Package 'minimaxALT'

September 1, 2025

Type Package

Title Generate Optimal Designs of Accelerated Life Test using PSO-Based Algorithm

Version 1.0.0

Encoding UTF-8

License GPL (>= 3)

Description A computationally efficient solution for generating optimal experimental designs in Accelerated Life Testing (ALT). Leveraging a Particle Swarm Optimization (PSO)-based hybrid algorithm, the package identifies optimal test plans that minimize estimation variance under specified failure models and stress profiles. For more detailed, see Lee et al. (2025), Optimal Robust Strategies for Accelerated Life Tests and Fatigue Testing of Polymer Composite Materials, submitted to Annals of Applied Statistics, <https://imstat.org/journals-and-publications/annals-of-applied-statistics/annals-of-applied-statistics-next-issues/>, and Hoang (2025), Model-Robust Minimax Design of Accelerated Life Tests via PSO-based Hybrid Algorithm, Master' Thesis, Unpublished.

SystemRequirements GNU Scientific Library (GSL), OpenMP

Imports Rcpp (>= 1.0.11), RcppArmadillo (>= 14.0.0.1), RcppGSL (>= 0.3.13), ggplot2 (>= 3.0.0), parallel (>= 4.0.0), stats, graphics

Depends R (>= 4.0.0)

LinkingTo Rcpp (>= 1.0.11), RcppArmadillo (>= 14.0.0.1), RcppGSL (>= 0.3.13)

RoxygenNote 7.3.2

URL https://github.com/hoanglinh171/minimaxALT

BugReports https://github.com/hoanglinh171/minimaxALT/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Hoai-Linh Hoang [aut, cre],
I-Chen Lee [aut],
Ping-Yang Chen [aut],
Ray-Bing Chen [aut],
Weng Kee Wong [aut]

Maintainer Hoai-Linh Hoang <hoailinh.hoang17@gmail.com>

Repository CRAN

Date/Publication 2025-09-01 09:30:02 UTC

Contents

`check_equivalence_theorem`

Check Equivalence Theorem for Optimal Design

Description

Evaluates whether a design satisfies the equivalence theorem.

Usage

```
check_equivalence_theorem(best_particle, model_set, design_info, seed)
```

Arguments

`best_particle`	A vector containing the best particle's position (i.e., stress levels and transformed allocated proportion).
`model_set`	A matrix of models, including parameters and distribution, that maximize the optimality criteria with the given best particle's position.
`design_info`	A list containing design parameters such as factor levels, number of units, and other settings.
`seed`	Seed for reproducibility

Value

max_directional_derivative Maximum directional derivative within design space.

model_set The model set that is input.

model_weight The weight assigned to each model in the model set.

equivalence_data Generated designs and their corresponding directional derivative given the optimal design `best_particle`. Each design is a combination of factors with value in [0, 1]. These designs are data for plotting equivalence theorem plot.

References

1. Müller, C. H., & Pázman, A. (1998). Applications of necessary and sufficient conditions for maximin efficient designs. Metrika, 48, 1–19.

2. Huang, M.-N. L., & Lin, C.-S. (2006). Minimax and maximin efficient designs for estimating the location-shift parameter of parallel models with dual responses. Journal of Multivariate Analysis, 97(1), 198–210.

Examples

```
design_info <- set_design_info(k_levels=2, j_factor=1, n_unit=300,
                               censor_time=183, p=0.1, use_cond=0, sigma=0.6)

best_particle <- c(0.682, 1, 0.706)

model_set <- rbind(
  c(0.01, 0.9, 1),
  c(0.01, 0.99, 2))

equi <- check_equivalence_theorem (best_particle=best_particle,
                                    model_set=model_set,
                                    design_info=design_info,
                                    seed = 42)

equi$max_directional_derivative
```

`find_optimal_alt` *Find Optimal ALT Design Using Hybrid Algorithm*

Description

Runs hybrid algorithm combining PSO and Nelder-Mead to find the optimal design of accelerated life test (ALT).

