

# Package: MixStable (via r-universe)

June 1, 2026

**Type** Package

**Title** Parameter Estimation for Stable Distributions and Their Mixtures

**Version** 0.1.0

**Description** Provides various functions for parameter estimation of one-dimensional stable distributions and their mixtures. It implements a diverse set of estimation methods, including quantile-based approaches, regression methods based on the empirical characteristic function (empirical, kernel, and recursive), and maximum likelihood estimation. For mixture models, it provides stochastic expectation-maximization (SEM) algorithms and Bayesian estimation methods using sampling and importance sampling to overcome the long burn-in period of Markov Chain Monte Carlo (MCMC) strategies. The package also includes tools and statistical tests for analyzing whether a dataset follows a stable distribution. Some of the implemented methods are described in Hajjaji, O., Manou-Abi, S. M., and Slaoui, Y. (2024) <[doi:10.1080/02664763.2024.2434627](https://doi.org/10.1080/02664763.2024.2434627)>.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** stabledist, mixtools, nortest, openxlsx, e1071, jsonlite, libstable4u, stats, graphics, MASS, utils

**Suggests** ggplot2, grDevices, moments, readxl, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Solym Manou-Abi [aut, cre], Adam Najib [aut], Yousri Slaoui [aut]

**Maintainer** Solym Manou-Abi <[solym.manou.abi@univ-poitiers.fr](mailto:solym.manou.abi@univ-poitiers.fr)>

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2025-11-03 19:20:18 UTC

**RemoteUrl** <https://github.com/cran/MixStable>

**RemoteRef** HEAD

**RemoteSha** d869cc7ba6913ffb48f211b359d2593b30b85d26

## Contents

aic . . . . .	5
analyse_stable_distribution . . . . .	5
bayesian_mixture_model . . . . .	6
bic . . . . .	6
build_mcculloch_interpolators . . . . .	7
calculate_log_likelihood . . . . .	7
CDF . . . . .	8
clip . . . . .	8
compare_em_vs_em_gibbs . . . . .	9
compare_estimators_on_simulations . . . . .	9
compare_methods_across_configs . . . . .	10
compare_methods_with_gibbs . . . . .	10
compute_model_metrics . . . . .	11
compute_quantile_ratios . . . . .	12
compute_serial_interval . . . . .	12
cosine_exp_alpha . . . . .	13
cosine_log_weighted_exp_alpha . . . . .	13
DONNEE_with_serial_interval . . . . .	14
ecf_components . . . . .	14
ecf_empirical . . . . .	15
ecf_estimate_all . . . . .	15
ecf_fn . . . . .	16
ecf_regression . . . . .	16
em_alpha_stable . . . . .	17
em_estimate_stable_from_cdf . . . . .	17
em_estimate_stable_from_cdf_with_gibbs . . . . .	18
em_estimate_stable_kernel_ecf . . . . .	19
em_estimate_stable_kernel_ecf_with_gibbs . . . . .	19
em_estimate_stable_recursive_ecf . . . . .	20
em_estimate_stable_recursive_ecf_with_gibbs . . . . .	20
em_estimate_stable_weighted_ols . . . . .	21
em_estimate_stable_weighted_ols_with_gibbs . . . . .	21
em_estimation_mixture . . . . .	22
em_fit_alpha_stable_mixture . . . . .	22
em_stable_mixture . . . . .	23
empirical_r0 . . . . .	23
ensure_positive_scale . . . . .	24
est_r0_ml . . . . .	24
est_r0_mle . . . . .	25
estimate_alpha_gamma . . . . .	25
estimate_beta_delta . . . . .	26
estimate_mixture_params . . . . .	26

estimate_stable_from_cdf . . . . .	27
estimate_stable_kernel_ecf . . . . .	27
estimate_stable_params . . . . .	28
estimate_stable_r . . . . .	28
estimate_stable_recursive_ecf . . . . .	29
estimate_stable_weighted_ols . . . . .	29
eta_func . . . . .	30
eta0 . . . . .	30
evaluate_estimation_method . . . . .	31
evaluate_fit . . . . .	31
export_analysis_report . . . . .	32
false_position_update . . . . .	33
fast_integrate . . . . .	33
fit_alpha_stable_mle . . . . .	34
fit_mle_mixture . . . . .	34
fit_stable_ecf . . . . .	35
generate_alpha_stable_mixture . . . . .	35
generate_mcculloch_table . . . . .	36
generate_mixture_data . . . . .	37
generate_synthetic_data . . . . .	37
gibbs_sampler . . . . .	38
grad_loglik_alpha . . . . .	38
grad_loglik_beta . . . . .	39
grad_loglik_delta . . . . .	39
grad_loglik_omega . . . . .	40
Im . . . . .	40
Int_Im . . . . .	41
Int_Re . . . . .	41
integrate_cosine . . . . .	42
integrate_cosine_log_weighted . . . . .	42
integrate_function . . . . .	43
integrate_sine . . . . .	43
integrate_sine_log_weighted . . . . .	44
integrate_sine_r_weighted . . . . .	45
integrate_sine_weighted . . . . .	45
kde_bandwidth_plugin . . . . .	46
L_stable . . . . .	46
log_likelihood_mixture . . . . .	47
Max_vrai . . . . .	47
mcculloch_lookup_estimate . . . . .	48
mcculloch_quantile_init . . . . .	48
metropolis_hastings . . . . .	49
mixture_stable_pdf . . . . .	49
mle_estimate . . . . .	50
mock_gibbs_sampling . . . . .	50
mock_lookup_alpha_beta . . . . .	51
N_epanechnikov . . . . .	51
N_gaussian . . . . .	52

N_uniform . . . . .	52
negative_log_likelihood . . . . .	53
normalized_grad_alpha . . . . .	53
normalized_objective_beta . . . . .	54
normalized_objective_delta . . . . .	54
normalized_objective_omega . . . . .	55
plot_comparison . . . . .	56
plot_distributions . . . . .	56
plot_effective_reproduction_number . . . . .	57
plot_final_mixture_fit . . . . .	57
plot_fit_vs_true . . . . .	58
plot_fit_vs_true_methods . . . . .	59
plot_method_comparison . . . . .	59
plot_mixture . . . . .	60
plot_mixture_fit . . . . .	60
plot_real_mixture_fit . . . . .	61
plot_results . . . . .	62
plot_trace . . . . .	63
plot_vs_normal_stable . . . . .	63
qcv_stat . . . . .	64
r_stable_pdf . . . . .	64
Re . . . . .	65
recursive_weight . . . . .	65
robust_ecf_regression . . . . .	66
robust_mle_estimate . . . . .	66
rstable . . . . .	67
RT . . . . .	67
run_all_estimations . . . . .	68
run_estimations_with_gibbs . . . . .	68
safe_integrate . . . . .	69
simple_em_real . . . . .	70
simulate_mixture . . . . .	70
sine_exp_ralpha . . . . .	71
sine_log_weighted_exp_ralpha . . . . .	71
sine_r_weighted_exp_ralpha . . . . .	72
sine_weighted_exp_ralpha . . . . .	72
skew_kurtosis . . . . .	73
stable_fit_init . . . . .	73
TableS2_serial_interval_mean_ . . . . .	74
test_normality . . . . .	75
unpack_params . . . . .	75
validate_params . . . . .	76
wasserstein_distance_mixture . . . . .	76

---

aic	<i>Akaike Information Criterion (AIC)</i>
-----	---

---

**Description**

Computes the AIC value given a log-likelihood and number of parameters.

**Usage**

```
aic(log_likelihood, num_params)
```

**Arguments**

log_likelihood	Log-likelihood value.
num_params	Number of estimated parameters.

**Value**

AIC value.

---

analyse_stable_distribution	<i>Perform full stability analysis and export results</i>
-----------------------------	---

---

**Description**

Executes the full pipeline for stability analysis: normality tests, QCV computation, parameter estimation, plotting, and report export.

**Usage**

```
analyse_stable_distribution(  
  x,  
  filename = "interval_analysis",  
  qcv_threshold = 1.8,  
  fig_path = NULL,  
  verbose = FALSE  
)
```

**Arguments**

x	Numeric vector of data.
filename	Base name for output files (JSON and Excel). Written to tempdir().
qcv_threshold	Threshold for QCV to consider distribution heavy-tailed.
fig_path	Optional path to save the plot (PNG). If NULL, uses tempdir().
verbose	Logical; if TRUE, prints progress messages. Default is FALSE.

**Value**

Invisibly returns a character string containing a Markdown-formatted summary report.

---

`bayesian_mixture_model`

*Bayesian mixture model using normal components (simplified)*

---

**Description**

Bayesian mixture model using normal components (simplified)

**Usage**

```
bayesian_mixture_model(data, draws = 1000, chains = 2)
```

**Arguments**

<code>data</code>	Numeric vector of observations.
<code>draws</code>	Number of iterations per chain.
<code>chains</code>	Number of chains to simulate.

**Value**

List containing model fit, chains, weights, means, and standard deviations.

---

`bic`

*Bayesian Information Criterion (BIC)*

---

**Description**

Computes the BIC value given a log-likelihood, number of parameters, and sample size.

**Usage**

```
bic(log_likelihood, num_params, n)
```

**Arguments**

<code>log_likelihood</code>	Log-likelihood value.
<code>num_params</code>	Number of estimated parameters.
<code>n</code>	Sample size.

**Value**

BIC value.

---

`build_mcculloch_interpolators`*Build interpolation functions from McCulloch table*

---

**Description**

Constructs nearest-neighbor interpolation functions for alpha and beta based on quantile ratios (v\_alpha, v\_beta) from the lookup table.

