Package 'FITclust'

August 21, 2026

Type Package

Title Fair Interpolated Transport for Group-Fair Clustering

Version 1.0.0

Date 2026-08-19

Author Jesse S. Ghashti [aut, cre],
Warren Hare [aut],
John R.J. Thompson [aut]

Maintainer Jesse S. Ghashti <jesse.ghashti@ubc.ca>

Description Implementation of Fair Interpolated Transport (FIT), an algorithm-agnostic preprocessing framework for group-fair clustering. Group-conditional empirical distributions are moved along Wasserstein-2 geodesics toward a shared barycenter at a tunable transport intensity, and the smallest intensity meeting a soft-fairness tolerance is selected. Three soft clustering families are provided, centroid based, graph based, and model based.

License GPL-2

Encoding UTF-8

Depends R (>= 3.5.0)

Imports stats, transport, Matrix, mvtnorm

Suggests ggplot2, knitr, rmarkdown, devtools, spelling,

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.3.3

Language en-US

Repository CRAN

Date/Publication 2026-08-21 13:40:48 UTC

Contents

`buildTransport` *Build a Transport Interpolation*

Description

Precomputes the Wasserstein-2 barycenter and the group transport maps for a data set, and returns a closure that produces the transported data at any transport intensity by McCann interpolation.

Usage

```
buildTransport(
  dataMat,
  groupVec,
  alphaVec,
  maxBaryIter = 30,
  tol = 1e-04,
  baryInit = "pooled",
  verbose = FALSE
)
```

Arguments

`dataMat`	A numeric matrix of observations, $n \times p$.
`groupVec`	An integer or factor vector of protected-group labels of length n.
`alphaVec`	A numeric vector of barycenter weights, one per group, summing to one.
`maxBaryIter`	Maximum number of barycenter fixed-point iterations.
`tol`	Convergence tolerance for the barycenter iteration.

baryInit	Initial barycenter support, passed to wassersteinBarycenter.
verbose	Logical, whether to print barycenter progress.

Details

At intensity t the transported coordinates of an observation in group g are (1 - t) times the original coordinates plus t times the transported target. A vector t applies a per-group intensity.

Value

A list with components fn (a function of a scalar or length-G intensity t returning the transported $n \times p$ matrix), barycenter, targets, idxByGroup, groups, baryConverged, and baryIter.

Examples

```
dataMat <- rbind(matrix(rnorm(40), ncol = 2), matrix(rnorm(60) + 2, ncol = 2))
groupVec <- rep(1:2, times = c(20, 30))
transport <- buildTransport(dataMat, groupVec, c(0.5, 0.5))
halfway <- transport$fn(0.5)
dim(halfway)
```

---

computeTransportMaps *Transport Maps from Groups to the Barycenter*

Description

Recovers the source-to-barycenter transport maps by barycentric projection. When the barycenter plans are supplied they are reused by transposition, since squared-cost optimal transport is symmetric under swapping source and target.

Usage

```
computeTransportMaps(xByGroup, barycenter, plans = NULL)
```

Arguments

xByGroup	A list of numeric matrices, one per protected group.
barycenter	The $n \times p$ barycenter support matrix.
plans	An optional list of transport plans returned by wassersteinBarycenter. When NULL the plans are recomputed.

Value

A list of numeric matrices, one per group, giving each group's transported target coordinates on the barycenter support.

Examples

```
xByGroup <- list(matrix(rnorm(40), ncol = 2), matrix(rnorm(60) + 1, ncol = 2))
bary <- wassersteinBarycenter(xByGroup, c(0.5, 0.5), maxIter = 5)
maps <- computeTransportMaps(xByGroup, bary$barycenter, bary$plans)
length(maps)
```

`distMatrix` *Squared Euclidean Distance Matrix*

Description

Computes the matrix of squared Euclidean distances between the rows of two matrices.

Usage

```
distMatrix(xMat, yMat)
```

Arguments

`xMat`	A numeric matrix of size $n \times p$.
`yMat`	A numeric matrix of size $m \times p$.

Value

An $n \times m$ numeric matrix whose entry in row i and column j is the squared Euclidean distance between row i of `xMat` and row j of `yMat`.

