Package 'topolow'

May 8, 2026

Title Force-Directed Euclidean Embedding of Dissimilarity Data

Version 2.0.1

Maintainer Omid Arhami <omid.arhami@uga.edu>

Description A robust implementation of Topolow algorithm. It embeds objects into a low-dimensional Euclidean space from a matrix of pairwise dissimilarities, even when the data do not satisfy metric or Euclidean axioms. The package is particularly well-suited for sparse, incomplete, and censored (thresholded) datasets such as antigenic relationships. The core is a physics-inspired, gradient-free optimization framework that models objects as particles in a physical system, where observed dissimilarities define spring rest lengths and unobserved pairs exert repulsive forces. The package also provides functions specific to antigenic mapping to transform cross-reactivity and binding affinity measurements into accurate spatial representations in a phenotype space.
Key features include:
* Robust Embedding from Sparse Data: Effectively creates complete and consistent maps (in optimal dimensions) even with high proportions of missing data (e.g., >95%).
* Physics-Inspired Optimization: Models objects (e.g., antigens, landmarks) as particles connected by springs (for measured dissimilarities) and subject to repulsive forces (for missing dissimilarities), and simulates the physical system using laws of mechanics, reducing the need for complex gradient computations.
* Automatic Dimensionality Detection: Employs a likelihood-based approach to determine the optimal number of dimensions for the embedding/map, avoiding distortions common in methods with fixed low dimensions.
* Noise and Bias Reduction: Naturally mitigates experimental noise and bias through its network-based, error-dampening mechanism.
* Antigenic Velocity Calculation (for antigenic data): Introduces and quantifies ``antigenic velocity," a vector that describes the rate and direction of antigenic drift for each pathogen isolate. This can help identify cluster transitions and potential lineage replacements.
* Broad Applicability: Analyzes data from various objects that their dissimilarity may be of interest, ranging from complex biological measurements such as continuous and relational phenotypes, antibody-antigen interactions, and protein folding to abstract concepts, such as customer perception of different brands.
Methods are described in the context of bioinformatics applications in Arhami and Rohani (2025a) <doi:10.1093/bioinformatics/btaf372>, and mathematical proofs and Euclidean embedding details are in Arhami and Rohani (2025b) <doi:10.48550/arXiv.2508.01733>.

License BSD_3_clause + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports future, lifecycle, ggplot2 (>= 3.4.0), dplyr (>= 1.1.0), data.table (>= 1.14.0), reshape2 (>= 1.4.4), stats, utils, parallel (>= 4.1.0), filelock, lhs, rlang,

Suggests coda (>= 0.19-4), Rtsne, ape, Racmacs (>= 1.1.2), vegan, umap, igraph, rgl (>= 1.0.0), scales, ggrepel, plotly (>= 4.10.0), gridExtra, covr, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/omid-arhami/topolow

BugReports https://github.com/omid-arhami/topolow/issues

LazyData true

Depends R (>= 4.1.0)

VignetteBuilder knitr

NeedsCompilation no

Author Omid Arhami [aut, cre, cph] (ORCID: <https://orcid.org/0009-0005-2681-6598>)

Repository CRAN

Date/Publication 2025-08-31 00:20:02 UTC

Contents

`analyze_network_structure` *Analyze Network Structure*

Description

Analyzes the connectivity of a dissimilarity matrix, returning node degrees and overall completeness.

Usage

```
analyze_network_structure(dissimilarity_matrix)
```

Arguments

`dissimilarity_matrix`
: Square symmetric matrix of dissimilarities.

Value

A list containing the network analysis results:

`adjacency`
: A logical `matrix` where `TRUE` indicates a measured dissimilarity.

`connectivity`
: A `data.frame` with node-level metrics, including the `completeness` (degree) for each point.

`summary`
: A list of overall network statistics, including `n_points`, `n_measurements`, and total `completeness`.

Examples

```
# Create a sample dissimilarity matrix
dist_mat <- matrix(runif(25), 5, 5)
rownames(dist_mat) <- colnames(dist_mat) <- paste0("Point", 1:5)
dist_mat[lower.tri(dist_mat)] <- t(dist_mat)[lower.tri(dist_mat)]
diag(dist_mat) <- 0
dist_mat[1, 3] <- NA; dist_mat[3, 1] <- NA

# Analyze the network structure
metrics <- analyze_network_structure(dist_mat)
print(metrics$summary$completeness)
```

`calculate_diagnostics` *Calculate MCMC-style Diagnostics for Sampling Chains*

Description

Calculates standard MCMC-style convergence diagnostics for multiple chains from an optimization or sampling run. It computes the R-hat (potential scale reduction factor) and effective sample size (ESS) to help assess if the chains have converged to a stable distribution.

Usage

```
calculate_diagnostics(chain_files, mutual_size = 500)
```

Arguments

`chain_files`
: Character vector. Paths to CSV files, where each file represents a chain of samples.

`mutual_size`
: Integer. Number of samples to use from the end of each chain for calculations.

Value

A list object of class `topolow_diagnostics` containing convergence diagnostics for the MCMC chains.

`rhat`	A numeric vector of the R-hat (potential scale reduction factor) statistic for each parameter. Values close to 1 indicate convergence.
`ess`	A numeric vector of the effective sample size for each parameter.
`chains`	A list of data frames, where each data frame is a cleaned and trimmed MCMC chain.
`param_names`	A character vector of the parameter names being analyzed.
`mutual_size`	The integer number of samples used from the end of each chain for calculations.

Examples

```
# This example demonstrates how to use the function with temporary files.
# Create dummy chain files in a temporary directory
temp_dir <- tempdir()
chain_files <- character(3)
par_names <- c("log_N", "log_k0", "log_cooling_rate", "log_c_repulsion")
sample_data <- data.frame(
  log_N = rnorm(100), log_k0 = rnorm(100),
  log_cooling_rate = rnorm(100), log_c_repulsion = rnorm(100),
  NLL = runif(100), Holdout_MAE = runif(100)
)
for (i in 1:3) {
  chain_files[i] <- file.path(temp_dir, paste0("chain", i, ".csv"))
  write.csv(sample_data, chain_files[i], row.names = FALSE)
}

# Calculate diagnostics
diag_results <- calculate_diagnostics(chain_files, mutual_size = 50)
print(diag_results)

# Clean up the temporary files and directory
unlink(chain_files)
unlink(temp_dir, recursive = TRUE)
```

`calculate_prediction_interval` *Calculate Prediction Interval for Dissimilarity Estimates*

Description

Computes prediction intervals for the estimated dissimilarities based on residual variation between true and predicted values.

Usage

```
calculate_prediction_interval(
  dissimilarity_matrix,
  predicted_dissimilarity_matrix,
  confidence_level = 0.95
)
```

Arguments

`dissimilarity_matrix`	Matrix of true dissimilarities.
`predicted_dissimilarity_matrix`	Matrix of predicted dissimilarities.
`confidence_level`	The confidence level for the interval (default: 0.95).

Value

A single numeric value representing the margin of error for the prediction interval.

`calculate_weighted_marginals` *Calculate Weighted Marginal Distributions*

Description

Calculates the marginal probability distribution for each model parameter. The distributions are weighted by the likelihood of each sample, making this useful for identifying the most probable parameter values from a set of Monte Carlo samples.

Usage

```
calculate_weighted_marginals(samples)
```

Arguments

`samples`	A data frame containing parameter samples (e.g., `log_N`, `log_k0`) and a negative log-likelihood column named `NLL`.

Details

This function uses the `weighted_kde` helper to perform kernel density estimation for each parameter, with weights derived from the normalized likelihoods of the samples.

Value

A named list where each element is a density object (a list with x and y components) corresponding to a model parameter.

x	Vector of parameter values
y	Vector of density estimates

`check_gaussian_convergence` *Model Diagnostics and Convergence Testing Check Multivariate Gaussian Convergence*

Description

Assesses the convergence of multivariate samples by monitoring the stability of the mean vector and covariance matrix over a sliding window. This is useful for checking if a set of parameter samples has stabilized.

Usage

```
check_gaussian_convergence(data, window_size = 300, tolerance = 0.01)
```

Arguments

data	Matrix or Data Frame. A matrix of samples where columns are parameters.
window_size	Integer. The size of the sliding window used to compute statistics.
tolerance	Numeric. The convergence threshold for the relative change in the mean and covariance.

Value

An object of class `topolow_convergence` containing diagnostics about the convergence of the multivariate samples. This list includes logical flags for convergence (`converged`, `mean_converged`, `cov_converged`) and the history of the mean and covariance changes.

Examples

```
# Create sample data for the example
chain_data <- as.data.frame(matrix(rnorm(500 * 4), ncol = 4))
colnames(chain_data) <- c("param1", "param2", "param3", "param4")

# Run the convergence check
conv_results <- check_gaussian_convergence(chain_data)
print(conv_results)

# The plot method for this object can be used to create convergence plots.
# plot(conv_results)
```

clean_data *Clean Data by Removing MAD-based Outliers*

Description

Removes outliers from numeric data using the Median Absolute Deviation (MAD) method. Outliers are replaced with NA values.

Usage

```
clean_data(x, k = 3, take_log = FALSE)
```

Arguments

x	Numeric vector to clean.
k	Numeric threshold for outlier detection (default: 3).
take_log	Logical. Deprecated parameter. Log transformation should be done before calling this function.

Value

A numeric vector of the same length as x, where detected outliers have been replaced with NA.

See Also

detect_outliers_mad for the underlying outlier detection.

Examples

```
# Clean parameter values
params <- c(0.01, 0.012, 0.011, 0.1, 0.009, 0.011, 0.15)
clean_params <- clean_data(params)
```

color_palettes *Color Palettes*

Description

Predefined color palettes optimized for visualization.

Usage

```
c25
```

Format

An object of class character of length 20.

coordinates_to_matrix *Convert Coordinates to a Distance Matrix*

Description

Calculates pairwise Euclidean distances between points in a coordinate space.

Usage

```
coordinates_to_matrix(positions)
```

Arguments

positions	Matrix or Data Frame of coordinates where rows are points and columns are dimensions.

Value

A symmetric matrix of pairwise Euclidean distances between points.

create_cv_folds *Create Cross-Validation Folds for a Dissimilarity Matrix*

Description

Creates k-fold cross-validation splits from a dissimilarity matrix while maintaining symmetry. Each fold in the output consists of a training matrix (with some values masked as NA) and a corresponding ground truth matrix for validation.

