutation	A numeric value for mutation probability. Default is 0.2.
elitism	An integer specifying the number of elite individuals to carry over to the next generation. Default is 5.
monitor	A logical value indicating whether to display progress. Default is FALSE.
seed	A numeric value used for the random number generator. Default is 123.
...	Additional control parameters for ga.

Value

A list containing:

f_cd	The optimal objective function value.
x_value	The optimal values of the decision variables.

See Also

qp_solver, qp_solver_DEoptim, qp_solver_nloptr

---

qp_solver_nloptr *Solve Quadratic Programming Problem using nloptr*

Description

This function solves a quadratic programming problem using the Sequential Least Squares Programming (SLSQP) algorithm from the nloptr package.

Usage

```
qp_solver_nloptr(
  n,
  A,
  b,
  c,
  x_max = 1,
  print_message = FALSE,
  xtol_rel = 1e-08,
  maxeval = 10000,
  print_level = 0,
  ...
)
```

Arguments

n	An integer representing the number of decision variables.
A	A matrix representing the quadratic coefficients.
b	A numeric vector representing the linear coefficients.
c	A numeric scalar representing the constant term in the objective function.
x_max	An integer representing the upper bound for the decision variables. Default is 1.
print_message	A logical value indicating whether to print optimization details. Default is FALSE.
xtol_rel	A numeric value specifying the relative tolerance for convergence. Default is 1e-8.
maxeval	An integer specifying the maximum number of function evaluations. Default is 10000.
print_level	An integer controlling the verbosity of output. Default is 0.
...	Additional control parameters for nloptr.

Value

A list containing:

f_cd	The optimal objective function value.
x_value	The optimal values of the decision variables.

See Also

qp_solver, qp_solver_DEoptim, qp_solver_GA

---

sample *Sample Dataset with Missing Values*

Description

This dataset contains a matrix of scores with 50 rows and 4 columns, representing 50 individuals and 4 test items. Some entries are NA, indicating missing data.

Usage

```
sample
```

Format

A 50 x 4 matrix:

Rows Each row represents an individual (total 50 individuals).

Columns Each column represents a test item or score (total 4 items).

Missing values Some entries are NA, representing missing data.

Source

Generated for demonstration purposes.

Examples

```
# Load the sample dataset
data(sample)

# Display the first few rows of the sample dataset
head(sample)
```

Index

∗ **datasets**
sample, 16

compute_alpha_max, 2, *4*, *6*, *10*
compute_alpha_min, *3*, 3, *6*, *10*
cronbach_alpha_enum, *6*, 7
cronbach_alpha_rough, *6, 7*, 7
cronbachs_alpha, 5, *8*

display_all, 8

examine_alpha_bound, *3, 4*, 9

generate_scores_mat_bernoulli, *6*, 11

qp_solver, *6*, *10*, 12, *13*, *15, 16*
qp_solver_DEoptim, *12*, 13, *15, 16*
qp_solver_GA, *12, 13*, 14, *16*
qp_solver_nloptr, *12, 13*, *15*, 15

sample, 16