Package 'MATES'

January 22, 2026

Title Multi-View Aggregated Two Sample Tests

Version 0.1

Maintainer Zexi Cai <zc2626@columbia.edu>

Depends R (>= 3.3.0)

Description Implements the Multi-view Aggregated Two-Sample (MATES) test, a powerful nonparametric method for testing equality of two multivariate distributions. The method constructs multiple graph-based statistics from various perspectives (views) including different distance metrics, graph types (nearest neighbor graphs, minimum spanning trees, and robust nearest neighbor graphs), and weighting schemes. These statistics are then aggregated through a quadratic form to achieve improved statistical power. The package provides both asymptotic closed-form inference and permutation-based testing procedures. For methodological details, see Cai and others (2026+) <doi:10.48550/arXiv.2412.16684>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/ZexiCAI/MATES

BugReports https://github.com/ZexiCAI/MATES/issues

LinkingTo Rcpp

Imports Rcpp, ade4, MASS, magrittr

NeedsCompilation yes

Author Zexi Cai [aut, cre] (ORCID: <https://orcid.org/0000-0002-0253-8045>),
Mingshuo Liu [aut, ctb] (ORCID:
<https://orcid.org/0000-0001-9270-0298>),
Wenbo Fei [aut, ctb] (ORCID: <https://orcid.org/0009-0000-6377-3252>),
Doudou Zhou [aut, ctb] (ORCID: <https://orcid.org/0000-0002-0830-2287>)

Repository CRAN

Date/Publication 2026-01-22 21:40:07 UTC

Contents

`MATES-package` *MATES*

Description

Implements the Multi-view Aggregated Two-Sample (MATES) test, a powerful nonparametric method for testing equality of two multivariate distributions. The method constructs multiple graph-based statistics from various perspectives (views) including different distance metrics, graph types (nearest neighbor graphs, minimum spanning trees, and robust nearest neighbor graphs), and weighting schemes. These statistics are then aggregated through a quadratic form to achieve improved statistical power. The package provides both asymptotic closed-form inference and permutation-based testing procedures. For methodological details, see Cai and others (2026+) doi: 10.48550/arXiv.2412.16684.

Author(s)

Maintainer: Zexi Cai <`zc2626@columbia.edu`> (ORCID)

Authors:

- Mingshuo Liu <`mshliu@ucdavis.edu`> (ORCID) [contributor]
- Wenbo Fei <`wf2270@columbia.edu`> (ORCID) [contributor]
- Doudou Zhou <`ddzhou@nus.edu.sg`> (ORCID) [contributor]

See Also

Useful links:

- `https://github.com/ZexiCAI/MATES`
- Report bugs at `https://github.com/ZexiCAI/MATES/issues`

asy_cov *Find the permutation covariance*

Description

This function takes a list of numeric matrices and uses a C++ backend to find the permutation covariance

Usage

```
asy_cov(R_list, m, n)
```

Arguments

`R_list`	A list of numeric matrices with length S
`m`	An integer representing the number of sample in X
`n`	An integer representing the number of sample in Y

Value

A numeric matrix with row and column 2*S

Examples

```
# Generate simulated data
set.seed(123)
X <- matrix(rnorm(20), ncol = 2)  # 10 samples, 2 dimensions
Y <- matrix(rnorm(20), ncol = 2)  # 10 samples, 2 dimensions
Z <- rbind(X, Y)
m <- nrow(X)
n <- nrow(Y)
N <- m + n

# Compute distance and similarity matrices
D <- as.matrix(dist(Z, method = "manhattan"))
S <- max(D) - D

# Compute rank matrix (simplified NNG approach)
R <- matrix(0, N, N)
k <- 3
for(i in 1:N) {
  neighbors <- order(D[i,])[2:(k+1)]  # k nearest neighbors
  R[i, neighbors] <- 1:k
}
R <- R + t(R)

# Create list with one rank matrix
R_list <- list(R)
```

```
# Calculate permutation covariance
cov_mat <- asy_cov(R_list, m = m, n = n)
print(cov_mat)
```

---

asy_mean *Find the permutation mean*

Description

This function takes a list of numeric matrices and uses a C++ backend to find the permutation mean.