Usage

```
create_cv_folds(
  dissimilarity_matrix,
  ground_truth_matrix = NULL,
  n_folds = 10,
  random_seed = NULL
)
```

Arguments

dissimilarity_matrix	The input dissimilarity matrix, which may contain noise.
ground_truth_matrix	An optional, noise-free dissimilarity matrix to be used as the ground truth for evaluation. If NULL, the input dissimilarity_matrix is used as the truth.
n_folds	The integer number of folds to create.
random_seed	An optional integer to set the random seed for reproducibility.

Value

A list of length `n_folds`. Each element of the list is itself a list containing two matrices: `truth` (the ground truth for that fold) and `train` (the training matrix with `NA` values for validation).

Note

This function has breaking changes from previous versions:

- Parameter `truth_matrix` renamed to `dissimilarity_matrix`
- Parameter `no_noise_truth` renamed to `ground_truth_matrix`
- Return structure now uses named elements (`$truth`, `$train`)

Examples

```
# Create a sample dissimilarity matrix
d_mat <- matrix(runif(100), 10, 10)
diag(d_mat) <- 0

# Create 5-fold cross-validation splits
folds <- create_cv_folds(d_mat, n_folds = 5, random_seed = 123)
```

`create_diagnostic_plots` *Create Diagnostic Plots for Multiple Sampling Chains*

Description

Creates trace and density plots for multiple sampling or optimization chains to help assess convergence and mixing. It displays parameter trajectories and their distributions across all chains.

Usage

```
create_diagnostic_plots(
  chain_files,
  mutual_size = 2000,
  output_file = "diagnostic_plots.png",
  output_dir,
  save_plot = FALSE,
  width = 3000,
  height = 3000,
  res = 300
)
```

Arguments

chain_files	A character vector of paths to CSV files, where each file contains data for one chain.
mutual_size	Integer. The number of samples to use from the end of each chain for plotting.
output_file	Character. The path for saving the plot. Required if save_plot is TRUE.
output_dir	Character. The directory for saving output files. Required if save_plot is TRUE.
save_plot	Logical. If TRUE, saves the plot to a file. Default: FALSE.
width, height, res	Numeric. The dimensions and resolution for the saved plot.

Value

A ggplot object of the combined plots.

Examples

```
# This example uses sample data files that would be included with the package.
chain_files <- c(
  system.file("extdata", "diag_chain1.csv", package = "topolow"),
  system.file("extdata", "diag_chain2.csv", package = "topolow"),
  system.file("extdata", "diag_chain3.csv", package = "topolow")
)

# Only run the example if the files are found
if (all(nzchar(chain_files))) {
  # Create diagnostic plot without saving to a file
  create_diagnostic_plots(chain_files, mutual_size = 50, save_plot = FALSE)
}
```

---

denv_data *Dengue Virus (DENV) Titer Data*

Description

A dataset containing neutralization titer data for Dengue virus. This data can be used to create antigenic maps and explore the antigenic relationships between different DENV strains.

Usage

```
denv_data
```

Format

A data frame with the following columns:

virus_strain Character, the name of the virus strain.

serum_strain Character, the name of the serum strain.

titer Character, the neutralization titer value. May include values like '<10' or '>1280'.

virusYear Numeric, the year the virus was isolated.

serumYear Numeric, the year the serum was collected.

cluster Factor, the cluster or serotype assignment for the strains.

color Character, a color associated with the cluster for plotting.

Source

Katzelnick, L.C., et al. (2019). An antigenically diverse, representative panel of dengue viruses for neutralizing antibody discovery and vaccine evaluation. *eLife*. doi:10.7554/eLife.42496

`error_calculator_comparison`
Error calculation and validation metrics for topolow Calculate Comprehensive Error Metrics

---

Description

Computes a comprehensive set of error metrics (in-sample, out-of-sample, completeness) between predicted and true dissimilarities for model evaluation.

Usage

```
error_calculator_comparison(
  predicted_dissimilarities,
  true_dissimilarities,
  input_dissimilarities = NULL
)
```

Arguments

`predicted_dissimilarities`
: Matrix of predicted dissimilarities from the model.

`true_dissimilarities`
: Matrix of true, ground-truth dissimilarities.

`input_dissimilarities`
: Matrix of input dissimilarities, which may contain NAs and is used to identify the pattern of missing values for out-of-sample error calculation. Optional - if not provided, defaults to `true_dissimilarities` (no holdout set).

Details

Input requirements and constraints:

- All input matrices must have matching dimensions.
- Row and column names must be consistent across matrices.
- NAs are allowed and handled appropriately.
- Threshold indicators (< or >) in the input matrix are processed correctly.

When `input_dissimilarities` is provided, it represents the training data where some values have been set to NA to create a holdout set. This allows calculation of:

- In-sample errors: for data available during training
- Out-of-sample errors: for data held out during training

When `input_dissimilarities` is NULL (default), all errors are treated as in-sample since no data was held out.

Value

A list containing:

`report_df`	A `data.frame` with detailed error metrics for each point-pair, including `InSampleError`, `OutSampleError`, and their percentage-based counterparts.
`Completeness`	A single numeric value representing the completeness statistic, which is the fraction of validation points for which a prediction could be made.

Examples

```
# Example 1: Normal evaluation (no cross-validation)
true_mat <- matrix(c(0, 1, 2, 1, 0, 3, 2, 3, 0), 3, 3)
pred_mat <- true_mat + rnorm(9, 0, 0.1)  # Add some noise

# Evaluate all predictions (input_dissimilarities defaults to true_dissimilarities)
errors1 <- error_calculator_comparison(pred_mat, true_mat)

# Example 2: Cross-validation evaluation
input_mat <- true_mat
input_mat[1, 3] <- input_mat[3, 1] <- NA  # Create holdout set

# Evaluate with train/test split
errors2 <- error_calculator_comparison(pred_mat, true_mat, input_mat)
```

`euclidean_embedding` *Main topolow algorithm implementation*

Description

[Stable]

topolow (topological stochastic pairwise reconstruction for Euclidean embedding) optimizes point positions in an N-dimensional space to match a target dissimilarity matrix. The algorithm uses a physics-inspired approach with spring and repulsive forces to find optimal point configurations while handling missing and thresholded measurements.

Usage

```
euclidean_embedding(
  dissimilarity_matrix,
  ndim,
  mapping_max_iter = 1000,
  k0,
  cooling_rate,
  c_repulsion,
  relative_epsilon = 1e-04,
  convergence_counter = 5,
  initial_positions = NULL,
  write_positions_to_csv = FALSE,
  output_dir,
  verbose = FALSE
)
```

Arguments

`dissimilarity_matrix`
: Matrix. A square, symmetric dissimilarity matrix. Can contain NA values for missing measurements and character strings with < or > prefixes for thresholded measurements.

`ndim`
: Integer. Number of dimensions for the embedding space.

`mapping_max_iter`
: Integer. Maximum number of map optimization iterations.

`k0`
: Numeric. Initial spring constant controlling spring forces.

`cooling_rate`
: Numeric. Rate of spring constant decay per iteration (0 < cooling_rate < 1).

`c_repulsion`
: Numeric. Repulsion constant controlling repulsive forces.

`relative_epsilon`
: Numeric. Convergence threshold for relative change in error. Default is 1e-4.

`convergence_counter`
: Integer. Number of iterations below threshold before declaring convergence. Default is 5.

`initial_positions`
: Matrix or NULL. Optional starting coordinates. If NULL, random initialization is used. Matrix should have nrow = nrow(dissimilarity_matrix) and ncol = ndim.

`write_positions_to_csv`
: Logical. Whether to save point positions to a CSV file. Default is FALSE.

`output_dir`
: Character. Directory to save the CSV file. Required if `write_positions_to_csv` is TRUE.

`verbose`
: Logical. Whether to print progress messages. Default is FALSE.

Details

The algorithm iteratively updates point positions using:

- Spring forces between points with measured dissimilarities.
- Repulsive forces between points without measurements.
- Conditional forces for thresholded measurements (< or >).
- An adaptive spring constant that decays over iterations.
- Convergence monitoring based on relative error change.
- Automatic matrix reordering to optimize convergence. Consider if downstream analyses depend on specific point ordering: The order of points in the output is adjusted to put high-dissimilarity points in the opposing ends.

This function replaces the deprecated `create_topolow_map()`. The core algorithm is identical, but includes performance improvements and enhanced validation.

Value

A `list` object of class `topolow`. This list contains the results of the optimization and includes the following components:

- `positions`: A `matrix` of the optimized point coordinates in the N-dimensional space.
- `est_distances`: A `matrix` of the Euclidean distances between points in the final optimized configuration.
- `mae`: The final Mean Absolute Error between the target dissimilarities and the estimated distances.
- `iter`: The total number of iterations performed before the algorithm terminated.
- `parameters`: A `list` containing the input parameters used for the optimization run.
- `convergence`: A `list` containing the final convergence status, including a logical `achieved` flag and the final `error` value.

See Also

`create_topolow_map()` for the deprecated predecessor function.

Examples

```
# Create a simple dissimilarity matrix
dist_mat <- matrix(c(0, 2, 3, 2, 0, 4, 3, 4, 0), nrow=3)

# Run topolow in 2D
result <- euclidean_embedding(
  dissimilarity_matrix = dist_mat,
  ndim = 2,
  mapping_max_iter = 100,
  k0 = 1.0,
  cooling_rate = 0.001,
  c_repulsion = 0.01,
  verbose = FALSE
)

# View results
head(result$positions)
print(result$mae)

# Example with thresholded measurements
thresh_mat <- matrix(c(0, ">2", 3, ">2", 0, "<5", 3, "<5", 0), nrow=3)
result_thresh <- euclidean_embedding(
  dissimilarity_matrix = thresh_mat,
  ndim = 2,
  mapping_max_iter = 50,
  k0 = 0.5,
  cooling_rate = 0.01,
  c_repulsion = 0.001
)
```

Euclidify *Automatic Euclidean Embedding with Parameter Optimization*

Description

A user-friendly wrapper function that automatically optimizes parameters and performs Euclidean embedding on a dissimilarity matrix. This function handles the entire workflow from parameter optimization to final embedding.

