Package: RHISEA (via r-universe)

May 16, 2026

Type Package

Title An R package for flexible and reproducible mixed-stock analysis in fisheries and ecology

Version 0.1.0

Maintainer Sosthene A. V. Akia `<sosthene.akia@dfo-mpo.gc.ca>`

Description RHISEA modernizes Millar's HISEA (1987,1990) methodology for reproducible mixed-stock analysis. RHISEA provides a flexible and reproducible framework for mixed-stock analysis in fisheries and ecology. Modernizes legacy workflows by integrating modern statistical classifiers (LDA, RF, MCLUST, SVM, etc.) and estimators (EM, ML, COOK). Includes simulation frameworks, bootstrap resampling, and complete analysis workflows for fisheries stock assessment and population structure inference.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports stats, utils, MASS, data.table, ggplot2, tidyr, caret, xgboost, randomForest, nnet, e1071, mclust, class, kknn

SystemRequirements Windows (for the included hisea.exe binary)

Suggests testthat (>= 3.0.0), roxygen2, knitr, rmarkdown, covr, readr, dplyr

VignetteBuilder knitr

URL `https://github.com/SostheneA/RHISEA`, `https://sosthenea.github.io/RHISEA/`

BugReports `https://github.com/SostheneA/RHISEA/issues`

Config/testthat/edition 3

LazyData true

Repository https://sosthenea.r-universe.dev

Date/Publication 2026-03-17 18:29:01 UTC

RemoteUrl https://github.com/SostheneA/RHISEA

RemoteRef HEAD

RemoteSha 7752da424d88030c7520ea3c42df5e17030d6b1b

Contents

`accel_em_algorithm` *Accelerated Expectation-Maximization (EM) Algorithm*

Description

Estimates mixing proportions (theta) using an EM algorithm with a simple acceleration step. The acceleration is attempted based on changes in theta over iterations and log-likelihood improvement. This is a simplified interpretation of HISEA's ACCEL subroutine.

Usage

```
accel_em_algorithm(
  likelihood,
  np,
  freq = NULL,
  max_iter = 100,
  tol = 1e-06,
```

```
  verbose = FALSE,
  save_theta_path = NULL
)
```

Arguments

`likelihood`	A numeric matrix (N_rows x N_populations) where ‘likelihood[i, j]‘ is $P(Data_i \| Stock_j)$ or a value proportional to it. N_rows can be N_samples (for Theta5) or N_populations (for Theta4).
`np`	Integer, the number of populations (stocks). Should match ‘ncol(likelihood)‘.
`freq`	Optional numeric vector of frequencies/weights for each "observation" (row in ‘likelihood‘). Length must match ‘nrow(likelihood)‘. Defaults to ‘rep(1, nrow(likelihood))‘.
`max_iter`	Integer, the maximum number of iterations for the EM algorithm.
`tol`	Numeric, the convergence tolerance. The algorithm stops if the maximum absolute change in any ‘theta‘ component is less than ‘tol‘.
`verbose`	Logical, if TRUE, prints iteration number and current theta estimates.
`save_theta_path`	Optional character string. If provided, the history of theta estimates at each iteration is saved to a CSV file at this path.

Value

A numeric vector of estimated stock proportions (theta), length ‘np‘.

Examples

```
lik <- matrix(runif(30), 10, 3)
lik <- lik / rowSums(lik)

theta_accel <- accel_em_algorithm(lik, np = 3)
print(theta_accel)
```

---

baseline *Baseline Data for RHISEA*

Description

A dataset containing baseline stock characteristics.

Usage

```
baseline
```

Format

A data frame (ou list) with...

Source

Your Lab / Gulf Fisheries Centre

`classify_samples` *Classify Samples Using LDF Coefficients*

Description

Assigns each sample (observation) to a population/stock based on the highest Linear Discriminant Function (LDF) score. It also computes posterior probabilities (or values proportional to them if priors are not explicitly used).