**Usage**

```
build_mcculloch_interpolators(table)
```

**Arguments**

table           Lookup table generated by generate\_mcculloch\_table.

**Value**

List with functions interp\_alpha and interp\_beta.

---

`calculate_log_likelihood`*Calculate simplified log-likelihood*

---

**Description**

Computes a placeholder log-likelihood value based on squared deviations from the mean. This is not a true likelihood function and is used for testing or comparison purposes.

**Usage**

```
calculate_log_likelihood(data, params)
```

**Arguments**

data            Numeric vector of observations.  
params          Vector of parameters (not used in this placeholder).

**Value**

Scalar value representing the pseudo log-likelihood.

---

CDF	<i>Estimate stable distribution parameters using classical ECF regression</i>
-----	---

---

**Description**

Estimate stable distribution parameters using classical ECF regression

**Usage**

CDF(x, u)

**Arguments**

x	Numeric vector of data.
u	Vector of frequency values.

**Value**

A list with estimated parameters: alpha, beta, gamma, and delta.

---

clip	<i>Clip values between lower and upper bounds</i>
------	---

---

**Description**

Clip values between lower and upper bounds

**Usage**

clip(x, lower, upper)

**Arguments**

x	Numeric input.
lower	Minimum allowed value.
upper	Maximum allowed value.

**Value**

Clipped numeric vector.

---

 compare\_em\_vs\_em\_gibbs

*Compare standard EM and EM with Gibbs sampling using kernel ECF*


---

**Description**

Compares log-likelihood and weight estimates between standard EM and EM with Gibbs sampling.

**Usage**

```
compare_em_vs_em_gibbs(data, n_runs = 20)
```

**Arguments**

data	Numeric vector. Observed data.
n_runs	Integer. Number of runs for comparison.

**Value**

Data frame with log-likelihoods and weights for each method across runs.

---

compare\_estimators\_on\_simulations

*Compare MLE, ECF, and McCulloch estimators on simulated data*


---

**Description**

Runs multiple simulations and compares the mean squared error (MSE) of parameter estimates from MLE, kernel ECF, and McCulloch methods.

**Usage**

```
compare_estimators_on_simulations(
  n_samples = 1000,
  n_runs = 30,
  interp_alpha = NULL,
  interp_beta = NULL
)
```

**Arguments**

n_samples	Integer. Number of samples per simulation.
n_runs	Integer. Number of simulation runs.
interp_alpha	Optional interpolation function for alpha (McCulloch).
interp_beta	Optional interpolation function for beta (McCulloch).

**Value**

Data frame of simulation results with MSE and estimated parameters.

---

compare\_methods\_across\_configs

*Compare McCulloch, ECF, and MLE methods across parameter configurations*

---

**Description**

Evaluates and compares three estimation methods (McCulloch, ECF, MLE) over a set of parameter configurations.

**Usage**

```
compare_methods_across_configs(parameter_configs, trials = 20, n = 1000)
```

**Arguments**

parameter\_configs      Named list of parameter sets (each a list with alpha, beta, gamma, delta).  
trials                    Integer. Number of trials per configuration.  
n                         Integer. Number of samples per trial.

**Value**

A named list of results with MSE values for each method and configuration.

---

compare\_methods\_with\_gibbs

*Compare estimation methods with and without Gibbs sampling*

---

**Description**

Visualizes and compares mixture fits from EM, ECF (kernel and empirical), and Gibbs sampling.

**Usage**

```
compare_methods_with_gibbs(  
  data,  
  em_params,  
  ecf_kernel_params,  
  ecf_empirical_params,  
  fig_dir = tempdir()  
)
```

**Arguments**

data	Numeric vector of observations.
em_params	List of EM-estimated parameters.
ecf_kernel_params	List of kernel ECF-estimated parameters.
ecf_empirical_params	List of empirical ECF-estimated parameters.
fig_dir	Optional path to save plots. Defaults to tempdir().

**Value**

NULL (plots are saved to fig\_dir)

---

compute\_model\_metrics *Compute log-likelihood, AIC, and BIC for alpha-stable model*

---

**Description**

Evaluates the model fit by computing the log-likelihood, AIC, and BIC for a given set of alpha-stable parameters.

**Usage**

```
compute_model_metrics(data, params)
```

**Arguments**

data	Numeric vector of observations.
params	Vector of parameters: alpha, beta, gamma, delta.

**Value**

List with log\_likelihood, AIC, BIC, and optional error message.

---

`compute_quantile_ratios`*Compute McCulloch quantile ratios from sample data*

---

**Description**

This function calculates four key ratios used in McCulloch's method for estimating stable distribution parameters: - `v_alpha`: shape indicator - `v_beta`: skewness indicator - `v_gamma`: scale proxy - `v_delta`: location proxy

**Usage**

```
compute_quantile_ratios(X)
```

**Arguments**

`X` Numeric vector of sample data.

**Value**

A list containing `v_alpha`, `v_beta`, `v_gamma`, and `v_delta`.

---

`compute_serial_interval`*Compute serial interval from CSV file*

---

**Description**

Calculates the serial interval from a CSV file containing date columns.

**Usage**

```
compute_serial_interval(filepath)
```

**Arguments**

`filepath` Character. Path to the CSV file.

**Value**

Numeric vector of serial intervals.

---

cosine\_exp\_ralpha      *Cosine exponential function*

---

**Description**

Cosine exponential function

**Usage**

```
cosine_exp_ralpha(r, x, alpha, beta, delta, omega)
```

**Arguments**

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric value of the integrand.

---

cosine\_log\_weighted\_exp\_ralpha  
*Cosine-log-weighted exponential with  $r^{(-\alpha)}$  term*

---

**Description**

Cosine-log-weighted exponential with  $r^{(-\alpha)}$  term

**Usage**

```
cosine_log_weighted_exp_ralpha(r, x, alpha, beta, delta, omega)
```

**Arguments**

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric value of the integrand.

---

DONNEE\_with\_serial\_interval

*Example serial interval data*

---

**Description**

A data frame containing serial interval information for transmission pairs.

**Usage**

```
data(DONNEE_with_serial_interval)
```

**Format**

A data frame with columns:

**ID** Integer. Unique identifier.

**x.lb** Date. Lower bound of infector symptom onset (YYYY-MM-DD).

**x.ub** Date. Upper bound of infector symptom onset (YYYY-MM-DD).

**y** Date. Infectee symptom onset (YYYY-MM-DD).

**serial\_interval** Integer. Serial interval in days.

**Source**

Simulated or real data for demonstration.

---

ecf\_components

*Extract magnitude and phase components from ECF*

---

**Description**

Extract magnitude and phase components from ECF

**Usage**

```
ecf_components(phi_vals)
```

**Arguments**

**phi\_vals** Complex vector of ECF values

**Value**

A list with two numeric vectors: **y\_vals** (log-log magnitude) and **arg\_vals** (phase).

---

ecf_empirical	<i>Compute empirical characteristic function</i>
---------------	--

---

**Description**

Compute empirical characteristic function

**Usage**

```
ecf_empirical(X, t_grid)
```

**Arguments**

X	Numeric vector of data
t_grid	Vector of frequency values

**Value**

Complex vector of empirical characteristic function values.

---

ecf_estimate_all	<i>Estimate all stable parameters from empirical characteristic function</i>
------------------	--

---

**Description**

Estimate all stable parameters from empirical characteristic function

**Usage**

```
ecf_estimate_all(X, t_grid = NULL, weights = NULL)
```

**Arguments**

X	Numeric vector of data.
t_grid	Optional vector of frequencies (default: seq(0.1, 1.0, length.out = 50)).
weights	Optional vector of weights.

**Value**

A list containing estimated parameters: alpha, beta, gamma, and delta.

---

ecf_fn	<i>Empirical Characteristic Function</i>
--------	--

---

**Description**

Empirical Characteristic Function

**Usage**

```
ecf_fn(x, u, method = "simple")
```

**Arguments**

x	Numeric vector of data.
u	Numeric vector of frequencies.
method	Method: "simple", "kernel", or "recursive".

**Value**

List with magnitude and phase of ECF.

---

ecf_regression	<i>Estimate stable parameters using weighted ECF regression</i>
----------------	---

---

**Description**

Estimate stable parameters using weighted ECF regression

**Usage**

```
ecf_regression(x, u)
```

**Arguments**

x	Numeric vector of data.
u	Vector of frequency values.

**Value**

A list with estimated parameters: alpha, beta, gamma, and del ta.

---

em_alpha_stable	<i>EM algorithm for alpha-stable mixture</i>
-----------------	--

---

**Description**

Estimates parameters of an alpha-stable mixture using EM with optional random initialization.

**Usage**

```
em_alpha_stable(
  data,
  n_components = 2,
  max_iter = 100,
  tol = 1e-04,
  random_init = TRUE,
  debug = TRUE
)
```

**Arguments**

data	Numeric vector of observations.
n_components	Integer. Number of mixture components.
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance.
random_init	Logical. Whether to use random initialization.
debug	Logical. Whether to print debug information.

**Value**

List with estimated weights and parameters (alpha, beta, gamma, delta).

---

em_estimate_stable_from_cdf	<i>EM algorithm for mixture of alpha-stable distributions using CDF-based ECF</i>
-----------------------------	---

---

**Description**

Performs EM estimation of a two-component alpha-stable mixture using a simplified empirical characteristic function derived from the cumulative distribution function (CDF).

