Package 'topolow'

July 1, 2025

Title Antigenic Mapping and Antigenic Velocity Algorithm

Version 1.0.0

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) <https://doi.org/10.1093/bioinformatics/btaf372>.

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. This is an internal function called by run_adaptive_sampling.

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,
  verbose = FALSE
)
```

Arguments

samples_file	Path to CSV with initial samples for this job.
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
verbose	Logical. Whether to print progress messages. Default: FALSE

Value

A data.frame containing all samples (initial and newly generated) with their parameters and evaluated performance metrics. The data frame includes columns for the log-transformed parameters, Holdout_MAE, and NLL. Returns NULL if the results file was not created.

`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

A `list` containing three noisy `matrix` objects:

`n1`	Matrix with the first realization of random Gaussian noise.
`n2`	Matrix with a second, different realization of random Gaussian noise.
`nb`	Matrix with both random noise and a systematic negative bias.

Examples

```
# 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)
```

`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

A list containing the network analysis results:

`adjacency`	A logical `matrix` where `TRUE` indicates a measured distance between two points, representing the network's adjacency matrix.
`connectivity`	A `data.frame` with node-level metrics, including the `completeness` (degree) for each point.
`summary`	A list of overall network statistics, including `n_points`, `n_measurements`, and total `completeness`.

Examples

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

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

calculate_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

A list containing year-over-year antigenic distance metrics:

dist_data	A data.frame where each row represents a point and its distance to the mean coordinate of the previous year.
summary	A list containing the overall_mean and overall_sd (standard deviation) of the annual distances across all years.

Examples

```
# Create sample coordinate data
coords <- data.frame(V1 = rnorm(10), V2 = rnorm(10), year = rep(2000:2004, 2))
annual_stats <- calculate_annual_distances(coords, ndim=2)
print(annual_stats$summary$overall_mean)
```

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

A list containing the calculated distance metrics. The content of the list depends on the `reference_row` parameter.

- If `reference_row` is specified, the list contains `summary_data`: a `data.frame` with distances from the specified reference point to all other points, summarized by season and cluster. The columns include `season_num`, `cluster`, `color`, `avg_euclidean_dist`, `count`, `total_count`, and `fraction`.
- If `reference_row` is `FALSE`, the list contains `dist_data`: a `data.frame` with all unique pairwise distances. The columns include `year_diff`, `euclidean_dist`, and `ref_year`.

Examples

```
# Create sample coordinate data
coords <- data.frame(V1 = rnorm(10), V2 = rnorm(10), year = rep(2000:2004, 2),
                     season = paste0(rep(2000:2004, 2), "-S"),
                     cluster = factor(rep(1:2, 5)), color = "blue")

# 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)
```

`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

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

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

Examples

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

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

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

calculate_prediction_interval

Calculate prediction interval for 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

A single numeric value representing the margin of error for the 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

A single numeric value representing the sum of squared differences after Procrustes transformation.

Examples

```
# Create sample map data
map1 <- data.frame(name = letters[1:10], X = rnorm(10), X.1 = rnorm(10))
map2 <- data.frame(name = letters[1:10], X = rnorm(10), X.1 = rnorm(10))
diff <- calculate_procrustes_difference(map1, map2)
```

---

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

A single numeric p-value from the Procrustes permutation test.

Examples

```
# Create sample map data
map1 <- data.frame(name = letters[1:10], X = rnorm(10), X.1 = rnorm(10))
map2 <- data.frame(name = letters[1:10], X = rnorm(10), X.1 = rnorm(10))
p_val <- calculate_procrustes_significance(map1, map2)
```

---

calculate_weighted_marginals

Calculate Weighted Marginal Distributions

Description

Calculates marginal distributions for each parameter with weights derived from log-likelihoods.

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.

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

An object of class `topolow_convergence` containing diagnostics about the convergence of the multivariate samples. This list includes:

converged	A logical flag, TRUE if both mean and covariance have converged.
mean_converged	A logical flag, TRUE if the mean vector has converged.
cov_converged	A logical flag, TRUE if the covariance matrix has converged.
final_mean	The mean vector calculated from the last window_size samples.
final_cov	The covariance matrix calculated from the last window_size samples.
mean_history	A matrix tracking the history of the running mean of each parameter.
cov_changes	A numeric vector of the relative changes in the Frobenius norm of the covariance matrix.
param_names	The names of the parameters (columns) from the input data.

Examples

