Package: FastCCCR (via r-universe)

May 26, 2026

Type Package

Title Fast Cell-Cell Communication Analysis with Statistical Framework

Version 1.0.0

Date 2026-01-26

Description A high-performance R package for cell-cell communication (CCC) analysis in single-cell RNA sequencing data. FastCCCR implements an innovative statistical framework based on exact null distribution computation, Cauchy combination of multiple statistical methods, and reference-based inference. The package is optimized with Rcpp for computational efficiency and supports both Seurat V4 and V5 objects. Key features include: (1) Multiple scoring methods with Cauchy combination, (2) Reference panel construction and query inference, (3) Support for multiple ligand-receptor databases (CellPhoneDB, CellChat, NicheNet), (4) High-performance vectorized and parallel computation.

License MIT + file LICENSE

URL `https://zaoqu-liu.github.io/FastCCCR/`, `https://github.com/Zaoqu-Liu/FastCCCR`

BugReports `https://github.com/Zaoqu-Liu/FastCCCR/issues`

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports R6, Rcpp (>= 1.0.0), data.table, Matrix, methods, stats, utils, graphics, grDevices, parallel, future, future.apply, progressr, cli, rlang

Suggests Seurat (>= 4.0.0), SeuratObject, SingleCellExperiment, SummarizedExperiment, ggplot2, igraph, ComplexHeatmap, circlize, testthat (>= 3.0.0), knitr, rmarkdown, covr

LinkingTo Rcpp, RcppArmadillo

SystemRequirements C++17

NeedsCompilation yes

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'RcppExports.R' 'globals.R' 'zzz.R' 'utils.R' 'distribution.R' 'dist-iid-set.R' 'dist-complex.R' 'dist-lr.R' 'data-preprocess.R' 'score.R' 'core.R' 'cauchy-combine.R' 'build-reference.R' 'infer-query.R' 'visualize.R' 'FastCCCR-package.R'

Repository https://zaoqu-liu.r-universe.dev

Date/Publication 2026-01-26 06:14:08 UTC

RemoteUrl https://github.com/Zaoqu-Liu/FastCCCR

RemoteRef main

RemoteSha e2454107b95439b3adb3114008e8fab419284898

Contents

FastCCCR-package 3
analyze_interactions_percents 5
batch_compute_gene_distributions_cpp 5
batch_compute_pvalues_cpp 6
build-reference 7
build_reference 7
calculate_cluster_mean 8
calculate_cluster_mean_distribution 8
calculate_cluster_means_cpp 9
calculate_cluster_percents 10
calculate_cluster_percents_cpp 10
calculate_cluster_quantile 11
calculate_cluster_quantile_distribution 11
calculate_cluster_quantiles_cpp 12
calculate_interaction_strengths_cpp 13
calculate_interactions_pvalue 13
calculate_interactions_strength 14
cauchy-combine 15
cauchy_combine 15
cauchy_combine_cpp 15
cauchy_combine_files 16
cauchy_combine_with_stat 16
check_interactions_by_DEG 17
cluster_markers_method 17
combine_complex_distribution 18
combine_complex_distribution_dt 18
core 19

create_distribution . . . 19
create_significant_df . . . 19
data-preprocess . . . 20
digitize_transform_cpp . . . 20
dist-complex . . . 20
dist-iid-set . . . 20
dist-lr . . . 21
Distribution . . . 21
fast_histogram_cpp . . . 24
fastccc . . . 25
fastccc_cauchy . . . 25
fft_convolve_cpp . . . 26
get_average_distribution . . . 27
get_count_data . . . 27
get_input_data . . . 28
get_input_data_from_files . . . 29
get_interactions . . . 29
get_minimum_distribution . . . 30
get_minimum_distribution_cpp . . . 30
get_network_data . . . 31
get_pmf_from_digit_samples . . . 31
get_pvalue . . . 32
get_pvalues_from_pmf_cpp . . . 32
get_strength_network_data . . . 33
get_threshold . . . 33
infer-query . . . 34
infer_query . . . 34
load_reference_config . . . 35
multiply_distributions_cpp . . . 35
plot_ccc_chord . . . 36
plot_ccc_heatmap . . . 36
plot_ccc_network . . . 37
score . . . 38
statistical_analysis_method . . . 38
sum_distributions_cpp . . . 39
visualize . . . 39

Index **40**

`FastCCCR-package` *FastCCCR: Fast Cell-Cell Communication Analysis*

Description

A high-performance R package for cell-cell communication (CCC) analysis in single-cell RNA sequencing data. FastCCCR implements an innovative statistical framework based on exact null distribution computation, Cauchy combination of multiple statistical methods, and reference-based inference.