**Usage**

```
em_estimate_stable_from_cdf(data, max_iter = 100, tol = 1e-04)
```

**Arguments**

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

**Value**

A list with estimated component parameters (alpha, beta, gamma, del ta) and mixture weight (w).

---

em\_estimate\_stable\_from\_cdf\_with\_gibbs

*EM algorithm for alpha-stable mixture using CDF-based ECF and Gibbs M-step*

---

**Description**

Performs EM estimation of a two-component alpha-stable mixture using a simplified empirical characteristic function derived from the cumulative distribution function (CDF), with the M-step replaced by Gibbs sampling.

**Usage**

```
em_estimate_stable_from_cdf_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

**Arguments**

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

**Value**

A list with estimated component parameters (alpha, beta, gamma, del ta) and mixture weight (w).

---

 em\_estimate\_stable\_kernel\_ecf

*EM algorithm for mixture of alpha-stable distributions using kernel ECF*

---

### Description

Performs EM estimation of a two-component alpha-stable mixture using kernel-smoothed empirical characteristic function (ECF).

### Usage

```
em_estimate_stable_kernel_ecf(data, max_iter = 100, tol = 1e-04)
```

### Arguments

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

### Value

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

em\_estimate\_stable\_kernel\_ecf\_with\_gibbs

*EM algorithm for alpha-stable mixture using kernel ECF and Gibbs M-step*

---

### Description

Performs EM estimation of a two-component alpha-stable mixture using kernel-smoothed empirical characteristic function (ECF), with the M-step replaced by Gibbs sampling.

### Usage

```
em_estimate_stable_kernel_ecf_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

### Arguments

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

### Value

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

em\_estimate\_stable\_recursive\_ecf

*EM algorithm for mixture of alpha-stable distributions using recursive ECF*

---

### Description

Performs Expectation-Maximization (EM) to estimate parameters of a two-component alpha-stable mixture model using recursive kernel smoothing on the empirical characteristic function.

### Usage

```
em_estimate_stable_recursive_ecf(data, max_iter = 100, tol = 1e-04)
```

### Arguments

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

### Value

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

em\_estimate\_stable\_recursive\_ecf\_with\_gibbs

*EM algorithm for alpha-stable mixture using recursive ECF and Gibbs M-step*

---

### Description

Performs EM estimation of a two-component alpha-stable mixture using recursive kernel smoothing on the empirical characteristic function (ECF), with the M-step replaced by Gibbs sampling.

### Usage

```
em_estimate_stable_recursive_ecf_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

### Arguments

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

### Value

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

 em\_estimate\_stable\_weighted\_ols

*EM algorithm for mixture of alpha-stable distributions using weighted OLS*

---

**Description**

Performs EM estimation of a two-component alpha-stable mixture using weighted least squares regression on the ECF.

**Usage**

```
em_estimate_stable_weighted_ols(data, max_iter = 100, tol = 1e-04)
```

**Arguments**

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

**Value**

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

em\_estimate\_stable\_weighted\_ols\_with\_gibbs

*EM algorithm for alpha-stable mixture using weighted OLS and Gibbs M-step*

---

**Description**

Performs EM estimation of a two-component alpha-stable mixture using weighted least squares regression on the ECF, with the M-step replaced by Gibbs sampling.

**Usage**

```
em_estimate_stable_weighted_ols_with_gibbs(data, max_iter = 100, tol = 1e-04)
```

**Arguments**

data	Numeric vector of observations.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on log-likelihood.

**Value**

A list with estimated component parameters (alpha, beta, gamma, delta) and mixture weight (w).

---

em\_estimation\_mixture *EM algorithm for two-component Gaussian mixture*

---

**Description**

Estimates parameters of a Gaussian mixture using the EM algorithm.

**Usage**

```
em_estimation_mixture(data, max_iter = 100, tol = 1e-06)
```

**Arguments**

data	Numeric vector of observations.
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance.

**Value**

A list containing estimated mixture parameters: pi, mu1, mu2, sigma1, sigma2.

---

em\_fit\_alpha\_stable\_mixture  
*EM algorithm for two-component alpha-stable mixture using MLE*

---

**Description**

Estimates parameters of a two-component alpha-stable mixture using MLE and EM.

**Usage**

```
em_fit_alpha_stable_mixture(  
  data,  
  max_iter = 200,  
  tol = 1e-04,  
  return_trace = FALSE  
)
```

**Arguments**

data	Numeric vector of observations.
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance on log-likelihood.
return_trace	Logical. Whether to return trace of responsibilities and log-likelihoods.

**Value**

List with estimated parameters and optional trace.

---

em_stable_mixture	<i>EM algorithm for alpha-stable mixture using a custom estimator</i>
-------------------	---

---

**Description**

Performs EM estimation using a user-defined parameter estimator and ECF frequencies.

**Usage**

```
em_stable_mixture(data, u, estimator_func, max_iter = 300, epsilon = 0.001)
```

**Arguments**

data	Numeric vector of observations.
u	Numeric vector of frequency values for ECF.
estimator_func	Function to estimate stable parameters.
max_iter	Integer. Maximum number of EM iterations.
epsilon	Numeric. Convergence threshold on log-likelihood.

**Value**

List with estimated weights, parameters, and log-likelihood.

---

empirical_r0	<i>Empirical R0 estimation using growth model</i>
--------------	---

---

**Description**

Estimates the basic reproduction number (R0) using a growth model (exponential or logistic) fitted to incidence data.

**Usage**

```
empirical_r0(incidence, serial_interval, growth_model = "exponential")
```

**Arguments**

incidence	Numeric vector of incidence counts.
serial_interval	Mean serial interval.
growth_model	Type of growth model: "exponential" or "logistic".

**Value**

Estimated R0 value.

---

ensure\_positive\_scale *Ensure positive scale parameter*

---

**Description**

Ensure positive scale parameter

**Usage**

```
ensure_positive_scale(val, min_scale = 1e-06, max_scale = 1e+06)
```

**Arguments**

val	Numeric input.
min_scale	Minimum scale.
max_scale	Maximum scale.

**Value**

A finite positive scale value.

---

est\_r0\_ml *Estimate R0 using maximum likelihood*

---

**Description**

Computes the basic reproduction number (R0) using a maximum likelihood approach based on secondary cases and generation time weights.

**Usage**

```
est_r0_ml(W, N)
```

**Arguments**

W	Generation time distribution.
N	Secondary case counts.

**Value**

Estimated R0 value.

---

est_r0_mle	<i>MLE estimation of R0 using generation time</i>
------------	---

---

**Description**

Estimates the basic reproduction number (R0) using maximum likelihood and a parametric model based on generation time and incidence data.

**Usage**

```
est_r0_mle(incidence, gen_time)
```

**Arguments**

incidence	Numeric vector of incidence counts.
gen_time	Mean generation time.

**Value**

Estimated R0 value.

---

estimate_alpha_gamma	<i>Estimate alpha and gamma from ECF modulus</i>
----------------------	--

---

**Description**

Estimate alpha and gamma from ECF modulus

**Usage**

```
estimate_alpha_gamma(t_grid, phi_vals, weights = NULL)
```

**Arguments**

t_grid	Vector of frequency values.
phi_vals	Complex vector of ECF values.
weights	Optional vector of weights.

**Value**

A list with elements alpha\_hat and gamma\_hat.

---

estimate\_beta\_delta     *Estimate beta and delta from ECF phase*

---

### Description

Estimate beta and delta from ECF phase

### Usage

```
estimate_beta_delta(t_grid, phi_vals, alpha_hat, gamma_hat, weights = NULL)
```

### Arguments

t_grid	Vector of frequency values.
phi_vals	Complex vector of ECF values.
alpha_hat	Estimated alpha.
gamma_hat	Estimated gamma.
weights	Optional vector of weights.

### Value

A list with elements beta\_hat and delta\_hat.

---

estimate\_mixture\_params  
                                   *Estimate mixture of two stable distributions*

---

### Description

Wrapper function to estimate parameters of a two-component alpha-stable mixture using MLE.

### Usage

```
estimate_mixture_params(data)
```

### Arguments

data	Numeric vector of observations.
------	---------------------------------

### Value

List with estimated weight and parameters for both components.

---

`estimate_stable_from_cdf`*Estimate stable parameters using CDF-based ECF regression*

---

**Description**

Estimate stable parameters using CDF-based ECF regression

**Usage**

```
estimate_stable_from_cdf(data, u)
```

**Arguments**

<code>data</code>	Numeric vector of observations
<code>u</code>	Vector of frequency values

**Value**

A list with estimated parameters: alpha, beta, gamma, and del ta.

---

`estimate_stable_kernel_ecf`*Estimate stable parameters using kernel-based ECF method*

---

**Description**

Estimate stable parameters using kernel-based ECF method

**Usage**

```
estimate_stable_kernel_ecf(data, u)
```

**Arguments**

<code>data</code>	Numeric vector of observations
<code>u</code>	Vector of frequency values

**Value**

A list with estimated parameters: alpha, beta, gamma, and del ta.

---

`estimate_stable_params`*Estimate single stable distribution parameters*

---

**Description**

Wrapper function to estimate stable parameters using one of three methods: "basic", "robust", or "lbfgs".

**Usage**

```
estimate_stable_params(data, method = "robust")
```

**Arguments**

<code>data</code>	Numeric vector of observations.
<code>method</code>	Estimation method: "basic", "robust", or "lbfgs".

**Value**

List with estimated alpha, beta, gamma, delta.

---

`estimate_stable_r`*Estimate stable parameters using method of moments*

---

**Description**

Provides a rough estimation of alpha-stable parameters using empirical moments.

**Usage**