Usage

```
Euclidify(
  dissimilarity_matrix,
  output_dir,
  ndim_range = c(2, 10),
  k0_range = c(0.1, 20),
  cooling_rate_range = c(1e-04, 0.1),
  c_repulsion_range = c(1e-04, 1),
```

```
  n_initial_samples = 50,
  n_adaptive_samples = 150,
  max_cores = NULL,
  folds = 20,
  mapping_max_iter = 500,
  clean_intermediate = TRUE,
  verbose = "standard",
  fallback_to_defaults = FALSE,
  save_results = FALSE
)
```

Arguments

`dissimilarity_matrix`	Square symmetric dissimilarity matrix. Can contain NA values for missing measurements and threshold indicators (< or >).
`output_dir`	Character. Directory for saving optimization files and results. Required - no default.
`ndim_range`	Integer vector of length 2. Range for number of dimensions (minimum, maximum). Default: c(2, 10)
`k0_range`	Numeric vector of length 2. Range for initial spring constant (minimum, maximum). Default: c(0.1, 15)
`cooling_rate_range`	Numeric vector of length 2. Range for cooling rate (minimum, maximum). Default: c(0.001, 0.07)
`c_repulsion_range`	Numeric vector of length 2. Range for repulsion constant (minimum, maximum). Default: c(0.001, 0.4)
`n_initial_samples`	Integer. Number of samples for initial parameter optimization. Default: 100
`n_adaptive_samples`	Integer. Number of samples for adaptive refinement. Default: 250
`max_cores`	Integer. Maximum number of cores to use. Default: NULL (auto-detect)
`folds`	Integer. Number of cross-validation folds. Default: 20
`mapping_max_iter`	Integer. Maximum iterations for final embedding. Half this value is used for parameter search. Default: 1000
`clean_intermediate`	Logical. Whether to remove intermediate files. Default: TRUE
`verbose`	Character. Verbosity level: "off" (no output), "standard" (progress updates), or "full" (detailed output including from internal functions). Default: "standard"
`fallback_to_defaults`	Logical. Whether to use default parameters if optimization fails. Default: TRUE
`save_results`	Logical. Whether to save the final positions as CSV. Default: FALSE

Value

A list containing:

positions	Matrix of optimized coordinates
est_distances	Matrix of estimated distances
mae	Mean absolute error
optimal_params	List of optimal parameters found, including cross-validation MAE during optimization
optimization_summary	Summary of the optimization process
data_characteristics	Summary of input data characteristics
runtime	Total runtime in seconds

Examples

```
# Example 1: Basic usage with small matrix
test_data <- data.frame(
object = rep(paste0("Obj", 1:4), each = 4),
reference = rep(paste0("Ref", 1:4), 4),
score = sample(c(1, 2, 4, 8, 16, 32, 64, "<1", ">12"), 16, replace = TRUE)
)
dist_mat <- list_to_matrix(
  data = test_data,  # Pass the data frame, not file path
  object_col = "object",
  reference_col = "reference",
  value_col = "score",
  is_similarity = TRUE
)
## Not run:
# Note: output_dir is required for actual use
result <- Euclidify(
  dissimilarity_matrix = dist_mat,
  output_dir = tempdir()  # Use temp directory for example
)
coordinates <- result$positions

## End(Not run)

# Example 2: Using custom parameter ranges
## Not run:
result <- Euclidify(
  dissimilarity_matrix = dist_mat,
  output_dir = tempdir(),
  n_initial_samples = 10,
  n_adaptive_samples = 7,
  verbose = "off"
)

## End(Not run)
```

```
# Example 3: Handling missing data
dist_mat_missing <- dist_mat
dist_mat_missing[1, 3] <- dist_mat_missing[3, 1] <- NA
## Not run:
result <- Euclidify(
  dissimilarity_matrix = dist_mat_missing,
  output_dir = tempdir(),
  n_initial_samples = 10,
  n_adaptive_samples = 7,
  verbose = "off"
)

## End(Not run)

# Example 4: Using threshold indicators
dist_mat_threshold <- dist_mat
dist_mat_threshold[1, 2] <- ">2"
dist_mat_threshold[2, 1] <- ">2"
## Not run:
result <- Euclidify(
  dissimilarity_matrix = dist_mat_threshold,
  output_dir = tempdir(),
  n_initial_samples = 10,
  n_adaptive_samples = 7,
  verbose = "off"
)

## End(Not run)

# Example 5: Parallel processing with custom cores
## Not run:
result <- Euclidify(
  dissimilarity_matrix = dist_mat,
  output_dir = tempdir(),
  max_cores = 4,
  n_adaptive_samples = 100,
  save_results = TRUE  # Save positions to CSV
)

## End(Not run)
```

`example_positions` *Example Antigenic Mapping Data*

Description

HI titers of Influenza antigens and antisera published in Smith et al., 2004 were used to find the antigenic relationships and coordinates of the antigens. It can be used for mapping. The data captures how different influenza virus strains (antigens) react with antisera from infected individuals.

Usage

```
example_positions
```

Format

A data frame with 285 rows and 11 variables:

V1 First dimension coordinate from 5D mapping

V2 Second dimension coordinate from 5D mapping

V3 Third dimension coordinate from 5D mapping

V4 Fourth dimension coordinate from 5D mapping

V5 Fifth dimension coordinate from 5D mapping

name Strain identifier

antigen Logical; TRUE if point represents an antigen

antiserum Logical; TRUE if point represents an antiserum

cluster Factor indicating antigenic cluster assignment (A/H3N2 1968-2003)

color Color assignment for visualization

year Year of strain isolation

Source

Smith et al., 2004

---

extract_numeric_values

Utility functions for the topolow package Extract Numeric Values from Mixed Data

---

Description

Extracts numeric values from data that may contain threshold indicators (e.g., "<10", ">1280") or regular numeric values.

Usage

```
extract_numeric_values(x)
```

Arguments

x	A vector that may contain numeric values, character strings with threshold indicators, or a mix of both.

Value

A numeric vector with threshold indicators converted to their numeric equivalents.

Examples

```
# Mixed data with threshold indicators
mixed_data <- c(10, 20, "<5", ">100", 50)
extract_numeric_values(mixed_data)
```

`ggsave_white_bg` *Save ggplot with white background*

Description

Wrapper around `ggplot2::ggsave` that ensures a white background by default.

Usage

```
ggsave_white_bg(..., bg = "white")
```

Arguments

`...`	Other arguments passed on to the graphics device function, as specified by `device`.
`bg`	Background colour. If `NULL`, uses the `plot.background` fill value from the plot theme.

Value

No return value, called for side effects.

`h3n2_data` *H3N2 Influenza HI Assay Data from Smith et al. 2004*

Description

Hemagglutination inhibition (HI) assay data for influenza A/H3N2 viruses spanning 35 years of evolution.

Usage

```
h3n2_data
```

Format

A data frame with the following variables:

virusStrain Character. Virus strain identifier

serumStrain Character. Antiserum strain identifier

titer Numeric. HI assay titer value

virusYear Numeric. Year virus was isolated

serumYear Numeric. Year serum was collected

cluster Factor. Antigenic cluster assignment

color Character. Color code for visualization

Source

Smith et al. (2004) Science, 305(5682), 371-376.

---

hiv_titers *HIV Neutralization Assay Data*

Description

IC50 neutralization measurements between HIV viruses and antibodies.

Usage

```
hiv_titers
```

Format

A data frame with the following variables:

Antibody Character. Antibody identifier

Virus Character. Virus strain identifier

IC50 Numeric. IC50 neutralization value

Source

Los Alamos HIV Database (https://www.hiv.lanl.gov/)

hiv_viruses *HIV Virus Metadata*

Description

Reference information for HIV virus strains used in neutralization assays.

Usage

```
hiv_viruses
```

Format

A data frame with the following variables:

Virus.name Character. Virus strain identifier

Country Character. Country of origin

Subtype Character. HIV subtype

Year Numeric. Year of isolation

Source

Los Alamos HIV Database (https://www.hiv.lanl.gov/)

initial_parameter_optimization *Parameter Space Sampling and Optimization Functions for topolow*

Description

Performs parameter optimization using Latin Hypercube Sampling (LHS) combined with k-fold cross-validation. Parameters are sampled from specified ranges using maximin LHS design to ensure good coverage of parameter space. Each parameter set is evaluated using k-fold cross-validation to assess prediction accuracy. To calculate one NLL per set of parameters, the function uses a pooled errors approach which combine all validation errors into one set, then calculate a single NLL. This approach has two main advantages: 1- It treats all validation errors equally, respecting the underlying error distribution assumption 2- It properly accounts for the total number of validation points

Note: As of version 2.0.0, this function returns log-transformed parameters directly, eliminating the need to call `log_transform_parameters()` separately.

Usage

```
initial_parameter_optimization(
  dissimilarity_matrix,
  mapping_max_iter = 1000,
  relative_epsilon,
  convergence_counter,
  scenario_name,
  N_min,
  N_max,
  k0_min,
  k0_max,
  c_repulsion_min,
  c_repulsion_max,
  cooling_rate_min,
  cooling_rate_max,
  num_samples = 20,
  max_cores = NULL,
  folds = 20,
  verbose = FALSE,
  write_files = FALSE,
  output_dir
)
```

Arguments

`dissimilarity_matrix`	Matrix. Input dissimilarity matrix. Must be square and symmetric.
`mapping_max_iter`	Integer. Maximum number of optimization iterations for each map.
`relative_epsilon`	Numeric. Convergence threshold for relative change in error.
`convergence_counter`	Integer. Number of iterations below threshold before declaring convergence.
`scenario_name`	Character. Name for output files and job identification.
`N_min`, `N_max`	Integer. Range for the number of dimensions parameter.
`k0_min`, `k0_max`	Numeric. Range for the initial spring constant parameter.
`c_repulsion_min`, `c_repulsion_max`	Numeric. Range for the repulsion constant parameter.
`cooling_rate_min`, `cooling_rate_max`	Numeric. Range for the cooling rate parameter.
`num_samples`	Integer. Number of LHS samples to generate. Default: 20.
`max_cores`	Integer. Maximum number of cores for parallel processing. Default: NULL (uses all but one).
`folds`	Integer. Number of cross-validation folds. Default: 20.
`verbose`	Logical. Whether to print progress messages. Default: FALSE.

write_files	Logical. Whether to save results to a CSV file. Default: FALSE.
output_dir	Character. Directory for output files. Required if write_files is TRUE.

Details

Initial Parameter Optimization using Latin Hypercube Sampling

The function performs these steps:

1. Generates LHS samples in the parameter space (original scale for sampling).
2. Creates k-fold splits of the input data.
3. For each parameter set, it trains the model on each fold's training set and evaluates on the validation set, calculating a pooled MAE and NLL across all folds.
4. Computations are run locally in parallel.
5. **NEW**: Automatically log-transforms the final results for direct use with adaptive sampling.

Value

A data.frame containing the log-transformed parameter sets and their performance metrics. Columns include: log_N, log_k0, log_cooling_rate, log_c_repulsion, Holdout_MAE, and NLL.