Main Functions

- `fastccc`: Simple wrapper for CCC analysis
- `fastccc_cauchy`: Full analysis with Cauchy combination
- `statistical_analysis_method`: Single method analysis
- `build_reference`: Build reference panel
- `infer_query`: Infer using reference

Visualization

- `plot_ccc_network`: Network visualization
- `plot_ccc_heatmap`: Heatmap visualization
- `plot_ccc_chord`: Chord diagram

Supported Databases

- CellPhoneDB v4.1.0, v5.0.0
- CellChat v1.6.1
- NicheNet v1.1.1, v2.1.5

Seurat Compatibility

FastCCCR supports both Seurat V4 and V5 objects. The package automatically detects the Seurat version and uses the appropriate methods.

Author(s)

Zaoqu Liu <`liuzaoqu@163.com`>

References

Liu, Z. et al. (2026). FastCCCR: Fast Cell-Cell Communication Analysis with Statistical Framework.

See Also

Useful links:

- `https://github.com/Zaoqu-Liu/FastCCCR`
- Report bugs at `https://github.com/Zaoqu-Liu/FastCCCR/issues`

`analyze_interactions_percents` *Analyze Interaction Percentages*

Description

Determine which interactions pass expression threshold

Usage

```
analyze_interactions_percents(
  cluster_percents,
  interactions,
  threshold = 0.1,
  sep = "|"
)
```

Arguments

`cluster_percents`	data.table of expression percentages
`interactions`	data.table of interactions
`threshold`	Minimum percentage threshold
`sep`	Separator for cell type pair names

Value

data.table of logical values (TRUE = passes threshold)

`batch_compute_gene_distributions_cpp` *Batch compute gene sum distributions*

Description

Compute sum distributions for all genes at once (FAST!)

Usage

```
batch_compute_gene_distributions_cpp(
  expr_mat,
  bin_edges_r,
  n_fft = 30L,
  max_value = 20,
  n_threads = 0L
)
```

Arguments

`expr_mat`	Expression matrix (genes x cells)
`n_fft`	Number of convolutions to compute
`max_value`	Maximum value to clip (for histogram)
`n_threads`	Number of threads (0 = auto)
`bin_edges`	Histogram bin edges

Value

List of lists: gene_sum_pmf[gene][n]

`batch_compute_pvalues_cpp`

Batch compute p-values for L-R interactions

Description

Compute p-values for all valid interactions at once (FAST!)

Usage

```
batch_compute_pvalues_cpp(
  ligand_pmfs,
  receptor_pmfs,
  observed_values,
  precision = 0.01
)
```

Arguments

`ligand_pmfs`	List of ligand PMF arrays (one per interaction)
`receptor_pmfs`	List of receptor PMF arrays (one per interaction)
`observed_values`	Vector of observed interaction strengths
`precision`	PMF precision (default 0.01)

Value

Vector of p-values

`build-reference` *Reference Building Functions*

Description

Functions for building CCC reference panels

`build_reference` *Build Reference Panel*

Description

Build a CCC reference panel from atlas data

Usage

```
build_reference(
  seurat_obj,
  reference_name,
  database,
  celltype_col,
  save_path = ".",
  min_percentile = 0.1,
  min_genes_per_cell = 50L,
  debug_mode = FALSE
)
```

Arguments

`seurat_obj`	Seurat object with reference data
`reference_name`	Name for the reference
`database`	Database name or path
`celltype_col`	Column name for cell type
`save_path`	Path to save reference
`min_percentile`	Minimum expression percentage
`min_genes_per_cell`	Minimum genes per cell for QC
`debug_mode`	Whether to save debug files

Value

Invisibly returns save path

`calculate_cluster_mean`

Calculate Cluster Mean Expression

Description

Calculate mean expression per cluster for each gene

Usage

```
calculate_cluster_mean(counts_dt, labels_dt)
```

Arguments

`counts_dt`	data.table of expression (cells x genes)
`labels_dt`	data.table with cell_type column

Value

data.table (clusters x genes)