Usage

```
asy_mean(R_list, m, n)
```

Arguments

`R_list`	A list of numeric matrices with length S
`m`	An integer representing the number of sample in X
`n`	An integer representing the number of sample in Y

Value

A numeric vector with length 2*S

Examples

```
# Generate simulated data
set.seed(123)
X <- matrix(rnorm(20), ncol = 2)  # 10 samples, 2 dimensions
Y <- matrix(rnorm(20), ncol = 2)  # 10 samples, 2 dimensions
Z <- rbind(X, Y)
m <- nrow(X)
n <- nrow(Y)
N <- m + n

# Compute distance and similarity matrices
D <- as.matrix(dist(Z, method = "manhattan"))
S <- max(D) - D

# Compute rank matrix (simplified NNG approach)
R <- matrix(0, N, N)
k <- 3
for(i in 1:N) {
  neighbors <- order(D[i,])[2:(k+1)]  # k nearest neighbors
  R[i, neighbors] <- 1:k
}
R <- R + t(R)
```

```
# Create list with one rank matrix
R_list <- list(R)

# Calculate permutation mean
mean_vec <- asy_mean(R_list, m = m, n = n)
print(mean_vec)
```

`degree_distribution` *Auxiliary function to compute rank matrix*

Description

This function is used in 'P_Knear_rank' to compute the degrees

Usage

```
degree_distribution(G, sampleIDs)
```

Arguments

`G`	Integer or numeric matrix with two columns, where each row represents a directed edge (`from, to`) in the k-NN graph
`sampleIDs`	Integer vector of node indices for which to compute degrees

Value

Numeric vector of degrees with the same length and order as `sampleIDs`

`MATES` *MATES test statistic with two samples (recommended for general use)*

Description

This function takes two data matrices (m x d and n x d) and other parameters to compute the MATES test statistic. It only implements the same distance, graph, and weight options across all views. For other combinations, please compute the corresponding view matrices (R_list) and use the MATES_stat function directly.

Usage

```
MATES(
  X,
  Y,
  S = 4,
  dt = "manhattan",
  gh = "NNG",
  wt = "kernel",
  pow = 0.8,
  perm = NULL
)
```

Arguments

`X`	A numeric matrix of size m x d
`Y`	A numeric matrix of size n x d
`S`	An integer representing the number of moments to use
`dt`	A character string indicating the distance metric to use ("manhattan" or "Lp")
`gh`	A character string indicating the graph type to use ("NNG", "MST", or "rNNG")
`wt`	A character string indicating the weight function to use ("kernel", "rank", "distance", or "plain")
`pow`	A numeric representing the number of neighbors to use for graph, if pow = 0, then use default value 10; otherwise use round(N^pow)
`perm`	An integer indicating the number of permutation (default is NULL, which uses closed form)

Value

A list with the MATES test statistic (test.stat) and p-value (pval)

Examples

```
# Generate two-sample data from different distributions
set.seed(123)
X <- matrix(rnorm(50, mean = 0), ncol = 5)  # 10 samples from N(0,1)
Y <- matrix(rnorm(50, mean = 0.5), ncol = 5)  # 10 samples from N(0.5,1)

# Perform MATES test
result <- MATES(X, Y, S = 4, dt = "manhattan", gh = "NNG", wt = "kernel", pow = 0.8)
print(result$test.stat)
print(result$pval)
```

`MATES_test` *MATES test statistic with pre-computed view matrices*

Description

This function takes a list of view matrices (R_list) and other parameters to compute the MATES test statistic.

Usage

```
MATES_test(UxUy, R_list, m, n, perm = NULL)
```

Arguments

`UxUy`	A numeric vector of length 2*S containing the Ux and Uy statistics for each view
`R_list`	A list of numeric matrices with length S
`m`	An integer representing the number of sample in X
`n`	An integer representing the number of sample in Y
`perm`	An integer indicating the number of permutation (default is NULL, which uses closed form)

Value

A list with the MATES test statistic (test.stat) and p-value (pval)