Examples

```
xMat <- matrix(rnorm(20), ncol = 2)
yMat <- matrix(rnorm(10), ncol = 2)
distMatrix(xMat, yMat)
```

`emGMM` *Gaussian Mixture Model by Expectation Maximization*

Description

Fits a full-covariance Gaussian mixture by the expectation maximization algorithm with ridge-regularized covariances and an optional warm start.

Usage

```
emGMM(
  dataMat,
  numClusters,
  initParams = NULL,
  numStart = 5,
  maxIter = 200,
  tol = 1e-06,
  regLambda = 1e-06
)
```

Arguments

`dataMat`	A numeric matrix of observations, $n \times p$.
`numClusters`	The number of mixture components, K.
`initParams`	An optional list with components `weights`, `means`, and `covariances` giving a warm start. When supplied a single run is performed.
`numStart`	The number of random restarts when `initParams` is `NULL`, each initialized from k-means.
`maxIter`	Maximum number of expectation maximization iterations.
`tol`	Convergence tolerance on the log-likelihood.
`regLambda`	Ridge value added to the diagonal of each component covariance for numerical stability.

Value

A list with components `weights` (the mixing proportions), `means` (the $K \times p$ component means), `covariances` (a list of $p \times p$ covariance matrices), `posterior` (the $n \times K$ posterior assignment matrix), `logLik` (the maximized log-likelihood), and `clusters` (the integer vector of hard assignments obtained by maximum posterior).

References

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977), Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B 39*(1), 1–38.

Examples

```
dataMat <- rbind(matrix(rnorm(60), ncol = 2), matrix(rnorm(60) + 4, ncol = 2))
fit <- emGMM(dataMat, numClusters = 2, numStart = 2)
table(fit$clusters)
```

`fcm` *Fuzzy c-means Clustering*

Description

Fits fuzzy c-means with an optional warm start from supplied centroids. When initial centroids are given a single run is performed, otherwise the best of several random restarts is returned.

Usage

```
fcm(
  dataMat,
  numClusters,
  fuzzifier = 2,
  initCenters = NULL,
  numStart = 5,
  maxIter = 100,
  tol = 1e-06
)
```

Arguments

`dataMat`	A numeric matrix of observations, $n \times p$.
`numClusters`	The number of clusters, K.
`fuzzifier`	The fuzziness exponent, greater than one.
`initCenters`	An optional $K \times p$ matrix of initial centroids. When supplied, `numStart` is ignored and one run is performed.
`numStart`	The number of random restarts when `initCenters` is `NULL`.
`maxIter`	Maximum number of iterations per run.
`tol`	Convergence tolerance on the largest centroid change.

Value

A list with components `membership` (the $n \times K$ soft membership matrix), `centers` (the $K \times p$ centroid matrix), `objective` (the fuzzy objective value), and `clusters` (the integer vector of hard assignments obtained by maximum membership).

References

Bezdek, J. C. (1981), *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York.

Examples

```
dataMat <- rbind(matrix(rnorm(40), ncol = 2), matrix(rnorm(40) + 4, ncol = 2))
fit <- fcm(dataMat, numClusters = 2, numStart = 3)
table(fit$clusters)
```

`fitSKM` *Fair Interpolated Transport, Centroid Variant*

Description

Applies the Fair Interpolated Transport framework with a fuzzy c-means clustering family. The data are transported toward the Wasserstein-2 barycenter along a grid of intensities, and the smallest intensity at which the soft-fairness violation falls within the tolerance is selected.

Usage

```
fitSKM(
  dataMat,
  groupVec,
  numClusters,
  deltaFair = 0.05,
  alpha = "uniform",
  tSeq = seq(0, 1, by = 0.02),
  fuzzifier = 2,
  numStartBaseline = 5,
  warmStart = TRUE,
  verbose = TRUE
)
```

Arguments

`dataMat`	A numeric matrix of observations, $n \times p$.
`groupVec`	An integer or factor vector of protected-group labels of length n.
`numClusters`	The number of clusters, K.
`deltaFair`	The soft-fairness tolerance in the interval from zero to one.
`alpha`	The barycenter weight specification, passed to `resolveAlpha`.
`tSeq`	A numeric grid of transport intensities in the unit interval.
`fuzzifier`	The fuzzy c-means fuzziness exponent.
`numStartBaseline`	The number of random restarts for the baseline fit at intensity zero.
`warmStart`	Logical, whether to warm-start each intensity from the previous solution along the grid.
`verbose`	Logical, whether to print barycenter progress.

