Package 'topolow'

June 24, 2025

Title Topolow: Antigenic Mapping and Antigenic Velocity Algorithm

Version 0.3.2

Description An implementation of the TopoLow algorithm, a novel, physics-inspired method for antigenic cartography. TopoLow addresses significant challenges in mapping antigenic relationships,
especially from sparse and noisy experimental data. The package transforms cross-reactivity and binding affinity measurements into accurate spatial representations in a phenotype space.
Key features include:
* Robust Mapping from Sparse Data: Effectively creates complete and consistent maps even with high
proportions of missing data (e.g., >95%).
* Physics-Inspired Optimization: Models antigens as particles connected by springs (for measured interactions) and subject to repulsive forces (for missing interactions), reducing the need for complex gradient computations.
* Automatic Dimensionality Detection: Employs a likelihood-based approach to determine the optimal
number of dimensions for the antigenic 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: 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 pathogens, including influenza, HIV, and Dengue viruses.
It can be applied to any continuous and relational phenotype under directional selection pressure.
Methods are described in Arhami and Rohani (2025) <doi:10.1101/2025.02.09.637307>.

License BSD_3_clause + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports ggplot2 (>= 3.4.0),
dplyr (>= 1.1.0),
data.table (>= 1.14.0),
reshape2 (>= 1.4.4),
stats,

utils,
plotly (>= 4.10.0),
Racmacs (>= 1.1.2),
parallel (>= 4.1.0),
coda (>= 0.19-4),
MASS,
vegan,
filelock,
igraph,
lhs,
umap,
gridExtra,
scales,
Rtsne,
ggrepel,
rgl (>= 1.0.0),
rlang,
ape

Suggests 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)

Contents

`adaptive_MC_sampling` *Perform Adaptive Monte Carlo Sampling*

Description

Main function implementing adaptive Monte Carlo sampling to explore parameter space. Updates sampling distribution based on evaluated likelihoods.

Usage

```
adaptive_MC_sampling(
  samples_file,
  distance_matrix,
  iterations = 1,
  batch_size = 1,
  mapping_max_iter,
  relative_epsilon,
  folds = 20,
  num_cores = 1,
  scenario_name,
  output_dir = NULL,
  verbose = FALSE
)
```

Arguments

`samples_file`	Path to CSV with initial samples
`distance_matrix`	Distance matrix to fit
`iterations`	Number of sampling iterations per job
`batch_size`	Samples per iteration (fixed to 1)
`mapping_max_iter`	Maximum map optimization iterations
`relative_epsilon`	Convergence threshold
`folds`	Number of CV folds
`num_cores`	Number of cores for parallel processing
`scenario_name`	Name for output files
`output_dir`	Character. Directory for output files. If NULL, uses current directory
`verbose`	Logical. Whether to print progress messages. Default: FALSE

Value

Data frame of samples with evaluated likelihoods

add_noise_bias *Add Noise and Bias to Matrix Data*

Description

Creates noisy versions of a distance matrix by adding random noise and/or systematic bias. Useful for testing robustness of algorithms to measurement errors and systematic biases.

Usage

```
add_noise_bias(matrix_data)
```

Arguments

matrix_data	Numeric matrix to add noise to

Details

The function generates three variants of the input matrix:

1. n1: Matrix with random Gaussian noise
2. n2: Different realization of random noise
3. nb: Matrix with both random noise and systematic negative bias

The noise level is scaled relative to the data mean to maintain realistic error magnitudes.

Value

List containing three matrices:

n1	Matrix with first noise realization
n2	Matrix with second noise realization
nb	Matrix with noise and negative bias

Examples

```
## Not run:
# Create sample distance matrix
dist_mat <- matrix(runif(100), 10, 10)
dist_mat[lower.tri(dist_mat)] <- t(dist_mat)[lower.tri(dist_mat)]
diag(dist_mat) <- 0

# Generate noisy versions
noisy_variants <- add_noise_bias(dist_mat)

## End(Not run)
```

`aggregate_parameter_optimization_results` *Aggregate Results from Parameter Optimization Jobs*

Description

Combines results from multiple parameter optimization jobs executed via SLURM into a single dataset. This function processes results from jobs submitted by `submit_parameter_jobs`.

Usage

```
aggregate_parameter_optimization_results(
  scenario_name,
  write_files = TRUE,
  dir = NULL
)
```

Arguments

`scenario_name`	Character. Name used in parameter optimization jobs.
`write_files`	Logical. Whether to save combined results (default: TRUE).
`dir`	Character. Directory for output files. If NULL, uses current directory

Details

The function looks for CSV files in the init_param_optimization directory that match the pattern params_{scenario_name}.csv. It combines all results into a single dataset, computes median values across folds, and optionally writes the aggregated results to a file.

The output file is saved as: model_parameters/{scenario_name}_model_parameters.csv

Value

Data frame of aggregated results containing median values across folds:

`N`	Number of dimensions
`k0`	Initial spring constant
`cooling_rate`	Spring decay rate
`c_repulsion`	Repulsion constant
`Holdout_MAE`	Median holdout mean absolute error
`NLL`	negative log likelihood of the fold

See Also

`initial_parameter_optimization` for running the optimization `submit_parameter_jobs` for job submission

Examples

```
## Not run:
# After running parameter optimization jobs:
results <- aggregate_parameter_optimization_results("optimization_run1")

## End(Not run)
```

`analyze_network_structure` *Calculate Network Analysis Metrics*

Description

Analyzes the connectivity pattern in a distance matrix by converting it to a network representation. Useful for assessing data completeness and structure.

Usage

```
analyze_network_structure(distance_matrix)
```

Arguments

`distance_matrix`	Square symmetric matrix of distances

Value

List containing:

`adjacency`	Logical matrix indicating presence of measurements
`connectivity`	Data frame with connectivity metrics per point
`summary`	List of overall network statistics

Examples

```
## Not run:
metrics <- analyze_network_structure(dist_mat)
print(metrics$summary$completeness)

## End(Not run)
```

`calculate_annual_distances` *Calculate Annual Distance Metrics*

Description

Calculates year-over-year antigenic distances and statistics. Compares each point to the mean coordinates of the previous year.

Usage

```
calculate_annual_distances(df_coords, ndim, na.rm = TRUE)
```

Arguments

`df_coords`	Data frame containing: - V1...Vn coordinate columns - year: Numeric years - name: Point identifiers (will use rownames if missing)
`ndim`	Number of coordinate dimensions
`na.rm`	Logical indicating whether to remove NA values

Value

List containing:

`dist_data`	Data frame with columns: • year: Collection year • distance: Distance from previous year mean
`summary`	List with: • overall_mean: Mean distance across all years • overall_sd: Standard deviation of distances

Examples

```
## Not run:
annual_stats <- calculate_annual_distances(coords, ndim=2)
print(annual_stats$summary$overall_mean)

## End(Not run)
```

`calculate_cumulative_distances` *Calculate Cumulative Distance Metrics*

Description

Calculates cumulative distance metrics either from a reference point or between all pairs. Handles both seasonal and year-based analyses.