Usage

```
classify_samples(samples, coefs, freq = NULL, type = "S")
```

Arguments

`samples`	A numeric matrix of samples to be classified. Rows are observations, columns are variables.
`coefs`	A numeric matrix of LDF coefficients, as returned by ‘compute_ldf_coefficients‘. Rows are populations, columns are coefficients for variables followed by the constant term.
`freq`	Optional numeric vector of frequency weights for the samples. Defaults to 1 for each sample if NULL. (Used mainly for HISEA compatibility).
`type`	Character, indicates the type of run (e.g., "S" for simulation). This parameter is from HISEA Fortran and affects a condition for processing, though its impact here is minor if ‘freq‘ are positive. Default "S".

Value

A list containing:

`class`	An integer vector of predicted class labels (1 to NP) for each sample.
`likelihood`	A numeric matrix where rows are samples and columns are populations. Contains posterior probabilities of class membership (assuming equal priors, rows sum to 1).

Examples

```
# 1. Prepare coefficients
s1 <- matrix(rnorm(20, mean = -10), ncol = 2)
s2 <- matrix(rnorm(20, mean = -15), ncol = 2)
coefs <- compute_ldf_coefficients(list(StockA = s1, StockB = s2))

# 2. Prepare unknown samples (mixture)
unknown_samples <- matrix(rnorm(10, mean = -12), ncol = 2)
colnames(unknown_samples) <- c("Var1", "Var2")

# 3. Classify
results <- classify_samples(samples = unknown_samples, coefs = coefs)

# Check predictions and probabilities
print(results$class)
print(results$likelihood)
```

`compute_cook_estimators` *Compute Cook Estimators for Stock Proportions*

Description

Computes three types of estimators based on classification results: 1. Raw proportions: Simple proportion of samples classified to each stock. 2. Cook's corrected estimator: Raw proportions corrected by the inverse of the misclassification matrix (Phi_inv). This can result in negative estimates. 3. Cook's constrained estimator: An iterative adjustment of Cook's corrected estimator to ensure proportions are non-negative and sum to 1.

Usage

```
compute_cook_estimators(class_predictions, PHIinv, np)
```

Arguments

`class_predictions`	A numeric vector of predicted class (stock) labels for the mixed sample. Values should be integers from 1 to 'np'.
`PHIinv`	The inverse of the misclassification matrix (Phi). Dimensions should be np x np.
`np`	Integer, the number of populations (stocks).

Value

A list containing three numeric vectors:

`raw`	Raw proportions of classification.
`cook`	Cook's corrected estimates.
`cook_constrained`	Cook's constrained estimates (non-negative, sum to 1).

Examples

```
# 2 stocks, 10 samples classified
preds <- c(1, 1, 1, 2, 2, 2, 1, 1, 1, 1) # Mostly stock 1
# Dummy Phi matrix (80% accuracy)
phi <- matrix(c(0.8, 0.2, 0.2, 0.8), 2, 2)
phi_inv <- solve(phi)

cooks <- compute_cook_estimators(preds, phi_inv, np = 2)
print(cooks$cook_constrained)
```

`compute_ldf_coefficients` *Compute Linear Discriminant Function (LDF) Coefficients*

Description

Computes the coefficients for Fisher's Linear Discriminant Function for each group (population/stock). This implementation assumes a common (pooled) covariance matrix across all groups, a standard assumption for LDA. The LDF is of the form: LDF_j(x) = C_j0 + C_j1*x1 + ... + C_jp*xp.

Usage

```
compute_ldf_coefficients(baseline)
```

Arguments

`baseline`	A list of numeric matrices, where each matrix represents the baseline (training) data for one population/stock. Each matrix should have observations as rows and variables as columns.

Value

A numeric matrix of LDF coefficients. Rows correspond to populations, and columns correspond to variables followed by the constant term. The dimensions are (np x (nv + 1)).

Examples

```
# Create dummy baseline data for 2 stocks with 2 variables (isotopes)
# Stock A
s1 <- matrix(c(-10, 2, -11, 2.1, -10.5, 2.2), ncol = 2)
colnames(s1) <- c("d13c", "d18o")
# Stock B
s2 <- matrix(c(-15, 5, -16, 5.1, -15.5, 5.2), ncol = 2)
colnames(s2) <- c("d13c", "d18o")

baseline_list <- list(StockA = s1, StockB = s2)

# Compute coefficients
```

```
ldf_coeffs <- compute_ldf_coefficients(baseline_list)
print(ldf_coeffs)
```

`create_hisea_summary_report`

Create HISEA Summary Report

Description

Generates a comprehensive summary from HISEA estimation results, including means, standard deviations, MSE (for simulations), and covariance/correlation for ML estimates (often for bootstrap).

