Package 'MTLRF'

August 21, 2026

Type Package

Title Random Forest Regression with Modified Topp-Leone Error Model

Version 1.0.0

Date 2026-08-18

Description Implements Random Forest regression under the Modified Topp-Leone (MTL) distribution error model. Provides core distribution functions (density, cumulative distribution, exact closed-form quantile, random generation, hazard, and survival), parameter estimation via closed-form Expectation-Maximization/Maximum Likelihood (EM/MLE) and Bayesian Markov Chain Monte Carlo (MCMC), non-parametric bootstrap confidence intervals (at 90%, 95%, and 99% levels), Highest Posterior Density (HPD) intervals, Heidelberger and Welch MCMC convergence diagnostics, model evaluation metrics (estimated values, bias, mean squared error, risk value), homoscedastic prediction intervals, and goodness-of-fit diagnostic tests (Kolmogorov-Smirnov and Anderson-Darling tests, Akaike Information Criterion, and Bayesian Information Criterion). References: Breiman (2001) <doi:10.1023/A:1010933404324>; Singh, Tyagi, Singh, and Tyagi (2025) <https://statassoc.or.th>; Topp and Leone (1955) <doi:10.1080/01621459.1955.10501259>; Wright and Ziegler (2017) <doi:10.18637/jss.v077.
mer, Best, Cowles, and Vines (2006) <https://CRAN.R-project.org/package=coda>; Heidelberger and Welch (1983) <doi:10.1287/opre.31.6.1109>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports ranger, coda, goftest, stats, graphics

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Shikhar Tyagi [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1606-0844>),
Aruna Rajballie [aut],
Vrijesh Tripathi [aut]

Maintainer Shikhar Tyagi <shikhar1093tyagi@gmail.com>

Repository CRAN

Date/Publication 2026-08-21 13:10:42 UTC

Contents

`mcmc_bayesian` *Bayesian MCMC Estimation, HPD Intervals, and Heidelberger-Welch Convergence Diagnostics*

Description

Estimates the Modified Topp-Leone (MTL) parameter `alpha` using Markov Chain Monte Carlo (MCMC) sampling, evaluates Highest Posterior Density (HPD) intervals at 90

Usage

```
mcmc_mtl(
  r,
  n_iter = 5000,
  burn_in = 1000,
  thin = 2,
  init_alpha = NULL,
  seed = NULL
)
```

Arguments

`r`	Vector of absolute residuals.
`n_iter`	Total number of MCMC iterations (default `5000`).
`burn_in`	Number of initial burn-in iterations to discard (default `1000`).
`thin`	Thinning interval for posterior chain (default `2`).
`init_alpha`	Initial value for α. If `NULL`, calculated via `mtl_mle`.
`seed`	Optional random seed.

Details

MCMC sampling is performed using an adaptive Metropolis-Hastings algorithm on the log-scale of α ($\phi = \log \alpha$) to ensure positivity and numerical stability. A weakly informative Gamma(0.001, 0.001) prior is placed on α. The posterior chain is analyzed using `HPDinterval` to extract HPD intervals at 90 Convergence is assessed via the Heidelberger and Welch diagnostic (`heidel.diag`), evaluating stationarity and halfwidth adequacy.

Value

A list containing:

posterior_mean Posterior mean estimate of α.

posterior_median Posterior median estimate of α.

posterior_sd Posterior standard deviation of α.

hpd_90 Highest Posterior Density interval at 90% level.

hpd_95 Highest Posterior Density interval at 95% level.

hpd_99 Highest Posterior Density interval at 99% level.

acceptance_rate MCMC acceptance rate (convergence probability).

heidel_diag Heidelberger and Welch's convergence diagnostic summary.

convergence_prob Estimated convergence probability.

mcmc_chain `mcmc` object of posterior samples.

Examples

```
set.seed(123)
r_sample <- rmtl(50, alpha = 2.0)
mcmc_fit <- mcmc_mtl(r_sample, n_iter = 2000, burn_in = 500)
print(mcmc_fit$posterior_mean)
print(mcmc_fit$hpd_95)
```

`mle_em` *Maximum Likelihood Estimation and Bootstrap Confidence Intervals for MTL Model*

Description

Computes the exact closed-form Maximum Likelihood Estimator (MLE) for the Modified Topp-Leone (MTL) parameter `alpha`, and evaluates non-parametric bootstrap confidence intervals at 90

Usage

```
mtl_mle(r)

mtl_loglik(alpha, r)

mtl_fisher(alpha, n)

bootstrap_ci(
  y,
  X,
  n_boot = 100,
  conf_levels = c(0.9, 0.95, 0.99),
  num.trees = 100,
  seed = NULL
)
```

Arguments

r	Vector of absolute residuals.
alpha	Shape parameter α.
n	Number of observations.
y	Numeric vector of response values.
X	Data frame of predictor variables.
n_boot	Number of bootstrap iterations (default 100).
conf_levels	Confidence levels for intervals (default c(0.90, 0.95, 0.99)).
num.trees	Number of trees for bootstrap forest fits (default 100).
seed	Optional integer seed for reproducibility.