```
# Assuming 'chain_data' is a data frame with samples
chain_data <- as.data.frame(matrix(rnorm(500 * 4), ncol = 4))
colnames(chain_data) <- c("log_N", "log_k0", "log_cooling_rate", "log_c_repulsion")
conv_results <- check_gaussian_convergence(chain_data)
print(conv_results)  # Shows summary
# The plot method for this object would create convergence plots.
# plot(conv_results)
```

`clean_data` *Clean Data by Removing MAD-based Outliers*

Description

Removes outliers from numeric data using the Median Absolute Deviation 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

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

See Also

`detect_outliers_mad` for the underlying outlier detection

Examples

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

# 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; It can be a matrix or a data frame.

Value

A symmetric `matrix` of pairwise Euclidean distances between points.

`create_and_optimize_RACMACS_map` *Create and Optimize a RACMACS Map*

Description

Creates and optimizes an antigenic map using the RACMACS package and keeps the best optimization result. This function wraps common 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,
  output_file = NULL,
  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).
`output_file`	Character. An optional, full path (including filename and extension) where the map coordinates will be saved as a CSV file. If NULL (the default), the coordinates are not saved to a file.
`num_cores`	Integer number of cores to use for parallel optimization (default: 1).

Value

A `racmap` object from the `Racmacs` package, containing the optimized coordinates and other map data.

Examples

```
# Create a dummy titer table for the example
ag_names <- paste("V", 1:5)
sr_names <- paste("S", 1:4)
titer_table <- matrix(
  sample(c(10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120), 20, replace = TRUE),
  nrow = length(ag_names), ncol = length(sr_names),
  dimnames = list(ag_names, sr_names)
)

# Create and optimize map without saving coordinates
map_obj <- create_and_optimize_RACMACS_map(titer_table)

# Create map and save coordinates to a temporary file.
```

```
# tempfile() creates a path in the session's temporary directory.
temp_coords_file <- tempfile(fileext = ".csv")
map_obj_saved <- create_and_optimize_RACMACS_map(
  titer_table,
  dim = 3,
  optimization_number = 100,
  output_file = temp_coords_file
)

# Check that the file was created
file.exists(temp_coords_file)

# Clean up the temporary file
unlink(temp_coords_file)
```

`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

A list of length `n_folds`. Each element is a list containing two matrices:

`truth`	The truth matrix for that fold.
`train`	The training matrix with some values replaced by `NA` for validation.

Examples

```
# Create a sample distance matrix
dist_matrix <- matrix(runif(100), 10, 10)
diag(dist_matrix) <- 0
# Create 5-fold CV splits
folds <- create_cv_folds(dist_matrix, n_folds = 5, random_seed = 123)
```

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,
  save_plot = FALSE,
  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. Required if save_plot is TRUE.
output_dir	Character. Directory for output files. Required if save_plot is TRUE.
save_plot	Logical. Whether to save plots to files. Default: FALSE
width, height, res	Plot dimensions and resolution for saving

Value

A ggplot object of the combined plots.

Examples

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

# Only run the example if the files are found
if (all(nzchar(chain_files))) {
  # Create diagnostic plot without saving to a file
  create_diagnostic_plots(chain_files, mutual_size = 2, save_plot = FALSE)
}
```

`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 = FALSE,
  output_dir,
  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 FALSE

output_dir	Character. Directory to save CSV file. Required if write_positions_to_csv 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 object of class topolow. This list contains the results of the optimization and includes the following components:

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

Examples

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

# Run TopoLow in 2D without writing to a file
result <- create_topolow_map(dist_mat, ndim=2, mapping_max_iter=100,
                      k0=1.0, cooling_rate=0.001, c_repulsion=0.01, verbose=FALSE)
```

```
# results
head(result$positions)
```

`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

```
# 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)
```

`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

A list containing:

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

example_positions *Example Antigenic Mapping Data*

Description

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

Usage