Usage

```
calculate_cumulative_distances(
  df_coords,
  ndim,
  reference_row = FALSE,
  na.rm = TRUE
)
```

Arguments

`df_coords`	Data frame containing: - V1...Vn coordinate columns - year: Numeric years - season: Character season identifiers. - cluster: Factor cluster assignments - color: Character color codes
`ndim`	Number of coordinate dimensions
`reference_row`	Integer index of reference row (or FALSE for all-pairs analysis)
`na.rm`	Logical indicating whether to remove NA values

Value

List containing either: If reference_row provided:

`summary_data` Data frame with columns:

- season_num: Numeric season identifier based on Influenza A.
- cluster: Cluster assignment
- color: Point color
- avg_euclidean_dist: Mean distance to reference
- count: Points per cluster
- total_count: Total points per season
- fraction: Proportion of points in cluster

If reference_row = FALSE:

`dist_data` Data frame with columns:

- year_diff: Years between points
- euclidean_dist: Distance between points
- ref_year: Reference year

Examples

```
## Not run:
# Calculate distances from reference point
ref_distances <- calculate_cumulative_distances(coords, ndim=2, reference_row=1)

# Calculate all pairwise distances
all_distances <- calculate_cumulative_distances(coords, ndim=2, reference_row=FALSE)

## End(Not run)
```

---

calculate_diagnostics *Calculate Adaptive Monte Carlo Sampling Diagnostics*

Description

Calculates standard Adaptive Monte Carlo Sampling diagnostics including R-hat (potential scale reduction) and effective sample size for multiple chains. Can be used with any iterative sampling or optimization procedure that produces chain-like output.

Usage

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

Arguments

chain_files	Character vector of paths to CSV files containing chains
mutual_size	Integer number of samples to use from end of each chain

Value

List containing:

rhat	R-hat statistic for each parameter
ess	Effective sample size for each parameter

Examples

```
## Not run:
chain_files <- c("chain1.csv", "chain2.csv", "chain3.csv")
diag <- calculate_diagnostics(chain_files, mutual_size = 1000)
print(diag)  # Shows R-hat and ESS
plot(diag)   # Creates density plots
print(diag$rhat) # Should be close to 1
print(diag$ess)  # Should be large enough (>400) for reliable inference

## End(Not run)
```

calculate_prediction_interval

Calculate prediction interval for distance estimates

Description

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

Usage

```
calculate_prediction_interval(
  distance_matrix,
  p_dist_mat,
  confidence_level = 0.95
)
```

Arguments

distance_matrix	Matrix of true distances
p_dist_mat	Matrix of predicted distances
confidence_level	Confidence level for interval (default: 0.95)

Value

Numeric margin of error for prediction interval

calculate_procrustes_difference

Calculate Procrustes Difference Between Maps

Description

Computes the quantitative difference between two maps using Procrustes analysis. The difference is calculated as the sum of squared differences after optimal rotation and scaling.

Usage

```
calculate_procrustes_difference(map1, map2)
```

Arguments

map1	Data frame with coordinates from first map (must have X, X.1 columns)
map2	Data frame with coordinates from second map (must have X, X.1 columns)

Value

Numeric sum of squared differences after Procrustes transformation

Examples

```
## Not run:
map1 <- read.csv("map1_coords.csv")
map2 <- read.csv("map2_coords.csv")
diff <- calculate_procrustes_difference(map1, map2)

## End(Not run)
```

`calculate_procrustes_significance` *Calculate Statistical Significance Between Maps Using Procrustes Analysis*

Description

Performs Procrustes analysis between two maps and calculates statistical significance of their differences using permutation tests. Handles common data cleaning steps like removing missing values and ensuring comparable point sets.

Usage

```
calculate_procrustes_significance(map1, map2)
```

Arguments

`map1`	Data frame with coordinates from first map (must have X, X.1 columns)
`map2`	Data frame with coordinates from second map (must have X, X.1 columns)

Value

Numeric p-value from Procrustes permutation test

Examples

```
## Not run:
map1 <- read.csv("map1_coords.csv")
map2 <- read.csv("map2_coords.csv")
p_val <- calculate_procrustes_significance(map1, map2)

## End(Not run)
```

calculate_weighted_marginals

Calculate Weighted Marginal Distributions

Description

Calculates marginal distributions for each parameter with weights derived from log-likelihoods. Uses parallel processing for efficiency.

Usage

```
calculate_weighted_marginals(samples)
```

Arguments

samples	Data frame containing: - log_N, log_k0, log_cooling_rate, log_c_repulsion: Parameter columns - NLL: Negative log-likelihood column

Details

Uses kernel density estimation weighted by normalized likelihoods. Parallelizes computation across parameter dimensions using mclapply.

Value

Named list of marginal distributions, each containing:

x	Vector of parameter values
y	Vector of density estimates

check_gaussian_convergence

Check Multivariate Gaussian Convergence

Description

Assesses convergence of multivariate samples by monitoring changes in mean vector and covariance matrix over a sliding window. Useful for checking stability of parameter distributions in optimization or sampling.

Usage

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

Arguments

data	Matrix or data frame of samples where columns are parameters
window_size	Integer size of sliding window for statistics
tolerance	Numeric convergence threshold for relative changes

Value

List containing:

converged	Logical indicating if convergence achieved
mean_converged	Logical for mean convergence
cov_converged	Logical for covariance convergence
final_mean	Vector of final mean values
final_cov	Final covariance matrix
mean_history	Matrix of mean values over iterations
cov_changes	Vector of covariance changes

Examples

```
## Not run:
data <- read.csv("chain_data.csv")
conv_results <- check_gaussian_convergence(data)
print(conv_results)  # Shows summary
plot(conv_results)   # Creates convergence plots

## End(Not run)
```

---

check_job_status *Check Status of Submitted Job*

Description

Check Status of Submitted Job

Usage

```
check_job_status(job_id)
```

Arguments

job_id	Character. SLURM job ID

Value

Character job status or NA if not found

clean_data *Clean Data by Removing MAD-based Outliers*

Description

Removes outliers from numeric data using the Median Absolute Deviation method. Outliers are replaced with NA values. This function is particularly useful for cleaning parameter tables where each column may contain outliers.

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. Whether to log transform data before outlier detection (default: FALSE)

Value

Numeric vector with outliers replaced by 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)

# Clean multiple parameter columns
param_table <- data.frame(
  k0 = runif(100),
  cooling_rate = runif(100),
  c_repulsion = runif(100)
)
clean_table <- as.data.frame(lapply(param_table, clean_data))
```

color_palettes *Color Palettes*

Description

Predefined color palettes optimized for visualization

Usage

```
c25

c25_claud

c25_old

c25_older
```

Format

An object of class `character` of length 20.

An object of class `character` of length 24.

An object of class `character` of length 25.

An object of class `character` of length 25.

coordinates_to_matrix *Convert coordinates to distance matrix*

Description

Calculates pairwise Euclidean distances between points in coordinate space

Usage

```
coordinates_to_matrix(positions)
```

Arguments

`positions`	Matrix of coordinates where rows are points and columns are dimensions

Value

Matrix of pairwise distances between points

copy_reproduction_examples
Copy Reproduction Examples to Working Directory

Description

Copies all reproduction examples, including data files and supporting materials, to a specified directory.

Usage

```
copy_reproduction_examples(dest_dir = file.path(getwd(), "examples"))
```

Arguments

dest_dir	Character. Destination directory (default: "reproduction_examples" in current working directory)

Value

Invisibly returns the path to the destination directory

create_and_optimize_RACMACS_map
Create and Optimize RACMACS Map

Description

Creates and optimizes an antigenic map using RACMACS and keeps the best optimization. This function wraps RACMACS functionality to provide a simplified interface for map creation and optimization.

Usage

```
create_and_optimize_RACMACS_map(
  titer_table,
  dim = 2,
  optimization_number = 400,
  scenario_name,
  num_cores = 1
)
```

Arguments

titer_table	Matrix or data frame of titer measurements
dim	Integer number of dimensions for the map (default: 2)
optimization_number	Integer number of optimization runs (default: 400)
scenario_name	Character string for output file naming
num_cores	Integer number of cores to use for optimization (default: 1)

Value

RACMACS map object containing optimized coordinates

Examples

```
## Not run:
# Create and optimize map from titer data
map <- create_and_optimize_RACMACS_map(titer_table)

# Create map with specific settings
map <- create_and_optimize_RACMACS_map(
  titer_table,
  dim = 3,
  optimization_number = 1000,
  scenario_name = "example_map"
)