Details

When no grid intensity satisfies the tolerance, the solution at the intensity minimizing the soft violation is returned with a warning. Warm starting follows the McCann interpolation path so that adjacent intensities give similar clusterings.

Value

A list describing the selected solution, with components including `clusters`, `centers`, `membership`, the baseline quantities `clustersBaseline`, `centersBaseline`, and `membershipBaseline`, the selected intensity `tOptimal`, the swept diagnostics in `history`, and the transported data `dataTransported`.

References

Bezdek, J. C. (1981), *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York.

McCann, R. J. (1997), A convexity principle for interacting gases, *Advances in Mathematics 128(1)*, 153–179.

See Also

`fitSMM`, `fitSSC`

Examples

```
dataMat <- rbind(matrix(rnorm(60), ncol = 2), matrix(rnorm(60) + 4, ncol = 2))
groupVec <- rep(1:2, times = 30)
fit <- fitSKM(dataMat, groupVec, numClusters = 2, tSeq = seq(0, 1, by = 0.1),
              verbose = FALSE)
fit$tOptimal
```

`fitSMM` *Fair Interpolated Transport, Model Variant*

Description

Applies the Fair Interpolated Transport framework with a Gaussian mixture clustering family. The mixture posteriors serve as the soft memberships, and the expectation maximization fit is warm-started along the intensity grid.

Usage

```
fitSMM(
  dataMat,
  groupVec,
  numClusters,
  deltaFair = 0.05,
  alpha = "uniform",
  tSeq = seq(0, 1, by = 0.02),
  numStartBaseline = 5,
  warmStart = TRUE,
  regLambda = 1e-06,
  verbose = TRUE
)
```

Arguments

`dataMat`	A numeric matrix of observations, $n \times p$.
`groupVec`	An integer or factor vector of protected-group labels of length n.
`numClusters`	The number of mixture components, K.
`deltaFair`	The soft-fairness tolerance in the interval from zero to one.
`alpha`	The barycenter weight specification, passed to `resolveAlpha`.
`tSeq`	A numeric grid of transport intensities in the unit interval.
`numStartBaseline`	The number of random restarts for the baseline fit.
`warmStart`	Logical, whether to warm-start the mixture from the previous intensity along the grid.
`regLambda`	Ridge value added to component covariances for stability.
`verbose`	Logical, whether to print barycenter progress.

Value

A list describing the selected solution, with components including `clusters`, the posterior `membership`, the mixture parameters `weights`, `means`, and `covariances`, the baseline quantities `clustersBaseline` and `membershipBaseline`, the selected intensity `tOptimal`, the swept diagnostics in `history`, and the transported data `dataTransported`.

References

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977), Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B 39*(1), 1–38.

McCann, R. J. (1997), A convexity principle for interacting gases, *Advances in Mathematics 128(1)*, 153–179.

See Also

`fitSSC`, `fitSKM`

Examples

```
dataMat <- rbind(matrix(rnorm(80), ncol = 2), matrix(rnorm(80) + 4, ncol = 2))
groupVec <- rep(1:2, times = 40)
fit <- fitSMM(dataMat, groupVec, numClusters = 2, tSeq = seq(0, 1, by = 0.1),
              verbose = FALSE)
fit$tOptimal
```

`fitSSC` *Fair Interpolated Transport, Graph Variant*

Description

Applies the Fair Interpolated Transport framework with a soft normalized spectral clustering family. The spectral embedding is recomputed at each intensity and aligned to the previous intensity for basis continuity, and fuzzy c-means is run on the embedding.