Usage

```
find_optimal_alt(
  design_type,
  distribution,
  design_info,
  pso_info,
  coef = NULL,
  coef_lower = NULL,
  coef_upper = NULL,
  init_values = NULL,
  highest_level = TRUE,
  n_threads = 1,
  verbose = TRUE,
  seed
)
```

Arguments

`design_type`	Integer. 1: Locally optimal design, 2: Minimax design.
`distribution`	Integer. The assumed failure time distribution, 1: Weibull, 2: Log-normal, 3: Model robust (both distribution Weibull and Log-normal).
`design_info`	A list from 'set_design_info()' containing design specifications.
`pso_info`	A list from 'pso_setting()' defining PSO hyperparameters.
`coef`	Optional. Fixed model coefficients. Required if `design_type` = `1`.
`coef_lower`	Optional. Lower bounds for model parameters. Required if `design_type` = `2`.
`coef_upper`	Optional. Upper bounds for model parameters. Required if `design_type` = `2`.
`init_values`	Optional. A list of initial values from 'initialize_values()'.
`highest_level`	Logical. Whether the highest stress level of the generated design is the upper bound of stress range `x_h`. Default value is `TRUE`.
`n_threads`	Integer. Number of threads for parallel processing.
`verbose`	Logical. If `TRUE`, print optimization progress.
`seed`	Integer. Seed for reproducibility

Value

g_best The global best design found by the hybrid algorithm.

coef_best The parameters corresponding to the global best design.

distribution_best The distribution corresponding to the global best design.

max_directional_derivative Maximum directional derivative within design space, evaluated using equivalence theorem.

fg_best The objective function value corresponding to the global best design.

fg_best_hist A vector tracking the best objective function value of each iteration.

p_best A matrix containing each particle's personal best design found during the optimization.

fp_best A vector containing the objective function values corresponding to each particle's personal best.

g_hist All particle positions of each iteration.

coef_best_hist The parameters corresponding to the global best designs of each iteration.

distribution_best_hist The distribution corresponding to the global best designs of each iteration.

model_set A matrix containing distribution and model parameters of global best particles of each iteration, duplicated models are removed.

model_weight The weight assigned to each model in the model set.

equivalence_data Generated designs and their corresponding directional derivative given the optimal design `g_best`. Each design is a combination of factors with value in [0, 1]. These designs are data for plotting equivalence theorem plot.

References

1. Chen P (2024). _globpso: Particle Swarm Optimization Algorithms and Differential Evolution for Minimization Problems_. R package version 1.2.1, <https://github.com/PingYangChen/globpso>.

2. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks (ICNN) (Vol. 4, pp. 1942–1948).

3. Lee, I. C., Chen, R. B., Wong, W. K., (in press). Optimal Robust Strategies for Accelerated Life Tests and Fatigue Testing of Polymer Composite Materials. Annals of Applied Statistics. <https://imstat.org/journals-and-publications/annals-of-applied-statistics/annals-of-applied-statistics-next-issues/>

4. Meeker, W. Q., & Escobar, L. A. (1998). Statistical methods for reliability data. New York: Wiley-Interscience.

5. Nelder, J. A. and Mead, R. (1965). A simplex algorithm for function minimization. Computer Journal, 7, 308–313. 10.1093/comjnl/7.4.308.

Examples

```
design_info <- set_design_info(k_levels=2, j_factor=1, n_unit=300,
                               censor_time=183, p=0.1, use_cond=0, sigma=0.6)

pso_info <- pso_setting(n_swarm=32, max_iter=128, early_stopping=10, tol=0.01)

res <- find_optimal_alt(design_type=1, distribution=1, design_info=design_info,
                        pso_info=pso_info, coef=c(0.001, 0.9), verbose = FALSE,
                        seed = 42)

summary(res)
plot(res, x_l=0, x_h=1)
```

initialize_values *Initialize Particle Swarm Optimization and Nelder-Mead Algorithm Values*

Description

Sets initial particles for PSO, initial locally optimal design, and initial parameters for Nelder-Mead algorithm.