Usage

```
create_hisea_summary_report(
  all_estimates_array,
  actual_proportions = NULL,
  run_type
)
```

Arguments

`all_estimates_array`	An array of dimension (nsamps, np, n_estimators) containing all estimation results. The 3rd dimension should have estimator names.
`actual_proportions`	Optional numeric vector of true stock proportions. Required if MSE is to be calculated (typically for "SIMULATION" run_type).
`run_type`	Character string, e.g., "SIMULATION", "ANALYSIS", "BOOTSTRAP". Used to decide if actual_proportions are expected for MSE.

Value

A list containing the summary statistics.

Examples

```
# 1. Create a dummy estimation array (10 samples, 2 stocks, 3 estimators)
nsamps <- 10; np <- 2; nest <- 3
est_array <- array(runif(nsamps * np * nest), dim = c(nsamps, np, nest))
dimnames(est_array) <- list(
  NULL,
  c("StockA", "StockB"),
  c("RAW", "COOK", "ML")
)
# Ensure proportions sum to 1
```

```
for(i in 1:nsamps) for(k in 1:nest) est_array[i,,k] <- est_array[i,,k]/sum(est_array[i,,k])

# 2. Generate report
summary_rep <- create_hisea_summary_report(
  all_estimates_array = est_array,
  actual_proportions = c(0.6, 0.4),
  run_type = "SIMULATION"
)
print(summary_rep$mean_estimates)
```

`em_algorithm` *Standard Expectation-Maximization (EM) Algorithm for Stock Proportions*

Description

Estimates mixing proportions (theta) from a likelihood matrix using the EM algorithm.

Usage

```
em_algorithm(
  likelihood,
  np,
  freq = NULL,
  max_iter = 100,
  tol = 1e-06,
  verbose = FALSE,
  save_theta_path = NULL
)
```

Arguments

`likelihood`	A numeric matrix (N_rows x N_populations) where 'likelihood[i, j]' is $P(Data_i \| Stock_j)$ or a value proportional to it. N_rows can be N_samples (for Theta5) or N_populations (for Theta4).
`np`	Integer, the number of populations (stocks). Should match 'ncol(likelihood)'.
`freq`	Optional numeric vector of frequencies/weights for each "observation" (row in 'likelihood'). Length must match 'nrow(likelihood)'. Defaults to 'rep(1, nrow(likelihood))'.
`max_iter`	Integer, the maximum number of iterations for the EM algorithm.
`tol`	Numeric, the convergence tolerance. The algorithm stops if the maximum absolute change in any 'theta' component is less than 'tol'.
`verbose`	Logical, if TRUE, prints iteration number and current theta estimates.
`save_theta_path`	Optional character string. If provided, the history of theta estimates at each iteration is saved to a CSV file at this path.

Value

A numeric vector of estimated stock proportions (theta), length ‘np‘.

Examples

```
# Dummy likelihood matrix (10 samples, 3 stocks)
lik <- matrix(runif(30), 10, 3)
lik <- lik / rowSums(lik) # Normalize rows

theta_em <- em_algorithm(lik, np = 3, max_iter = 20)
print(theta_em)
```

`estimate_millar_theta4`

Estimate Millar Theta4 Parameter

Description

This function estimates the theta4 parameter using the Millar method.

Usage

```
estimate_millar_theta4(
  class_predictions_mixed_sample,
  PHI_matrix,
  np,
  use_accelerated_em = TRUE,
  ...
)
```

Arguments

`class_predictions_mixed_sample`	Numeric vector of predicted classes for the mixed sample
`PHI_matrix`	Confusion matrix used for correction
`np`	Integer number of populations
`use_accelerated_em`	Logical; whether to use the accelerated EM version
`...`	Additional arguments passed to internal functions

Value

Numeric value of the estimated theta4 parameter

Examples

```
preds <- c(1, 1, 2, 1, 2)
phi_mat <- matrix(c(0.9, 0.1, 0.1, 0.9), 2, 2)

theta4 <- estimate_millar_theta4(preds, phi_mat, np = 2)
print(theta4)
```