Usage

```
fitSSC(
  dataMat,
  groupVec,
  numClusters,
  deltaFair = 0.05,
  alpha = "uniform",
  tSeq = seq(0, 1, by = 0.02),
  numNeighbors = 7,
  fuzzifier = 2,
  numStartBaseline = 5,
  warmStart = TRUE,
  verbose = TRUE
)
```

Arguments

`dataMat`	A numeric matrix of observations, $n \times p$.
`groupVec`	An integer or factor vector of protected-group labels of length n.
`numClusters`	The number of clusters, K.
`deltaFair`	The soft-fairness tolerance in the interval from zero to one.
`alpha`	The barycenter weight specification, passed to `resolveAlpha`.
`tSeq`	A numeric grid of transport intensities in the unit interval.
`numNeighbors`	The neighbor rank for local scaling in the embedding.
`fuzzifier`	The fuzzy c-means fuzziness exponent applied on the embedding.
`numStartBaseline`	The number of random restarts for the baseline fit.
`warmStart`	Logical, whether to align embeddings and warm-start centroids across the grid.
`verbose`	Logical, whether to print barycenter progress.

Value

A list describing the selected solution, with components including `clusters`, `membership`, the embedding `embedding` and affinity `affinity`, the baseline quantities `clustersBaseline`, `membershipBaseline`, `embeddingBaseline`, and `affinityBaseline`, the selected intensity `tOptimal`, the swept diagnostics in `history`, and the transported data `dataTransported`.

References

Bezdek, J. C. (1981), *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York.

McCann, R. J. (1997), A convexity principle for interacting gases, *Advances in Mathematics 128(1)*, 153–179.

Zelnik-Manor, L. and Perona, P. (2004), Self-tuning spectral clustering, in *Advances in Neural Information Processing Systems*, Vol. 17.

See Also

`fitSMM`, `fitSKM`

Examples

```
dataMat <- rbind(matrix(rnorm(60), ncol = 2), matrix(rnorm(60) + 5, ncol = 2))
groupVec <- rep(1:2, times = 30)
fit <- fitSSC(dataMat, groupVec, numClusters = 2, tSeq = seq(0, 1, by = 0.1),
              verbose = FALSE)
fit$tOptimal
```

`hardViolation` *Hard Group-Fairness Violation*

Description

Computes the worst-case absolute deviation of any cluster's group share from the population group proportion, over all clusters and groups, for a hard partition.

Usage

```
hardViolation(clusters, groupVec)
```

Arguments

`clusters`	An integer vector of cluster labels, one per observation.
`groupVec`	An integer or factor vector of protected-group labels.

Value

A single non-negative numeric value. Zero indicates exact proportional representation in every cluster.

See Also

`softViolation`

Examples

```
clusters <- rep(1:2, each = 50)
groupVec <- rep(1:2, times = 50)
hardViolation(clusters, groupVec)
```

`procrustesAlign` *Procrustes Alignment of Spectral Embeddings*

Description

Rotates one spectral embedding to best match a reference embedding in Frobenius norm, which stabilizes the eigenvector basis across a sequence of related problems.

Usage

```
procrustesAlign(newEmbed, refEmbed)
```

Arguments

`newEmbed`	A numeric matrix, the embedding to rotate.
`refEmbed`	A numeric matrix of the same dimension, the reference embedding.

Value

The rotated embedding, a numeric matrix of the same dimension as `newEmbed`.

Examples

```
refEmbed <- matrix(rnorm(30), ncol = 3)
newEmbed <- refEmbed %*% qr.Q(qr(matrix(rnorm(9), 3)))
aligned <- procrustesAlign(newEmbed, refEmbed)
dim(aligned)
```

`resolveAlpha` *Resolve Barycenter Weights*

Description

Converts a weight specification into a length-G vector of barycenter weights that sum to one, where G is the number of protected groups.

Usage

```
resolveAlpha(alpha, groupVec, groups)
```

Arguments

alpha	Either a character string, one of "uniform", "proportional", or "inverseProportional", or a numeric vector of length G giving custom non-negative weights.
groupVec	An integer or factor vector of protected-group labels, one entry per observation.
groups	A vector of the unique group labels, in the order the returned weights should follow.

Details

Proportional weights are the group frequencies. Inverse proportional weights are proportional to the reciprocals of the group frequencies and then renormalized. A custom numeric alpha is checked for length, non-negativity, and a positive sum, then renormalized.