```
example_positions
```

Format

A data frame with 285 rows and 11 variables:

V1 First dimension coordinate from 5D mapping

V2 Second dimension coordinate from 5D mapping

V3 Third dimension coordinate from 5D mapping

V4 Fourth dimension coordinate from 5D mapping

V5 Fifth dimension coordinate from 5D mapping

name Strain identifier

antigen Logical; TRUE if point represents an antigen

antiserum Logical; TRUE if point represents an antiserum

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

color Color assignment for visualization

year Year of strain isolation

Source

Smith et al., 2004

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

A data.frame with n_points rows and n_dim columns. Column names are "Dim1" through "DimN" where N is n_dim.

Examples

```
# Generate basic dataset
data <- generate_complex_data(n_points = 500, n_dim = 10,
                              n_clusters = 4, cluster_spread = 1)

# The function returns a data frame, which can be inspected
head(data)
```

`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. If `output_dir` is provided, the generated datasets are saved as RDS files.

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. If NULL (the default), no files are saved.
`prefix`	Character string to prefix output files (optional)
`save_plots`	Logical whether to save network visualization plots. Requires `output_dir` to be set.

Value

A list containing the generated synthetic data and metadata:

`matrices`	A list of generated symmetric distance matrices for each dimension.
`panels`	A list of generated assay panels (non-symmetric matrices) for each dimension.
`metadata`	A `data.frame` with the generation parameters for each dataset.

Examples

```
# Generate datasets without saving to disk
results <- generate_synthetic_datasets(
  n_dims_list = c(2, 3),
  seeds = c(123, 456),
  n_points = 50
)

# Generate datasets and save to a temporary directory
temp_out_dir <- tempdir()
results_saved <- generate_synthetic_datasets(
  n_dims_list = c(2),
  seeds = c(123),
  n_points = 10,
  missingness_levels = list(low=0.5, high=0.8),
  output_dir = temp_out_dir,
  save_plots = TRUE
)
list.files(temp_out_dir)
# Clean up the directory
unlink(temp_out_dir, recursive = TRUE)
```

---

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

A character vector of unique strings with year suffixes

ggsave_white_bg *Save ggplot with white background*

Description

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

Usage

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

Arguments

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

Value

No return value, called for side effects.

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

Description

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

Usage

```
h3n2_data
```

Format

A data frame with the following variables:

virusStrain Character. Virus strain identifier

serumStrain Character. Antiserum strain identifier

titer Numeric. HI assay titer value

virusYear Numeric. Year virus was isolated

serumYear Numeric. Year serum was collected

cluster Factor. Antigenic cluster assignment

color Character. Color code for visualization

Source

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

hiv_titers *HIV Neutralization Assay Data*

Description

IC50 neutralization measurements between HIV viruses and antibodies.

Usage

```
hiv_titers
```

Format

A data frame with the following variables:

Antibody Character. Antibody identifier

Virus Character. Virus strain identifier

IC50 Numeric. IC50 neutralization value

Source

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

hiv_viruses *HIV Virus Metadata*

Description

Reference information for HIV virus strains used in neutralization assays.

Usage

```
hiv_viruses
```

Format

A data frame with the following variables:

Virus.name Character. Virus strain identifier

Country Character. Country of origin

Subtype Character. HIV subtype

Year Numeric. Year of isolation

Source

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

`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

A `matrix` with an increased number of `NA` values, maintaining symmetry.

Examples

```
# 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)
```

`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
)
```

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.
output_dir	Character. Directory where output files will be saved. Required if write_files is TRUE.

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. Computations are run locally in parallel.

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

Value

A data.frame containing the parameter sets and their performance metrics (Holdout_MAE and NLL). The columns of the data frame are N, k0, cooling_rate, c_repulsion, Holdout_MAE, and NLL. If write_files is TRUE, this data frame is also saved to a CSV file as a side effect.