`estimate_ml_theta5` *Estimate Stock Composition Using Direct Maximum Likelihood (Theta5)*

Description

Applies an EM algorithm to obtain the direct Maximum Likelihood Estimate (MLE) of stock proportions (Theta_5 in HISEA). This uses the likelihood of each individual mixed fishery sample fish belonging to each stock. The likelihood maximized is:

Usage

```
estimate_ml_theta5(
  individual_likelihoods,
  np,
  freq = NULL,
  use_accelerated_em = TRUE,
  ...
)
```

Arguments

`individual_likelihoods`	A numeric matrix (N_mixed_samples x NP) where ‘individual_likelihoods[i,j]‘ is \$P(Data_i | Stock_j)\$, the likelihood of observing the data for mixed sample fish ‘i‘ given it came from stock ‘j‘.
`np`	Integer, the number of populations (stocks).
`freq`	Optional numeric vector of frequencies/weights for each sample (row) in ‘individual_likelihoods‘. Defaults to ‘rep(1, nrow(individual_likelihoods))‘.
`use_accelerated_em`	Logical, if TRUE, uses ‘accel_em_algorithm‘, otherwise ‘em_algorithm‘. Default TRUE.
`...`	Additional arguments passed to the chosen EM algorithm (e.g., ‘max_iter‘, ‘tol‘, ‘verbose‘, ‘save_theta_path‘).

Value

A numeric vector of estimated stock proportions (Theta5).

Examples

```
# Direct ML estimation from individual likelihoods
lik <- matrix(runif(50), 25, 2)
lik <- lik / rowSums(lik)

theta5 <- estimate_ml_theta5(lik, np = 2)
print(theta5)
```

`get_phi` *Get Phi Matrix*

Description

Calculate the phi matrix for stock composition analysis.

Usage

```
get_phi(predicted_class, true_labels, np)
```

Arguments

`predicted_class`	Integer vector of predictions
`true_labels`	Factor of true labels
`np`	Number of populations (levels)

Value

Matrix of phi values

Examples

```
# Simulated classification results
# 4 individuals: 2 from Stock 1, 2 from Stock 2
true_labels <- c(1, 1, 2, 2)
predicted_class <- c(1, 2, 2, 2) # One error: individual 2 misclassified

# Compute Phi matrix (2 stocks)
phi_matrix <- get_phi(
  predicted_class = predicted_class,
  true_labels = true_labels,
  np = 2
)

# The result shows the classification accuracy per stock
print(phi_matrix)
```

`mixture` *Mixture dataset for RHISEA examples*

Description

A synthetic dataset containing isotope values for unknown individuals.

Usage

```
data(mixture)
```

Format

A data frame with X rows and 2 variables:

d13c_ukn	Carbon isotope ratio for unknown samples
d18o_ukn	Oxygen isotope ratio for unknown samples

`ordvec` *Generate Ordered Random Values (Order Statistics based)*

Description

Produces an ordered vector of ‘num‘ random values, scaled to be approximately between 0 and ‘upper‘. This function can be used as a basis for certain types of random sampling schemes. The logic is similar to generating order statistics from an exponential distribution.

Usage

```
ordvec(num, upper)
```

Arguments

`num`	Integer, the number of ordered values to generate.
`upper`	Numeric, the maximum value for the scaled output. The generated values will be in the approximate range (0, ‘upper‘].

Value

A numeric vector of length ‘num‘, containing ordered values scaled approximately between 0 and ‘upper‘.

Examples

```
ordvec(5, 100) # Generates 5 ordered numbers, max value around 100
ordvec(10, 10) # Generates 10 ordered numbers, max value around 10
```

`plot_hisea_theta` *Plot HISEA theta estimates with error bars utils::globalVariables(c("Stock", "Mean", "Estimator", "SD", "ActualValue"))*

Description

Plot HISEA theta estimates with error bars utils::globalVariables(c("Stock", "Mean", "Estimator", "SD", "ActualValue"))

Usage

```
plot_hisea_theta(summary_report, stock_names = NULL, actual_proportions = NULL)
```

Arguments

`summary_report`	Output from `create_hisea_summary_report`.
`stock_names`	Optional character vector for stock labels.
`actual_proportions`	Optional numeric vector of true proportions.