```
estimate_stable_r(x)
```

**Arguments**

<code>x</code>	Numeric vector of data.
----------------	-------------------------

**Value**

List with estimated alpha, beta, gamma, delta.

---

`estimate_stable_recursive_ecf`*Estimate stable parameters using recursive ECF method*

---

**Description**

Estimate stable parameters using recursive ECF method

**Usage**

```
estimate_stable_recursive_ecf(data, u)
```

**Arguments**

<code>data</code>	Numeric vector of observations.
<code>u</code>	Vector of frequency values.

**Value**

A list with estimated parameters: alpha, beta, gamma, and del ta.

---

`estimate_stable_weighted_ols`*Estimate stable parameters using weighted OLS on recursive ECF*

---

**Description**

Estimate stable parameters using weighted OLS on recursive ECF

**Usage**

```
estimate_stable_weighted_ols(data, u)
```

**Arguments**

<code>data</code>	Numeric vector of observations
<code>u</code>	Vector of frequency values

**Value**

A list with estimated parameters: alpha, beta, gamma, and del ta.

---

eta_func	<i>General eta function</i>
----------	-----------------------------

---

**Description**

General eta function

**Usage**

```
eta_func(t, alpha, gamma)
```

**Arguments**

t	Frequency vector.
alpha	Stability parameter.
gamma	Scale parameter.

**Value**

Numeric vector of eta values.

---

eta0	<i>Helper function for eta0 computation</i>
------	---

---

**Description**

Helper function for eta0 computation

**Usage**

```
eta0(u, alpha, gamma, eps = 0.05)
```

**Arguments**

u	Frequency vector.
alpha	Stability parameter.
gamma	Scale parameter.
eps	Tolerance for $\alpha \approx 1$ .

**Value**

Numeric vector of eta0 values.

---

`evaluate_estimation_method`*Evaluate estimation method using MSE over multiple trials*

---

**Description**

Computes the mean squared error (MSE) of a parameter estimation function over several trials.

**Usage**

```
evaluate_estimation_method(  
    estimator_fn,  
    true_params,  
    n = 1000,  
    trials = 20,  
    seed = 42  
)
```

**Arguments**

<code>estimator_fn</code>	Function. Estimation function to evaluate.
<code>true_params</code>	List. True parameters: alpha, beta, gamma, delta.
<code>n</code>	Integer. Number of samples per trial.
<code>trials</code>	Integer. Number of trials to run.
<code>seed</code>	Integer. Random seed for reproducibility.

**Value**

Numeric value representing the average MSE across trials.

---

`evaluate_fit`*Evaluate fit quality using RMSE and log-likelihood*

---

**Description**

Computes RMSE and log-likelihood scores for different mixture models compared to histogram data.

**Usage**

```
evaluate_fit(data, methods, x_vals)
```

**Arguments**

data	Numeric vector of observations.
methods	Named list of parameter sets (each with two components and a weight).
x_vals	Numeric vector. Evaluation grid.

**Value**

Named list of scores per method (RMSE and LogLikelihood).

---

export\_analysis\_report

*Export analysis report to JSON and Excel*

---

**Description**

Saves the results of a stability analysis to a JSON file and an Excel workbook.

**Usage**

```
export_analysis_report(
  data,
  stable_params,
  qcv,
  skew_kurt,
  normality,
  verdict,
  filename = "stable_report"
)
```

**Arguments**

data	Numeric vector of data.
stable_params	List of estimated stable parameters.
qcv	QCV statistic.
skew_kurt	List with skewness and kurtosis.
normality	List of normality test results.
verdict	Final verdict string.
filename	Base name for output files.

**Value**

Invisibly returns a list containing the paths of the exported JSON and Excel files.

---

false\_position\_update *False position method update step*

---

**Description**

Performs a single iteration of the false position (regula falsi) method for root-finding. This method approximates the root of a function by using a linear interpolation between two points where the function changes sign.

**Usage**

```
false_position_update(a, b_n, f_a, f_b, objective_func)
```

**Arguments**

a	Numeric scalar. Left bound of the interval.
b_n	Numeric scalar. Right bound of the interval (current step).
f_a	Numeric scalar. Value of the objective function evaluated at 'a'.
f_b	Numeric scalar. Value of the objective function evaluated at 'b_n'.
objective_func	Function. The objective function whose root is being sought.

**Value**

Numeric scalar. Updated estimate for the root after one iteration.

---

fast\_integrate *Fast numerical integration using trapezoidal rule*

---

**Description**

Fast numerical integration using trapezoidal rule

**Usage**

```
fast_integrate(func, a = -6, b = 6, N = 100)
```

**Arguments**

func	Function to integrate.
a	Lower bound.
b	Upper bound.
N	Number of points.

**Value**

Approximated integral value.

---

fit\_alpha\_stable\_mle    *Fit Alpha-Stable Distribution using MLE (L-BFGS-B)*

---

**Description**

Estimates the parameters of a single alpha-stable distribution using maximum likelihood and the L-BFGS-B optimization method.

**Usage**

```
fit_alpha_stable_mle(data)
```

**Arguments**

data                  Numeric vector of observations.

**Value**

Numeric vector of estimated parameters: alpha, beta, gamma, delta.

---

fit\_mle\_mixture        *Fit MLE Mixture of Two Stable Distributions*

---

**Description**

Estimates parameters of a two-component alpha-stable mixture using maximum likelihood and the L-BFGS-B optimization method.

**Usage**

```
fit_mle_mixture(data)
```

**Arguments**

data                  Numeric vector of observations.

**Value**

Numeric vector of estimated parameters: weight, alpha1, beta1, gamma1, delta1, alpha2, beta2, gamma2, delta2.

---

fit_stable_ecf	<i>Estimate stable parameters using filtered and weighted ECF regression</i>
----------------	--

---

**Description**

Estimate stable parameters using filtered and weighted ECF regression

**Usage**

```
fit_stable_ecf(data, frequencies)
```

**Arguments**

data	Numeric vector of observations.
frequencies	Vector of frequency values.

**Value**

A list with estimated parameters: alpha, beta, gamma, and delta.

---

generate_alpha_stable_mixture	<i>Generate samples from a predefined alpha-stable mixture</i>
-------------------------------	--

---

**Description**

Simulates samples from a mixture of alpha-stable distributions using specified parameters.

**Usage**

```
generate_alpha_stable_mixture(  
  weights,  
  alphas,  
  betas,  
  gammas,  
  deltas,  
  size = 1000  
)
```

**Arguments**

weights	Numeric vector of mixture weights.
alphas	Numeric vector of alpha parameters.
betas	Numeric vector of beta parameters.
gammas	Numeric vector of gamma parameters.
deltas	Numeric vector of delta parameters.
size	Integer. Number of samples to generate.

**Value**

A list with:

**samples** Numeric vector of generated samples.

**labels** Integer vector indicating the component each sample came from.

---

`generate_mcculloch_table`

*Generate McCulloch lookup table from simulated stable samples*

---

**Description**

Simulates alpha-stable samples across a grid of alpha and beta values, and computes quantile-based ratios used for McCulloch estimation. The result is a lookup table indexed by (alpha, beta) keys.

**Usage**

```
generate_mcculloch_table(alpha_grid, beta_grid, size = 1e+05)
```

**Arguments**

alpha_grid	Vector of alpha values to simulate.
beta_grid	Vector of beta values to simulate.
size	Number of samples per simulation.

**Value**

Named list of quantile ratios indexed by "alpha\_beta".

---

generate\_mixture\_data *Simulates a mixture of alpha-stable distributions with randomly sampled parameters.*

---

**Description**

Simulates a mixture of alpha-stable distributions with randomly sampled parameters.

**Usage**

```
generate_mixture_data(K = 2, N = 1000)
```

**Arguments**

**K** Integer. Number of mixture components.  
**N** Integer. Total number of samples to generate.

**Value**

A list containing:

**data** Numeric vector of generated samples.

**params** List of parameters for each component (alpha, beta, gamma, delta, pi).

---

generate\_synthetic\_data

*Generate synthetic data from two alpha-stable components*

---

**Description**

Creates a synthetic dataset composed of two alpha-stable distributions.

**Usage**

```
generate_synthetic_data(n = 1000)
```

**Arguments**

**n** Integer. Total number of samples to generate.

**Value**

Numeric vector of shuffled synthetic data.

---

gibbs_sampler	<i>Gibbs sampler for Gaussian mixture model</i>
---------------	---

---

**Description**

Performs Gibbs sampling for a two-component Gaussian mixture model. Updates latent assignments, component means, variances, and mixing proportions.

**Usage**

```
gibbs_sampler(data, iterations = 1000)
```

**Arguments**

data	Numeric vector of observations.
iterations	Number of Gibbs iterations.

**Value**

List of sampled parameters per iteration.

---

grad_loglik_alpha	<i>Log-likelihood gradient with respect to alpha</i>
-------------------	--

---

**Description**

Log-likelihood gradient with respect to alpha

**Usage**

```
grad_loglik_alpha(alpha, beta, delta, omega, x)
```

**Arguments**

alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).
x	Numeric vector. Observations at which the integral is computed.

**Value**

Numeric scalar: estimated gradient (or NA\_real\_ on invalid input).

---

grad_loglik_beta	<i>Log-likelihood gradient with respect to beta</i>
------------------	---

---

**Description**

Log-likelihood gradient with respect to beta

**Usage**