Note

Breaking Change in v2.0.0: This function now returns log-transformed parameters directly. The returned data frame has columns log_N, log_k0, log_cooling_rate, log_c_repulsion instead of the original scale parameters. This eliminates the need to call log_transform_parameters() separately before using run_adaptive_sampling().

Breaking Change in v2.0.0: The parameter distance_matrix has been renamed to dissimilarity_matrix. Please update your code accordingly.

See Also

euclidean_embedding for the core optimization algorithm.

Examples

```
# This example can exceed 5 seconds on some systems.
# 1. Create a simple synthetic dataset for the example
synth_coords <- matrix(rnorm(60), nrow = 20, ncol = 3)
dist_mat <- coordinates_to_matrix(synth_coords)

# 2. Run the optimization on the synthetic data
results <- initial_parameter_optimization(
  dissimilarity_matrix = dist_mat,
  mapping_max_iter = 100,
  relative_epsilon = 1e-3,
  convergence_counter = 2,
  scenario_name = "test_opt_synthetic",
  N_min = 2, N_max = 5,
  k0_min = 1, k0_max = 10,
```

```
  c_repulsion_min = 0.001, c_repulsion_max = 0.05,
  cooling_rate_min = 0.001, cooling_rate_max = 0.02,
  num_samples = 4,
  max_cores = 1,  # Avoid parallel processing in check environment
  verbose = FALSE
)
```

`list_to_matrix` *topolow Data Preprocessing Functions*

Description

Converts data from long/list format (one measurement per row) to a symmetric dissimilarity matrix. The function handles both similarity and dissimilarity data, with optional conversion from similarity to dissimilarity.

Usage

```
list_to_matrix(
  data,
  object_col,
  reference_col,
  value_col,
  is_similarity = FALSE
)
```

Arguments

`data`	Data frame in long format with columns for objects, references, and values.
`object_col`	Character. Name of the column containing object identifiers.
`reference_col`	Character. Name of the column containing reference identifiers.
`value_col`	Character. Name of the column containing measurement values.
`is_similarity`	Logical. Whether values are similarities (TRUE) or dissimilarities (FALSE). If TRUE, similarities will be converted to dissimilarities by subtracting from the maximum value per reference. Default: FALSE.

Details

Convert List Format Data to Dissimilarity Matrix

The function expects data in long format with at least three columns:

- A column for object names
- A column for reference names
- A column containing the (dis)similarity values

When `is_similarity = TRUE`, the function converts similarities to dissimilarities by subtracting each similarity value from the maximum similarity value within each reference group. Threshold indicators (< or >) are handled appropriately and inverted during similarity-to-dissimilarity conversion.

Value

A symmetric matrix of dissimilarities with row and column names corresponding to the union of unique objects and references in the data. NA values represent unmeasured pairs, and the diagonal is set to 0.

Examples

```
# Example with dissimilarity data
data_dissim <- data.frame(
  object = c("A", "B", "A", "C"),
  reference = c("X", "X", "Y", "Y"),
  dissimilarity = c(2.5, 1.8, 3.0, 4.2)
)

mat_dissim <- list_to_matrix(
  data = data_dissim,
  object_col = "object",
  reference_col = "reference",
  value_col = "dissimilarity",
  is_similarity = FALSE
)

# Example with similarity data (will be converted to dissimilarity)
data_sim <- data.frame(
  object = c("A", "B", "A", "C"),
  reference = c("X", "X", "Y", "Y"),
  similarity = c(7.5, 8.2, 7.0, 5.8)
)

mat_from_sim <- list_to_matrix(
  data = data_sim,
  object_col = "object",
  reference_col = "reference",
  value_col = "similarity",
  is_similarity = TRUE
)
```

`log_transform_parameters`

Log Transform Parameter Samples

Description

Reads parameter samples from a CSV file and applies a log transformation to specified parameter columns (e.g., N, k0, cooling_rate, c_repulsion).

Note: As of version 2.0.0, this function is primarily for backward compatibility with existing parameter files. The `initial_parameter_optimization()` function now returns log-transformed parameters directly, eliminating the need for this separate transformation step in the normal workflow.

Usage

```
log_transform_parameters(samples_file, output_file = NULL)
```

Arguments

`samples_file`	Character. Path to the CSV file containing the parameter samples.
`output_file`	Character. Optional path to save the transformed data as a new CSV file.

Details

This function is maintained for users who have existing parameter files from older versions of the package or who need to work with parameter files that contain original-scale parameters. In the current workflow:

- `initial_parameter_optimization()` –> returns log-transformed parameters directly
- `run_adaptive_sampling()` –> works with log-transformed parameters
- `euclidean_embedding()` –> works with original-scale parameters

If you are working with the current workflow (using `Euclidify()` or calling `initial_parameter_optimization()` directly), you typically do not need to call this function.

Value

A `data.frame` with the log-transformed parameters. If `output_file` is specified, the data frame is also written to a file and returned invisibly.

Note

Backward Compatibility Note: This function is maintained for compatibility with existing workflows and parameter files. For new workflows, consider using `initial_parameter_optimization()` which returns log-transformed parameters directly.

Examples

```
# This example uses a sample file included with the package.
sample_file <- system.file("extdata", "sample_params.csv", package = "topolow")

# Ensure the file exists before running the example
if (nzchar(sample_file)) {
  # Transform the data from the sample file and return as a data frame
```

```
  transformed_data <- log_transform_parameters(sample_file, output_file = NULL)

  # Display the first few rows of the transformed data
  print(head(transformed_data))
}
```

`make_interactive` *Create Interactive Plot*

Description

Converts a static ggplot visualization to an interactive plotly visualization with customizable tooltips and interactive features.

Usage

```
make_interactive(plot, tooltip_vars = NULL)
```

Arguments

`plot`	ggplot object to convert
`tooltip_vars`	Vector of variable names to include in tooltips

Details

The function enhances static plots by adding:

- Hover tooltips with data values
- Zoom capabilities
- Pan capabilities
- Click interactions
- Double-click to reset

If tooltip_vars is NULL, the function attempts to automatically determine relevant variables from the plot's mapping.

Value

A `plotly` object with interactive features.

Examples

```
if (interactive() && requireNamespace("plotly", quietly = TRUE)) {
# Create sample data and plot
data <- data.frame(
  V1 = rnorm(100), V2 = rnorm(100), name=1:100,
  antigen = rep(c(0,1), 50), antiserum = rep(c(1,0), 50),
  year = rep(2000:2009, each=10), cluster = rep(1:5, each=20)
)

# Create temporal plot
p1 <- plot_temporal_mapping(data, ndim=2)

# Make interactive with default tooltips
p1_interactive <- make_interactive(p1)

# Create cluster plot with custom tooltips
p2 <- plot_cluster_mapping(data, ndim=2)
p2_interactive <- make_interactive(p2,
  tooltip_vars = c("cluster", "year", "antigen")
)
}
```

---

new_aesthetic_config *Plot Aesthetic Configuration Class*

Description

S3 class for configuring plot visual aesthetics including points, colors, labels and text elements.

Usage

```
new_aesthetic_config(
  point_size = 3.5,
  point_alpha = 0.8,
  point_shapes = c(antigen = 16, antiserum = 0),
  color_palette = c25,
  gradient_colors = list(low = "blue", high = "red"),
  show_labels = FALSE,
  show_title = FALSE,
  label_size = 3,
  title_size = 14,
  subtitle_size = 12,
  axis_title_size = 12,
  axis_text_size = 10,
  legend_text_size = 10,
  legend_title_size = 12,
  show_legend = TRUE,
  legend_position = "right",
```

```
  arrow_head_size = 0.2,
  arrow_alpha = 0.6
)
```

Arguments

point_size	Base point size
point_alpha	Point transparency
point_shapes	Named vector of shapes for different point types
color_palette	Color palette name or custom palette
gradient_colors	List with low and high colors for gradients
show_labels	Whether to show point labels
show_title	Whether to show plot title (default: FALSE)
label_size	Label text size
title_size	Title text size
subtitle_size	Subtitle text size
axis_title_size	Axis title text size
axis_text_size	Axis text size
legend_text_size	Legend text size
legend_title_size	Legend title text size
show_legend	Whether to show the legend
legend_position	Legend position ("none", "right", "left", "top", "bottom")
arrow_head_size	Size of the arrow head for velocity arrows (in cm)
arrow_alpha	Transparency of arrows (0 = invisible, 1 = fully opaque)

Value

An S3 object of class `aesthetic_config`, which is a list containing the specified configuration parameters for plot aesthetics.

new_annotation_config *Visualization functions for the topolow package Plot Annotation Configuration Class*

Description

S3 class for configuring point annotations in plots, including labels, connecting lines, and visual properties.

Usage

```
new_annotation_config(
  notable_points = NULL,
  size = 4.9,
  color = "black",
  alpha = 0.9,
  fontface = "plain",
  box = FALSE,
  segment_size = 0.3,
  segment_alpha = 0.6,
  min_segment_length = 0,
  max_overlaps = Inf,
  outline_size = 0.4
)
```

Arguments

notable_points	Character vector of notable points to highlight
size	Numeric. Size of annotations for notable points
color	Character. Color of annotations for notable points
alpha	Numeric. Alpha transparency of annotations
fontface	Character. Font face of annotations ("plain", "bold", "italic", etc.)
box	Logical. Whether to draw a box around annotations
segment_size	Numeric. Size of segments connecting annotations to points
segment_alpha	Numeric. Alpha transparency of connecting segments
min_segment_length	Numeric. Minimum length of connecting segments
max_overlaps	Numeric. Maximum number of overlaps allowed for annotations
outline_size	Numeric. Size of the outline for annotations

Value

An S3 object of class annotation_config, which is a list containing the specified configuration parameters for plot annotations.

new_dim_reduction_config *Dimension Reduction Configuration Class*

Description

S3 class for configuring dimension reduction parameters including method selection and algorithm-specific parameters.

Usage

```
new_dim_reduction_config(
  method = "pca",
  n_components = 2,
  scale = TRUE,
  center = TRUE,
  pca_params = list(tol = sqrt(.Machine$double.eps), rank. = NULL),
 umap_params = list(n_neighbors = 15, min_dist = 0.1, metric = "euclidean", n_epochs =
    200),
  tsne_params = list(perplexity = 30, mapping_max_iter = 1000, theta = 0.5),
  compute_loadings = FALSE,
  random_state = NULL
)
```

Arguments

method	Dimension reduction method ("pca", "umap", "tsne")
n_components	Number of components to compute
scale	Scale the data before reduction
center	Center the data before reduction
pca_params	List of PCA-specific parameters
umap_params	List of UMAP-specific parameters
tsne_params	List of t-SNE-specific parameters
compute_loadings	Compute and return loadings
random_state	Random seed for reproducibility

Value

An S3 object of class dim_reduction_config, which is a list containing the specified configuration parameters for dimensionality reduction.

new_layout_config *Plot Layout Configuration Class*

Description

S3 class for configuring plot layout including dimensions, margins, grids and coordinate systems.

