Package: SpatMix (via r-universe)

October 9, 2026

Type Package

Title Spatial Mixture Models for Clustering

Version 0.1.0

Description Fits spatial mixture models, including spatial Gaussian mixtures and mixtures of spatial factor analyzers, to complete or incomplete data. Spatial decay can be represented by monotone I-splines or a normalized sigmoid. Missing entries are handled using a built-in partial expectation-maximization procedure for matrix-variate data. The spatial covariance and spatial factor analyzer models are described in Lu and colleagues (2026a) ``Spatial Covariance Constraints for Gaussian Mixture Models'' <doi:10.48550/arXiv.2601.07979> and Lu and colleagues (2026b) ``Mixtures of spatial factor analyzers for tensor-variate data'' <doi:10.48550/arXiv.2607.07887>.

License MIT + file LICENSE

URL https://github.com/LHZMix/SpatMix

BugReports https://github.com/LHZMix/SpatMix/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports MASS, matrixStats, Rcpp, splines2, stats, withr

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Hanzhang Lu [aut, cre], Jeffrey L. Andrews [aut, ths]

Maintainer Hanzhang Lu <hanzhang.lu@ubc.ca>

Repository https://cran.r-universe.dev

Date/Publication 2026-09-09 15:30:19 UTC

RemoteUrl https://github.com/cran/SpatMix

RemoteRef HEAD

RemoteSha 372be67633a803a3b50fdf7093cc2bf63a25fc0a

Contents

spatmix 2

Index **8**

`spatmix` *Fit a spatial mixture model*

Description

Fits a spatial Gaussian mixture or a mixture with a non-spatial factor-analyzer covariance structure to complete or incomplete data. The row covariance is a linear spatial covariance model with decay represented by monotone I-splines or a normalized sigmoid function. When `X` contains missing values, conditional means and covariance blocks are updated by the matrix partial EM (MPEM) algorithm.

Usage

```
spatmix(
  X,
  G,
  r = NULL,
  coords,
  nknots = 6L,
  degree = 3L,
  common_noise = TRUE,
  mean_structure = c("unconstrained", "constrained"),
  max_iter = 1000L,
  tol = 0.1,
  seed = 1L,
  init = c("kmeans", "covariance", "random"),
  mpem_sweeps = 1L,
  cov_warm_start = 1,
  spatial_max_iter = 5000L,
  spatial_tol = 1e-06,
  verbose = interactive(),
  spatial_decay = c("ispline", "sigmoid"),
  decay_group = 1,
  sigmoid_ctrl = list()
)
```

Arguments

X	A numeric p by q by n array, whose dimensions correspond to spatial locations, non-spatial variables, and observations. Spatial-only data may instead be supplied as an n by p matrix, with observations in rows and spatial locations in columns. NA values invoke the MPEM updates.
G	Number of mixture components.
r	Number of column-covariance factors per component. Required for array input with a non-spatial dimension; use NULL or 0 for spatial-only data.
coords	A numeric vector or p-row numeric matrix of point coordinates, or a list of numeric vectors defining the axes of a complete Cartesian grid. For grid coordinates, prod(lengths(coords)) must equal p.
nknots	Number of internal I-spline knots, given as a scalar or one value per grid axis. Internal knots are placed at empirical quantiles of the positive pairwise distances. Point coordinates use one set of knots. Ignored for sigmoid decay.
degree	I-spline degree, given as a scalar or one value per grid axis. Point coordinates use one degree. Ignored for sigmoid decay.
common_noise	If TRUE, use one row-noise parameter within each component; otherwise estimate a location-specific diagonal.
mean_structure	Mean structure for each mixture component. "unconstrained" estimates a separate mean at each spatial location; "constrained" estimates a mean that is constant across spatial locations for each non-spatial variable.
max_iter	Maximum number of outer EM iterations.
tol	Absolute convergence tolerance for successive log-likelihood values.
seed	Seed used for initialization.
init	Initialization method. "kmeans" applies k-means to the observations, "covariance" applies k-means to squared centered observations for covariance-driven clusters, and "random" uses random initial responsibilities.
mpem_sweeps	Number of conditional-moment coordinate sweeps per MPEM update. Used only when X contains missing values.
cov_warm_start	Weight in [0, 1] assigned to the previous conditional covariance block at the start of each MPEM update; the remaining weight is assigned to a diagonal approximation. Used only when X contains missing values.
spatial_max_iter	Maximum projected-gradient iterations for each I-spline spatial update. Ignored for sigmoid decay.
spatial_tol	Tolerance for the I-spline or sigmoid decay update.
verbose	If TRUE, print the log-likelihood at each iteration.
spatial_decay	Spatial decay representation: "ispline" for monotone I-splines or "sigmoid" for a normalized one-parameter sigmoid function.
decay_group	Labels specifying which grid-coordinate axes share decay parameters. The default 1 places all axes in one group; NULL gives each axis a distinct group; and, for example, c(1, 1, 2) groups the first two axes separately from the third.