`calculate_cluster_mean_distribution`

Calculate Cluster Mean Distribution

Description

Compute null distribution of mean expression per cluster

Usage

```
calculate_cluster_mean_distribution(
  counts_dt,
  labels_dt,
  n_fft = NULL,
  mode = "continuous",
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

`counts_dt`	data.table of expression (cells x genes)
`labels_dt`	data.table with cell_type column
`n_fft`	Maximum n for exact convolution. NULL = auto (30 for continuous, 100 for digit)
`mode`	Distribution mode ("continuous" or "digit")
`parallel`	Whether to use parallel computation
`n_cores`	Number of cores for parallel

Value

data.table where each cell contains a Distribution object

`calculate_cluster_means_cpp`

Calculate cluster means (vectorized)

Description

Calculate mean expression per cluster

Usage

```
calculate_cluster_means_cpp(expr, labels, n_clusters)
```

Arguments

`expr`	Expression matrix (genes x cells)
`labels`	Integer vector of cluster labels (1-indexed)
`n_clusters`	Number of clusters

Value

Matrix of means (genes x clusters)

`calculate_cluster_percents` *Calculate Cluster Expression Percentages*

Description

Calculate percentage of cells expressing each gene per cluster

Usage

```
calculate_cluster_percents(counts_dt, labels_dt)
```

Arguments

`counts_dt`	data.table of expression
`labels_dt`	data.table with cell_type column

Value

data.table (clusters x genes)

`calculate_cluster_percents_cpp` *Calculate cluster expression percentages*

Description

Calculate percentage of cells expressing each gene per cluster

Usage

```
calculate_cluster_percents_cpp(expr, labels, n_clusters)
```

Arguments

`expr`	Expression matrix (genes x cells)
`labels`	Integer vector of cluster labels (1-indexed)
`n_clusters`	Number of clusters

Value

Matrix of percentages (genes x clusters)

`calculate_cluster_quantile` *Calculate Cluster Quantile Expression*

Description

Calculate quantile expression per cluster

Usage

```
calculate_cluster_quantile(counts_dt, labels_dt, qt = 0.9)
```

Arguments

`counts_dt`	data.table of expression
`labels_dt`	data.table with cell_type column
`qt`	Quantile (0-1)

Value

data.table (clusters x genes)

`calculate_cluster_quantile_distribution` *Calculate Cluster Quantile Distribution*

Description

Compute null distribution of quantile expression per cluster

Usage

```
calculate_cluster_quantile_distribution(
  counts_dt,
  labels_dt,
  quantile = 0.9,
  mode = "continuous",
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

`counts_dt`	data.table of expression
`labels_dt`	data.table with cell_type column
`quantile`	Quantile to compute (0-1)
`mode`	Distribution mode ("continuous" or "digit")
`parallel`	Whether to use parallel computation
`n_cores`	Number of cores

Value

data.table where each cell contains a Distribution object

`calculate_cluster_quantiles_cpp`
Calculate cluster quantiles (vectorized)

Description

Calculate quantile expression per cluster

Usage

```
calculate_cluster_quantiles_cpp(expr, labels, n_clusters, quantile = 0.9)
```

Arguments

`expr`	Expression matrix (genes x cells)
`labels`	Integer vector of cluster labels (1-indexed)
`n_clusters`	Number of clusters
`quantile`	Quantile to compute (0-1)

Value

Matrix of quantiles (genes x clusters)

`calculate_interaction_strengths_cpp`
Calculate interaction strengths (vectorized)

Description

Calculate L-R interaction strengths for all cell type pairs

Usage

```
calculate_interaction_strengths_cpp(
  mean_expr,
  ligand_idx,
  receptor_idx,
  method = 1L
)
```

Arguments

`mean_expr`	Mean expression matrix (genes x clusters)
`ligand_idx`	Ligand gene indices (0-indexed)
`receptor_idx`	Receptor gene indices (0-indexed)
`method`	Method: 1 = arithmetic, 2 = geometric

Value

3D array (n_interactions x n_clusters x n_clusters)

`calculate_interactions_pvalue`
Calculate Interaction P-values

Description

Compute p-values for L-R interactions

Usage

```
calculate_interactions_pvalue(
  mean_pmf_dt,
  interactions,
  interactions_strength,
  percents_analysis,
  method = "Arithmetic",
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

`mean_pmf_dt`	data.table of cluster distributions (clusters x genes)
`interactions`	data.table of interactions
`interactions_strength`	data.table of interaction strengths
`percents_analysis`	data.table of logical (passes threshold)
`method`	"Arithmetic" or "Geometric"
`parallel`	Whether to use parallel processing (reserved for future)
`n_cores`	Number of cores for parallel processing (reserved for future)

Value

data.table of p-values (cell type pairs x interactions)

`calculate_interactions_strength`

Calculate Interaction Strengths

Description

Calculate L-R interaction strengths for all cell type pairs. Uses C++ implementation for ~20x speedup.

Usage