Usage

```
new_layout_config(
  width = 8,
  height = 8,
  dpi = 300,
  aspect_ratio = 1,
  show_grid = TRUE,
  grid_type = "major",
  grid_color = "grey80",
  grid_linetype = "dashed",
  show_axis = TRUE,
  axis_lines = TRUE,
  plot_margin = margin(1, 1, 1, 1, "cm"),
  coord_type = "fixed",
  background_color = "white",
  panel_background_color = "white",
  panel_border = TRUE,
  panel_border_color = "black",
  save_plot = FALSE,
  save_format = "png",
  reverse_x = 1,
  reverse_y = 1,
  x_limits = NULL,
  y_limits = NULL,
  arrow_plot_threshold = 0.1
)
```

Arguments

width	Plot width in inches
height	Plot height in inches
dpi	Plot resolution
aspect_ratio	Plot aspect ratio
show_grid	Show plot grid
grid_type	Grid type ("none", "major", "minor", "both")
grid_color	Grid color
grid_linetype	Grid line type

show_axis	Show axes
axis_lines	Show axis lines
plot_margin	Plot margins in cm
coord_type	Coordinate type ("fixed", "equal", "flip", "polar")
background_color	Plot background color
panel_background_color	Panel background color
panel_border	Show panel border
panel_border_color	Panel border color
save_plot	Logical. Whether to save the plot to a file.
save_format	Plot save format ("png", "pdf", "svg", "eps")
reverse_x	Numeric multiplier for x-axis direction (1 or -1)
reverse_y	Numeric multiplier for y-axis direction (1 or -1)
x_limits	Numeric vector of length 2 specifying c(min, max) for x-axis. If NULL, limits are set automatically.
y_limits	Numeric vector of length 2 specifying c(min, max) for y-axis. If NULL, limits are set automatically.
arrow_plot_threshold	Threshold for velocity arrows to be drawn in the same antigenic distance unit (default: 0.10)

Value

An S3 object of class `layout_config`, which is a list containing the specified configuration parameters for plot layout.

`parameter_sensitivity_analysis`
Parameter Sensitivity Analysis

Description

Analyzes the sensitivity of the model performance (measured by MAE) to changes in a single parameter. This function bins the parameter range to identify the minimum MAE for each bin, helping to understand how robust the model is to parameter choices.

Usage

```
parameter_sensitivity_analysis(
  param,
  samples,
  bins = 30,
  mae_col = "Holdout_MAE",
  threshold_pct = 5,
  min_samples = 1
)
```

Arguments

param	The character name of the parameter to analyze.
samples	A data frame containing parameter samples and performance metrics.
bins	The integer number of bins to divide the parameter range into.
mae_col	The character name of the column containing the Mean Absolute Error (MAE) values.
threshold_pct	A numeric percentage above the minimum MAE to define an acceptable performance threshold.
min_samples	The integer minimum number of samples required in a bin for it to be included in the analysis.

Details

The function performs these steps:

1. Cleans the input data using Median Absolute Deviation (MAD) to remove outliers.
2. Bins the parameter values into equal-width bins.
3. Calculates the minimum MAE within each bin to create an empirical performance curve.
4. Identifies a performance threshold based on a percentage above the global minimum MAE.
5. Returns an S3 object for plotting and further analysis.

Value

An object of class "parameter_sensitivity" containing:

param_values	Vector of parameter bin midpoints
min_mae	Vector of minimum MAE values per bin
param_name	Name of analyzed parameter
threshold	Threshold value (default: min. +5%)
min_value	Minimum MAE value across all bins
sample_counts	Number of samples per bin

plot.parameter_sensitivity *Plot Parameter Sensitivity Analysis*

Description

The S3 plot method for `parameter_sensitivity` objects. It creates a visualization showing how the model's performance (minimum MAE) changes across the range of a single parameter. A threshold line is included to indicate the region of acceptable performance.

Usage

```
## S3 method for class 'parameter_sensitivity'
plot(
  x,
  width = 3.5,
  height = 3.5,
  save_plot = FALSE,
  output_dir,
  y_limit_factor = NULL,
  ...
)
```

Arguments

`x`	A `parameter_sensitivity` object, typically from `parameter_sensitivity_analysis()`.
`width`	The numeric width of the output plot in inches.
`height`	The numeric height of the output plot in inches.
`save_plot`	A logical indicating whether to save the plot to a file.
`output_dir`	A character string specifying the directory for output files. Required if `save_plot` is TRUE.
`y_limit_factor`	A numeric factor to set the upper y-axis limit as a percentage above the threshold value (e.g., 1.10 for 10% above). If NULL, scaling is automatic.
`...`	Additional arguments (not currently used).

Value

A `ggplot` object representing the sensitivity plot.

`plot.profile_likelihood` *Plot Method for profile_likelihood Objects*

Description

Creates a visualization of the profile likelihood for a parameter, showing the maximum likelihood estimates and the 95% confidence interval. It supports mathematical notation for parameter names for clearer plot labels.

Usage

```
## S3 method for class 'profile_likelihood'
plot(x, LL_max, width = 3.5, height = 3.5, save_plot = FALSE, output_dir, ...)
```

Arguments

`x`	A `profile_likelihood` object returned by `profile_likelihood()`.
`LL_max`	The global maximum log-likelihood value from the entire sample set, used as the reference for calculating the confidence interval.
`width, height`	Numeric. The width and height of the output plot in inches.
`save_plot`	Logical. If TRUE, the plot is saved to a file.
`output_dir`	Character. The directory where the plot will be saved. Required if `save_plot` is TRUE.
`...`	Additional arguments passed to the plot function.

Details

The 95% confidence interval is determined using the likelihood ratio test, where the cutoff is based on the chi-squared distribution: $LR(\theta_{ij}) = -2[logL_{max}(\theta_{ij}) - logL_{max}(\hat{\theta})]$. The interval includes all parameter values θ_{ij} for which $LR(\theta_{ij}) \leq \chi^2_{1,0.05} \approx 3.84$.

Value

A ggplot object representing the profile likelihood plot.

Examples

```
# This example can take more than 5 seconds to run.
# Create a sample data frame of MCMC samples
samples <- data.frame(
  log_N = log(runif(50, 2, 10)),
  log_k0 = log(runif(50, 1, 5)),
  log_cooling_rate = log(runif(50, 0.01, 0.1)),
  log_c_repulsion = log(runif(50, 0.1, 1)),
  NLL = runif(50, 20, 100)
)
```

```
# Calculate profile likelihood for the "log_N" parameter
pl_result <- profile_likelihood("log_N", samples, grid_size = 10)

# Provide the global maximum log-likelihood from the samples
LL_max <- max(-samples$NLL)

# The plot function requires the ggplot2 package
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(pl_result, LL_max, width = 4, height = 3)
}
```

`plot.topolow_convergence` *Plot Method for topolow Convergence Diagnostics*

Description

Creates visualizations of convergence diagnostics from a sampling run, including parameter mean trajectories and covariance matrix stability over iterations. This helps assess whether parameter estimation has converged.

Usage

```
## S3 method for class 'topolow_convergence'
plot(x, param_names = NULL, ...)
```

Arguments

`x`	A `topolow_convergence` object from `check_gaussian_convergence()`.
`param_names`	Optional character vector of parameter names for plot titles. If NULL, names are taken from the input object.
`...`	Additional arguments (not currently used).

Details

The function generates two types of plots:

1. Parameter mean plots: Shows how the mean value for each parameter changes over iterations. Stabilization of these plots indicates convergence.
2. Covariance change plot: Shows the relative change in the Frobenius norm of the covariance matrix. A decreasing trend approaching zero indicates stable relationships between parameters.

Value

A grid of plots showing convergence metrics.

See Also

check_gaussian_convergence for generating the convergence object.

Examples

```
# Example with simulated data
chain_data <- data.frame(
  param1 = rnorm(1000, mean = 1.5, sd = 0.1),
  param2 = rnorm(1000, mean = -0.5, sd = 0.2)
)

# Check convergence
results <- check_gaussian_convergence(chain_data)

# Plot diagnostics
plot(results)

# With custom parameter names
plot(results, param_names = c("Parameter 1 (log)", "Parameter 2 (log)"))
```

---

plot.topolow_diagnostics

Plot Method for topolow parameter estimation Diagnostics

Description

Creates trace and density plots for multiple chains to assess convergence and mixing. This is an S3 method that dispatches on topolow_diagnostics objects.

Usage

```
## S3 method for class 'topolow_diagnostics'
plot(
  x,
  output_dir,
  output_file = "topolow_param_diagnostics.png",
  save_plot = FALSE,
  ...
)
```

Arguments

x	A topolow_diagnostics object from calculate_diagnostics().
output_dir	Character. Directory for output files. Required if save_plot is TRUE.
output_file	Character path for saving the plot.
save_plot	Logical. Whether to save the plot.
...	Additional arguments passed to create_diagnostic_plots.

Value

A ggplot object of the combined plots.

`plot_3d_mapping` *Create 3D Visualization*

Description

Creates an interactive or static 3D visualization using rgl. Supports both temporal and cluster-based coloring schemes with configurable point appearances and viewing options.

Usage

```
plot_3d_mapping(
  df,
  ndim,
  dim_config = new_dim_reduction_config(),
  aesthetic_config = new_aesthetic_config(),
  layout_config = new_layout_config(),
  interactive = TRUE,
  output_dir
)
```

Arguments

`df`	Data frame containing: - V1, V2, ... Vn: Coordinate columns - antigen: Binary indicator for antigen points - antiserum: Binary indicator for antiserum points - cluster: (Optional) Factor or integer cluster assignments - year: (Optional) Numeric year values for temporal coloring
`ndim`	Number of dimensions in input coordinates (must be >= 3)
`dim_config`	Dimension reduction configuration object
`aesthetic_config`	Aesthetic configuration object
`layout_config`	Layout configuration object
`interactive`	Logical; whether to create an interactive plot
`output_dir`	Character. Directory for output files. Required if `interactive` is `FALSE`.