Within each mixture component, axes in the same group share both the decay parameters and the corresponding spatial covariance coefficient. Point coordinates produce one joint distance matrix, so `decay_group` has no nontrivial effect for that layout. Axes that share an I-spline decay must have the same number of basis functions.

`sigmoid_ctrl` Optional list with elements `init`, `lower`, `upper`, and `shift`. The first three may be scalars, values supplied per grid axis, or values supplied per decay group; `shift` must be a scalar. Sigmoid distances are normalized to `[0, 2]` separately for each axis. By default, the upper bound is determined from the nearest positive normalized distance.

Details

`coords` supports two spatial layouts. A numeric vector or matrix gives the point coordinates of the `p` spatial locations and produces one Euclidean distance matrix. A list of coordinate vectors, such as `list(x = 1:8, y = 1:8)`, defines a complete Cartesian grid and produces one axis-wise spatial term per list element. Grid locations must follow the row order returned by `do.call(expand.grid, coords)`, with the first axis varying fastest.

Value

An object of class `spatmixfit`. It is a list containing:

- `call`: the matched function call.
- `converged`, `iterations`, and `log_likelihood`: convergence information and the log-likelihood sequence.
- `BIC` and `n_parameters`: the criterion `2 * log-likelihood - log(n) * n_parameters` and its parameter count. Larger `BIC` values indicate better models.
- `proportions`, `responsibility`, and `cluster`: fitted mixing proportions, an `n` by `G` responsibility matrix, and hard cluster assignments.
- `means`: a `p` by `q` by `G` array of component means.
- `row_covariance`, `row_precision`, `col_covariance`, and `col_precision`: fitted row and column covariance matrices and their precisions.
- `alpha`: spatial covariance coefficients, with one column per component.
- `coordinate_beta`: a list of fitted decay parameters, with one element per decay group.
- `sigmoid`: fitted sigmoid parameters when `spatial_decay = "sigmoid"`, and `NULL` otherwise. It is a matrix for one decay group and a list of matrices for multiple groups.
- `loading` and `uniqueness`: factor-analyzer loadings and uniquenesses for the non-spatial covariance.
- `imputation` and `has_missing`: the completed data in the same layout as the input and an indicator of whether `X` contained missing values.
- `knots` and `coordinate_dimensions`: the I-spline knots and spatial layout dimensions.
- `settings`: the model settings used for the fit.