## End(Not run)
```

---

create_cv_folds *Create Cross-validation Folds for Distance Matrix*

Description

Creates k-fold cross-validation splits of a distance matrix while maintaining symmetry. Each fold has a training matrix with some values masked for validation.

Usage

```
create_cv_folds(
  truth_matrix,
  no_noise_truth = NULL,
  n_folds = 10,
  random_seed = NULL
)
```

Arguments

truth_matrix	Matrix of true distances
no_noise_truth	Optional matrix of noise-free distances. If provided, used as truth.
n_folds	Integer number of folds to create
random_seed	Integer random seed for reproducibility

Value

List of lists, each containing:

truth	Truth matrix for this fold
train	Training matrix with masked validation entries

Examples

```
## Not run:
# Create 5-fold CV splits
folds <- create_cv_folds(dist_matrix, n_folds = 5, random_seed = 123)

## End(Not run)
```

`create_diagnostic_plots` *Create Diagnostic Plots for Multiple Chains*

Description

Creates trace and density plots for multiple Adaptive Monte Carlo Sampling or optimization chains to assess convergence and mixing. Displays parameter trajectories and distributions across chains.

Usage

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

Arguments

`chain_files`	Character vector of paths to CSV files containing chain data
`mutual_size`	Integer number of samples to use from end of each chain
`output_file`	Character path for saving plot
`output_dir`	Character. Directory for output files. If NULL, uses current directory
`save_plot`	Logical. Whether to save plots to files. Default: TRUE
`width`, `height`, `res`	Plot dimensions and resolution for saving

Value

Invisible NULL, saves plot to file

Examples

```
## Not run:
chain_files <- c("chain1.csv", "chain2.csv", "chain3.csv")
create_diagnostic_plots(chain_files, mutual_size = 2000,
  output_file = "chain_diagnostics.png")

## End(Not run)
```

`create_slurm_script` *Create SLURM Job Script*

Description

Creates a SLURM batch script with specified parameters and resource requests.

Usage

```
create_slurm_script(
  job_name,
  script_path,
  args,
  num_cores,
  output_file,
  error_file,
  time = "8:00:00",
  memory = "4G",
  partition = "rohani_p",
  r_module = "R/4.4.1-foss-2022b",
  working_dir = NULL,
  extra_sbatch_args = NULL
)
```

Arguments

`job_name`	Name of the job
`script_path`	Path to R script to execute
`args`	Vector of command line arguments
`num_cores`	Number of CPU cores to request
`output_file`	Path for job output file
`error_file`	Path for job error file
`time`	Time limit (default: "8:00:00")
`memory`	Memory request (default: "14G")
`partition`	SLURM partition (default: "rohani_p")
`r_module`	Character. R module to load (default: "R/4.4.1-foss-2022b")
`working_dir`	Working directory (default: NULL)
`extra_sbatch_args`	Additional SBATCH arguments (default: NULL)

Value

Path to created script file

`create_topolow_map` *Main TopoLow algorithm implementation*

Description

TopoLow (Topological Optimization for Low-Dimensional Mapping) optimizes point positions in n-dimensional space to match a target distance 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

```
create_topolow_map(
  distance_matrix,
  ndim,
  mapping_max_iter,
  k0,
  cooling_rate,
  c_repulsion,
  relative_epsilon = 1e-04,
  convergence_counter = 5,
  initial_positions = NULL,
  write_positions_to_csv = TRUE,
  verbose = FALSE
)
```

Arguments

`distance_matrix`	Matrix. Square, symmetric distance 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(distance_matrix) and ncol = ndim.
`write_positions_to_csv`	Logical. Whether to save point positions to CSV file. Default is TRUE.
`verbose`	Logical. Whether to print progress messages. Default is TRUE.

Details

The algorithm iteratively updates point positions using:

- Spring forces between points with measured distances
- Repulsive forces between points without measurements
- Modified forces for thresholded measurements (< or >)
- Adaptive spring constant that decays over iterations
- Convergence monitoring based on relative error change

Valid parameter ranges and constraints:

- ndim: Positive integer, typically 2-20.
- k0: Initial spring constant, positive numeric > 0. Typical range: 0.1-30 Controls initial force strength
- cooling_rate: Spring and repulsion decay rate, numeric between 0 and 1. Typical range: 0.0001-0.1 Controls how quickly spring forces weaken
- c_repulsion: Repulsion constant, positive numeric > 0. Typical range: 0.00001-0.1 Controls strength of repulsive forces
- relative_epsilon: Positive numeric, typically 1e-9 to 1e-3 Smaller values require more iterations but give higher precision
- convergence_counter: Positive integer, typically 5-20 Higher values do not necessarily lead to a better convergence

Value

A list with class "topolow" containing:

- positions: Matrix of optimized point coordinates
- est_distances: Matrix of distances in the optimized configuration
- mae: Mean absolute error between target and optimized distances
- iter: Number of iterations performed
- parameters: List of input parameters used
- convergence: List with convergence status and final error

Examples

```
## Not run:
# Create a simple distance matrix
dist_mat <- matrix(c(0, 2, 3, 2, 0, 4, 3, 4, 0), nrow=3)

# Run TopoLow in 2D
result <- create_topolow_map(dist_mat, ndim=2, mapping_max_iter=1000,
                      k0=1.0, cooling_rate=0.001, c_repulsion=0.1)

# Plot results
plot(result$positions)

## End(Not run)
```

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

dist_to_titer_table *Convert Distance Matrix to Titer Panel Format*

Description

Converts a distance matrix to a titer panel format, handling threshold measurements and logarithmic transformations common in antigenic cartography. The function identifies reference points (typically antisera) and challenge points (typically antigens) based on row/column name prefixes.

Usage

```
dist_to_titer_table(input_matrix, base = exp(1), tens = 1)
```

Arguments

input_matrix	Matrix of distances, with row/column names prefixed with "V/" for antigens and "S/" for sera
base	Numeric. Base for logarithmic transformation. Default exp(1). For HI Assay 2
tens	Numeric. Scaling factor for final titers. Default 1. For HI Assay 10

Details

The function:

1. Identifies antigen and serum entries from matrix row/column names
2. Creates titer table from antigen-serum pairs
3. Handles threshold indicators (< and >) in distance values
4. Applies appropriate transformations to convert distances to titers

Transformation steps:

1. Extract numeric values from thresholded measurements
2. Convert distances to titers via logarithmic transformation
3. Apply scaling factor
4. Reapply threshold indicators to transformed values

Value

A matrix of titers with:

- Rows corresponding to antigen strains (without "V/" prefix)
- Columns corresponding to antisera (without "S/" prefix)
- Values as character strings including threshold indicators where applicable
- NA values replaced with "*"

Examples

```
## Not run:
# Create sample distance matrix
dist_mat <- matrix(c(0, 2, ">3", 2, 0, 4, "3", 4, 0), nrow=3)
rownames(dist_mat) <- c("V/strain1", "V/strain2", "S/serum1")
colnames(dist_mat) <- c("V/strain1", "V/strain2", "S/serum1")

# Convert to titer panel
titer_panel <- dist_to_titer_table(dist_mat)

## End(Not run)
```

---

error_calculator_comparison

Calculate comprehensive error metrics between predicted and true distances

---

Description

Computes various error metrics including in-sample and out-of-sample errors, and Completeness statistics for model evaluation.

Usage

```
error_calculator_comparison(p_dist_mat, truth_matrix, input_matrix)
```

Arguments

p_dist_mat	Matrix of predicted distances
truth_matrix	Matrix of true distances
input_matrix	Matrix of input distances (may contain NAs and is used to find the NAs' pattern)

Details

Input requirements and constraints:

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

Value

List containing:

report_df	Data frame with error metrics per point
Completeness	Numeric Completeness statistic

---

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

Arhami and Rohani 2025 doi:

`find_mode` *Find Mode of Density Distribution*

Description

Calculates the mode (maximum point) of a kernel density estimate.

Usage