Examples

```
# Generate simulated data
set.seed(123)
X <- matrix(rnorm(20), ncol = 2)  # 10 samples, 2 dimensions
Y <- matrix(rnorm(20), ncol = 2)  # 10 samples, 2 dimensions
Z <- rbind(X, Y)
m <- nrow(X)
n <- nrow(Y)
N <- m + n

# Compute distance and similarity matrices
D <- as.matrix(dist(Z, method = "manhattan"))
S <- max(D) - D

# Compute rank matrix (simplified NNG approach)
R <- matrix(0, N, N)
k <- 3
for(i in 1:N) {
  neighbors <- order(D[i,])[2:(k+1)]  # k nearest neighbors
  R[i, neighbors] <- 1:k
}
```

```
R <- R + t(R)

# Create list with one rank matrix
R_list <- list(R)

# Calculate test statistics (Ux and Uy)
sample1ID <- 1:m
sample2ID <- (m+1):N
Ux <- sum(R[sample1ID, sample1ID])
Uy <- sum(R[sample2ID, sample2ID])
UxUy <- c(Ux, Uy)

# Perform MATES test
result <- MATES_test(UxUy, R_list, m = m, n = n)
print(result$test.stat)
print(result$pval)
```

---

optimalwithrank_curnode *This function is used in 'P_Knear_rank' #' Compute k-rNNG graph*

Description

This function builds one-step neighbor update for penalized K nearest neighbor graphs with rank The output is a list containing the graph and the degree distribution

Usage

```
optimalwithrank_curnode(k, cur_neis, neighbor, degree, lambda, rowrank)
```

Arguments

`k`	Integer; desired number of neighbors (out-degree) for the current node
`cur_neis`	Integer vector of current neighbors of the node
`neighbor`	Integer vector of candidate neighbor indices for this node,
`degree`	Numeric vector of current degrees for all nodes
`lambda`	A numeric representing the penalty parameter
`rowrank`	Numeric vector of rank-based penalties for this node

Value

A list with two elements:

new_neis Integer vector of length `k` giving the updated neighbors of the node.

degree Updated numeric degree vector for all nodes.

References

Zhu, Y., & Chen, H. (2023). A new robust graph for graph-based methods. *arXiv preprint arXiv:2307.15205.*

`Out_direct` *Auxiliary function to compute rank matrix*

Description

get outdirect nodes for each node

Usage

```
Out_direct(K, nodes)
```

Arguments

`K`	Integer or numeric matrix with two columns, where each row represents a directed edge `(from, to)` in the k-NN graph
`nodes`	Integer vector of node indices for which to extract outgoing neighbors

Value

A list where entry `out[[i]]` is the vector of neighbors `j` such that there is an edge `(i, j)` in `K`

`P_Knear_rank` *Compute k-rNNG graph*

Description

This function builds penalized K nearest neighbor graphs with rank The output is a list containing the graph and the degree distribution

Usage

```
P_Knear_rank(M, K = round(nrow(M)^0.8), lambda = 0.3)
```

Arguments

`M`	A numeric matrix representing the distance matrix
`K`	An integer representing the number of neighbors to use
`lambda`	A numeric representing the penalty parameter

Value

A list containing the truncated KNN graph (trun_KNN) and the degree distribution (degree)

References

Zhu, Y., & Chen, H. (2023). A new robust graph for graph-based methods. *arXiv preprint arXiv:2307.15205.*

`rank_mats` *Compute rank matrix*

Description

This function computes the rank matrix based on the specified graph type and number of neighbors.

Usage

```
rank_mats(S, Dd, gtype, k)
```

Arguments

`S`	A numeric matrix representing the similarity matrix
`Dd`	A dist object representing the distance matrix
`gtype`	A character string indicating the graph type to use ("NNG", "MST", or "rNNG")
`k`	A numeric representing the number of neighbors to use for graph

Value

A numeric matrix representing the rank matrix

References

Zhu, Y., & Chen, H. (2023). A new robust graph for graph-based methods. *arXiv preprint arXiv:2307.15205.*

`Rise_Rank` *RISE rank matrix*

Description

The rank function to calculate rank of elements of a matrix. Two possible methods: the overall rank and the row-wise rank.

Usage

```
Rise_Rank(S, method = "overall")
```

Arguments

`S`	A numeric matrix representing the similarity matrix
`method`	A character string indicating the ranking method to use ("overall" or "row")

Value

A numeric matrix representing the rank matrix

Index