See Also

create_topolow_map for the core optimization algorithm

Examples

```
# This example is wrapped in \donttest{} because it can exceed 5 seconds,
# 1. Create a structured, synthetic dataset for the example
# Generate coordinates for a more realistic test case
synth_coords <- generate_complex_data(n_points = 20, n_dim = 3)
# Convert coordinates to a distance matrix
dist_mat <- coordinates_to_matrix(synth_coords)
```

```
# 2. Run the optimization on the synthetic data
# ensuring it passes CRAN's automated checks.
results <- initial_parameter_optimization(
  distance_matrix = dist_mat,
  mapping_max_iter = 100,
  relative_epsilon = 1e-3,
  convergence_counter = 2,
  scenario_name = "test_opt_synthetic",
  N_min = 2, N_max = 5,
  k0_min = 1, k0_max = 10,
  c_repulsion_min = 0.001, c_repulsion_max = 0.05,
  cooling_rate_min = 0.001, cooling_rate_max = 0.02,
  num_samples = 4,
  max_cores = 2,
  verbose = FALSE
)
print(results)
```

`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. If `output_file` is specified, the transformed data is saved. Otherwise, the transformed data frame is returned.

Usage

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

Arguments

`samples_file`	Character. Path to CSV file containing samples.
`output_file`	Character. Optional path (including filename) for saving transformed data as a CSV. If NULL (the default), the function returns the transformed data frame without writing a file.

Value

A `data.frame` with log-transformed parameters. If `output_file` is specified, the function also writes the data frame to the specified path and returns it invisibly.

Examples

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

# Ensure the file exists before running the example
if (nzchar(sample_file)) {
```

```
  # Transform the data from the sample file and return as a data frame
  transformed_data <- log_transform_parameters(sample_file, output_file = NULL)

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

`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. `NA` values represent unmeasured pairs.

Examples

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

# Basic conversion
mat <- 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)
```

`make_interactive` *Create Interactive Plot*

Description

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

Usage

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

Arguments

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

Details

The function enhances static plots by adding:

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

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

Value

A `plotly` object with interactive features.

Examples

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

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

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

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

`new_aesthetic_config` *Plot Aesthetic Configuration Class*

Description

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

Usage

```
new_aesthetic_config(
  point_size = 3.5,
  point_alpha = 0.8,
  point_shapes = c(antigen = 16, antiserum = 0),
  color_palette = c25,
  gradient_colors = list(low = "blue", high = "red"),
  show_labels = FALSE,
  show_title = FALSE,
  label_size = 3,
  title_size = 14,
  subtitle_size = 12,
  axis_title_size = 12,
  axis_text_size = 10,
  legend_text_size = 10,
  legend_title_size = 12,
  show_legend = TRUE,
  legend_position = "right",
  arrow_head_size = 0.2,
  arrow_alpha = 0.6
)
```

Arguments

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

Value

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

`new_annotation_config` *Plot Annotation Configuration Class*

Description

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

Usage

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

Arguments

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

Value

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

`new_dim_reduction_config` *Dimension Reduction Configuration Class*

Description

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

Usage

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

Arguments

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

Value

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

new_layout_config *Plot Layout Configuration Class*

Description

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

Usage

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

Arguments

width	Plot width in inches
height	Plot height in inches
dpi	Plot resolution
aspect_ratio	Plot aspect ratio
show_grid	Show plot grid
grid_type	Grid type ("none", "major", "minor", "both")
grid_color	Grid color
grid_linetype	Grid line type
show_axis	Show axes
axis_lines	Show axis lines
plot_margin	Plot margins in cm

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

Value

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

`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

A `matrix` of the same dimensions as the input, but with non-virus-vs-antiserum distances set to `NA`.

`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 = FALSE,
  output_dir,
  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: FALSE
output_dir	Character. Directory for output files. Required if save_plot is TRUE.
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 = FALSE, output_dir, ...)
```

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: FALSE
`output_dir`	Character. Directory for output files. Required if `save_plot` is TRUE.
`...`	Additional arguments passed to plot

Value

A ggplot object

Examples

```
# These examples take more than 5 seconds to run, so they are not run by default.
# Use parallel processing (the default) to speed up.

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

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

# Plot with maximum likelihood from samples
```

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

`plot.topolow_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_dir,
  output_file = "mc_diagnostics.png",
  save_plot = FALSE,
  width = 3000,
  height = 3000,
  res = 300,
  ...
)
```

Arguments

`x`	A topolow_amcs_diagnostics object
`output_dir`	Character. Directory for output files. Required if `save_plot` is `TRUE`.
`output_file`	Character path for saving plot
`save_plot`	Logical. Whether to save the plot.
`width, height, res`	Plot dimensions and resolution
`...`	Additional arguments passed to plot functions

Value

A ggplot object of the combined plots.

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