```
calculate_interactions_strength(
  mean_counts,
  interactions,
  method = "Arithmetic",
  sep = "|"
)
```

Arguments

`mean_counts`	data.table of mean expression (clusters x genes)
`interactions`	data.table of interactions
`method`	"Arithmetic" or "Geometric"
`sep`	Separator for cell type pair names

Value

data.table (cell type pairs x interactions)

`cauchy-combine` *Cauchy Combination Functions*

Description

Functions for combining p-values using Cauchy distribution

`cauchy_combine` *Cauchy Combination of P-values*

Description

Combine p-values from multiple methods using Cauchy distribution

Usage

```
cauchy_combine(pval_list, weights = NULL)
```

Arguments

`pval_list`	List of p-value data.tables
`weights`	Optional weight vector (NULL for equal weights)

Value

data.table of combined p-values

`cauchy_combine_cpp` *Cauchy combination of p-values*

Description

Combine p-values using Cauchy distribution

Usage

```
cauchy_combine_cpp(pvals, weights = NULL)
```

Arguments

`pvals`	Matrix of p-values (rows = tests, cols = methods)
`weights`	Weight vector (NULL for equal weights)

Value

Vector of combined p-values

`cauchy_combine_files` *Cauchy Combine from Files*

Description

Combine p-value files in a directory

Usage

```
cauchy_combine_files(result_dir, task_id = NULL, pattern = "*_pvals.tsv")
```

Arguments

`result_dir`	Directory containing p-value files
`task_id`	Task ID to filter files (NULL for all)
`pattern`	File pattern to match

Value

data.table of combined p-values

`cauchy_combine_with_stat` *Cauchy Combine with Statistic*

Description

Combine p-values and compute Cauchy statistic

Usage

```
cauchy_combine_with_stat(pval_list, weights = NULL)
```

Arguments

`pval_list`	List of p-value data.tables
`weights`	Optional weight vector

Value

List with combined p-values and statistics

`check_interactions_by_DEG` *Check Interactions by DEG*

Description

Additional filtering based on differential expression

Usage

```
check_interactions_by_DEG(mean_counts, mean_pmf_dt, interactions, pvals)
```

Arguments

`mean_counts`	data.table of mean expression
`mean_pmf_dt`	data.table of distributions
`interactions`	data.table of interactions
`pvals`	data.table of p-values

Value

Updated p-values with DEG filtering

`cluster_markers_method` *Cluster Markers Method*

Description

Calculate cluster markers using FastCCC statistics

Usage

```
cluster_markers_method(
  seurat_obj,
  celltype_col,
  cluster_method = "Mean",
  quantile = 0.9
)
```

Arguments

`seurat_obj`	Seurat object
`celltype_col`	Column for cell type
`cluster_method`	Method ("Mean" or "Quantile")
`quantile`	Quantile value

Value

List with mean counts, distributions, and p-values

`combine_complex_distribution` *Combine Complex Distribution*

Description

Add complex expression scores to cluster scores

Usage

```
combine_complex_distribution(mean_counts, complex_table, agg_func = "min")
```

Arguments

`mean_counts`	data.table of cluster means
`complex_table`	data.table of complex composition
`agg_func`	Aggregation function ("min" or "mean")

Value

data.table with complex scores added

`combine_complex_distribution_dt` *Combine Complex Distribution*

Description

Add complex distributions to cluster distributions

Usage

```
combine_complex_distribution_dt(dist_dt, complex_table, complex_func = "min")
```

Arguments

`dist_dt`	data.table of distributions (clusters x genes)
`complex_table`	data.table of complex composition
`complex_func`	Function to combine subunit distributions ("min" or "avg")

Value

data.table with complex distributions added

`core` *Core Functions for FastCCCR*

Description

Main entry functions for cell-cell communication analysis

`create_distribution` *Create Distribution from Samples*

Description

Create Distribution from Samples

Usage

```
create_distribution(samples, mode = "continuous")
```

Arguments

`samples`	Numeric vector of samples
`mode`	"continuous" or "digit"

Value

Distribution object

`create_significant_df` *Create Significant Interactions DataFrame*

Description

Create summary of significant interactions

Usage

```
create_significant_df(pvals, database_path, threshold = 0.05)
```

Arguments

`pvals`	data.table of p-values
`database_path`	Path to database
`threshold`	P-value threshold

Value

data.table of significant interactions

`data-preprocess` *Data Preprocessing Functions for FastCCCR*

Description

Functions for loading and preprocessing data

`digitize_transform_cpp` *Digitize transform for rank-based analysis*

Description

Transform expression values to rank-based bins

Usage