Details

The function supports two main visualization modes:

1. Interactive mode: Creates a manipulatable 3D plot window
2. Static mode: Generates a static image from a fixed viewpoint

Color schemes are automatically selected based on available data:

- If cluster data is present: Uses discrete colors per cluster
- If year data is present: Uses continuous color gradient
- Otherwise: Uses default point colors

For data with more than 3 dimensions, dimension reduction is applied first.

Note: This function requires the rgl package and OpenGL support. If rgl is not available, the function will return a 2D plot with a message explaining how to enable 3D visualization.

Value

Invisibly returns the rgl scene ID for further manipulation if rgl is available, or a 2D ggplot object as a fallback.

See Also

plot_temporal_mapping for 2D temporal visualization plot_cluster_mapping for 2D cluster visualization make_interactive for converting 2D plots to interactive versions

Examples

```
# Create sample data
set.seed(123)
data <- data.frame(
  V1 = rnorm(100), V2 = rnorm(100), V3 = rnorm(100), V4 = rnorm(100), name = 1:100,
  antigen = rep(c(0,1), 50), antiserum = rep(c(1,0), 50),
  cluster = rep(1:5, each=20), year = rep(2000:2009, each=10)
)

# Create a static plot and save to a temporary file
# This example requires an interactive session and the 'rgl' package.
if (interactive() && requireNamespace("rgl", quietly = TRUE)) {
  temp_dir <- tempdir()
  # Basic interactive plot (will open a new window)
  if(interactive()) {
    plot_3d_mapping(data, ndim=4)
  }

# Custom configuration for temporal visualization
aesthetic_config <- new_aesthetic_config(
  point_size = 5,
  point_alpha = 0.8,
  gradient_colors = list(
    low = "blue",
    high = "red"
  )
)

layout_config <- new_layout_config(
  width = 12,
  height = 12,
  background_color = "black",
```

```
  show_axis = TRUE
)
  # Create customized static plot and save it
plot_3d_mapping(data, ndim=4,
  aesthetic_config = aesthetic_config,
  layout_config = layout_config,
  interactive = FALSE, output_dir = temp_dir
)
  list.files(temp_dir)
  unlink(temp_dir, recursive = TRUE)
}
```

`plot_cluster_mapping` *Create Clustered Mapping Plots*

Description

Antigenic Mapping and Antigenic Velocity Function. Creates a visualization of points colored by cluster assignment using dimension reduction, with optional antigenic velocity arrows. Points are colored by cluster with different shapes for antigens and antisera.

Usage

```
plot_cluster_mapping(
  df_coords,
  ndim,
  dim_config = new_dim_reduction_config(),
  aesthetic_config = new_aesthetic_config(),
  layout_config = new_layout_config(),
  annotation_config = new_annotation_config(),
  output_dir,
  show_shape_legend = TRUE,
  cluster_legend_title = "Cluster",
  draw_arrows = FALSE,
  annotate_arrows = TRUE,
  phylo_tree = NULL,
  sigma_t = NULL,
  sigma_x = NULL,
  clade_node_depth = NULL,
  show_one_arrow_per_cluster = FALSE,
  cluster_legend_order = NULL
)
```

Arguments

`df_coords`	Data frame containing: - V1, V2, ... Vn: Coordinate columns - antigen: Binary indicator for antigen points - antiserum: Binary indicator for antiserum points - cluster: Factor or integer cluster assignments

`ndim` Number of dimensions in input coordinates

`dim_config` Dimension reduction configuration object specifying method and parameters

`aesthetic_config`
Aesthetic configuration object controlling plot appearance

`layout_config` Layout configuration object controlling plot dimensions and style. Use x_limits and y_limits in layout_config to set axis limits.

`annotation_config`
Annotation configuration object for labeling notable points

`output_dir` Character. Directory for output files. Required if `layout_config$save_plot` is `TRUE`.

`show_shape_legend`
Logical. Whether to show the shape legend (default: TRUE)

`cluster_legend_title`
Character. Custom title for the cluster legend (default: "Cluster")

`draw_arrows` logical; if TRUE, compute and draw antigenic drift vectors

`annotate_arrows`
logical; if TRUE, show names of the points having arrows

`phylo_tree` Optional; phylo object in Newick format. Does not need to be rooted. If provided, used to compute antigenic velocity arrows.

`sigma_t` Optional; numeric; bandwidth for the Gaussian kernel discounting on time in years or the time unit of the data. If NULL, uses Silverman's rule of thumb.

`sigma_x` Optional; numeric; bandwidth for the Gaussian kernel discounting on antigenic distance in antigenic units. If NULL, uses Silverman's rule of thumb.

`clade_node_depth`
Optional; integer; number of levels of parent nodes to define clades. Antigens from different clades will be excluded from the calculation antigenic velocity arrows. (Default: Automatically calculated mode of leaf-to-backbone distance of the tree)

`show_one_arrow_per_cluster`
Shows only the largest antigenic velocity arrow in each cluster

`cluster_legend_order`
in case you prefer a certain order for clusters in the legend, provide a list with that order here; e.g., c("cluster 2", "cluster 1")

Details

The function performs these steps:

1. Validates input data structure and types
2. Applies dimension reduction if ndim > 2
3. Creates visualization with cluster-based coloring
4. Applies specified aesthetic and layout configurations
5. Applies custom axis limits if specified in layout_config

Different shapes distinguish between antigens and antisera points, while color represents cluster assignment. The color palette can be customized through the aesthetic_config.

Value

A ggplot object containing the cluster mapping visualization.

See Also

plot_temporal_mapping for temporal visualization plot_3d_mapping for 3D visualization new_dim_reduction_config for dimension reduction options new_aesthetic_config for aesthetic options new_layout_config for layout options new_annotation_config for annotation options

Examples

```
# Basic usage with default configurations
data <- data.frame(
  V1 = rnorm(100), V2 = rnorm(100), V3 = rnorm(100), name = 1:100,
  antigen = rep(c(0,1), 50), antiserum = rep(c(1,0), 50),
  cluster = rep(1:5, each=20)
)
p1 <- plot_cluster_mapping(data, ndim=3)

# Save plot to a temporary directory
temp_dir <- tempdir()
# Custom configurations with specific color palette and axis limits
aesthetic_config <- new_aesthetic_config(
  point_size = 4,
  point_alpha = 0.7,
  color_palette = c("red", "blue", "green", "purple", "orange"),
  show_labels = TRUE,
  label_size = 3
)

layout_config_save <- new_layout_config(save_plot = TRUE,
  width = 10,
  height = 8,
  coord_type = "fixed",
  show_grid = TRUE,
  grid_type = "major",
  x_limits = c(-10, 10),
  y_limits = c(-8, 8)
)

p_saved <- plot_cluster_mapping(data, ndim=3,
  layout_config = layout_config_save,
  aesthetic_config = aesthetic_config,
  output_dir = temp_dir
)

list.files(temp_dir)
unlink(temp_dir, recursive = TRUE)
```

plot_network_structure *Plot Network Structure*

Description

Creates a visualization of the dissimilarity matrix as a network graph, showing data availability patterns and connectivity between points.

Usage

```
plot_network_structure(
  network_results,
  output_file = NULL,
  width = 3000,
  height = 3000,
  dpi = 300
)
```

Arguments

network_results	The list output from `analyze_network_structure()`.
output_file	Character. An optional full path to save the plot. If NULL, the plot is not saved.
width	Numeric. Width in pixels for saved plot (default: 3000).
height	Numeric. Height in pixels for saved plot (default: 3000).
dpi	Numeric. Resolution in dots per inch (default: 300).

Value

A `ggplot` object representing the network graph.

Examples

```
# Create a sample dissimilarity matrix
adj_mat <- matrix(runif(25), 5, 5)
rownames(adj_mat) <- colnames(adj_mat) <- paste0("Point", 1:5)
adj_mat[lower.tri(adj_mat)] <- t(adj_mat)[lower.tri(adj_mat)]
diag(adj_mat) <- 0
net_analysis <- analyze_network_structure(adj_mat)

# Create and display the plot
plot_network_structure(net_analysis)
```

plot_temporal_mapping *Create Temporal Mapping Plot*

Description

Antigenic Mapping and Antigenic Velocity Function. Creates a visualization of points colored by time (year) using dimension reduction, with optional antigenic velocity arrows. Points are colored on a gradient scale based on their temporal values, with different shapes for antigens and antisera.

Usage

```
plot_temporal_mapping(
  df_coords,
  ndim,
  dim_config = new_dim_reduction_config(),
  aesthetic_config = new_aesthetic_config(),
  layout_config = new_layout_config(),
  annotation_config = new_annotation_config(),
  output_dir,
  show_shape_legend = TRUE,
  draw_arrows = FALSE,
  annotate_arrows = TRUE,
  phylo_tree = NULL,
  sigma_t = NULL,
  sigma_x = NULL,
  clade_node_depth = NULL
)
```

Arguments

df_coords	Data frame containing: - V1, V2, ... Vn: Coordinate columns - antigen: Binary indicator for antigen points - antiserum: Binary indicator for antiserum points - year: Numeric year values for temporal coloring
ndim	Number of dimensions in input coordinates
dim_config	Dimension reduction configuration object specifying method and parameters
aesthetic_config	Aesthetic configuration object controlling plot appearance
layout_config	Layout configuration object controlling plot dimensions and style. Use x_limits and y_limits in layout_config to set axis limits.
annotation_config	Annotation configuration object for labeling notable points
output_dir	Character. Directory for output files. Required if `layout_config$save_plot` is `TRUE`.
show_shape_legend	Logical. Whether to show the shape legend (default: TRUE)

`draw_arrows` logical; if TRUE, compute and draw antigenic drift vectors

`annotate_arrows` logical; if TRUE, show names of the points having arrows

`phylo_tree` Optional; phylo object in Newick format. Does not need to be rooted. If provided, used to compute antigenic velocity arrows.

`sigma_t` Optional; numeric; bandwidth for the Gaussian kernel discounting on time in years or the time unit of the data. If NULL, uses Silverman's rule of thumb.

`sigma_x` Optional; numeric; bandwidth for the Gaussian kernel discounting on antigenic distancein antigenic units. If NULL, uses Silverman's rule of thumb.

`clade_node_depth` Optional; integer; number of levels of parent nodes to define clades. Antigens from different clades will be excluded from the calculation antigenic velocity arrows. (Default: Automatically calculated mode of leaf-to-backbone distance of the tree)

Details

The function performs these steps:

1. Validates input data structure and types
2. Applies dimension reduction if ndim > 2
3. Creates visualization with temporal color gradient
4. Applies specified aesthetic and layout configurations
5. Applies custom axis limits if specified in layout_config

Different shapes distinguish between antigens and antisera points, while color represents temporal progression.