Value

A ggplot object.

Examples

```
# 1. Generate a summary report first
nsamps <- 5; np <- 2; nest <- 2
est_array <- array(runif(nsamps * np * nest), dim = c(nsamps, np, nest))
dimnames(est_array) <- list(NULL, c("Pop1", "Pop2"), c("RAW", "ML"))
rep <- create_hisea_summary_report(est_array, run_type = "ANALYSIS")

# 2. Plot results
if (requireNamespace("ggplot2", quietly = TRUE)) {
  p <- plot_hisea_theta(rep, actual_proportions = c(0.5, 0.5))
  print(p)
}
```

`print_hisea_summary` *Print HISEA Summary Report to Console and/or File*

Description

Print HISEA Summary Report to Console and/or File

Usage

```
print_hisea_summary(
  summary_report,
  nsamps,
  actual_proportions,
  means_vars = NULL,
  sds_vars = NULL,
  standard_sizes = NULL,
  stocks_names,
  output_file = "hisea_summary_report.txt",
  header = NULL,
  run_type = "SIMULATION",
  nv_used = NULL,
  seed_val = NULL,
  nmix_simulated = NULL
)
```

Arguments

`summary_report`	Output from `create_hisea_summary_report`.
`nsamps`	Number of simulation/bootstrap replicates.
`actual_proportions`	True composition vector.
`means_vars`	Matrix of mean variable values.
`sds_vars`	Matrix of SD of variable values.
`standard_sizes`	Vector of baseline sample sizes.
`stocks_names`	Character vector of stock names.
`output_file`	Path to the output text file.
`header`	Optional custom header line.
`run_type`	Character string of run type.
`nv_used`	Integer, number of variables.
`seed_val`	Integer, seed value.
`nmix_simulated`	Integer, size of simulated mixture.

Value

'data.frame' summary statistics (invisibly)

Examples

```
# 1. Prepare dummy summary data
nsamps <- 10; np <- 2
est_array <- array(runif(nsamps * np * 2), dim = c(nsamps, np, 2))
dimnames(est_array) <- list(NULL, c("Stock1", "Stock2"), c("RAW", "ML"))
rep <- create_hisea_summary_report(est_array, run_type = "SIMULATION")

# 2. Print to console (using a temp file to avoid clutter)
print_hisea_summary(
  summary_report = rep,
  nsamps = 10,
  actual_proportions = c(0.5, 0.5),
  stocks_names = c("Stock1", "Stock2"),
  output_file = tempfile(fileext = ".txt"),
  run_type = "SIMULATION"
)
```

`read_baseline` *Read HISEA Baseline File*

Description

Reads a HISEA-formatted baseline (standard) data file. The file is expected to contain data for multiple stocks/populations, with data for each stock separated by a delimiter string (e.g., "NEXT STOCK"). Blank lines and lines starting with '#' (comments) are ignored.

Usage

```
read_baseline(filepath, nv)
```

Arguments

`filepath`	Character string, the path to the baseline '.std' file.
`nv`	Integer, the number of variables (columns) expected for each observation in the baseline data.

Value

A list of numeric matrices. Each matrix in the list corresponds to a stock/population, with rows being observations and columns being variables. Returns an empty list if no valid data is found or errors occur during parsing.

Examples

```
# Create a dummy HISEA standard file
tmp_std <- tempfile(fileext = ".std")
writeLines(c(
  "# Stock A",
  "1.2  3.4",
  "5.6  7.8",
  "NEXT STOCK",
  "# Stock B",
  "9.0  1.1"
), tmp_std)

# Read the baseline (2 variables expected)
baseline_data <- read_baseline(tmp_std, nv = 2)

# Check the results
print(baseline_data)
length(baseline_data) # Should be 2
```

`read_mixture` *Read HISEA Mixture File*

Description

Reads a HISEA-formatted mixture data file. It attempts to extract lines containing only numeric data (typical for HISEA mixture files) and converts them into a numeric matrix.