```
# 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)"))
```

`plot_3d_mapping` *Create 3D Visualization*

Description

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

Usage

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

Arguments

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

Details

The function supports two main visualization modes:

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

Color schemes are automatically selected based on available data:

- If cluster data is present: Uses discrete colors per cluster

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

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

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

Value

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

See Also

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

Examples

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

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

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

layout_config <- new_layout_config(
  width = 12,
  height = 12,
  background_color = "black",
  show_axis = TRUE
)
  # Create customized static plot and save it
plot_3d_mapping(data, ndim=4,
  aesthetic_config = aesthetic_config,
  layout_config = layout_config,
```

```
  interactive = FALSE, output_dir = temp_dir
)
  list.files(temp_dir)
  unlink(temp_dir, recursive = TRUE)
}
```

`plot_cluster_mapping` *Create Clustered Mapping Plots*

Description

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

Usage

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

Arguments

`df_coords`	Data frame containing: - V1, V2, ... Vn: Coordinate columns - antigen: Binary indicator for antigen points - antiserum: Binary indicator for antiserum points - cluster: Factor or integer cluster assignments
`ndim`	Number of dimensions in input coordinates
`dim_config`	Dimension reduction configuration object specifying method and parameters
`aesthetic_config`	Aesthetic configuration object controlling plot appearance
`layout_config`	Layout configuration object controlling plot dimensions and style. Use x_limits and y_limits in layout_config to set axis limits.
`annotation_config`	Annotation configuration object for labeling notable points

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

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

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

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

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

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

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

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

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

show_one_arrow_per_cluster
Shows only the largest antigenic velocity arrow in each cluster

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

Details

The function performs these steps:

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

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

Value

A ggplot object containing the cluster mapping visualization.

See Also

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

Examples

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

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

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

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

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

---

plot_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,
  output_file = NULL,
  aesthetic_config = new_aesthetic_config(),
```

```
  layout_config = new_layout_config()
)
```

Arguments

heatmap_data	List output from prepare_heatmap_data()
output_file	Character. Full path (including filename and extension) where the plot will be saved. If NULL, the plot is not saved.
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

```
# Create sample heatmap data
dist_mat <- matrix(rnorm(100), 10, 10)
hmap_data <- prepare_heatmap_data(dist_mat)

# Create and display the plot object
plot_distance_heatmap(hmap_data)
```

---

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,
  output_file = NULL,
  aesthetic_config = new_aesthetic_config(),
  layout_config = new_layout_config()
)
```

Arguments

network_results	List output from analyze_network_structure()
output_file	Character. Full path (including filename and extension) where the plot will be saved. If NULL, the plot is not saved.
aesthetic_config	Plot aesthetic configuration object
layout_config	Plot layout configuration object

Value

A ggplot object representing the network graph.

Examples

```
# Create sample network data
adj_mat <- matrix(sample(c(0,1), 25, replace=TRUE), 5, 5)
# Add row and column names, which are required by the analysis function
rownames(adj_mat) <- colnames(adj_mat) <- paste0("Point", 1:5)
# Ensure the matrix is symmetric for the analysis
adj_mat[lower.tri(adj_mat)] <- t(adj_mat)[lower.tri(adj_mat)]
diag(adj_mat) <- 0
net_analysis <- analyze_network_structure(adj_mat)

# Create plot and return the plot object
plot_network_structure(net_analysis)
```

---

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,
  show_shape_legend = TRUE,
  draw_arrows = FALSE,
  annotate_arrows = TRUE,
  phylo_tree = NULL,
  sigma_t = NULL,
```