Details

For independent absolute residuals $r_1, \ldots, r_n > 0$, the log-likelihood function of the MTL distribution is:

$$\ell(\alpha) = n \log 2 + n \log \alpha + (\alpha - 1) \sum_{i=1}^{n} \log(2r_i + r_i^2) - (2\alpha + 1) \sum_{i=1}^{n} \log(1 + r_i)$$

Setting the score function $\partial \ell / \partial \alpha = 0$ yields the exact closed-form MLE:

$$\hat{\alpha} = \frac{n}{-\sum_{i=1}^{n} \log\left(1 - \frac{1}{(1+r_i)^2}\right)}$$

Under the EM/MLE estimation method, because asymptotic distribution may be impacted by Stage 1 non-parametric Random Forest estimation, non-parametric bootstrap resampling is utilized to compute empirical confidence intervals at 90

Value

`mtl_mle` returns the scalar MLE $\hat{\alpha}$. `mtl_loglik` returns the log-likelihood value. `mtl_fisher` returns the expected Fisher information. `bootstrap_ci` returns a list containing bootstrap distributions and confidence intervals at 90%, 95%, and 99% levels for `alpha`, `bias`, `mse`, and `risk`.

Examples

```
set.seed(123)
r <- rmtl(50, alpha = 2.5)
hat_alpha <- mtl_mle(r)
print(hat_alpha)
```

`mtl` *The Modified Topp-Leone (MTL) Distribution*

Description

Density, distribution function, quantile function, random generation, hazard function, and survival function for the Modified Topp-Leone (MTL) distribution with shape parameter `alpha`.

Usage

```
dmtl(x, alpha, log = FALSE)

pmtl(q, alpha, lower.tail = TRUE, log.p = FALSE)

qmtl(p, alpha, lower.tail = TRUE, log.p = FALSE)

rmtl(n, alpha)

smtl(x, alpha, log = FALSE)

hmtl(x, alpha, log = FALSE)
```

Arguments

`x`, `q`	Vector of quantiles.
`alpha`	Shape parameter ($\alpha > 0$).
`log`, `log.p`	Logical or character string indicating boolean evaluation (`TRUE` or `"TRUE"` / `FALSE` or `"FALSE"`). If `TRUE`, probabilities/densities are given as `log(p)`.
`lower.tail`	Logical or character string indicating boolean evaluation. If `TRUE` (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`p`	Vector of probabilities.
`n`	Number of observations. If `length(n) > 1`, the length is taken to be the number required.

Details

The Modified Topp-Leone (MTL) distribution is a continuous probability distribution defined on $(0, \infty)$ with shape parameter $\alpha > 0$. Its probability density function (PDF) is given by:

$$f(x;\alpha) = \frac{2\alpha(2x+x^2)^{\alpha-1}}{(1+x)^{2\alpha+1}} = 2\alpha(1+x)^{-2\alpha-1}(2x+x^2)^{\alpha-1}, \quad x > 0, \alpha > 0$$

Its cumulative distribution function (CDF) is given by:

$$F(x;\alpha) = \left(1 - \frac{1}{(1+x)^2}\right)^{\alpha} = \left(\frac{2x+x^2}{(1+x)^2}\right)^{\alpha}, \quad x > 0$$

Its survival function (SF) is:

$$S(x;\alpha) = 1 - F(x;\alpha) = 1 - \left(1 - \frac{1}{(1+x)^2}\right)^{\alpha}, \quad x > 0$$

Its hazard rate function (HRF) is:

$$h(x;\alpha) = \frac{f(x;\alpha)}{S(x;\alpha)} = \frac{2\alpha(2x+x^2)^{\alpha-1}}{(1+x)^{2\alpha+1}\left[1 - \left(1 - \frac{1}{(1+x)^2}\right)^{\alpha}\right]}, \quad x > 0$$

Its exact closed-form quantile function (QF) for probability $p \in (0,1)$ is:

$$Q(p;\alpha) = \left(1 - p^{1/\alpha}\right)^{-1/2} - 1$$

In log-stable computation, $Q(p;\alpha) = \left(-\text{expm1}\left(\frac{\log p}{\alpha}\right)\right)^{-1/2} - 1$. Random generation is performed by direct inversion sampling using `runif`.

Value

`dmtl` gives the density, `pmtl` gives the distribution function, `qmtl` gives the exact quantile function, `rmtl` generates random deviates, `hmtl` gives the hazard function, and `smtl` gives the survival function.