```
grad_loglik_beta(alpha, beta, delta, omega, x)
```

**Arguments**

alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).
x	Numeric vector. Observations at which the integral is computed.

**Value**

Numeric scalar: estimated gradient (or NA\_real\_ on invalid input).

---

grad_loglik_delta	<i>Log-likelihood gradient with respect to delta (scale)</i>
-------------------	--

---

**Description**

Log-likelihood gradient with respect to delta (scale)

**Usage**

```
grad_loglik_delta(alpha, beta, delta, omega, x)
```

**Arguments**

alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).
x	Numeric vector. Observations at which the integral is computed.

**Value**

Numeric scalar: estimated gradient (or NA\_real\_ on invalid input).

---

grad_loglik_omega	<i>Log-likelihood gradient with respect to omega (location)</i>
-------------------	---

---

**Description**

Log-likelihood gradient with respect to omega (location)

**Usage**

```
grad_loglik_omega(alpha, beta, delta, omega, x)
```

**Arguments**

alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).
x	Numeric vector. Observations at which the integral is computed.

**Value**

Numeric scalar: estimated gradient (or NA\_real\_ on invalid input).

---

Im	<i>Imaginary part of the ECF integral</i>
----	---

---

**Description**

Imaginary part of the ECF integral

**Usage**

```
Im(r, u, x, bn)
```

**Arguments**

r	Integration variable.
u	Frequency.
x	Data point.
bn	Bandwidth.

**Value**

Imaginary component value.

---

Int_Im	<i>Integrate imaginary component over <math>\mathbb{R}</math></i>
--------	---

---

**Description**

Integrate imaginary component over  $\mathbb{R}$

**Usage**

Int\_Im(u, x, bn)

**Arguments**

u	Frequency.
x	Data point.
bn	Bandwidth.

**Value**

Integrated imaginary value.

---

Int_Re	<i>Integrate real component over <math>\mathbb{R}</math></i>
--------	--

---

**Description**

Integrate real component over  $\mathbb{R}$

**Usage**

Int\_Re(u, x, bn)

**Arguments**

u	Frequency.
x	Data point.
bn	Bandwidth.

**Value**

Integrated real value.

---

integrate\_cosine      *Integrate cosine exponential*

---

**Description**

Computes the integral of the cosine exponential function for stable distributions.

**Usage**

```
integrate_cosine(x, alpha, beta, delta, omega)
```

**Arguments**

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric vector of integral values.

---

integrate\_cosine\_log\_weighted  
*Integrate cosine-log-weighted exponential*

---

**Description**

Computes the integral of the cosine-log-weighted exponential function for stable distributions.

**Usage**

```
integrate_cosine_log_weighted(x, alpha, beta, delta, omega)
```

**Arguments**

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric vector of integral values.

---

integrate\_function     *Robust integration helper function*

---

**Description**

Applies ‘safe\_integrate’ to a given integrand over a vector of x values.

**Usage**

```
integrate_function(f, x, alpha, beta, delta, omega, upper = 50, eps = 1e-08)
```

**Arguments**

f	Function. The integrand function to be evaluated.
x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).
upper	Numeric scalar. Upper limit of integration (default = 50).
eps	Numeric scalar. Tolerance for numerical integration (default = 1e-8).

**Value**

Numeric vector of integral values evaluated at each element of x.

---

integrate\_sine     *Integrate sin exponential*

---

**Description**

Computes the integral of the sin exponential function for stable distributions.

**Usage**

```
integrate_sine(x, alpha, beta, delta, omega)
```

**Arguments**

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric vector of integral values.

---

`integrate_sine_log_weighted`

*Integrate sine-log-weighted exponential*

---

**Description**

Computes the integral of the sine-log-weighted exponential function for stable distributions.

**Usage**

```
integrate_sine_log_weighted(x, alpha, beta, delta, omega)
```

**Arguments**

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric vector of integral values.

---

integrate\_sine\_r\_weighted  
*Integrate sine-r-weighted exponential*

---

**Description**

Computes the integral of the sine-r-weighted exponential function for stable distributions.

**Usage**

```
integrate_sine_r_weighted(x, alpha, beta, delta, omega)
```

**Arguments**

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric vector of integral values.

---

integrate\_sine\_weighted  
*Integration wrappers for specific integrands*

---

**Description**

Integration wrappers for specific integrands

**Usage**

```
integrate_sine_weighted(x, alpha, beta, delta, omega)
```

**Arguments**

x	Numeric vector. Observations at which the integral is computed.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric vector of integral values.

---

kde\_bandwidth\_plugin    *KDE bandwidth selection using plugin method*

---

**Description**

KDE bandwidth selection using plugin method

**Usage**

kde\_bandwidth\_plugin(X, alpha)

**Arguments**

X	Numeric vector of data.
alpha	Stability parameter.

**Value**

Bandwidth value.

---

L\_stable                    *Negative log-likelihood for stable distribution using dstable*

---

**Description**

Computes the negative log-likelihood of a stable distribution given parameters and data.

**Usage**

L\_stable(param, obs)

**Arguments**

param	Numeric vector of parameters: alpha, beta, gamma, delta.
obs	Numeric vector of observations.

**Value**

Scalar value of negative log-likelihood.

---

log\_likelihood\_mixture  
*Log-likelihood for mixture of stable distributions*

---

**Description**

Log-likelihood for mixture of stable distributions

**Usage**

```
log_likelihood_mixture(params, data)
```

**Arguments**

params	Numeric vector of parameters.
data	Numeric vector of observations.

**Value**

Negative log-likelihood (for minimization).

---

Max\_vrai *Maximum likelihood estimation using Nelder-Mead*

---

**Description**

Estimates parameters of a stable distribution using the Nelder-Mead method with penalty constraints.

**Usage**

```
Max_vrai(x)
```

**Arguments**

x	Numeric vector of observations.
---	---------------------------------

**Value**

List with estimated alpha, beta, gamma, delta.

---

 mcculloch\_lookup\_estimate

*Estimate stable parameters using McCulloch lookup*


---

### Description

Estimates alpha and beta using quantile ratios and interpolation functions. Gamma and delta are derived directly from quantiles.

### Usage

```
mcculloch_lookup_estimate(
  X,
  interpolators = NULL,
  interp_alpha = NULL,
  interp_beta = NULL
)
```

### Arguments

X	Numeric vector of data.
interpolators	Optional list with interp_alpha and interp_beta functions.
interp_alpha	Optional function to interpolate alpha.
interp_beta	Optional function to interpolate beta.

### Value

List with estimated alpha, beta, gamma, delta.

---

 mcculloch\_quantile\_init

*Initialization using McCulloch quantile method*


---

### Description

Estimates all four stable parameters using quantile ratios and mock lookup. Useful for initializing optimization algorithms.

### Usage

```
mcculloch_quantile_init(X)
```

### Arguments

X	Numeric vector of data.
---	-------------------------

**Value**

List with estimated alpha, beta, gamma, delta.

---

metropolis\_hastings     *Metropolis-Hastings MCMC for stable mixture clustering*

---

**Description**

Performs Metropolis-Hastings sampling to estimate parameters of a two-component alpha-stable mixture model. Proposals are generated for each parameter and accepted based on likelihood ratios. Cluster assignments are updated at each iteration.

**Usage**

```
metropolis_hastings(fct, iterations, lok, aa = c(1, 1), proposal_std = 0.1)
```

**Arguments**

fct	Function to estimate initial parameters.
iterations	Number of MCMC iterations.
lok	Numeric vector of observations.
aa	Prior parameters for Dirichlet distribution.
proposal_std	Standard deviation for proposal distributions.

**Value**

List of sampled parameters and weights across iterations.

---

mixture\_stable\_pdf     *Mixture of two stable PDFs*

---

**Description**

Mixture of two stable PDFs

**Usage**

```
mixture_stable_pdf(x, p1, p2, w)
```

**Arguments**

x	Numeric vector.
p1	List of parameters for first stable distribution.
p2	List of parameters for second stable distribution.
w	Mixture weight ( $0 < w < 1$ ).

**Value**

Numeric vector of mixture PDF values.

---

mle_estimate	<i>Simple MLE estimation with default starting values</i>
--------------	---

---

**Description**

Estimates stable distribution parameters using Nelder-Mead optimization from default or user-provided starting values.

**Usage**

```
mle_estimate(X, x0 = NULL)
```

**Arguments**

X	Numeric vector of observations.
x0	Optional starting values.

**Value**

List with estimated alpha, beta, gamma, delta.

---

mock_gibbs_sampling	<i>Mock Gibbs sampling for alpha-stable mixture estimation</i>
---------------------	--

---

**Description**

Performs a simplified Gibbs sampling procedure to estimate parameters of a two-component alpha-stable mixture. Samples are drawn from prior distributions and evaluated using log-likelihood. The best parameter set is selected based on likelihood.

**Usage**