```
find_mode(density)
```

Arguments

`density`	List containing density estimate with components: **x** Vector of values **y** Vector of density estimates

Value

Numeric value of the mode

`generate_complex_data` *Generate Complex High-Dimensional Data for Testing*

Description

Generates synthetic high-dimensional data with clusters and trends for testing dimensionality reduction methods. Creates data with specified properties:

- Multiple clusters along a trend line
- Variable density regions
- Controllable noise levels
- Optional visualization

The function generates cluster centers along a trend line, adds points around those centers with specified spread, and incorporates random noise to create high and low density areas. The data is useful for testing dimensionality reduction and visualization methods.

Usage

```
generate_complex_data(
  n_points = 500,
  n_dim = 10,
  n_clusters = 4,
  cluster_spread = 1,
  fig_name = NA
)
```

Arguments

n_points	Integer number of points to generate
n_dim	Integer number of dimensions
n_clusters	Integer number of clusters
cluster_spread	Numeric controlling cluster variance
fig_name	Character path to save visualization (optional)

Value

Data frame with generated coordinates in n_dim dimensions. Column names are "Dim1" through "DimN" where N is n_dim.

Examples

```
## Not run:
# Generate basic dataset
data <- generate_complex_data(n_points = 500, n_dim = 10,
                              n_clusters = 4, cluster_spread = 1)

# Generate and visualize dataset
data <- generate_complex_data(n_points = 500, n_dim = 10,
                              n_clusters = 4, cluster_spread = 1,
                              fig_name = "cluster_viz.png")

## End(Not run)
```

---

generate_synthetic_datasets

Generate Synthetic Distance Matrices with Missing Data

Description

Creates synthetic distance matrices with controlled levels of missingness and noise for testing and validating mapping algorithms. Generates multiple datasets with different dimensionalities and missingness patterns.

Usage

```
generate_synthetic_datasets(
  n_dims_list,
  seeds,
  n_points,
  missingness_levels = list(S = 0.67, M = 0.77, L = 0.87),
  output_dir = NULL,
  prefix = "sim",
  save_plots = FALSE
)
```

Arguments

n_dims_list	Numeric vector of dimensions to generate data for
seeds	Integer vector of random seeds (same length as n_dims_list)
n_points	Integer number of points to generate
missingness_levels	Named list of missingness percentages (default: list(S=0.67, M=0.77, L=0.87))
output_dir	Character path to directory for saving outputs (optional)
prefix	Character string to prefix output files (optional)
save_plots	Logical whether to save network visualization plots

Value

List containing:

matrices	List of generated distance matrices
panels	List of generated assay panels
metadata	Data frame with generation parameters

Examples

```
## Not run:
# Generate datasets with different dimensions
results <- generate_synthetic_datasets(
  n_dims_list = c(2, 5, 10),
  seeds = c(123, 456, 789),
  n_points = 250,
  output_dir = "sim_data"
)

# Custom missingness levels
results <- generate_synthetic_datasets(
  n_dims_list = c(2, 5),
  seeds = c(123, 456),
  n_points = 200,
  missingness_levels = list(low=0.5, high=0.8)
)

## End(Not run)
```

---

generate_unique_string

Generate unique string identifiers with year suffix

Description

Generate unique string identifiers with year suffix

Usage

```
generate_unique_string(n, length = 8, lower_bound = 1, upper_bound = 20)
```

Arguments

n	Number of strings to generate
length	Length of random part of string (default: 8)
lower_bound	Lower bound for year suffix (default: 1)
upper_bound	Upper bound for year suffix (default: 20)

Value

Character vector of unique strings with year suffixes

---

ggsave *Save ggplot with white background*

Description

Wrapper around ggplot2::ggsave that ensures white background. This function masks ggplot2::ggsave.

Usage

```
ggsave(..., 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.

---

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/)

`increase_na_percentage` *Increase Missing Values in a Matrix*

Description

Strategically introduces NA values into a distance matrix while maintaining symmetry. New NA values are added preferentially farther from the diagonal to simulate real-world measurement patterns where distant pairs are more likely to be unmeasured.

Usage

```
increase_na_percentage(mat, target_na_percentage)
```

Arguments

`mat`	Matrix to modify
`target_na_percentage`	Numeric between 0 and 1 specifying desired proportion of NAs

Details

The function:

1. Calculates needed additional NAs to reach target percentage
2. Creates probability matrix favoring off-diagonal elements
3. Randomly selects positions weighted by distance from diagonal
4. Maintains matrix symmetry by mirroring NAs

Value

Matrix with increased NA values, maintaining symmetry

Examples

```
## Not run:
# Create sample distance matrix
dist_mat <- matrix(runif(100), 10, 10)
dist_mat[lower.tri(dist_mat)] <- t(dist_mat)[lower.tri(dist_mat)]
diag(dist_mat) <- 0

# Increase NAs to 70%
sparse_mat <- increase_na_percentage(dist_mat, 0.7)

## End(Not run)
```

`initial_parameter_optimization` *Run Parameter Optimization Via Latin Hypercube Sampling*

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

Usage

```
initial_parameter_optimization(
  distance_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 = NULL,
  time = "8:00:00",
  memory = "3G",
  use_slurm = FALSE,
  cider = FALSE
)
```

Arguments

`distance_matrix`
: Matrix or data frame. Input distance matrix. Must be square and symmetric. Can contain NA values for missing measurements.

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

`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 number of dimensions parameter.

`k0_min`, `k0_max` Numeric. Range for initial spring constant parameter.

`c_repulsion_min`, `c_repulsion_max`
: Numeric. Range for repulsion constant parameter.

`cooling_rate_min`, `cooling_rate_max`
: Numeric. Range for spring decay parameter.

`num_samples` Integer. Number of LHS samples to generate (default: 20).

`max_cores` Integer. Maximum number of cores to use for parallel processing. If NULL, uses all available cores minus 1 (default: NULL).

`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 CSV. Default: FALSE. Only set TRUE on SLURM

`output_dir` Character. Directory where output and temporary files will be saved. If NULL, uses current working directory. Directory will be created if it doesn't exist.

`time` Character. Walltime for SLURM jobs in HH:MM:SS format. Default: "8:00:00".

`memory` Character. Memory allocation for SLURM jobs. Default: "3G".

`use_slurm` Logical. Whether to submit jobs via SLURM. Default: FALSE.

`cider` Logical. Whether to use cider queue in SLURM. Default: FALSE.

Details

The function performs these steps:

1. Generates LHS samples in parameter space
2. Creates k-fold splits of input data
3. For each parameter set and fold:
   - Trains model on training set
   - Evaluates on validation set
   - Calculates MAE and negative log likelihood
4. Can run computation locally or distribute via SLURM

Parameters ranges are transformed to log scale where appropriate to handle different scales effectively.