Usage

```
initialize_values(init_swarm = NULL, init_local = NULL, init_coef_mat = NULL)
```

Arguments

`init_swarm`	Optional matrix of initial particle positions. If not defined, particle positions are randomly generated using `runif` with pre-determined number of particles `n_swarm` and particle size.
`init_local`	Optional vector of initial locally optimal design. If not defined, the initial vector representing locally optimal design is `c(1, 0.6, 0.3)`.
`init_coef_mat`	Optional matrix of initial parameters to implement multi-start Nelder-Mead algorithm. The number of rows is the number of starts, and each row is the corresponding initial parameters. If not defined, the initial matrix of parameters is generated by sigmoid transformation of `10 * as.matrix(expand.grid(rep(list(c(1, -1)), j_factor + 1)))`.

Value

A list of initialized values.

Examples

```
init_local <- c(1, 0.6, 0.3)

init_coef_mat <- rbind(
  c(1e-6, 0.99),
  c(1e-2, 1),
  c(1.01e-6, 0.9999))

j_factor <- 1
k_levels <- 3
n_swarm <- 32
d_swarm <- (j_factor + 1) * k_levels - 1
init_swarm <- matrix(runif(n_swarm*d_swarm), nrow=n_swarm, byrow=TRUE)

init_values <- initialize_values(init_swarm=init_swarm,
                                  init_local=init_local,
                                  init_coef_mat=init_coef_mat)
```

pso_setting *Set PSO Optimization Settings*

Description

Define hyperparameters for particle swarm optimization (PSO).

Usage

```
pso_setting(
  n_swarm = 32,
  max_iter = 128,
  early_stopping = 10,
  tol = 0.01,
  c1 = 2.05,
  c2 = 2.05,
  w0 = 1.2,
  w1 = 0.2,
  w_var = 0.8,
  vk = 4
)
```

Arguments

`n_swarm`	Integer. Number of particles in the swarm.
`max_iter`	Integer. Maximum number of iterations.
`early_stopping`	Integer. The frequency, i.e. number of iterations, of validating the design optimality using equivalence theorem. The optimization process stops once maximum directional derivative is approximately 1.
`tol`	Numeric. Convergence tolerance. The algorithm stops if `abs(max_directional_derivative - 1) < tol`.
`c1`	Numeric. Cognitive acceleration coefficient. Default value is 2.05.
`c2`	Numeric. Social acceleration coefficient. Default value is 2.05.
`w0`	Numeric. Starting inertia weight. Default value is 1.2.
`w1`	Numeric. Ending inertia weight. Default value is 0.2.
`w_var`	Numeric. A number between $[0, 1]$ that controls the percentage of iterations during which PSO linearly decrease inertia weight from `w0` to `w1`. Default value is 0.8.
`vk`	Numeric. Velocity clamping factor. Default value is 4.

Value

A list of PSO hyperparameters.

Examples

```
pso_info <- pso_setting(n_swarm=32, max_iter=128, early_stopping=10, tol=0.01)
```

---

set_design_info *Set ALT Design Information*

Description

Configures the settings for an accelerated life test.

Usage

```
set_design_info(
  k_levels,
  j_factor,
  n_unit,
  censor_time,
  p,
  use_cond,
  sigma,
  x_l = 0,
  x_h = 1,
  opt_type = "C",
  reparam = TRUE
)
```

Arguments

`k_levels`	Integer. Number of stress levels.
`j_factor`	Integer. Number of stress factors.
`n_unit`	Integer. Total number of test units.
`censor_time`	Numeric. Test duration or censoring time.
`p`	Numeric. $0 < p < 1$. Lifetime percentile to be estimated at the use condition, i.e. stress levels are 0.
`use_cond`	Vector. Stress levels at the use condition.
`sigma`	Numeric. Scale parameter of the lifetime distribution.
`x_l`	Numeric. Lower bound of stress range. Default is 0.
`x_h`	Numeric. Upper bound of stress range. Default is 1.
`opt_type`	Character. Optimality criterion, currently only C-optimality is supported. Default is "C".
`reparam`	Logical. Whether reparameterization is applied to model parameters. Reparameterization is supported for all design types, while non-reparameterization is only available for locally optimal design `design_type = 1`. Default is TRUE.

Value

A list of design specifications

Examples

```
design_info <- set_design_info(k_levels=3, j_factor=1, n_unit=300,
                           censor_time=183, p=0.1, use_cond=c(0), sigma=0.6)
```

Index