```
mock_gibbs_sampling(data, n_samples = 500, verbose = FALSE)
```

**Arguments**

data	Numeric vector of observations.
n_samples	Number of Gibbs samples to draw.
verbose	Logical, whether to print best log-likelihood.

**Value**

List containing best\_params and all sampled parameter sets.

---

`mock_lookup_alpha_beta`*Mock lookup for alpha and beta (fallback)*

---

**Description**

Provides a fallback estimation of alpha and beta from quantile ratios using simple linear approximations.

**Usage**`mock_lookup_alpha_beta(v_alpha, v_beta)`**Arguments**

<code>v_alpha</code>	Quantile-based shape ratio.
<code>v_beta</code>	Quantile-based skewness ratio.

**Value**

List with estimated alpha and beta.

---

`N_epanechnikov`*Epanechnikov kernel*

---

**Description**

Epanechnikov kernel

**Usage**`N_epanechnikov(z)`**Arguments**

<code>z</code>	Numeric input.
----------------	----------------

**Value**

Kernel value.

---

N\_gaussian

*Gaussian kernel*

---

**Description**

Gaussian kernel

**Usage**

N\_gaussian(z)

**Arguments**

z                    Numeric input.

**Value**

Kernel value.

---

N\_uniform

*Uniform kernel*

---

**Description**

Uniform kernel

**Usage**

N\_uniform(z)

**Arguments**

z                    Numeric input.

**Value**

Kernel value.

---

 negative\_log\_likelihood

*Negative log-likelihood for single stable distribution*


---

**Description**

Negative log-likelihood for single stable distribution

**Usage**

```
negative_log_likelihood(params, data)
```

**Arguments**

params	Numeric vector (alpha, beta, gamma, delta).
data	Numeric vector of observations.

**Value**

Negative log-likelihood.

---

normalized_grad_alpha	<i>Normalized gradient for alpha parameter Computes the normalized gradient of the log-likelihood with respect to the alpha parameter over a set of observations. This is useful for optimization routines where scale-invariant updates are preferred.</i>
-----------------------	---

---

**Description**

Normalized gradient for alpha parameter Computes the normalized gradient of the log-likelihood with respect to the alpha parameter over a set of observations. This is useful for optimization routines where scale-invariant updates are preferred.

**Usage**

```
normalized_grad_alpha(alpha, beta, delta, omega, x)
```

**Arguments**

alpha	Stability parameter.
beta	Skewness parameter.
delta	Scale parameter.
omega	Location parameter.
x	Numeric vector of observations.

**Value**

Scalar value representing the normalized gradient.

---

normalized\_objective\_beta

*Normalized objective for beta parameter*

---

**Description**

Computes the normalized gradient of the log-likelihood with respect to the beta parameter using the latest values of alpha, delta, and omega stored in global vectors.

**Usage**

normalized\_objective\_beta(beta, X1, L\_alpha, L\_delta, L\_omega)

**Arguments**

beta	Skewness parameter.
X1	Numeric vector of observations.
L_alpha	Vector of alpha values (global memory).
L_delta	Vector of delta values (global memory).
L_omega	Vector of omega values (global memory).

**Value**

Scalar value representing the normalized gradient.

---

normalized\_objective\_delta

*Normalized objective for delta parameter*

---

**Description**

Computes the normalized gradient of the log-likelihood with respect to the delta parameter using the latest values of alpha, beta, and omega stored in global vectors.

**Usage**

normalized\_objective\_delta(delta, X1, L\_alpha, L\_beta, L\_omega)

**Arguments**

delta	Location parameter.
X1	Numeric vector of observations.
L_alpha	Vector of alpha values.
L_beta	Vector of beta values.
L_omega	Vector of omega values.

**Value**

Scalar value representing the normalized gra

---

normalized\_objective\_omega

*Normalized objective for omega parameter*

---

**Description**

Computes the normalized gradient of the log-likelihood with respect to the omega parameter using the latest values of alpha, beta, and delta stored in global vectors.

**Usage**

```
normalized_objective_omega(omega, X1, L_alpha, L_beta, L_delta)
```

**Arguments**

omega	Location parameter.
X1	Numeric vector of observations.
L_alpha	Vector of alpha values.
L_beta	Vector of beta values.
L_delta	Vector of delta values.

**Value**

Scalar value representing the normalized gradient.

---

plot_comparison	<i>Compare EM-estimated mixture with a non-optimized reference model</i>
-----------------	--

---

**Description**

Plots the fitted EM mixture density alongside a manually defined reference mixture to visually compare estimation quality.

**Usage**

```
plot_comparison(data, p1, p2, w)
```

**Arguments**

data	Numeric vector of observations.
p1	Vector of parameters for component 1 (alpha, beta, gamma, delta).
p2	Vector of parameters for component 2.
w	Mixing weight for component 1.

**Value**

Invisibly returns the file path to the saved PNG image of the comparison plot.

---

plot_distributions	<i>Plot histogram with normal and stable PDF overlays</i>
--------------------	---

---

**Description**

Displays a histogram of the data with overlaid normal and alpha-stable probability density functions.

**Usage**

```
plot_distributions(x, params_stable)
```

**Arguments**

x	Numeric vector of data.
params_stable	List with alpha, beta, gamma, delta.

**Value**

NULL (plot is displayed as a side effect)

---

`plot_effective_reproduction_number`*Plot effective reproduction number ( $R_e$ ) over time*

---

**Description**

Computes and visualizes the effective reproduction number ( $R_t$ ) over time using incidence data and generation time distribution. Optionally overlays a smoothed spline curve.

**Usage**

```
plot_effective_reproduction_number(  
  GT,  
  S,  
  inc,  
  dates,  
  est_r0_ml,  
  RT,  
  output_file = "RN_avec_dates_EM-ML.pdf"  
)
```

**Arguments**

GT	Numeric vector. Generation time distribution.
S	Numeric vector. Secondary cases.
inc	Numeric vector. Incidence time series.
dates	Date vector corresponding to observations.
est_r0_ml	Function to estimate $R_0$ .
RT	Function to compute $R_t$ .
output_file	Output PDF filename.

**Value**

NULL (plot is saved as PDF)

---

`plot_final_mixture_fit`*Plot final fitted mixture of alpha-stable distributions*

---

**Description**

Plots the final fitted mixture PDF over the data histogram.

**Usage**

```
plot_final_mixture_fit(data, p1, p2, w)
```

**Arguments**

data	Numeric vector of observations.
p1	Vector of parameters for component 1.
p2	Vector of parameters for component 2.
w	Mixing weight for component 1.

**Value**

NULL (plot is saved as PNG)

---

plot_fit_vs_true	<i>Plot true vs estimated mixture density</i>
------------------	---

---

**Description**

Compares the true mixture density with the estimated one by overlaying both on the histogram of the observed data.

**Usage**

```
plot_fit_vs_true(true_params, est_params, data, bins = 100)
```

**Arguments**

true_params	List of true mixture components with alpha, beta, gamma, delta, pi.
est_params	List of estimated mixture components.
data	Numeric vector of observations.
bins	Number of histogram bins.

**Value**

NULL (plot is displayed)

---

`plot_fit_vs_true_methods`

*Compare estimated mixture densities from two methods against the true density*

---

**Description**

Plots the true mixture density and compares it with two estimated densities obtained from different methods (e.g., MLE vs ECF).

**Usage**

```
plot_fit_vs_true_methods(  
  data,  
  true_params,  
  method1 = "MLE",  
  method2 = "ECF",  
  bins = 100  
)
```

**Arguments**

<code>data</code>	Numeric vector of observations.
<code>true_params</code>	List of true mixture components.
<code>method1</code>	First estimation method ("MLE", "ECF", etc.).
<code>method2</code>	Second estimation method.
<code>bins</code>	Number of histogram bins.

**Value**

NULL (plot is displayed)

---

`plot_method_comparison`

*Plot RMSE and Log-Likelihood comparison across methods*

---

**Description**

Generates bar plots comparing RMSE and log-likelihood values across different estimation methods.

**Usage**

```
plot_method_comparison(scores)
```

**Arguments**

scores            Named list of score objects with RMSE and LogLikelihood.

**Value**

NULL (plot is saved as PNG)

---

plot\_mixture            *Plot mixture of two alpha-stable distributions*

---

**Description**

Plots the histogram of data and overlays the estimated mixture density and its components.

**Usage**

```
plot_mixture(data, params1, params2, w, label = "EM")
```

**Arguments**

data            Numeric vector of observations.  
 params1        Vector of parameters for component 1 (alpha, beta, gamma, delta).  
 params2        Vector of parameters for component 2.  
 w              Mixing weight for component 1.  
 label         Optional label for output file.

**Value**

NULL (plot is saved as PNG)

---

plot\_mixture\_fit        *Plot mixture fit with individual components*

---

**Description**

Displays the estimated mixture density and optionally its individual components over a histogram of the data.

**Usage**

```
plot_mixture_fit(
  data,
  estimated_params,
  bins = 200,
  plot_components = TRUE,
  save_path = NULL,
  show_plot = TRUE,
  title = "Mixture of Alpha-Stable Distributions"
)
```

**Arguments**

data	Numeric vector of observations.
estimated_params	List with weights, alphas, betas, gammas, deltas.
bins	Number of histogram bins.
plot_components	Logical, whether to show individual components.
save_path	Optional PNG filename.
show_plot	Logical, whether to display the plot.
title	Plot title.

**Value**

Invisibly returns the file path to the saved plot (if `save_path` is provided), or `NULL` otherwise.

---

`plot_real_mixture_fit` *Plot fitted mixture on real dataset*

---

**Description**

Plots the fitted mixture model over a histogram of real-world data, showing both components and the overall mixture.

**Usage**

```
plot_real_mixture_fit(X, result)
```

**Arguments**

X	Numeric vector of observations.
result	List containing <code>params1</code> , <code>params2</code> , and <code>lambda1</code> .

**Value**

`NULL` (plot is saved as PNG)

---

plot\_results                      *Plot posterior mixture density from MCMC samples*

---

### Description

Visualizes the estimated two-component alpha-stable mixture density using parameters obtained from MCMC sampling. Optionally overlays the true density for comparison.

### Usage