Value

A named numeric vector of length G that sums to one.

Examples

```
groupVec <- rep(1:2, times = c(30, 70))
resolveAlpha("proportional", groupVec, sort(unique(groupVec)))
```

---

softViolation *Soft Group-Fairness Violation*

Description

Computes the worst-case absolute deviation of any cluster's soft group share from the population group proportion, over all clusters and groups, for a soft membership matrix.

Usage

```
softViolation(membership, groupVec)
```

Arguments

membership	A non-negative numeric membership matrix of size $n \times K$, whose rows give each observation's soft assignment across the K clusters.
groupVec	An integer or factor vector of protected-group labels of length n.

Value

A single non-negative numeric value.

See Also

hardViolation

Examples

```
membership <- matrix(runif(200), ncol = 2)
membership <- membership / rowSums(membership)
groupVec <- rep(1:2, times = 50)
softViolation(membership, groupVec)
```

`spectralEmbed` *Self-Tuning Spectral Embedding*

Description

Constructs a self-tuning spectral embedding with local scaling, forms the symmetric normalized Laplacian, and returns the row-normalized top eigenvectors.

Usage

```
spectralEmbed(dataMat, numClusters, numNeighbors = 7)
```

Arguments

`dataMat`	A numeric matrix of observations, $n \times p$.
`numClusters`	The embedding dimension, K.
`numNeighbors`	The neighbor rank used for local scaling.

Value

A list with components `embedding` (the $n \times K$ row-normalized eigenvector matrix), `affinity` (the $n \times n$ affinity matrix), `scale` (the per-observation local scale), and `eigenvalues` (the K smallest eigenvalues used).

References

Zelnik-Manor, L. and Perona, P. (2004), Self-tuning spectral clustering, in *Advances in Neural Information Processing Systems*, Vol. 17.

Examples

```
dataMat <- rbind(matrix(rnorm(40), ncol = 2), matrix(rnorm(40) + 5, ncol = 2))
embed <- spectralEmbed(dataMat, numClusters = 2)
dim(embed$embedding)
```

validateGrid *Validate a Transport-Intensity Grid*

Description

Checks that a transport-intensity grid is numeric and lies in the unit interval, then returns it sorted with duplicates removed.

Usage

```
validateGrid(tSeq)
```

Arguments

tSeq	A numeric vector of transport intensities.

Value

The validated grid, sorted in increasing order with duplicates removed.

Examples

```
validateGrid(c(0, 0.5, 0.5, 1))
```

wassersteinBarycenter *Wasserstein-2 Barycenter on a Fixed Support*

Description

Computes the Wasserstein-2 barycenter of a collection of group-conditional empirical distributions using the fixed-point iteration of Alvarez-Esteban et al. (2016).

Usage

```
wassersteinBarycenter(
  xByGroup,
  alpha,
  init = c("pooled", "weightedMean"),
  maxIter = 30,
  tol = 1e-04,
  verbose = FALSE
)
```

Arguments

xByGroup	A list of numeric matrices, one per protected group, each with p columns.
alpha	A numeric vector of barycenter weights of length equal to `length(xByGroup)`, summing to one.
init	A character string selecting the initial support, either `"pooled"` (all group rows stacked) or `"weightedMean"`.
maxIter	Maximum number of fixed-point iterations.
tol	Convergence tolerance on the largest change in support between iterations.
verbose	Logical, whether to print per-iteration progress.

Value

A list with components `barycenter` (the $n \times p$ support matrix), `plans` (the list of optimal transport plans from the final iteration), `iter` (the number of iterations run), and `converged` (logical).

References

Alvarez-Esteban, P. C., del Barrio, E., Cuesta-Albertos, J. A. and Matr´an, C. (2016), A fixed-point approach to barycenters in Wasserstein space, *Journal of Mathematical Analysis and Applications 441*(2), 744–762.

Examples

```
xByGroup <- list(matrix(rnorm(40), ncol = 2), matrix(rnorm(60) + 1, ncol = 2))
fit <- wassersteinBarycenter(xByGroup, c(0.5, 0.5), maxIter = 5)
dim(fit$barycenter)
```

Index