Value

If write_files=FALSE, returns a data frame with columns:

`N` Number of dimensions used

`k0` Initial spring constant

`cooling_rate` Spring decay rate

c_repulsion	Repulsion constant
Holdout_MAE	Mean absolute error on validation sets
NLL	Negative log likelihood

If write_files=TRUE, results are saved to CSV files in the format: {scenario_name}_model_parameters.csv

See Also

create_topolow_map for the core optimization algorithm

Examples

```
## Not run:
# Generate sample distance matrix
dist_mat <- matrix(runif(100), 10, 10)
dist_mat[lower.tri(dist_mat)] <- t(dist_mat)[lower.tri(dist_mat)]
diag(dist_mat) <- 0

# Run local optimization with 50 samples
results <- initial_parameter_optimization(
  distance_matrix = dist_mat,
  mapping_max_iter = 1000,
  relative_epsilon = 1e-4,
  convergence_counter = 10,
  scenario_name = "test_opt",
  N_min = 2, N_max = 10,
  k0_min = 1, k0_max = 30,
  c_repulsion_min = 0.00001, c_repulsion_max = 0.2,
  cooling_rate_min = 0.00001, cooling_rate_max = 0.2,
  num_samples = 50,
  max_cores = 4  # Limit to 4 cores
)

# Run with SLURM using 100 samples
initial_parameter_optimization(
  distance_matrix = dist_mat,
  mapping_max_iter = 1000,
  scenario_name = "slurm_opt",
  N_min = 2, N_max = 10,
  num_samples = 100,
  use_slurm = TRUE
)

## End(Not run)
```

---

log_transform_parameters

Log Transform Parameter Samples

Description

Reads samples from a CSV file and log transforms specific parameters (N, k0, cooling_rate, c_repulsion) if they exist in the data. Handles validation and error checking.

Usage

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

Arguments

samples_file	Character. Path to CSV file containing samples
output_file	Character. Optional path for saving transformed data. If NULL, overwrites input file

Value

Data frame with log-transformed parameters

Examples

```
## Not run:
# Transform and save to new file
log_transform_parameters("input_samples.csv", "transformed_samples.csv")

# Transform and overwrite original
log_transform_parameters("samples.csv")

## End(Not run)
```

---

long_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

```
long_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.

Examples

```
## Not run:
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 <- long_to_matrix(data,
                      chnames = "antigen",
                      rnames = "serum",
                      values_column = "distance")

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

```
                              sort = TRUE)

## End(Not run)
```

`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

plotly object with interactive features

Examples

```
## Not run:
# Create sample data and plot
data <- data.frame(
  V1 = rnorm(100),
  V2 = rnorm(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")
)

## End(Not run)
```

`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 aesthetic_config object

`new_annotation_config` *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 annotation_config object

---

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

A dim_reduction_config object

---

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_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_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

A layout_config object

---

only_virus_vs_as *Filter matrix to only virus vs antiserum distances*

Description

Filter matrix to only virus vs antiserum distances

Usage

```
only_virus_vs_as(dist_matrix, selected_names)
```

Arguments

`dist_matrix`	Distance matrix
`selected_names`	Names of selected reference points

Value

Filtered distance matrix

`parameter_sensitivity_analysis` *Parameter Sensitivity Analysis*

Description

Analyzes the sensitivity of model performance (MAE) to changes in a parameter. Uses binning to identify the minimum MAE across parameter ranges and calculates thresholds for acceptable parameter values.

Usage

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

Arguments

`param`	Character name of parameter to analyze
`samples`	Data frame containing parameter samples and performance metrics
`bins`	Integer number of bins for parameter range (default: 40)
`mae_col`	Character name of column containing MAE values (default: "Holdout_MAE")
`threshold_pct`	Numeric percentage above minimum for threshold calculation (default: 5)
`min_samples`	Integer minimum number of samples required in a bin (default: 1)

Details

The function performs these steps:

1. Cleans the input data using MAD-based outlier detection
2. Bins the parameter values into equal-width bins
3. Calculates the minimum MAE within each bin. Analogous to "poorman's likelihood" approach, minimum MAE within each bin is an empirical estimate of the performance surface at this parameter value when other parameters are at their optimal values.
4. Identifies a threshold of acceptable performance (default: Topolow min. +5% MAE)
5. Returns an object for visualization and further analysis

Value

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: Topolow min. +5%)
`min_value`	Minimum MAE value across all bins
`sample_counts`	Number of samples per bin

`plot.parameter_sensitivity` *Plot Method for Parameter Sensitivity Analysis*

Description

Creates a visualization of parameter sensitivity showing minimum MAE values across parameter ranges with threshold indicators.

Usage

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

Arguments

`x`	A parameter_sensitivity object
`width`	Numeric width of output plot in inches (default: 3.5)
`height`	Numeric height of output plot in inches (default: 3.5)
`save_plot`	Logical. Whether to save plot to file. Default: TRUE
`output_dir`	Character. Directory for output files. If NULL, uses current directory
`y_limit_factor`	Numeric. Factor to set the upper y-axis limit as a percentage above the threshold value (e.g., 1.10 for 10% above). Default: NULL (automatic scaling)
`...`	Additional arguments passed to plot

Value

A ggplot object

plot.profile_likelihood *Plot Method for Profile Likelihood Objects*

Description

Creates a visualization of profile likelihood for a parameter showing maximum likelihood estimates and confidence intervals. Supports mathematical notation for parameter names and configurable output settings.

Confidence interval is found using the likelihood ratio test: $LR(\theta_{ij}) = -2[logL_{max}(\theta_{ij}) - logL_{max}(\hat{\theta})]$ where $\hat{\theta}$ is the maximum likelihood estimate for all parameters. The 95% confidence interval is: $\{\theta_{ij} : LR(\theta_{ij}) \leq \chi^2_{1,0.05} = 3.84\}$

Usage

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

Arguments

x	A profile_likelihood object
LL_max	Numeric maximum log-likelihood value
width	Numeric width of output plot in inches (default: 3.5)
height	Numeric height of output plot in inches (default: 3.5)
save_plot	Logical. Whether to save plot to file. Default: TRUE
output_dir	Character. Directory for output files. If NULL, uses current directory
...	Additional arguments passed to plot

Value

A ggplot object

Examples

```
## Not run:
# Calculate profile likelihood
pl_result <- profile_likelihood("log_N", mcmc_samples)

# Plot with maximum likelihood from samples
LL_max <- max(-samples$NLL)
plot(pl_result, LL_max, width = 4, height = 3)

## End(Not run)
```

plot.topolow_amcs_diagnostics *Plot Method for Adaptive Monte Carlo Sampling Diagnostics*

Description

Creates trace and density plots for multiple chains to assess convergence and mixing.

Usage

```
## S3 method for class 'topolow_amcs_diagnostics'
plot(
  x,
  output_file = "mc_diagnostics.png",
  width = 3000,
  height = 3000,
  res = 300,
  ...
)
```

Arguments

x	A topolow_amcs_diagnostics object
output_file	Character path for saving plot
width, height, res	Plot dimensions and resolution
...	Additional arguments passed to plot functions

Value

Invisible NULL, saves plot to file

plot.topolow_convergence *Plot Method for Convergence Diagnostics*

Description

Creates visualization of convergence diagnostics from Monte Carlo sampling, including parameter mean trajectories and covariance matrix stability over iterations. Helps assess whether parameter estimation has converged to stable distributions.