Examples

```
# Evaluate PDF and CDF at x = 1, alpha = 2
dmtl(1, alpha = 2)
pmtl(1, alpha = 2)

# Quantile function
qmtl(0.95, alpha = 2)

# Survival and hazard functions
smtl(1, alpha = 2)
hmtl(1, alpha = 2)

# Generate random samples
set.seed(123)
r_samples <- rmtl(10, alpha = 2)
print(r_samples)
```

`mtl_diagnostics` *Goodness-of-Fit and Model Selection Diagnostics for MTL Error Model*

Description

Performs goodness-of-fit hypothesis testing (Kolmogorov-Smirnov and Anderson-Darling tests) and evaluates information-theoretic criteria (AIC and BIC) for a fitted RF-MTL model.

Usage

```
mtl_diagnostics(object, ...)
```

Arguments

`object`	A fitted object of class `"rf_mtl"`.
`...`	Additional arguments (currently unused).

Details

Goodness-of-fit evaluates whether absolute residuals $r_i = |y_i - \hat{y}_i|$ follow the Modified Topp-Leone distribution with estimated parameter $\hat{\alpha}$.

Information criteria are defined as:

$$\text{AIC} = 2k - 2\ell(\hat{\alpha}), \quad \text{BIC} = k \log n - 2\ell(\hat{\alpha})$$

where $k = 1$ (single shape parameter) and $\ell(\hat{\alpha})$ is the maximized MTL log-likelihood.

Value

An S3 object of class `"mtl_diagnostics"`, containing:

ks_stat Kolmogorov-Smirnov test statistic D_n.

ks_pvalue Asymptotic p-value for the KS test.

ad_stat Anderson-Darling test statistic A_n^2.

ad_pvalue p-value for the AD test via `ad.test`.

loglik Maximized log-likelihood value.

aic Akaike Information Criterion.

bic Bayesian Information Criterion.

alpha Estimated shape parameter $\hat{\alpha}$.

n Number of training observations.

Examples

```
set.seed(123)
df <- data.frame(y = 5 + 2 * rnorm(50), x1 = rnorm(50), x2 = runif(50))
fit <- rf_mtl(y ~ x1 + x2, data = df, method = "em", num.trees = 50, n_boot = 20)
diag_res <- mtl_diagnostics(fit)
print(diag_res)
```

`predict.rf_mtl` *Predict Method for RF-MTL Model Fits*

Description

Generates point predictions and exact closed-form prediction intervals for new observations using a fitted `"rf_mtl"` object.

Usage

```
## S3 method for class 'rf_mtl'
predict(
  object,
  newdata = NULL,
  level = 0.95,
  interval = c("none", "prediction"),
  ...
)
```

Arguments

`object`	A fitted object of class `"rf_mtl"`.
`newdata`	Data frame of new predictor observations. If omitted, fitted training values are returned.
`level`	Confidence level for prediction intervals (default `0.95`).
`interval`	Type of interval calculation: `"none"` (default) or `"prediction"`.
`...`	Additional arguments passed to `predict.ranger`.

Details

Point predictions are generated by ensembled Random Forest regression trees. Prediction intervals at confidence level `level` (e.g. 0.95) are calculated via the exact closed-form Modified Topp-Leone quantile function:

$$\mathrm{PI}_{1-\gamma}(\mathbf{x}^*) = \left[\hat{f}(\mathbf{x}^*) - Q_{1-\gamma}(\hat{\alpha}), \quad \hat{f}(\mathbf{x}^*) + Q_{1-\gamma}(\hat{\alpha})\right]$$

where $\gamma = 1 - \text{level}$ and $Q_{1-\gamma}(\hat{\alpha}) = \left(1 - (1-\gamma)^{1/\hat{\alpha}}\right)^{-1/2} - 1$.

Value

If `interval = "none"`, a numeric vector of point predictions is returned. If `interval = "prediction"`, a data frame containing columns `fit` (point prediction), `lwr` (lower prediction bound), and `upr` (upper prediction bound) is returned.

Examples

```
set.seed(123)
df <- data.frame(y = 5 + 2 * rnorm(50), x1 = rnorm(50), x2 = runif(50))
fit <- rf_mtl(y ~ x1 + x2, data = df, method = "em", num.trees = 50, n_boot = 20)
preds <- predict(fit, newdata = df[1:5, ], interval = "prediction", level = 0.95)
print(preds)
```

`rf_mtl` *Random Forest Regression with Modified Topp-Leone (MTL) Error Model*

Description

Fits a Random Forest regression model combined with a Modified Topp-Leone (MTL) distribution error model. Supports parameter estimation via closed-form EM/MLE algorithm (with nonparametric bootstrap confidence intervals at 90