```
digitize_transform_cpp(x, n_bins = 50L)
```

Arguments

`x`	Numeric vector (non-zero values)
`n_bins`	Number of bins

Value

Integer vector of bin indices

`dist-complex` *Complex Distribution Functions*

Description

Functions for computing distributions of protein complexes

`dist-iid-set` *IID Set Distribution Functions*

Description

Functions for computing distributions of cluster statistics

`dist-lr` *L-R Distribution and P-value Functions*

Description

Functions for computing L-R interaction p-values

`Distribution` *Distribution Class for FastCCCR*

Description

R6 class representing probability distributions for CCC analysis

An R6 class representing probability mass functions (PMF) with optimized operations for cell-cell communication analysis. Supports both discrete PMF arrays and normal distribution approximations.

Details

The Distribution class supports the following operations:

- Addition: Sum of two independent random variables (FFT convolution)
- Power: Sum of n i.i.d. random variables
- Division: Average of n i.i.d. random variables
- And (&): Special operation for L+R combination

Public fields

`dtype` Distribution type: "normal" or "other"

`loc` Location parameter (mean for normal)

`scale` Scale parameter (sd for normal)

`pmf_array` PMF array

`is_align` Whether PMF is aligned to standard bins

`is_normal_type` Whether distribution is normal type

`is_analytic` Whether distribution has analytic CDF

`is_complex_analytic` Whether from complex combination

`cdf_analytic_func` Analytic CDF function (for normal)

`min_cdf_non_zero` Lower bound of support

`min_cdf_one` Upper bound of support

`support_length` Length of support

`ligand` Ligand distribution (for complex analytic)

`receptor` Receptor distribution (for complex analytic)

`mode` Distribution mode: "continuous" or "digit"

Methods

Public methods:

- `Distribution$new()`
- `Distribution$get_mean()`
- `Distribution$get_std()`
- `Distribution$get_var()`
- `Distribution$get_pmf_array()`
- `Distribution$get_cdf_array()`
- `Distribution$add()`
- `Distribution$power()`
- `Distribution$divide()`
- `Distribution$and_op()`
- `Distribution$mul_op()`
- `Distribution$copy()`
- `Distribution$print()`
- `Distribution$clone()`

Method `new()`: Create a new Distribution object

Usage:

```
Distribution$new(
  dtype = "other",
  loc = NULL,
  scale = NULL,
  samples = NULL,
  pmf_array = NULL,
  is_align = NULL,
  mode = "continuous",
  eps = 1e-04
)
```

Arguments:

`dtype` Type: "normal", "gaussian", or "other"

`loc` Location parameter (mean)

`scale` Scale parameter (sd)

`samples` Sample data to construct PMF

`pmf_array` Pre-computed PMF array

`is_align` Whether PMF is aligned

`mode` Distribution mode ("continuous" or "digit")

`eps` Tolerance for PMF sum check

Returns: A new Distribution object

Method `get_mean()`: Get mean of distribution

Usage:

```
Distribution$get_mean()
```

Returns: Mean value

Method `get_std()`**:** Get standard deviation

Usage:

```
Distribution$get_std()
```

Returns: SD value

Method `get_var()`**:** Get variance

Usage:

```
Distribution$get_var()
```

Returns: Variance value

Method `get_pmf_array()`**:** Get PMF array

Usage:

```
Distribution$get_pmf_array()
```

Returns: PMF array

Method `get_cdf_array()`**:** Get CDF array

Usage:

```
Distribution$get_cdf_array()
```

Returns: CDF array

Method `add()`**:** Add two distributions (convolution)

Usage:

```
Distribution$add(other)
```

Arguments:

`other` Another Distribution object

Returns: New Distribution representing sum

Method `power()`**:** Sum of n i.i.d. copies

Usage:

```
Distribution$power(n)
```

Arguments:

`n` Number of copies

Returns: New Distribution

Method `divide()`**:** Average of n i.i.d. copies

Usage:

```
Distribution$divide(n)
```

Arguments:

`n` Number of copies

Returns: New Distribution

Method `and_op()`**:** Special "and" operation for L+R

Usage:

`Distribution$and_op(other)`

Arguments:

`other` Another Distribution

Returns: New Distribution

Method `mul_op()`**:** Geometric mean of two distributions (sqrt(X*Y))

Usage:

`Distribution$mul_op(other)`

Arguments:

`other` Another Distribution

Details: Uses C++ implementation for ~80x speedup over R version. Computes distribution of sqrt(X*Y) given distributions of X and Y.

Returns: New Distribution representing sqrt(XY)

Method `copy()`**:** Copy the distribution

Usage:

`Distribution$copy()`

Returns: A deep copy of this Distribution

Method `print()`**:** Print method

Usage:

`Distribution$print()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

`Distribution$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

`fast_histogram_cpp` *Fast histogram for PMF computation*

Description

Compute histogram with specified breaks

Usage