```
  sigma_x = NULL,
  clade_node_depth = NULL
)
```

Arguments

df_coords	Data frame containing: - V1, V2, ... Vn: Coordinate columns - antigen: Binary indicator for antigen points - antiserum: Binary indicator for antiserum points - year: Numeric year values for temporal coloring
ndim	Number of dimensions in input coordinates
dim_config	Dimension reduction configuration object specifying method and parameters
aesthetic_config	Aesthetic configuration object controlling plot appearance
layout_config	Layout configuration object controlling plot dimensions and style. Use x_limits and y_limits in layout_config to set axis limits.
annotation_config	Annotation configuration object for labeling notable points
output_dir	Character. Directory for output files. Required if `layout_config$save_plot` is `TRUE`.
show_shape_legend	Logical. Whether to show the shape legend (default: TRUE)
draw_arrows	logical; if TRUE, compute and draw antigenic drift vectors
annotate_arrows	logical; if TRUE, show names of the points having arrows
phylo_tree	Optional; phylo object in Newick format. Does not need to be rooted. If provided, used to compute antigenic velocity arrows.
sigma_t	Optional; numeric; bandwidth for the Gaussian kernel discounting on time in years or the time unit of the data. If NULL, uses Silverman's rule of thumb.
sigma_x	Optional; numeric; bandwidth for the Gaussian kernel discounting on antigenic distancein antigenic units. If NULL, uses Silverman's rule of thumb.
clade_node_depth	Optional; integer; number of levels of parent nodes to define clades. Antigens from different clades will be excluded from the calculation antigenic velocity arrows. (Default: Automatically calculated mode of leaf-to-backbone distance of the tree)

Details

The function performs these steps:

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

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

Value

A ggplot object containing the temporal mapping visualization.

See Also

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

Examples

```
# Basic usage with default configurations
data <- data.frame(
  V1 = rnorm(100), V2 = rnorm(100), V3 = rnorm(100), name = 1:100,
  antigen = rep(c(0,1), 50), antiserum = rep(c(1,0), 50),
  year = rep(2000:2009, each=10)
)
# Plot without saving
p1 <- plot_temporal_mapping(data, ndim=3)

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

---

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

A list of data prepared for generating a heatmap of the distance matrix:

`matrix_data`	The distance `matrix`, potentially reordered by clustering.
`row_order`	An integer vector of the row indices after clustering. If `cluster_rows` is `FALSE`, this is the original order.
`col_order`	An integer vector of the column indices after clustering. If `cluster_cols` is `FALSE`, this is the original order.
`stats`	A list of summary statistics for the distance matrix, including `mean`, `sd`, `min`, `max`, and `completeness`.

Examples

```
# Create a sample distance matrix
dist_mat <- matrix(runif(25), 5, 5)

# Prepare data for a heatmap
heatmap_data <- prepare_heatmap_data(dist_mat)
print(heatmap_data$stats$completeness)
```

---

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

Value

The original profile_likelihood object (invisibly).

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)

Value

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

Examples

```
# Create a simple distance matrix and run the optimization
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=50,
                             k0=1.0, cooling_rate=0.001, c_repulsion=0.1,
                             verbose = FALSE)
# Print the result object
print(result)
```

---

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

Value

No return value, called for side effects (prints a summary to the console).

---

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

Value

No return value, called for side effects (prints a summary to the console).

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

A list containing two elements:

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

Examples

```
# Locate the example data file included in the package
file_path <- system.file("extdata", "example_titer_data.csv", package = "topolow")

# Check if the file exists before running the example
if (file.exists(file_path)) {
  # Process the example titer data
  results <- process_antigenic_data(
    file_path,
    antigen_col = "virusStrain",
    serum_col = "serumStrain",
    value_col = "titer",
    is_titer = TRUE,
    metadata_cols = c("cluster", "color")
  )

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

`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

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

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

# Print the results
print(pl)
```

`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

A list containing two elements:

`pruned_data` A `data.frame` containing only the measurements for the well-connected subset of points.

`stats` A `list` of pruning statistics including:

- `original_points`: Number of unique antigens and sera before pruning.
- `remaining_points`: Number of unique antigens and sera after pruning.
- `iterations`: Number of pruning iterations performed.
- `min_connections`: The minimum connection threshold used.
- `is_connected`: A logical indicating if the final network is fully connected.