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 to use in plot titles. If NULL (default), uses the param_names from the topolow_convergence object.
...	Additional arguments passed to underlying plot functions (currently not 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 of parameter distributions.
2. Covariance change plot: Shows relative changes in the covariance matrix using the Frobenius norm (also called Hilbert-Schmidt norm), which is defined as the square root of the sum of the absolute squares of all matrix elements: $\sqrt{\sum |a_{ij}|^2}$. 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

```
## Not run:
# Example with simulated data
chain_data <- data.frame(
  log_N = rnorm(1000, mean = 1.5, sd = 0.1),
  log_k0 = 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("Dimensions (log)", "Spring constant (log)"))

## End(Not run)
```

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 = NULL
)
```

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. If NULL, uses current directory

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 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

```
## Not run:
# Create sample data
set.seed(123)
data <- data.frame(
  V1 = rnorm(100),
  V2 = rnorm(100),
  V3 = rnorm(100),
  V4 = rnorm(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)
)

# Basic interactive plot
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
plot_3d_mapping(data, ndim=4,
  aesthetic_config = aesthetic_config,
  layout_config = layout_config,
  interactive = FALSE
)

# Dimension reduction with UMAP
dim_config <- new_dim_reduction_config(
  method = "umap",
  n_components = 3,
```

```
  umap_params = list(
    n_neighbors = 20,
    min_dist = 0.2
  )
)

plot_3d_mapping(data, ndim=4,
  dim_config = dim_config,
  interactive = TRUE
)

## End(Not run)
```

`plot_cluster_mapping` *Create Clustered Mapping Plots*

Description

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 = NULL,
  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. If NULL, uses current directory

`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

ggplot object containing the cluster mapping visualization

See Also

plot_temporal_mapping for temporal visualization plot_3d_mapping for 3D visualization plot_combined for creating multiple visualizations 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

```
## Not run:
# Basic usage with default configurations
data <- data.frame(
  V1 = rnorm(100),
  V2 = rnorm(100),
  V3 = rnorm(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)

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

p2 <- plot_cluster_mapping(
  data,
  ndim = 3,
  aesthetic_config = aesthetic_config,
  layout_config = layout_config
)

## End(Not run)
```

---

plot_combined *Create Combined Visualization*

Description

Creates multiple coordinated visualizations of the same data using different methods and arrangements. Supports combining temporal, cluster, and 3D visualizations in flexible layouts.

Usage

```
plot_combined(
  df_coords,
  ndim,
  plot_types = c("temporal", "cluster"),
  dim_config = new_dim_reduction_config(),
  aesthetic_config = new_aesthetic_config(),
  layout_config = new_layout_config(),
  arrange = "grid",
  output_dir = 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: (Optional) Factor or integer cluster assignments - year: (Optional) Numeric year values for temporal coloring
ndim	Number of dimensions in input coordinates
plot_types	Vector of plot types to create ("temporal", "cluster", "3d")
dim_config	Dimension reduction configuration object
aesthetic_config	Aesthetic configuration object
layout_config	Layout configuration object
arrange	How to arrange multiple plots ("grid", "vertical", "horizontal")
output_dir	Character. Directory for output files. If NULL, uses current directory

Details

This function provides a high-level interface for creating multiple coordinated views of the same data. It supports:

Plot Types:

- temporal: Time-based color gradients
- cluster: Cluster-based discrete colors
- 3d: Three-dimensional interactive or static views (requires rgl package)

Arrangement Options:

- grid: Automatic square-like arrangement
- vertical: Plots stacked vertically
- horizontal: Plots arranged horizontally

All plots share consistent:

- Color schemes

- Point styles
- Axis scales
- Theme elements

Note: If "3d" is specified but the rgl package is not available, the function will skip the 3D plot and display a message.

Value

Combined plot object (grid arrangement of plots)

See Also

plot_temporal_mapping for individual temporal plots plot_cluster_mapping for individual cluster plots plot_3d_mapping for individual 3D plots make_interactive for creating interactive versions save_plot for saving plots to files

Examples

```
## Not run:
# Create sample data
set.seed(123)
data <- data.frame(
  V1 = rnorm(100),
  V2 = rnorm(100),
  V3 = rnorm(100),
  V4 = rnorm(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)
)

# Basic combined plot
p1 <- plot_combined(data, ndim=4,
  plot_types = c("temporal", "cluster")
)

# Advanced configuration
dim_config <- new_dim_reduction_config(
  method = "umap",
  n_components = 2,
  scale = TRUE,
  umap_params = list(
    n_neighbors = 15,
    min_dist = 0.1
  )
)

aesthetic_config <- new_aesthetic_config(
  point_size = 3,
  point_alpha = 0.7,
  point_shapes = c(antigen = 17, antiserum = 1),
  gradient_colors = list(
    low = "navy",
    high = "red"
```

```
  ),
  show_labels = TRUE,
  label_size = 3
)

layout_config <- new_layout_config(
  width = 12,
  height = 8,
  aspect_ratio = 1,
  show_grid = TRUE,
  grid_type = "major",
  background_color = "white",
  panel_border = TRUE
)

# Create comprehensive visualization
p2 <- plot_combined(data, ndim=4,
  plot_types = c("temporal", "cluster", "3d"),
  dim_config = dim_config,
  aesthetic_config = aesthetic_config,
  layout_config = layout_config,
  arrange = "grid"
)

# Save combined plot
save_plot(p2, "combined_visualization.pdf")

# Create interactive versions
p3 <- plot_combined(data, ndim=4,
  plot_types = c("temporal", "cluster"),
  arrange = "horizontal"
)

p3_interactive <- make_interactive(p3,
  tooltip_vars = c("year", "cluster", "antigen")
)

# Example with different layouts
# Vertical arrangement
p4 <- plot_combined(data, ndim=4,
  plot_types = c("temporal", "cluster", "3d"),
  arrange = "vertical"
)

# Horizontal arrangement with temporal and cluster only
p5 <- plot_combined(data, ndim=4,
  plot_types = c("temporal", "cluster"),
  arrange = "horizontal"
)

# Grid arrangement with custom layout
layout_config$width <- 15
layout_config$height <- 15
p6 <- plot_combined(data, ndim=4,
  plot_types = c("temporal", "cluster", "3d"),
  layout_config = layout_config,
  arrange = "grid"
```

```
)

# Example workflow for publication-quality figures
# 1. Create base visualization
p7 <- plot_combined(data, ndim=4,
  plot_types = c("temporal", "cluster")
)

# 2. Customize for publication
layout_config <- new_layout_config(
  width = 8,
  height = 6,
  dpi = 600,
  save_format = "pdf",
  background_color = "white",
  panel_border = TRUE,
  grid_type = "major"
)

# 3. Save high-resolution version
save_plot(p7, "publication_figure.pdf", layout_config)

## End(Not run)
```

`plot_distance_heatmap` *Plot Distance Matrix Heatmap*

Description

Creates heatmap visualization of distance matrix showing patterns and structure in the measurements.

Usage

```
plot_distance_heatmap(
  heatmap_data,
  scenario_name,
  aesthetic_config = new_aesthetic_config(),
  layout_config = new_layout_config()
)
```

Arguments

`heatmap_data`	List output from prepare_heatmap_data()
`scenario_name`	Character string for output file naming
`aesthetic_config`	Plot aesthetic configuration object
`layout_config`	Plot layout configuration object

Value

A ggplot object containing:

- Heatmap visualization of the distance matrix
- Color gradient representing distance values
- Title showing matrix completeness percentage

Examples

```
## Not run:
# Basic heatmap
hmap_data <- prepare_heatmap_data(dist_mat)
p <- plot_distance_heatmap(hmap_data, "scenario1")

# Heatmap with clustering
hmap_data <- prepare_heatmap_data(dist_mat, cluster_rows = TRUE)
p2 <- plot_distance_heatmap(hmap_data, "scenario1")

# Custom configuration
aesthetic_config <- new_aesthetic_config(
  gradient_colors = list(low = "navy", high = "red")
)
p3 <- plot_distance_heatmap(hmap_data, "scenario1",
                            aesthetic_config = aesthetic_config)

## End(Not run)
```

---

plot_network_structure

Plot Network Structure Analysis

Description

Creates visualization of distance matrix network structure showing data availability patterns and connectivity.