Usage

```
read_mixture(filepath = "hisea.mix", nv)
```

Arguments

`filepath`	Character string, the path to the mixture '.mix' file.
`nv`	Integer, the number of variables (columns) expected for each observation in the mixture data.

Value

A numeric matrix where rows are observations and columns are variables. Returns an empty matrix (with 'nv' columns) if no valid data is found or errors occur.

Examples

```
# Create a dummy HISEA mixture file
tmp_mix <- tempfile(fileext = ".mix")
writeLines(c(
  "1.1  2.2",
  "3.3  4.4",
  "5.5  6.6"
), tmp_mix)

# Read the mixture (2 variables expected)
mixture_matrix <- read_mixture(tmp_mix, nv = 2)

# Check the results
print(mixture_matrix)
nrow(mixture_matrix) # Should be 3
```

`run_hisea_all` *Run HISEA Mixed Stock Analysis*

Description

This is the main wrapper function and core set of utilities for running the HISEA mixed-stock analysis framework, allowing simulation, analysis, or bootstrap estimation of stock composition from mixture samples.

Supported operation modes: - **SIMULATION**: Simulate mixtures based on known proportions and evaluate performance of classification and estimators. - **ANALYSIS**: Apply trained classifier to real mixture data to estimate stock proportions. - **BOOTSTRAP**: Resample real mixture to evaluate variability of estimates.

Supported classifiers: LDA, QDA, Random Forest, SVM, k-NN, ANN, XGBoost, Naive Bayes, Mclust, MLR. Supported estimators: RAW, Cook, Constrained Cook, EM (Millar), Maximum Likelihood.

Includes integrated 10-fold cross-validation and model quality evaluation (accuracy, kappa, F1, etc.).

Usage

```
run_hisea_all(
  type = "ANALYSIS",
  np,
  nv,
  seed_val = 123456,
  var_cols_std = NULL,
  var_cols_mix = NULL,
  stock_col = NULL,
  nsamps = 1000,
  Nmix = 100,
```

```
  actual = NULL,
  baseline_input = NULL,
  mix_input = NULL,
  export_csv = FALSE,
  output_dir = ".",
  verbose = FALSE,
  method_class = "LDA",
  stocks_names = NULL,
  resample_baseline = FALSE,
  resampled_baseline_sizes = NULL,
  phi_method = c("standard", "cv"),
  mclust_model_names = NULL,
  mclust_perform_cv = TRUE,
  ...
)
```

Arguments

`type`	Character. "SIMULATION", "ANALYSIS" or "BOOTSTRAP".
`np`	Integer. Number of populations (stocks).
`nv`	Integer. Number of variables.
`seed_val`	Integer. Random seed for reproducibility.
`var_cols_std`	Character vector of column names for baseline variables.
`var_cols_mix`	Character vector of column names for mixture variables.
`stock_col`	Character name of stock column in baseline data.
`nsamps`	Integer. Number of replicates.
`Nmix`	Integer. Sample size of the simulated mixture (for SIMULATION only).
`actual`	Numeric vector. True proportions used in simulation.
`baseline_input`	Data frame or file path for baseline data.
`mix_input`	Data frame or file path for mixture data.
`export_csv`	Logical. Whether to export summary and confusion matrix to CSV.
`output_dir`	Character. Output directory.
`verbose`	Logical. Print progress messages.
`method_class`	Character. Classification method (e.g., "LDA", "RF", "SVM", etc.).
`stocks_names`	Character vector. Optional vector of stock names.
`resample_baseline`	Logical. Resample the baseline for each replicate.
`resampled_baseline_sizes`	Integer vector. Sizes of resamples per stock.
`phi_method`	Character. "standard" or "cv" (cross-validation-based confusion matrix).
`mclust_model_names`	Character vector. Models to test with Mclust.
`mclust_perform_cv`	Logical. Whether to cross-validate Mclust.
`...`	Additional arguments passed to the underlying classification models (e.g., ntree for Random Forest, cost for SVM).