```
fast_histogram_cpp(x, breaks)
```

Arguments

`x`	Numeric vector
`breaks`	Break points

Value

Counts per bin

`fastccc` *Simplified FastCCC Function*

Description

Simple wrapper for FastCCC analysis

Usage

```
fastccc(seurat_obj, database = "CPDBv5.0.0", celltype_col = "cell_type", ...)
```

Arguments

`seurat_obj`	Seurat object
`database`	Database name (default: "CPDBv5.0.0")
`celltype_col`	Cell type column name
`...`	Additional arguments passed to fastccc_cauchy

Value

List with analysis results

`fastccc_cauchy` *FastCCC with Cauchy Combination*

Description

Run FastCCC analysis with multiple methods and Cauchy combination

Usage

```
fastccc_cauchy(
  seurat_obj,
  database,
  celltype_col,
  single_unit_summary_list = c("Mean", "Median", "Q3", "Quantile_0.9"),
  complex_aggregation_list = c("Minimum", "Average"),
  LR_combination_list = c("Arithmetic", "Geometric"),
  min_percentile = 0.1,
  save_path = NULL,
  use_DEG = FALSE,
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

`seurat_obj`	Seurat object
`database`	Database name or path
`celltype_col`	Column name for cell type labels
`single_unit_summary_list`	Methods for single-unit summary
`complex_aggregation_list`	Methods for complex aggregation
`LR_combination_list`	Methods for L-R combination
`min_percentile`	Minimum expression percentage threshold
`save_path`	Path to save results (NULL for current directory)
`use_DEG`	Whether to filter by DEG
`parallel`	Whether to use parallel computation
`n_cores`	Number of cores for parallel

Value

List with results

`fft_convolve_cpp` *FFT Convolution*

Description

Fast convolution using FFT

Usage

```
fft_convolve_cpp(x, y)
```

Arguments

`x`	First vector
`y`	Second vector

Value

Convolution result

`get_average_distribution` *Get Average Distribution*

Description

Compute distribution of average of k random variables

Usage

```
get_average_distribution(dist_list)
```

Arguments

`dist_list`	List of Distribution objects

Value

Distribution object

`get_count_data` *Get Count Data Only*

Description

Extract expression data without interaction filtering

Usage

```
get_count_data(seurat_obj, celltype_col)
```

Arguments

`seurat_obj`	Seurat object
`celltype_col`	Column name for cell type

Value

List with counts_dt and labels_dt

`get_input_data` *Get Input Data from Seurat Object*

Description

Extract and preprocess data from Seurat object for CCC analysis

Usage

```
get_input_data(
  seurat_obj,
  database,
  celltype_col,
  convert_type = "hgnc_symbol",
  filter_genes = FALSE,
  min_cells = 1L
)
```

Arguments

`seurat_obj`	Seurat object
`database`	Database name or path
`celltype_col`	Column name for cell type labels
`convert_type`	Gene name type ("hgnc_symbol")
`filter_genes`	Whether to filter low-expressed genes
`min_cells`	Minimum cells expressing a gene

Value

List containing counts_dt, labels_dt, complex_table, interactions

---

`get_input_data_from_files`
Load Data from Files

Description

Load expression and metadata from files

Usage

```
get_input_data_from_files(
  counts_file,
  meta_file,
  database,
  celltype_col = "cell_type",
  convert_type = "hgnc_symbol"
)
```

Arguments

`counts_file`	Path to counts file (CSV/TSV)
`meta_file`	Path to metadata file
`database`	Database name or path
`celltype_col`	Column name for cell type
`convert_type`	Gene name type

Value

List with processed data

`get_interactions` *Get Interactions from Database*

Description

Load ligand-receptor interactions from database

Usage

```
get_interactions(database, select_list = NULL)
```

Arguments

`database`	Database name or path
`select_list`	Optional vector of interaction IDs to select

Value

data.table with interaction information

`get_minimum_distribution` *Get Minimum Distribution*

Description

Get Minimum Distribution

Usage

```
get_minimum_distribution(dist_list)
```

Arguments

`dist_list`	List of Distribution objects

Value

Distribution object

`get_minimum_distribution_cpp` *Minimum Distribution*

Description

Compute distribution of min(X1, X2, ..., Xk)