Usage

```
plot_network_structure(
  network_results,
  scenario_name,
  aesthetic_config = new_aesthetic_config(),
  layout_config = new_layout_config()
)
```

Arguments

network_results	List output from analyze_network_structure()
scenario_name	Character string for output file naming
aesthetic_config	Plot aesthetic configuration object
layout_config	Plot layout configuration object

Value

ggplot object

Examples

```
## Not run:
net_analysis <- analyze_network_structure(dist_mat)
p <- plot_network_structure(net_analysis, "scenario1")

## End(Not run)
```

`plot_temporal_mapping` *Create Temporal Mapping Plot*

Description

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 = NULL,
  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. If NULL, uses current directory

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

ggplot object containing the temporal mapping visualization

See Also

plot_cluster_mapping for cluster-based visualization plot_3d_mapping for 3D visualization plot_combined for creating multiple visualizations 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

```
## Not run:
# Basic usage with default configurations
data <- data.frame(
  V1 = rnorm(100),
  V2 = rnorm(100),
```

```
  V3 = rnorm(100),
  antigen = rep(c(0,1), 50),
  antiserum = rep(c(1,0), 50),
  year = rep(2000:2009, each=10)
)
# Default axis limits
p1 <- plot_temporal_mapping(data, ndim=3)

# Custom axis limits via layout configuration
layout_config <- new_layout_config(
  x_limits = c(-10, 10),
  y_limits = c(-8, 8)
)
p2 <- plot_temporal_mapping(data, ndim=3,
                            layout_config=layout_config)

## End(Not run)
```

`prepare_heatmap_data` *Generate Distance Matrix Heatmap Data*

Description

Prepares distance matrix data for heatmap visualization by handling missing values and calculating relevant statistics.

Usage

```
prepare_heatmap_data(
  distance_matrix,
  cluster_rows = FALSE,
  cluster_cols = FALSE
)
```

Arguments

`distance_matrix`	Square symmetric matrix of distances
`cluster_rows`	Logical; whether to cluster rows
`cluster_cols`	Logical; whether to cluster columns

Value

List containing:

`matrix_data`	Processed matrix for visualization
`row_order`	Optional row ordering from clustering
`col_order`	Optional column ordering from clustering
`stats`	List of matrix statistics

Examples

```
## Not run:
heatmap_data <- prepare_heatmap_data(dist_mat)
print(heatmap_data$stats$completeness)

## End(Not run)
```

---

print.parameter_sensitivity *Print Method for Parameter Sensitivity Objects*

Description

Print Method for Parameter Sensitivity Objects

Usage

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

Arguments

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

---

print.profile_likelihood *Print Method for Profile Likelihood Objects*

Description

Print Method for Profile Likelihood Objects

Usage

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

Arguments

x	Profile likelihood object
...	Additional arguments passed to print

print.topolow *Print method for topolow objects*

Description

Provides a concise display of key optimization results including dimensions, iterations, error metrics and convergence status.

Usage

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

Arguments

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

Examples

```
## Not run:
dist_mat <- matrix(c(0, 2, 3, 2, 0, 4, 3, 4, 0), nrow=3)
result <- create_topolow_map(dist_mat, ndim=2, mapping_max_iter=100,
                             k0=1.0, cooling_rate=0.001, c_repulsion=0.1)
print(result)

## End(Not run)
```

print.topolow_amcs_diagnostics *Print Method for Adaptive Monte Carlo Sampling Diagnostics*

Description

Print Method for Adaptive Monte Carlo Sampling Diagnostics

Usage

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

Arguments

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

print.topolow_convergence

Print Method for Convergence Diagnostics

Description

Print Method for Convergence Diagnostics

Usage

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

Arguments

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

process_antigenic_data

Process Raw Antigenic Assay Data

Description

Processes raw antigenic assay data from CSV files into standardized long and matrix formats. Handles both titer data (which needs conversion to distances) and direct distance measurements like IC50. Preserves threshold indicators (<, >) and handles repeated measurements by averaging.

Usage

```
process_antigenic_data(
  file_path,
  antigen_col,
  serum_col,
  value_col,
  is_titer = TRUE,
  metadata_cols = NULL,
  id_prefix = FALSE,
  base = NULL,
  scale_factor = 10
)
```

Arguments

file_path	Character. Path to CSV file 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_titer	Logical. Whether values are titers (TRUE) or distances like IC50 (FALSE).
metadata_cols	Character vector. Names of additional columns to preserve.
id_prefix	Logical. Whether to prefix IDs with V/ and S/ (default: TRUE).
base	Numeric. Base for logarithm transformation (default: 2 for titers, e for IC50).
scale_factor	Numeric. Scale factor for titers (default: 10).

Details

The function handles these key steps:

1. Reads and validates input data
2. Transforms values to log scale
3. Converts titers to distances if needed
4. Averages repeated measurements
5. Creates standardized long format
6. Creates distance matrix
7. Preserves metadata and threshold indicators
8. Preserves virusYear and serumYear columns if present

Input requirements and constraints:

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

Value

List containing:

long	Data frame in long format with standardized columns
matrix	Distance matrix

Examples

```
## Not run:
# Process titer data (e.g., HI assay)
results <- process_antigenic_data(
  "smith2004.csv",
  antigen_col = "virusStrain",
  serum_col = "serumStrain",
  value_col = "titer",
  is_titer = TRUE,
  metadata_cols = c("cluster", "color")
)

# Process IC50 data
results <- process_antigenic_data(
  "hiv_assays.csv",
  antigen_col = "Virus",
```

```
  serum_col = "Antibody",
  value_col = "IC50",
  is_titer = FALSE
)

## End(Not run)
```

`process_antigenic_data_notransform` *Process Raw Antigenic Assay Data without transformations*

Description

Processes raw antigenic assay data from CSV files into standardized long and matrix formats. Handles both titer data (which needs conversion to distances) and direct distance measurements like IC50. Preserves threshold indicators (<, >) and handles repeated measurements by averaging.

Usage

```
process_antigenic_data_notransform(
  file_path,
  antigen_col,
  serum_col,
  value_col,
  is_titer = TRUE,
  metadata_cols = NULL,
  id_prefix = FALSE,
  base = NULL,
  scale_factor = 10
)
```

Arguments

`file_path`	Character. Path to CSV file 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_titer`	Logical. Whether values are titers (TRUE) or distances like IC50 (FALSE).
`metadata_cols`	Character vector. Names of additional columns to preserve.
`id_prefix`	Logical. Whether to prefix IDs with V/ and S/ (default: TRUE).
`base`	Numeric. Base for logarithm transformation (default: 2 for titers, e for IC50).
`scale_factor`	Numeric. Scale factor for titers (default: 10).

Details

The function handles these key steps:

1. Reads and validates input data
2. Transforms values to log scale

3. Converts titers to distances if needed
4. Averages repeated measurements
5. Creates standardized long format
6. Creates distance matrix
7. Preserves metadata and threshold indicators
8. Preserves virusYear and serumYear columns if presen

Input requirements and constraints:

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

Value

List containing:

`long`	Data frame in long format with standardized columns
`matrix`	Distance matrix

Examples

```
## Not run:
# Process titer data (e.g., HI assay)
results <- process_antigenic_data(
  "smith2004.csv",
  antigen_col = "virusStrain",
  serum_col = "serumStrain",
  value_col = "titer",
  is_titer = TRUE,
  metadata_cols = c("cluster", "color")
)

# Process IC50 data
results <- process_antigenic_data(
  "hiv_assays.csv",
  antigen_col = "Virus",
  serum_col = "Antibody",
  value_col = "IC50",
  is_titer = FALSE
)

## End(Not run)
```

`profile_likelihood` *Profile Likelihood Analysis*

Description

Calculates profile likelihood for a parameter by evaluating conditional maximum likelihood across a grid of parameter values. Uses local sample windowing to estimate conditional likelihoods. This implementation is not a classical profile likelihood calculation, but rather an "empirical profile likelihood" which estimates the profile likelihood at each point based on the many observations previously sampled in 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`	Character name of parameter to analyze
`samples`	Data frame containing parameter samples and log-likelihoods
`grid_size`	Integer number of grid points (default: 48)
`bandwidth_factor`	Numeric factor for local sample window (default: 0.03)
`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 bandwidth window
2. Calculates conditional maximum likelihood from these samples
3. Tracks sample counts to assess estimate reliability
4. Handles boundary conditions and sparse regions

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

plot.profile_likelihood for visualization

Examples

```
## Not run:
# Calculate profile likelihood for parameter "log_N"
pl <- profile_likelihood("log_N", mcmc_samples,
                         grid_size = 60,
                         bandwidth_factor = 0.02)

# Plot results
plot(pl)

## End(Not run)
```

---

prune_distance_network

Prune Distance Data for Network Quality

Description

Iteratively removes viruses and antibodies with insufficient connections to create a well-connected network subset. The pruning continues until all remaining points have at least the specified minimum number of connections.