Value

A `ggplot` object containing the temporal mapping visualization.

See Also

`plot_cluster_mapping` for cluster-based visualization `plot_3d_mapping` for 3D visualization `new_dim_reduction_config` for dimension reduction options `new_aesthetic_config` for aesthetic options `new_layout_config` for layout options `new_annotation_config` for annotation options

Examples

```
# Basic usage with default configurations
data <- data.frame(
  V1 = rnorm(100), V2 = rnorm(100), V3 = rnorm(100), name = 1:100,
  antigen = rep(c(0,1), 50), antiserum = rep(c(1,0), 50),
  year = rep(2000:2009, each=10)
)
# Plot without saving
```

```
p1 <- plot_temporal_mapping(data, ndim=3)

# Save plot to a temporary directory
temp_dir <- tempdir()
layout_config_save <- new_layout_config(save_plot = TRUE,
                       x_limits = c(-10, 10),
                       y_limits = c(-8, 8))
p_saved <- plot_temporal_mapping(data, ndim = 3, layout_config = layout_config_save,
                                 output_dir = temp_dir)
list.files(temp_dir) # Check that file was created
unlink(temp_dir, recursive = TRUE) # Clean up
```

`print.parameter_sensitivity` *Print Method for Parameter Sensitivity Objects*

Description

The S3 print method for `parameter_sensitivity` objects. It displays a concise summary of the analysis results, including the parameter analyzed, the minimum error found, and the performance threshold.

Usage

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

Arguments

`x`	A `parameter_sensitivity` object.
`...`	Additional arguments passed to the print function (not currently used).

Value

Invisibly returns the original object. Called for its side effect of printing a summary to the console.

`print.profile_likelihood` *Print Method for profile_likelihood Objects*

Description

Provides a concise summary of a `profile_likelihood` object.

Usage

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

Arguments

x	A profile_likelihood object.
...	Additional arguments passed to print.

Value

The original profile_likelihood object (invisibly). Called for its side effect of printing a summary to the console.

---

print.topolow *Print method for topolow objects*

Description

Provides a concise display of key optimization results from euclidean_embedding.

Usage

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

Arguments

x	A topolow object returned by euclidean_embedding().
...	Additional arguments passed to print (not used).

Value

The original topolow object (invisibly). This function is called for its side effect of printing a summary to the console.

Examples

```
# Create a simple dissimilarity matrix and run the optimization
dist_mat <- matrix(c(0, 2, 3, 2, 0, 4, 3, 4, 0), nrow=3)
result <- euclidean_embedding(dist_mat, ndim=2, mapping_max_iter=50,
                           k0=1.0, cooling_rate=0.001, c_repulsion=0.1,
                           verbose = FALSE)
# Print the result object
print(result)
```

print.topolow_convergence

Print Method for topolow Convergence Diagnostics

Description

Print Method for topolow Convergence Diagnostics

Usage

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

Arguments

x	A topolow_convergence object.
...	Additional arguments passed to print.

Value

No return value; called for its side effect of printing a summary.

print.topolow_diagnostics

Print Method for topolow parameter estimation Diagnostics

Description

Print Method for topolow parameter estimation Diagnostics

Usage

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

Arguments

x	A topolow_diagnostics object.
...	Additional arguments passed to print.

Value

No return value; called for its side effect of printing a summary.

process_antigenic_data

Process Raw Antigenic Assay Data

Description

Processes raw antigenic assay data from data frames into standardized long and matrix formats. Handles both similarity data (like titers, which need conversion to distances) and direct dissimilarity measurements like IC50. Preserves threshold indicators (<, >) and handles repeated measurements by averaging.

Usage

```
process_antigenic_data(
  data,
  antigen_col,
  serum_col,
  value_col,
  is_similarity = FALSE,
  metadata_cols = NULL,
  base = NULL,
  scale_factor = 1
)
```

Arguments

data	Data frame containing raw data.
antigen_col	Character. Name of column containing virus/antigen identifiers.
serum_col	Character. Name of column containing serum/antibody identifiers.
value_col	Character. Name of column containing measurements (titers or distances).
is_similarity	Logical. Whether values are measures of similarity such as titers or binding affinities (TRUE) or dissimilarities like IC50 (FALSE). Default: FALSE.
metadata_cols	Character vector. Names of additional columns to preserve.
base	Numeric. Base for logarithm transformation (default: 2 for similarities, e for dissimilarities).
scale_factor	Numeric. Scale factor for similarities. This is the base value that all other dilutions are multiples of. E.g., 10 for HI assay where titers are 10, 20, 40,... Default: 1.

Details

The function handles these key steps:

1. Validates input data and required columns
2. Transforms values to log scale

3. Converts similarities to distances using Smith's method if needed
4. Averages repeated measurements
5. Creates standardized long format
6. Creates symmetric distance matrix
7. Preserves metadata and threshold indicators
8. Preserves virusYear and serumYear columns if present

Input requirements and constraints:

- Data frame must contain required columns
- Column names must match specified parameters
- Values can include threshold indicators (< or >)
- Metadata columns must exist if specified
- Allowed Year-related column names are "virusYear" and "serumYear"

Value

A list containing two elements:

`long`	A `data.frame` in long format with standardized columns, including the original identifiers, processed values, and calculated distances. Any specified metadata is also included.
`matrix`	A numeric `matrix` representing the processed symmetric distance matrix, with antigens and sera on columns and rows.

Examples

```
# Example 1: Processing HI titer data (similarities)
antigen_data <- data.frame(
  virus = c("A/H1N1/2009", "A/H1N1/2010", "A/H1N1/2011", "A/H1N1/2009", "A/H1N1/2010"),
  serum = c("anti-2009", "anti-2009", "anti-2009", "anti-2010", "anti-2010"),
  titer = c(1280, 640, "<40", 2560, 1280),  # Some below detection limit
  cluster = c("A", "A", "B", "A", "A"),
  color = c("red", "red", "blue", "red", "red")
)

# Process HI titer data (similarities -> distances)
results <- process_antigenic_data(
  data = antigen_data,
  antigen_col = "virus",
  serum_col = "serum",
  value_col = "titer",
  is_similarity = TRUE,  # Titers are similarities
  metadata_cols = c("cluster", "color"),
  scale_factor = 10  # Base dilution factor
)

# View the long format data
print(results$long)
```

```
# View the distance matrix
print(results$matrix)

# Example 2: Processing IC50 data (already dissimilarities)
ic50_data <- data.frame(
  virus = c("HIV-1", "HIV-2", "HIV-3"),
  antibody = c("mAb1", "mAb1", "mAb2"),
  ic50 = c(0.05, ">10", 0.2)
)

results_ic50 <- process_antigenic_data(
  data = ic50_data,
  antigen_col = "virus",
  serum_col = "antibody",
  value_col = "ic50",
  is_similarity = FALSE  # IC50 values are dissimilarities
)
```

`profile_likelihood` *Profile Likelihood Analysis*

Description

Calculates the profile likelihood for a given parameter by evaluating the conditional maximum likelihood across a grid of parameter values. This "empirical profile likelihood" estimates the likelihood surface based on samples from Monte Carlo simulations.

Usage

```
profile_likelihood(
  param,
  samples,
  grid_size = 40,
  bandwidth_factor = 0.05,
  start_factor = 0.5,
  end_factor = 1.5,
  min_samples = 5
)
```

Arguments

`param`	The character name of the parameter to analyze (e.g., "log_N").
`samples`	A data frame containing parameter samples and a log-likelihoods column named "NLL".
`grid_size`	The integer number of grid points for the analysis.
`bandwidth_factor`	A numeric factor for the local sample window size.

`start_factor`, `end_factor`
: Numeric range multipliers for parameter grid (default: 0.5, 1.2)

`min_samples`
: Integer minimum samples required for reliable estimate (default: 10)

Details

For each value in the parameter grid, the function:

1. Identifies nearby samples using a bandwidth window.
2. Calculates the conditional maximum likelihood from these samples.
3. Tracks sample counts to assess the reliability of the estimate.

Value

Object of class "profile_likelihood" containing:

`param`
: Vector of parameter values

`ll`
: Vector of log-likelihood values

`param_name`
: Name of analyzed parameter

`bandwidth`
: Bandwidth used for local windows

`sample_counts`
: Number of samples per estimate

See Also

The S3 methods `print.profile_likelihood` and `summary.profile_likelihood` for viewing results.

Examples

```
# Create a sample data frame of parameter samples
mcmc_samples <- data.frame(
  log_N = log(runif(50, 2, 10)),
  log_k0 = log(runif(50, 1, 5)),
  log_cooling_rate = log(runif(50, 0.01, 0.1)),
  log_c_repulsion = log(runif(50, 0.1, 1)),
  NLL = runif(50, 20, 100)
)

# Calculate profile likelihood for the "log_N" parameter
pl <- profile_likelihood("log_N", mcmc_samples,
                         grid_size = 10, # Smaller grid for a quick example
                         bandwidth_factor = 0.05)

# Print the results
print(pl)
```

`run_adaptive_sampling` *Performs adaptive Monte Carlo sampling*

Description

Performs adaptive Monte Carlo sampling to explore and refine the parameter space, running locally in parallel. Samples are drawn adaptively based on previously evaluated likelihoods to focus sampling in high-likelihood regions. Results from all parallel jobs accumulate in a single output file.

Usage

```
run_adaptive_sampling(
  initial_samples_file,
  scenario_name,
  dissimilarity_matrix,
  max_cores = NULL,
  num_samples = 10,
  mapping_max_iter = 1000,
  relative_epsilon = 1e-04,
  folds = 20,
  output_dir,
  verbose = FALSE
)
```

Arguments

`initial_samples_file`	Character. Path to a CSV file containing initial samples.
`scenario_name`	Character. Name for the output files.
`dissimilarity_matrix`	Matrix. The input dissimilarity matrix.
`max_cores`	Integer. Number of cores to use for parallel execution. If NULL, uses all available cores minus 1.
`num_samples`	Integer. Number of new samples to generate via adaptive sampling.
`mapping_max_iter`	Integer. Maximum number of map optimization iterations.
`relative_epsilon`	Numeric. Convergence threshold for relative change in error. Default is 1e-4.
`folds`	Integer. Number of cross-validation folds.
`output_dir`	Character. Required directory for output files.
`verbose`	Logical. Whether to print progress messages. Default is FALSE.

Value

No return value. Called for its side effect of writing results to a CSV file in `output_dir`.