Usage

```
get_minimum_distribution_cpp(pmf_list)
```

Arguments

`pmf_list`	List of PMF vectors

Value

PMF of minimum

`get_network_data` *Get Network Data*

Description

Extract network data from results

Usage

```
get_network_data(results, pval_threshold = 0.05)
```

Arguments

`results`	Results from fastccc
`pval_threshold`	P-value threshold

Value

List with nodes and edges data frames

`get_pmf_from_digit_samples` *Get PMF Array from Digitized Samples*

Description

Get PMF Array from Digitized Samples

Usage

```
get_pmf_from_digit_samples(samples, n_bins = 50L)
```

Arguments

`samples`	Digitized samples (integer 0 to n_bins)
`n_bins`	Number of bins

Value

PMF array

`get_pvalue` *Get p-value from Distribution*

Description

Get p-value from Distribution

Usage

```
get_pvalue(value, dist, mode = "continuous")
```

Arguments

`value`	Observed value
`dist`	Distribution object or PMF array
`mode`	"continuous" or "digit"

Value

P-value

`get_pvalues_from_pmf_cpp` *Calculate P-values from PMF (vectorized)*

Description

Calculate p-values for multiple observed values

Usage

```
get_pvalues_from_pmf_cpp(values, pmf, precision = 0.01)
```

Arguments

`values`	Numeric vector of observed values
`pmf`	PMF array
`precision`	Precision (bin width)

Value

Vector of p-values

`get_strength_network_data` *Get Strength Network Data*

Description

Extract network data based on interaction strength

Usage

```
get_strength_network_data(results, pval_threshold = 0.05)
```

Arguments

`results`	Results containing both pvals and strength
`pval_threshold`	P-value threshold

Value

List with nodes and edges

`get_threshold` *Get threshold from Distribution (for significance)*

Description

Get threshold from Distribution (for significance)

Usage

```
get_threshold(dist, alpha = 0.05, mode = "digit")
```

Arguments

`dist`	Distribution object
`alpha`	Significance level (right-tail area)
`mode`	"continuous" or "digit"

Value

Threshold value

`infer-query` *Query Inference Functions*

Description

Functions for inferring CCC using reference panels

`infer_query` *Infer Query Using Reference*

Description

Infer cell-cell communication for query data using reference

Usage

```
infer_query(
  seurat_obj,
  reference_path,
  database,
  celltype_col,
  celltype_mapping = NULL,
  save_path = NULL,
  min_genes_per_cell = 50L,
  debug_mode = FALSE,
  k = NULL
)
```

Arguments

`seurat_obj`	Query Seurat object
`reference_path`	Path to reference panel
`database`	Database name or path
`celltype_col`	Column name for cell type
`celltype_mapping`	Optional named list mapping reference to query cell types
`save_path`	Path to save results
`min_genes_per_cell`	Minimum genes per cell
`debug_mode`	Whether to output debug files
`k`	Scaling factor for confidence interval (NULL for auto)

Value

List with inference results

`load_reference_config` *Load Reference Config*

Description

Load Reference Config

Usage

```
load_reference_config(ref_path)
```

Arguments

`ref_path`	Reference path

Value

List with configuration

`multiply_distributions_cpp` *Multiply Distributions*

Description

Compute distribution of sqrt(X*Y) given distributions of X and Y

Usage

```
multiply_distributions_cpp(pmf1, pmf2)
```

Arguments

`pmf1`	PMF of first distribution
`pmf2`	PMF of second distribution

Value

PMF of sqrt(X*Y)

`plot_ccc_chord` *Plot CCC Chord Diagram*

Description

Create chord diagram of cell-cell communication

Usage

```
plot_ccc_chord(results, pval_threshold = 0.05, min_interactions = 1, ...)
```

Arguments

`results`	Results from fastccc
`pval_threshold`	P-value threshold
`min_interactions`	Minimum interactions for connection
`...`	Additional arguments passed to circlize::chordDiagram

Value

NULL (plots to device)

`plot_ccc_heatmap` *Plot CCC Heatmap*

Description

Create heatmap visualization of interaction strengths

Usage

```
plot_ccc_heatmap(
  results,
  show_pval = TRUE,
  pval_threshold = 0.05,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  top_n = 50,
  ...
)
```