Usage

```
prune_distance_network(
  data,
  virus_col,
  antibody_col,
  min_connections,
  iterations = 100
)
```

Arguments

data	Data frame in long format containing: - Column for viruses/antigens - Column for antibodies/antisera - Distance measurements (can contain NAs) - Optional metadata columns
virus_col	Character name of virus/antigen column
antibody_col	Character name of antibody/antiserum column
min_connections	Integer minimum required connections per point
iterations	Integer maximum pruning iterations (default 100)

Value

List containing:

pruned_data	Data frame of pruned measurements
stats	List of pruning statistics including:

- original_points: Number of points before pruning
- remaining_points: Number of points after pruning
- iterations: Number of pruning iterations performed
- min_connections: Minimum connections in final set

Examples

```
## Not run:
# Basic pruning keeping points with at least 10 connections
pruned <- prune_distance_network(hiv_data,
                                 virus_col = "Virus",
                                 antibody_col = "Antibody",
                                 min_connections = 10)

# Check pruning statistics
print(pruned$stats)

## End(Not run)
```

---

run_adaptive_sampling *Run Adaptive Monte Carlo Sampling*

Description

Performs adaptive Monte Carlo sampling to explore parameter space, either locally or distributed via SLURM. Samples are drawn adaptively based on previous evaluations to focus sampling in high-likelihood regions. Results from all jobs accumulate in a single output file.

Usage

```
run_adaptive_sampling(
  initial_samples_file,
  scenario_name,
  distance_matrix,
  num_parallel_jobs = 5,
  max_cores = NULL,
  num_samples = 10,
  mapping_max_iter = 1000,
  relative_epsilon = 1e-04,
  folds = 20,
  time = "8:00:00",
  memory = "10G",
  output_dir = NULL,
  use_slurm = FALSE,
  cider = FALSE,
  verbose = FALSE
)
```

Arguments

initial_samples_file	Character. Path to CSV file containing initial samples. Must contain columns: log_N, log_k0, log_cooling_rate, log_c_repulsion, NLL
scenario_name	Character. Name for output files.
distance_matrix	Matrix. Distance matrix of the input data.
num_parallel_jobs	Integer. Number of parallel jobs (cores on local machine or SLURM jobs).
max_cores	Integer. Maximum number of cores to use for parallel processing. If NULL, uses all available cores minus 1 (default: NULL).
num_samples	Integer. Number of new samples to be added to the CSV file containing initial samples through Adaptive Monte Carlo sampling (default: 10).
mapping_max_iter	Integer. Maximum iterations per map optimization.
relative_epsilon	Numeric. Convergence threshold.
folds	Integer. Number of CV folds (default: 10).
time	Character. Walltime for SLURM jobs in HH:MM:SS format. Default: "8:00:00".
memory	Character. Memory allocation for SLURM jobs. Default: "10G".
output_dir	Character. Directory for output files. If NULL, uses current directory.
use_slurm	Logical. Whether to use SLURM (default: FALSE).
cider	Logical. Whether to use cider queue (default: FALSE).
verbose	Logical. Whether to print progress messages. Default: FALSE.

Value

NULL. Results are written to: model_parameters/{scenario_name}_model_parameters.csv

---

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 = NULL
)
```

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. If NULL, uses current directory

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

Invisible NULL

Examples

```
## Not run:
# Create sample plot
data <- data.frame(
  V1 = rnorm(100),
  V2 = rnorm(100),
  antigen = rep(c(0,1), 50),
  antiserum = rep(c(1,0), 50),
  year = rep(2000:2009, each=10)
)
p <- plot_temporal_mapping(data, ndim=2)

# Basic save
save_plot(p, "temporal_plot.png")

# Save with custom layout
layout_config <- new_layout_config(
  width = 12,
  height = 8,
  dpi = 600,
  save_format = "pdf"
)

save_plot(p, "high_res_plot", layout_config)

# Save 3D plot
```

```
p3d <- plot_3d_mapping(data, ndim=3, interactive=FALSE)
save_plot(p3d, "3d_plot.png", layout_config)

## End(Not run)
```

`scatterplot_fitted_vs_true` *Plot Fitted vs True Distances*

Description

Creates diagnostic plots comparing fitted distances from a model against true distances. Generates both a scatter plot with prediction intervals and a residuals plot.

Usage

```
scatterplot_fitted_vs_true(
  distance_matrix,
  p_dist_mat,
  scenario_name = NA,
  ndim = NA,
  save_plot = TRUE,
  output_dir = NULL,
  confidence_level = 0.95
)
```

Arguments

`distance_matrix`	Matrix of true distances
`p_dist_mat`	Matrix of predicted/fitted distances
`scenario_name`	Character string for output file naming
`ndim`	Integer number of dimensions used in the model
`save_plot`	Logical. Whether to save plots to files. Default: TRUE
`output_dir`	Character. Directory for output files. If NULL, uses current directory
`confidence_level`	Numeric confidence level for prediction intervals (default: 0.95)

Value

Invisibly returns NULL, creates two plot files:

- {scenario_name}*prediction_scatter_dim*{ndim}.png
- {scenario_name}*residuals_vs_fitted_dim*{ndim}.png

Examples

```
## Not run:
# Create scatter and residual plots
scatterplot_fitted_vs_true(truth_matrix, predicted_matrix,
                           scenario_name = "example",
                           ndim = 5)

## End(Not run)
```

`submit_job` *Submit Job to SLURM or Run Locally*

Description

Submits a job to SLURM if available, otherwise runs locally. Provides consistent interface for both execution modes.

Usage

```
submit_job(script_file, use_slurm = TRUE, cider = FALSE)
```

Arguments

`script_file`	Path to script file
`use_slurm`	Logical; whether to use SLURM if available
`cider`	Logical; whether to use cider_qos queue

Value

Exit status code (invisible)

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

Description

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

Usage

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

Arguments

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

Examples

```
## Not run:
dist_mat <- matrix(c(0, 2, 3, 2, 0, 4, 3, 4, 0), nrow=3)
result <- create_topolow_map(dist_mat, ndim=2, mapping_max_iter=100,
                             k0=1.0, cooling_rate=0.001, c_repulsion=0.1)
summary(result)

## End(Not run)
```

`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

Matrix in assay panel format

`unweighted_kde` *Unweighted Kernel Density Estimation*

Description

Standard kernel density estimation for univariate data with various bandwidth selection rules.

Usage

```
unweighted_kde(x, n = 512, from = min(x), to = max(x), bw = "nrd0")
```

Arguments

`x`	Numeric vector of samples
`n`	Integer number of evaluation points
`from, to`	Numeric range for evaluation points
`bw`	Bandwidth selection ("nrd0", "nrd", "ucv", "bcv", "sj" or numeric)

Value

List containing:

x	Vector of evaluation points
y	Vector of density estimates
bw	Selected bandwidth

`weighted_kde` *Weighted Kernel Density Estimation*

Description

Performs weighted kernel density estimation for univariate data. Useful for analyzing parameter distributions with importance weights.

Usage

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

Arguments

x	Numeric vector of samples
weights	Numeric vector of weights
n	Integer number of evaluation points
from, to	Numeric range for evaluation points

Value

List containing:

x	Vector of evaluation points
y	Vector of density estimates

Index