Examples

```
# 1. Locate the example initial samples file included with the package
# In a real scenario, this file would be from an 'initial_parameter_optimization' run.
initial_file <- system.file(
  "extdata", "initial_samples_example.csv",
  package = "topolow"
)

# 2. Create a temporary directory for the function's output
# This function requires a writable directory for its results.
temp_out_dir <- tempdir()

# 3. Create a sample dissimilarity matrix for the function to use
dissim_mat <- matrix(runif(100, 1, 10), 10, 10)
diag(dissim_mat) <- 0

# 4. Run the adaptive sampling only if the example file is found
if (nzchar(initial_file)) {
  run_adaptive_sampling(
    initial_samples_file = initial_file,
    scenario_name = "adaptive_test_example",
    dissimilarity_matrix = dissim_mat,
    output_dir = temp_out_dir,
    max_cores = 1,
    num_samples = 1,
    verbose = FALSE
  )

  # 5. Verify output files were created
  print("Output files from adaptive sampling:")
  print(list.files(temp_out_dir, recursive = TRUE))

  # 6. Clean up the temporary directory
  unlink(temp_out_dir, recursive = TRUE)
}
```

`save_plot` *Save Plot to File*

Description

Saves a plot (ggplot or rgl scene) to file with specified configuration. Supports multiple output formats and configurable dimensions.

Usage

```
save_plot(plot, filename, layout_config = new_layout_config(), output_dir)
```

Arguments

plot	ggplot or rgl scene object to save
filename	Output filename (with or without extension)
layout_config	Layout configuration object controlling output parameters
output_dir	Character. Directory for output files. This argument is required.

Details

Supported file formats:

- PNG: Best for web and general use
- PDF: Best for publication quality vector graphics
- SVG: Best for web vector graphics
- EPS: Best for publication quality vector graphics

The function will:

1. Auto-detect plot type (ggplot or rgl)
2. Use appropriate saving method
3. Apply layout configuration settings
4. Add file extension if not provided

Value

No return value, called for side effects (saves a plot to a file).

Examples

```
# The sole purpose of save_plot is to write a file, so its example must demonstrate this.
# For CRAN tests we wrap the example in \donttest{} to avoid writing files.

# Create a temporary directory for saving all plots
temp_dir <- tempdir()

# --- Example 1: Basic ggplot save ---
# Create sample data with 3 dimensions to support both 2D and 3D plots
data <- data.frame(
  V1 = rnorm(10), V2 = rnorm(10), V3 = rnorm(10), name=1:10,
  antigen = rep(c(0,1), 5), antiserum = rep(c(1,0), 5),
  year = 2000:2009
)
p <- plot_temporal_mapping(data, ndim=2)
save_plot(p, "temporal_plot.png", output_dir = temp_dir)

# --- Example 2: Save with custom layout ---
layout_config <- new_layout_config(
  width = 12,
  height = 8,
  dpi = 600,
```

```
  save_format = "pdf"
)
save_plot(p, "high_res_plot.pdf", layout_config, output_dir = temp_dir)

# --- Verify files and clean up ---
list.files(temp_dir)
unlink(temp_dir, recursive = TRUE)
```

`scatterplot_fitted_vs_true` *Plot Fitted vs. True Dissimilarities*

Description

Creates diagnostic plots comparing fitted dissimilarities from a model against the true dissimilarities. It generates both a scatter plot with an identity line and prediction intervals, and a residuals plot.

Usage

```
scatterplot_fitted_vs_true(
  dissimilarity_matrix,
  p_dissimilarity_mat,
  scenario_name,
  ndim,
  save_plot = FALSE,
  output_dir,
  confidence_level = 0.95
)
```

Arguments

`dissimilarity_matrix`	Matrix of true dissimilarities.
`p_dissimilarity_mat`	Matrix of predicted/fitted dissimilarities.
`scenario_name`	Character string for output file naming. Used if `save_plot` is TRUE.
`ndim`	Integer number of dimensions used in the model.
`save_plot`	Logical. Whether to save plots to files. Default: FALSE.
`output_dir`	Character. Directory for output files. Required if `save_plot` is TRUE.
`confidence_level`	Numeric confidence level for prediction intervals (default: 0.95).

Value

A list containing the `scatter_plot` and `residuals_plot` ggplot objects.

Examples

```
# Create sample data
true_dist <- matrix(runif(100, 1, 10), 10, 10)
pred_dist <- true_dist + rnorm(100)

# Create plots without saving
plots <- scatterplot_fitted_vs_true(
  dissimilarity_matrix = true_dist,
  p_dissimilarity_mat = pred_dist,
  save_plot = FALSE
 )

# You can then display a plot, for instance:
# plots$scatter_plot
```

`summary.topolow` *Summary method for topolow objects*

Description

Provides a more detailed summary of the optimization results from `euclidean_embedding`, including parameters, convergence, and performance metrics.

Usage

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

Arguments

`object`	A `topolow` object returned by `euclidean_embedding()`.
`...`	Additional arguments passed to summary (not used).

Value

No return value. This function is called for its side effect of printing a detailed summary to the console.

Examples

```
# Create a simple dissimilarity matrix and run the optimization
dist_mat <- matrix(c(0, 2, 3, 2, 0, 4, 3, 4, 0), nrow=3)
result <- euclidean_embedding(dist_mat, ndim=2, mapping_max_iter=50,
                           k0=1.0, cooling_rate=0.001, c_repulsion=0.1,
                           verbose = FALSE)
# Summarize the result object
summary(result)
```

symmetric_to_nonsymmetric_matrix *Convert distance matrix to assay panel format*

Description

Convert distance matrix to assay panel format

Usage

```
symmetric_to_nonsymmetric_matrix(dist_matrix, selected_names)
```

Arguments

dist_matrix	Distance matrix
selected_names	Names of reference points

Value

A non-symmetric matrix in assay panel format, where rows are test antigens and columns are reference antigens.

table_to_matrix *Convert Table Format Data to Dissimilarity Matrix*

Description

Converts data from table/matrix format (objects as rows, references as columns) to a symmetric dissimilarity matrix. The function creates a matrix where both rows and columns contain the union of all object and reference names.

Usage

```
table_to_matrix(data, is_similarity = FALSE)
```

Arguments

data	Matrix or data frame where rownames represent objects, columnnames represent references, and cells contain (dis)similarity values.
is_similarity	Logical. Whether values are similarities (TRUE) or dissimilarities (FALSE). If TRUE, similarities will be converted to dissimilarities by subtracting from the maximum value per column (reference). Default: FALSE.

Details

The function takes a table where:

- Rows represent objects
- Columns represent references
- Values represent (dis)similarities

It creates a symmetric matrix where both rows and columns contain the union of all object names (row names) and reference names (column names). The original measurements are preserved, and the matrix is made symmetric by filling both (i,j) and (j,i) positions with the same value.

When `is_similarity = TRUE`, similarities are converted to dissimilarities by subtracting each value from the maximum value in its respective column (reference). Threshold indicators (< or >) are handled and inverted during conversion.

Value

A symmetric matrix of dissimilarities with row and column names corresponding to the union of all object and reference names. NA values represent unmeasured pairs, and the diagonal is set to 0.

Examples

```
# Example with dissimilarity data in table format
dissim_table <- matrix(c(1.2, 2.1, 3.4, 1.8, 2.9, 4.1),
                       nrow = 2, ncol = 3,
                       dimnames = list(c("Obj1", "Obj2"),
                                       c("Ref1", "Ref2", "Ref3")))

mat_dissim <- table_to_matrix(dissim_table, is_similarity = FALSE)

# Example with similarity data (will be converted to dissimilarity)
sim_table <- matrix(c(8.8, 7.9, 6.6, 8.2, 7.1, 5.9),
                    nrow = 2, ncol = 3,
                    dimnames = list(c("Obj1", "Obj2"),
                                    c("Ref1", "Ref2", "Ref3")))

mat_from_sim <- table_to_matrix(sim_table, is_similarity = TRUE)
```

---

titers_list_to_matrix *Convert Long Format Data to Distance Matrix*

Description

Converts a dataset from long format to a symmetric distance matrix. The function handles antigenic cartography data where measurements may exist between antigens and antisera points. Row and column names can be optionally sorted by a time variable.

Usage

```
titers_list_to_matrix(
  data,
  chnames,
  chorder = NULL,
  rnames,
  rorder = NULL,
  values_column,
  rc = FALSE,
  sort = FALSE
)
```

Arguments

`data`	Data frame in long format
`chnames`	Character. Name of column holding the challenge point names.
`chorder`	Character. Optional name of column for challenge point ordering.
`rnames`	Character. Name of column holding reference point names.
`rorder`	Character. Optional name of column for reference point ordering.
`values_column`	Character. Name of column containing distance/difference values. It should be from the nature of "distance" (e.g., antigenic distance or IC50), not "similarity" (e.g., HI Titer.)
`rc`	Logical. If TRUE, reference points are treated as a subset of challenge points. If FALSE, they are treated as distinct sets. Default is FALSE.
`sort`	Logical. Whether to sort rows/columns by chorder/rorder. Default FALSE.

Details

The function expects data in long format with at least three columns:

- A column for challenge point names
- A column for reference point names
- A column containing the distance/difference values

Optionally, ordering columns can be provided to sort the output matrix. The 'rc' parameter determines how to handle shared names between references and challenges.

Value

A symmetric `matrix` of distances with row and column names corresponding to the unique points in the data. `NA` values represent unmeasured pairs.

Examples

```
data <- data.frame(
  antigen = c("A", "B", "A"),
  serum = c("X", "X", "Y"),
  distance = c(2.5, 1.8, 3.0),
  year = c(2000, 2001, 2000)
)

# Basic conversion
mat <- titers_list_to_matrix(data,
                     chnames = "antigen",
                     rnames = "serum",
                     values_column = "distance")

# With sorting by year
mat_sorted <- titers_list_to_matrix(data,
                            chnames = "antigen",
                            chorder = "year",
                            rnames = "serum",
                            rorder = "year",
                            values_column = "distance",
                            sort = TRUE)
```

---

weighted_kde *Weighted Kernel Density Estimation*

Description

Performs weighted kernel density estimation for univariate data. This is useful for analyzing parameter distributions where each sample has an associated importance weight (e.g., a likelihood).

Usage

```
weighted_kde(x, weights, n = 512, from = min(x), to = max(x))
```

Arguments

x	A numeric vector of samples.
weights	A numeric vector of weights corresponding to each sample in x.
n	The integer number of points at which to evaluate the density.
from, to	The range over which to evaluate the density.

Value

A list containing the evaluation points (x) and the estimated density values (y).

Index