Examples

```
# Create a sparse dataset with 12 viruses and 12 antibodies
viruses <- paste0("V", 1:12)
antibodies <- paste0("A", 1:12)
all_pairs <- expand.grid(Virus = viruses, Antibody = antibodies, stringsAsFactors = FALSE)

# Sample 70 pairs to create a sparse matrix
set.seed(42)
assay_data <- all_pairs[sample(nrow(all_pairs), 70), ]

# Ensure some viruses/antibodies are poorly connected for the example
assay_data <- assay_data[!(assay_data$Virus %in% c("V11", "V12")),]
assay_data <- assay_data[!(assay_data$Antibody %in% c("A11", "A12")),]

# Add back single connections for the poorly connected nodes
poor_connections <- data.frame(
  Virus = c("V11", "V1", "V12", "V2"),
  Antibody = c("A1", "A11", "A2", "A12")
)
assay_data <- rbind(assay_data, poor_connections)

# View connection counts before pruning
# Virus V11 and V12, and Antibody A11 and A12 have only 1 connection
table(assay_data$Virus)
table(assay_data$Antibody)

# Prune the network to keep only nodes with at least 2 connections
pruned_result <- prune_distance_network(
  data = assay_data,
  virus_col = "Virus",
  antibody_col = "Antibody",
  min_connections = 2
)

# View connection counts after pruning
# The poorly connected nodes have been removed
table(pruned_result$pruned_data$Virus)
table(pruned_result$pruned_data$Antibody)

# Check the summary statistics
print(pruned_result$stats)
```

`run_adaptive_sampling` *Run Adaptive Monte Carlo Sampling*

Description

Performs adaptive Monte Carlo sampling to explore parameter space, running locally in parallel. Samples are drawn adaptively based on previous evaluations to focus sampling in high-likelihood regions. Results from all parallel jobs accumulate in a single output file. This function always writes to the file system and therefore requires the `output_dir` argument.

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,
  output_dir,
  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 local jobs (chains) to run.
`max_cores`	Integer. Maximum number of cores to use for parallel processing across all jobs. 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: 20).
`output_dir`	Character. Directory for output job files. The project's working directory is a straightforward example. This argument is required.
`verbose`	Logical. Whether to print progress messages. Default: FALSE.

Value

No return value, called for side effects. The function writes the results of the adaptive sampling to a CSV file within the specified `output_dir`.

Examples

```
# 1. Locate the example initial samples file included with the package
initial_file <- system.file(
  "extdata", "initial_samples_example.csv",
  package = "topolow"
)

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

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

# 4. Run the adaptive sampling only if the example file is found
if (nzchar(initial_file)) {
  run_adaptive_sampling(
    initial_samples_file = initial_file,
    scenario_name = "adaptive_test_example",
    distance_matrix = dist_mat,
    output_dir = temp_out_dir,
    num_parallel_jobs = 2, # Use small values for a quick example
    num_samples = 2,
    verbose = FALSE
  )

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

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

`save_plot` *Save Plot to File*

Description

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

Usage

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

Arguments

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

Details

Supported file formats:

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

The function will:

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

Value

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

Examples

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

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

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

# --- Example 2: Save with custom layout ---
layout_config <- new_layout_config(
  width = 12,
  height = 8,
  dpi = 600,
  save_format = "pdf"
)
save_plot(p, "high_res_plot.pdf", layout_config, output_dir = temp_dir)
```

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

`scatterplot_fitted_vs_true` *Plot Fitted vs True 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,
  ndim,
  save_plot = FALSE,
  output_dir,
  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. Used if `save_plot` is TRUE.
`ndim`	Integer number of dimensions used in the model
`save_plot`	Logical. Whether to save plots to files. Default: TRUE
`output_dir`	Character. Directory for output files. Required if `save_plot` is `TRUE`.
`confidence_level`	Numeric confidence level for prediction intervals (default: 0.95)

Value

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

Examples

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

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

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

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

Description

Provides a 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)

Value

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

Examples

```
# Create a simple distance matrix and run the optimization
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=50,
                             k0=1.0, cooling_rate=0.001, c_repulsion=0.1,
                             verbose = FALSE)
# Summarize the result object
summary(result)
```

`symmetric_to_nonsymmetric_matrix`
Convert distance matrix to assay panel format

Description

Convert distance matrix to assay panel format

Usage

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

Arguments

dist_matrix	Distance matrix
selected_names	Names of reference points

Value

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

---

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. Uses parallel processing for efficiency. 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