Arguments

`results`	Results from fastccc
`show_pval`	Whether to overlay p-value significance
`pval_threshold`	P-value threshold
`cluster_rows`	Whether to cluster rows
`cluster_cols`	Whether to cluster columns
`top_n`	Show only top N interactions by variance
`...`	Additional arguments passed to ComplexHeatmap

Value

ComplexHeatmap object

`plot_ccc_network` *Plot CCC Network*

Description

Create network visualization of cell-cell communication

Usage

```
plot_ccc_network(
  results,
  pval_threshold = 0.05,
  min_interactions = 1,
  layout = "circle",
  vertex_size_scale = 1,
  edge_width_scale = 1,
  ...
)
```

Arguments

`results`	Results from fastccc or related functions
`pval_threshold`	P-value threshold for significance
`min_interactions`	Minimum interactions to show edge
`layout`	Network layout algorithm
`vertex_size_scale`	Scale factor for vertex size
`edge_width_scale`	Scale factor for edge width
`...`	Additional arguments passed to plot

Value

igraph plot (invisibly returns igraph object)

`score` *Score Calculation Functions for FastCCCR*

Description

Functions for calculating expression scores

`statistical_analysis_method` *FastCCC Single Method*

Description

Run FastCCC with a single statistical method

Usage

```
statistical_analysis_method(
  seurat_obj,
  database,
  celltype_col,
  single_unit_summary = "Mean",
  complex_aggregation = "Minimum",
  LR_combination = "Arithmetic",
  min_percentile = 0.1,
  style = NULL,
  use_DEG = FALSE,
  save_path = NULL,
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

`seurat_obj`	Seurat object
`database`	Database name or path
`celltype_col`	Column name for cell type labels
`single_unit_summary`	Single-unit summary method
`complex_aggregation`	Complex aggregation method

`LR_combination`	L-R combination method
`min_percentile`	Minimum expression percentage
`style`	Preset style ("cpdb" for CellPhoneDB-like)
`use_DEG`	Whether to filter by DEG
`save_path`	Path to save results
`parallel`	Whether to use parallel
`n_cores`	Number of cores

Value

List with results

`sum_distributions_cpp` *Sum Distributions (for average)*

Description

Compute sum of k distributions using repeated convolution

Usage

```
sum_distributions_cpp(pmf_list)
```

Arguments

`pmf_list`	List of PMF vectors

Value

PMF of sum

`visualize` *Visualization Functions for FastCCCR*

Description

Functions for visualizing CCC analysis results

Index

analyze_interactions_percents, 5

batch_compute_gene_distributions_cpp, 5
batch_compute_pvalues_cpp, 6
build-reference, 7
build_reference, *4*, 7

calculate_cluster_mean, 8
calculate_cluster_mean_distribution, 8
calculate_cluster_means_cpp, 9
calculate_cluster_percents, 10
calculate_cluster_percents_cpp, 10
calculate_cluster_quantile, 11
calculate_cluster_quantile_distribution, 11
calculate_cluster_quantiles_cpp, 12
calculate_interaction_strengths_cpp, 13
calculate_interactions_pvalue, 13
calculate_interactions_strength, 14
cauchy-combine, 15
cauchy_combine, 15
cauchy_combine_cpp, 15
cauchy_combine_files, 16
cauchy_combine_with_stat, 16
check_interactions_by_DEG, 17
cluster_markers_method, 17
combine_complex_distribution, 18
combine_complex_distribution_dt, 18
core, 19
create_distribution, 19
create_significant_df, 19

data-preprocess, 20
digitize_transform_cpp, 20
dist-complex, 20
dist-iid-set, 20
dist-lr, 21
Distribution, 21

fast_histogram_cpp, 24
fastccc, *4*, 25
fastccc_cauchy, *4*, 25
FastCCCR (FastCCCR-package), 3
FastCCCR-package, 3
fft_convolve_cpp, 26

gene, *6*
get_average_distribution, 27
get_count_data, 27
get_input_data, 28
get_input_data_from_files, 29
get_interactions, 29
get_minimum_distribution, 30
get_minimum_distribution_cpp, 30
get_network_data, 31
get_pmf_from_digit_samples, 31
get_pvalue, 32
get_pvalues_from_pmf_cpp, 32
get_strength_network_data, 33
get_threshold, 33

infer-query, 34
infer_query, *4*, 34

load_reference_config, 35

multiply_distributions_cpp, 35

n, *6*

plot_ccc_chord, *4*, 36
plot_ccc_heatmap, *4*, 36
plot_ccc_network, *4*, 37

score, 38
statistical_analysis_method, *4*, 38
sum_distributions_cpp, 39

visualize, 39