Usage

```
rf_mtl(
  formula,
  data,
  method = c("em", "mcmc"),
  num.trees = 500,
  n_boot = 100,
  n_iter = 5000,
  burn_in = 1000,
  conf_levels = c(0.9, 0.95, 0.99),
  na.action = stats::na.omit,
  seed = NULL,
  ...
)
```

Arguments

`formula`	Object of class `formula` specifying model formula.
`data`	Data frame containing the variables in the model.
`method`	Estimation method: `"em"` (default) or `"mcmc"`.
`num.trees`	Number of trees in the Random Forest ensemble (default `500`).

n_boot	Number of bootstrap iterations for EM confidence intervals (default 100).
n_iter	Number of MCMC iterations for Bayesian estimation (default 5000).
burn_in	Number of burn-in MCMC iterations (default 1000).
conf_levels	Confidence levels for intervals (default c(0.90, 0.95, 0.99)).
na.action	Function specifying action for missing values (default na.omit).
seed	Optional integer seed for reproducibility.
...	Additional arguments passed to ranger.

Details

The RF-MTL regression model assumes:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \quad i = 1, \ldots, n$$

where $f(\mathbf{x})$ is non-parametrically estimated by a Random Forest predictor, and absolute errors $|\varepsilon_i|$ follow the Modified Topp-Leone distribution with shape parameter $\alpha > 0$.

In Stage 1, Random Forest regression is fitted using ranger under the mean squared error criterion to predict point estimates $\hat{y}_i = \hat{f}(\mathbf{x}_i)$. In Stage 2, absolute residuals $r_i = |y_i - \hat{y}_i|$ are modeled using the MTL error distribution.

Parameter estimation can be performed via:

- **EM / MLE Algorithm** (method = "em")**:** Computes exact analytical closed-form MLE $\hat{\alpha}$ and uses non-parametric bootstrap resampling to evaluate confidence intervals at 90
- **Bayesian MCMC** (method = "mcmc")**:** Draws MCMC posterior samples for α via adaptive Metropolis-Hastings, computes Highest Posterior Density (HPD) intervals at 90

Missing values are processed according to the specified na.action. Logical parameters accept both standard R boolean flags (TRUE/FALSE) and character representations ("TRUE"/"FALSE").

Value

An S3 object of class "rf_mtl", containing:

rf_model The fitted ranger Random Forest object.

alpha Estimated Modified Topp-Leone shape parameter α.

method Estimation method used ("em" or "mcmc").

fitted_values Fitted point predictions $\hat{y}$.

residuals Raw residuals $y - \hat{y}$.

abs_residuals Absolute residuals $|y - \hat{y}|$.

bias Estimated model bias.

mse Mean squared error.

rmse Root mean squared error.

mae Mean absolute error.

risk Model risk value (empirical squared loss).

intervals Bootstrap confidence intervals (for EM) or HPD intervals (for MCMC) at 90%, 95%, and 99% levels.

heidel_diag Heidelberger and Welch convergence diagnostic summary (if `method = "mcmc"`).

convergence_prob MCMC convergence probability / acceptance rate (if `method = "mcmc"`).

bootstrap_results Detailed bootstrap outputs if `method = "em"`.

mcmc_results Detailed MCMC sampler outputs if `method = "mcmc"`.

formula Model formula.

data Processed training data frame.

response_name Name of response variable.

predictor_names Vector of predictor variable names.

Examples

```
# Fit RF-MTL model on synthetic regression data
set.seed(123)
df <- data.frame(
  y = 5 + 2 * rnorm(50),
  x1 = rnorm(50),
  x2 = runif(50)
)
fit_em <- rf_mtl(y ~ x1 + x2, data = df, method = "em", num.trees = 50, n_boot = 30, seed = 123)
print(fit_em)
```

---

utils_s3 *S3 Methods for RF-MTL Model Objects*

Description

Print, summary, and plot methods for fitted `"rf_mtl"` objects.

Usage

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

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

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

## S3 method for class 'rf_mtl'
plot(x, which = 1:3, ...)
```

Arguments

x, object	A fitted object of class "rf_mtl".
which	Vector of plot types to produce: 1 (Residual density fit), 2 (Q-Q plot), 3 (Fitted vs Observed). Default is 1:3.
...	Additional arguments passed to generic print or plot functions.

Value

print.rf_mtl and print.summary_rf_mtl return the input object invisibly. summary.rf_mtl returns a list of summary statistics. plot.rf_mtl returns NULL invisibly and displays diagnostic graphics.

Examples

```
set.seed(123)
df <- data.frame(y = 5 + 2 * rnorm(50), x1 = rnorm(50), x2 = runif(50))
fit <- rf_mtl(y ~ x1 + x2, data = df, method = "em", num.trees = 50, n_boot = 20)
print(fit)
summary(fit)
```

Index