Value

A list of length 8 containing the statistical summary of the estimation (the same as the ‘estimation_summary‘ element in the saved file):

mean_estimates Matrix [np x nsamps] of mean estimated proportions.

sd_estimates Standard deviations of the estimates.

mse_estimates Mean Squared Error (if applicable).

var_emp Empirical variance of the estimates.

covar_ml Maximum Likelihood covariance matrix.

cor_ml Correlation matrix.

covar_inv_ml Inverse of the covariance matrix.

det_covar_ml Determinant of the covariance matrix (checks for singularity).

Saved Results Structure

The function automatically saves a ‘.rda‘ file in ‘output_dir‘ containing a master list named ‘out‘. This list includes:

estimation_summary The list of 8 statistical metrics described above.

classification_model The trained classifier object (e.g., LDA, RF).

baseline_classification_quality Accuracy, Kappa, and F1 scores.

phi_matrix The confusion matrix used for bias correction.

mixture_classification_details Predicted classes and posterior probabilities.

Examples

```
data(baseline)
data(mixture)

res <- run_hisea_all(
  baseline_input = baseline,
  mix_input      = mixture,
  stock_col      = "population",
  var_cols_std   = c("d13c", "d18o"),
  var_cols_mix   = c("d13c_ukn", "d18o_ukn"),
  output_dir     = tempdir(),
  np = 2, nv = 2, nsamps = 5, Nmix = 50, method_class = "LDA"
)
print(res$mean_estimates)
```

`run_hisea_estimates` *Run HISEA Estimations Only*

Description

Run HISEA Estimations Only

Usage

```
run_hisea_estimates(
  pseudo_classes,
  likelihoods,
  phi_matrix,
  np,
  type = "ANALYSIS",
  nsamps = 1000,
  stocks_names = NULL,
  export_csv = FALSE,
  output_dir = ".",
  verbose = FALSE
)
```

Arguments

`pseudo_classes`	Vector of predicted classes (integers)
`likelihoods`	Matrix of prediction probabilities
`phi_matrix`	Confusion matrix (Phi)
`np`	Number of populations
`type`	"ANALYSIS", "SIMULATION" or "BOOTSTRAP"
`nsamps`	Number of samples (default: 1000)
`stocks_names`	Names of stocks/populations
`export_csv`	Export results to CSV
`output_dir`	Output directory
`verbose`	Print progress messages

Value

List containing estimates and metrics

Examples

```
# Advanced estimation using probability matrices
dummy_probs <- matrix(runif(150), ncol = 3)
dummy_probs <- dummy_probs / rowSums(dummy_probs)
dummy_classes <- max.col(dummy_probs)

results <- run_hisea_estimates(
  pseudo_classes = dummy_classes,
  likelihoods = dummy_probs,
  phi_matrix = diag(3),
  np = 3, type = "ANALYSIS"
)
print(results$mean_estimates)
```

`simulate_mixture` *Simulate a Mixed Fishery Sample from Known Stock Proportions*

Description

Creates a synthetic fishery mixture by sampling individuals from different baseline stocks according to specified true proportions. Sampling from each stock is done with replacement.

Usage

```
simulate_mixture(baseline_data_list, actual_proportions, N_mixture_size)
```

Arguments

`baseline_data_list`	A list of numeric matrices. Each matrix represents the baseline (standard) data for one stock/population, with observations as rows and variables as columns.
`actual_proportions`	A numeric vector of true proportions for each stock in the mixture. Must sum to 1 and have the same length as ‘baseline_data_list‘.
`N_mixture_size`	Integer, the total number of individuals to draw for the synthetic mixture.

Value

A numeric matrix representing the simulated mixture sample. Rows are individuals, and columns are variables. Returns an error if the mixture cannot be formed (e.g., due to empty baseline stocks needed for sampling).

Examples

```
# 1. Prepare dummy baseline data for 2 stocks
stock1 <- matrix(rnorm(40, mean = 0, sd = 1), ncol = 2)
stock2 <- matrix(rnorm(60, mean = 2, sd = 1), ncol = 2)
baseline_list <- list(StockA = stock1, StockB = stock2)

# 2. Set true proportions and mixture size
true_props <- c(0.6, 0.4)
n_mix <- 100

# 3. Generate the synthetic mixture
simulated_sample <- simulate_mixture(baseline_list, true_props, n_mix)

# 4. Verify output
head(simulated_sample)
nrow(simulated_sample) # Should be 100
```

Index