```
plot_results(
  M2_w1,
  M2_alpha1,
  M2_beta1,
  M2_delta1,
  M2_omega1,
  M2_w2,
  M2_alpha2,
  M2_beta2,
  M2_delta2,
  M2_omega2,
  xx,
  xx_true = NULL,
  yy_true = NULL
)
```

### Arguments

M2_w1	Numeric scalar. Mixture weight of first component.
M2_alpha1	Numeric vector. Stability parameter samples of first component.
M2_beta1	Numeric vector. Skewness parameter samples of first component.
M2_delta1	Numeric vector. Location parameter samples of first component.
M2_omega1	Numeric vector. Scale parameter samples of first component.
M2_w2	Numeric scalar. Mixture weight of second component.
M2_alpha2	Numeric vector. Stability parameter samples of second component.
M2_beta2	Numeric vector. Skewness parameter samples of second component.
M2_delta2	Numeric vector. Location parameter samples of second component.
M2_omega2	Numeric vector. Scale parameter samples of second component.
xx	Numeric vector of grid values for density evaluation.
xx_true	Optional numeric vector for true density x-values.
yy_true	Optional numeric vector for true density y-values.

### Value

Invisibly returns the file path to the saved PNG image of the posterior mixture density plot.

---

plot_trace	<i>Plot trace of a parameter across MCMC iterations</i>
------------	---

---

**Description**

Displays the evolution of a parameter across MCMC iterations to assess convergence.

**Usage**

```
plot_trace(samples, param_name)
```

**Arguments**

samples	List of sample objects from MCMC.
param_name	Name of the parameter to trace.

**Value**

NULL (plot is displayed as a side effect)

---

plot_vs_normal_stable	<i>Plot comparison between normal and stable distributions</i>
-----------------------	--

---

**Description**

Displays a histogram of the data with overlaid normal and alpha-stable PDFs.

**Usage**

```
plot_vs_normal_stable(x, params_stable, fig_path = NULL)
```

**Arguments**

x	Numeric vector of data.
params_stable	List with alpha, beta, gamma, delta.
fig_path	Optional path to save the plot (PNG). If NULL, uses tempdir().

**Value**

NULL (saves plot to file)

---

qcv_stat	<i>QCV statistic for tail heaviness</i>
----------	---

---

**Description**

Computes the QCV (Quantile Conditional Variance) statistic to assess tail heaviness of a distribution.

**Usage**

```
qcv_stat(x)
```

**Arguments**

x	Numeric vector of data.
---	-------------------------

**Value**

Numeric: QCV value.

---

r_stable_pdf	<i>Robust stable PDF computation</i>
--------------	--------------------------------------

---

**Description**

Robust stable PDF computation

**Usage**

```
r_stable_pdf(x, alpha, beta, scale, location)
```

**Arguments**

x	Points at which to evaluate the density.
alpha	Stability parameter (0,2].
beta	Skewness parameter [-1,1].
scale	Scale (>0).
location	Location parameter.

**Value**

Numeric vector of density values.

---

Re	<i>Real part of the ECF integral</i>
----	--------------------------------------

---

**Description**

Real part of the ECF integral

**Usage**

Re(r, u, x, bn)

**Arguments**

r	Integration variable.
u	Frequency.
x	Data point.
bn	Bandwidth.

**Value**

Real component value.

---

recursive_weight	<i>Recursive weight function</i>
------------------	----------------------------------

---

**Description**

Recursive weight function

**Usage**

recursive\_weight(l)

**Arguments**

l	Index.
---	--------

**Value**

Weight value.

---

robust\_ecf\_regression *Estimate stable parameters using robust ECF regression*

---

**Description**

Estimate stable parameters using robust ECF regression

**Usage**

```
robust_ecf_regression(x, u)
```

**Arguments**

x	Numeric vector of data.
u	Vector of frequency values.

**Value**

A list with estimated parameters: alpha, beta, gamma, and delta.

---

robust\_mle\_estimate *Robust MLE estimation with multiple starting points*

---

**Description**

Performs multiple MLE estimations with randomized starting points and selects the best result based on log-likelihood.

**Usage**

```
robust_mle_estimate(data, n_starts = 5)
```

**Arguments**

data	Numeric vector of observations.
n_starts	Number of random initializations.

**Value**

List with best estimated alpha, beta, gamma, delta.

---

rstable	<i>Generate random samples from stable distribution</i>
---------	---

---

**Description**

Generate random samples from stable distribution

**Usage**

```
rstable(n, alpha, beta, gamma = 1, delta = 0, pm = 1)
```

**Arguments**

n	Number of samples.
alpha	Stability parameter (0,2].
beta	Skewness parameter [-1,1].
gamma	Scale (>0).
delta	Location.
pm	Parameterization (0 or 1).

**Value**

Numeric vector of samples.

---

RT	<i>Compute effective reproduction number Rt</i>
----	---

---

**Description**

Estimates the effective reproduction number (Rt) over time using incidence data and a generation time distribution.

**Usage**

```
RT(inc, G)
```

**Arguments**

inc	Numeric vector of incidence counts.
G	Numeric vector representing the generation time distribution.

**Value**

Numeric vector of Rt values.

---

run\_all\_estimations     *Run all EM-based estimations without Gibbs sampling (CRAN-safe)*

---

### Description

Executes multiple EM algorithms (recursive ECF, kernel ECF, weighted OLS, CDF-based) on the input data. Optionally saves mixture plots to a temporary directory.

### Usage

```
run_all_estimations(X1, bw_sj, max_iter = 200, tol = 1e-04, save_plots = FALSE)
```

### Arguments

X1	Numeric vector of data.
bw_sj	Numeric. Bandwidth value (e.g., Silverman's rule).
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance.
save_plots	Logical. Whether to save PNG plots to tempdir().

### Value

Invisibly returns a list containing fitted parameters for each EM method.

---

run\_estimations\_with\_gibbs  
                                   *Run all EM-based estimations with Gibbs sampling (CRAN-safe)*

---

### Description

Executes EM algorithms with Gibbs sampling on the input data. Optionally saves mixture plots to a temporary directory.

### Usage

```
run_estimations_with_gibbs(  
  data,  
  bw_sj,  
  max_iter = 100,  
  tol = 1e-04,  
  save_plots = FALSE  
)
```

**Arguments**

data	Numeric vector of observations.
bw_sj	Numeric. Bandwidth value (e.g., Silverman's rule).
max_iter	Integer. Maximum number of EM iterations.
tol	Numeric. Convergence tolerance.
save_plots	Logical. Whether to save PNG plots to tempdir().

**Value**

Invisibly returns a list containing fitted parameters for each EM method with Gibbs.

---

safe_integrate	<i>Safe integration wrapper with multiple fallback strategies</i>
----------------	---

---

**Description**

Performs numerical integration with progressively more conservative settings to improve robustness. Useful when standard integration may fail due to oscillatory or heavy-tailed functions.

**Usage**

```
safe_integrate(
  integrand_func,
  lower = 1e-08,
  upper = 50,
  max_attempts = 3,
  ...
)
```

**Arguments**

integrand_func	Function to integrate. Must accept a numeric vector as input and return numeric values.
lower	Numeric scalar. Lower bound of the integration interval (default = 1e-8 to avoid singularities at 0).
upper	Numeric scalar. Upper bound of the integration interval (default = 50).
max_attempts	Integer. Maximum number of fallback attempts with modified settings if the initial integration fails.
...	Additional arguments passed to integrand_func.

**Value**

Numeric scalar. The estimated value of the integral, or 0 if all attempts fail.

---

simple_em_real	<i>Simple 2-component EM using ECF initialization</i>
----------------	---

---

**Description**

Initializes EM using k-means clustering and ECF-based parameter estimation.

**Usage**

```
simple_em_real(X, max_iter = 10)
```

**Arguments**

X	Numeric vector of observations.
max_iter	Integer. Maximum number of iterations.

**Value**

List with lambda1 and estimated parameters for both components.

---

simulate_mixture	<i>Simulate mixture data from alpha-stable components</i>
------------------	---

---

**Description**

Generates synthetic data from a mixture of alpha-stable distributions using specified weights and parameters.

**Usage**

```
simulate_mixture(n, weights, params)
```

**Arguments**

n	Number of samples to generate.
weights	Vector of mixture weights.
params	List of parameter sets for each component (each with alpha, beta, gamma, delta).

**Value**

Numeric vector of simulated samples.

---

sine_exp_ralpha	<i>Sine exponential function</i>
-----------------	----------------------------------

---

**Description**

Sine exponential function

**Usage**

```
sine_exp_ralpha(r, x, alpha, beta, delta, omega)
```

**Arguments**

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric value of the integrand.

---

sine_log_weighted_exp_ralpha	<i>Sine-log-weighted exponential with <math>r^{(-\alpha)}</math> term</i>
------------------------------	---

---

**Description**

Sine-log-weighted exponential with  $r^{(-\alpha)}$  term

**Usage**

```
sine_log_weighted_exp_ralpha(r, x, alpha, beta, delta, omega)
```

**Arguments**

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric value of the integrand.

---

sine\_r\_weighted\_exp\_alpha  
*Sine-r-weighted exponential function*

---

**Description**

Sine-r-weighted exponential function

**Usage**

sine\_r\_weighted\_exp\_alpha(r, x, alpha, beta, delta, omega)

**Arguments**

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $0 < \alpha \leq 2$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric value of the integrand.

---

sine\_weighted\_exp\_alpha  
*Sine-weighted exponential with r^alpha term*

---

**Description**

Sine-weighted exponential with r^alpha term

**Usage**

sine\_weighted\_exp\_alpha(r, x, alpha, beta, delta, omega)

**Arguments**

r	Integration variable.
x	Observation.
alpha	Numeric scalar. Stability parameter of the stable distribution ( $(0 < \alpha \leq 2)$ ), controlling tail thickness.
beta	Numeric scalar. Skewness parameter of the stable distribution ( $-1 \leq \beta \leq 1$ ).
delta	Numeric scalar. Location parameter of the stable distribution.
omega	Numeric scalar. Scale parameter of the stable distribution ( $\omega > 0$ ).

**Value**

Numeric value of the integrand.

---

skew_kurtosis	<i>Calculate skewness and kurtosis</i>
---------------	--

---

**Description**

Computes the skewness and kurtosis of a numeric vector.

**Usage**

```
skew_kurtosis(x)
```

**Arguments**

x	Numeric vector of data.
---	-------------------------

**Value**

List with skewness and kurtosis.

---

stable_fit_init	<i>Initialize stable distribution parameters</i>
-----------------	--

---

**Description**

Initialize stable distribution parameters

**Usage**

```
stable_fit_init(x)
```

**Arguments**

x                    Numeric vector of data.

**Value**

A list with estimated parameters (alpha, beta, gamma, delta).

---

TableS2\_serial\_interval\_mean\_

*Example transmission pair data with mean serial interval*

---

**Description**

A data frame of infector-infectee pairs with demographic and serial interval information.

**Usage**

```
data(TableS2_serial_interval_mean_)
```

**Format**

A data frame with columns:

**infector\_id** Character. Infector ID.

**infector\_age** Numeric. Age of infector.

**infector\_sex** Character. Sex of infector.

**infector\_onsetDate** Date. Infector symptom onset.

**infector\_labConfirmDate** Date. Infector lab confirmation.

**infectee\_id** Character. Infectee ID.

**infectee\_age** Numeric. Age of infectee.

**infectee\_sex** Character. Sex of infectee.

**infectee\_onsetDate** Date. Infectee symptom onset.

**infectee\_labConfirmDate** Date. Infectee lab confirmation.

**serial\_interval\_mean\_based** Numeric. Mean-based serial interval.

**Source**

Simulated or real data for demonstration.

---

test_normality	<i>Test normality using multiple statistical tests</i>
----------------	--

---

**Description**

Applies Shapiro-Wilk, Jarque-Bera, Anderson-Darling, and Kolmogorov-Smirnov tests to assess normality of a dataset.

**Usage**

```
test_normality(x)
```

**Arguments**

x                    Numeric vector of data.

**Value**

List of test statistics and p-values.

---

unpack_params	<i>Helper function to unpack parameters</i>
---------------	---

---

**Description**

Helper function to unpack parameters

**Usage**

```
unpack_params(p)
```

**Arguments**

p                    A list containing alpha, beta, gamma, delta.

**Value**

A numeric vector of parameters.

---

validate_params	<i>Validate and clip parameters for stable distribution</i>
-----------------	---

---

**Description**

Ensures parameters are within valid bounds for stable distribution computations.

**Usage**

```
validate_params(alpha, beta, delta, r = NULL)
```

**Arguments**

alpha	Stability parameter (must be > 0 and < 2).
beta	Skewness parameter (clipped to [-1, 1]).
delta	Scale parameter (must be > 0).
r	Optional numeric vector. Must be positive if provided.

**Value**

Clipped beta value.

---

wasserstein_distance_mixture	<i>Wasserstein distance between two mixture distributions</i>
------------------------------	---

---

**Description**

Wasserstein distance between two mixture distributions

**Usage**

```
wasserstein_distance_mixture(params1, params2, size = 5000)
```

**Arguments**

params1	List of parameters for first mixture.
params2	List of parameters for second mixture.
size	Number of samples to approximate distance.

**Value**

Wasserstein distance.

# Index

- \* **datasets**
  - DONNEE\_with\_serial\_interval, 14
  - TableS2\_serial\_interval\_mean\_, 74
- aic, 5
- analyse\_stable\_distribution, 5
  
- bayesian\_mixture\_model, 6
- bic, 6
- build\_mcculloch\_interpolators, 7
  
- calculate\_log\_likelihood, 7
- CDF, 8
- clip, 8
- compare\_em\_vs\_em\_gibbs, 9
- compare\_estimators\_on\_simulations, 9
- compare\_methods\_across\_configs, 10
- compare\_methods\_with\_gibbs, 10
- compute\_model\_metrics, 11
- compute\_quantile\_ratios, 12
- compute\_serial\_interval, 12
- cosine\_exp\_alpha, 13
- cosine\_log\_weighted\_exp\_alpha, 13
  
- DONNEE\_with\_serial\_interval, 14
  
- ecf\_components, 14
- ecf\_empirical, 15
- ecf\_estimate\_all, 15
- ecf\_fn, 16
- ecf\_regression, 16
- em\_alpha\_stable, 17
- em\_estimate\_stable\_from\_cdf, 17
- em\_estimate\_stable\_from\_cdf\_with\_gibbs, 18
- em\_estimate\_stable\_kernel\_ecf, 19
- em\_estimate\_stable\_kernel\_ecf\_with\_gibbs, 19
- em\_estimate\_stable\_recursive\_ecf, 20
- em\_estimate\_stable\_recursive\_ecf\_with\_gibbs, 20
  
- em\_estimate\_stable\_weighted\_ols, 21
- em\_estimate\_stable\_weighted\_ols\_with\_gibbs, 21
- em\_estimation\_mixture, 22
- em\_fit\_alpha\_stable\_mixture, 22
- em\_stable\_mixture, 23
- empirical\_r0, 23
- ensure\_positive\_scale, 24
- est\_r0\_ml, 24
- est\_r0\_mle, 25
- estimate\_alpha\_gamma, 25
- estimate\_beta\_delta, 26
- estimate\_mixture\_params, 26
- estimate\_stable\_from\_cdf, 27
- estimate\_stable\_kernel\_ecf, 27
- estimate\_stable\_params, 28
- estimate\_stable\_r, 28
- estimate\_stable\_recursive\_ecf, 29
- estimate\_stable\_weighted\_ols, 29
- eta0, 30
- eta\_func, 30
- evaluate\_estimation\_method, 31
- evaluate\_fit, 31
- export\_analysis\_report, 32
  
- false\_position\_update, 33
- fast\_integrate, 33
- fit\_alpha\_stable\_mle, 34
- fit\_mle\_mixture, 34
- fit\_stable\_ecf, 35
  
- generate\_alpha\_stable\_mixture, 35
- generate\_mcculloch\_table, 36
- generate\_mixture\_data, 37
- generate\_synthetic\_data, 37
- gibbs\_sampler, 38
- grad\_loglik\_alpha, 38
- grad\_loglik\_beta, 39
- grad\_loglik\_delta, 39
- grad\_loglik\_omega, 40

Im, 40  
Int\_Im, 41  
Int\_Re, 41  
integrate\_cosine, 42  
integrate\_cosine\_log\_weighted, 42  
integrate\_function, 43  
integrate\_sine, 43  
integrate\_sine\_log\_weighted, 44  
integrate\_sine\_r\_weighted, 45  
integrate\_sine\_weighted, 45  
  
kde\_bandwidth\_plugin, 46  
  
L\_stable, 46  
log\_likelihood\_mixture, 47  
  
Max\_vrai, 47  
mcculloch\_lookup\_estimate, 48  
mcculloch\_quantile\_init, 48  
metropolis\_hastings, 49  
mixture\_stable\_pdf, 49  
mle\_estimate, 50  
mock\_gibbs\_sampling, 50  
mock\_lookup\_alpha\_beta, 51  
  
N\_epanechnikov, 51  
N\_gaussian, 52  
N\_uniform, 52  
negative\_log\_likelihood, 53  
normalized\_grad\_alpha, 53  
normalized\_objective\_beta, 54  
normalized\_objective\_delta, 54  
normalized\_objective\_omega, 55  
  
plot\_comparison, 56  
plot\_distributions, 56  
plot\_effective\_reproduction\_number, 57  
plot\_final\_mixture\_fit, 57  
plot\_fit\_vs\_true, 58  
plot\_fit\_vs\_true\_methods, 59  
plot\_method\_comparison, 59  
plot\_mixture, 60  
plot\_mixture\_fit, 60  
plot\_real\_mixture\_fit, 61  
plot\_results, 62  
plot\_trace, 63  
plot\_vs\_normal\_stable, 63  
  
qcv\_stat, 64  
  
r\_stable\_pdf, 64  
Re, 65  
recursive\_weight, 65  
robust\_ecf\_regression, 66  
robust\_mle\_estimate, 66  
rstable, 67  
RT, 67  
run\_all\_estimations, 68  
run\_estimations\_with\_gibbs, 68  
  
safe\_integrate, 69  
simple\_em\_real, 70  
simulate\_mixture, 70  
sine\_exp\_alpha, 71  
sine\_log\_weighted\_exp\_alpha, 71  
sine\_r\_weighted\_exp\_alpha, 72  
sine\_weighted\_exp\_alpha, 72  
skew\_kurtosis, 73  
stable\_fit\_init, 73  
  
TableS2\_serial\_interval\_mean\_, 74  
test\_normality, 75  
  
unpack\_params, 75  
  
validate\_params, 76  
  
wasserstein\_distance\_mixture, 76