Examples

```
sigmoid <- function(d, beta, shift = 3) {
  z0 <- plogis(-shift)
  (plogis(beta * d - shift) - z0) /
    (plogis(2 * beta - shift) - z0)
}

spatial_cov <- function(coords, beta, alpha) {
  if (!is.list(coords)) coords <- list(coords)
  dims <- lengths(coords)
  decay <- lapply(seq_along(coords), function(j) {
    d <- as.matrix(dist(coords[[j]]))
    d <- 2 * d / max(d)
    matrices <- lapply(seq_along(coords), function(k) {
      if (j == k) sigmoid(d, beta) else matrix(1, dims[k], dims[k])
    })
    Reduce(kronecker, rev(matrices))
  })
  p <- prod(dims)
  J <- matrix(1, p, p) - diag(p)
  alpha[1] * J + alpha[2] * Reduce("+", decay) + alpha[3] * diag(p)
}

## Example 1
set.seed(1)
n <- 150
coords <- 1:10
beta <- c(3, 7)
alpha <- list(c(1, -0.30, 1.2), c(1, -0.45, 1.4))

Xi <- list()
for (g in 1:2) Xi[[g]] <- spatial_cov(coords, beta[g], alpha[[g]])

X <- rbind(
  MASS::mvrnorm(n, rep(0, 10), Xi[[1]]),
  MASS::mvrnorm(n, rep(2.5, 10), Xi[[2]])
)
truth <- rep(1:2, each = n)

fit <- spatmix(
  X, G = 2, coords = coords, spatial_decay = "sigmoid",
  sigmoid_ctrl = list(init = 4, lower = 0.1, upper = 12),
  max_iter = 50, tol = 0.01, verbose = FALSE
)

ord <- order(colMeans(fit$means[, 1, ]))
table(truth, fitted = match(fit$cluster, ord))
round(rbind(truth = beta, fitted = fit$sigmoid[1, ord]), 2)
round(cbind(
  truth.1 = alpha[[1]], fitted.1 = fit$alpha[, ord[1]],
  truth.2 = alpha[[2]], fitted.2 = fit$alpha[, ord[2]]
), 2)
```

```
## Example 2
set.seed(16)
n <- 60
coords <- list(x = c(0, 0.5, 2), y = c(0, 0.5, 2))
p <- prod(lengths(coords))
q <- 2
beta <- c(2, 4)
alpha <- list(c(1, -0.30, 1.2), c(1, -0.45, 1.4))

Xi <- list()
for (g in 1:2) Xi[[g]] <- spatial_cov(coords, beta[g], alpha[[g]])

Lambda <- list(
  matrix(c(1, 0.6), ncol = 1),
  matrix(c(-0.5, 0.9), ncol = 1)
)
Psi <- list(c(0.5, 0.4), c(0.4, 0.7))
Omega <- list()
for (g in 1:2) Omega[[g]] <- tcrossprod(Lambda[[g]]) + diag(Psi[[g]])

M <- list(matrix(0, p, q), matrix(3, p, q))
z <- rbind(
  MASS::mvrnorm(n, as.vector(M[[1]]), kronecker(Omega[[1]], Xi[[1]])),
  MASS::mvrnorm(n, as.vector(M[[2]]), kronecker(Omega[[2]], Xi[[2]]))
)
X <- array(t(z), dim = c(p, q, 2 * n))

fit <- spatmix(
  X, G = 2, r = 1, coords = coords, nknots = 1, degree = 2,
  spatial_decay = "ispline", decay_group = 1,
  mean_structure = "constrained", max_iter = 15, tol = 0.01,
  spatial_max_iter = 20, verbose = FALSE
)

ord <- order(sapply(1:2, function(g) mean(fit$means[, , g])))
table(truth = rep(1:2, each = n), fitted = match(fit$cluster, ord))

d <- seq(0, 2, length.out = 200)
basis <- splines2::iSpline(
  d, knots = fit$knots[[1]], degree = 2, intercept = TRUE,
  Boundary.knots = range(d)
)
fitted_decay <- basis %*% fit$coordinate_beta[[1]][, ord]
plot(d, sigmoid(d, beta[1]), type = "l", lwd = 2,
     col = "firebrick", ylim = c(0, 1),
     xlab = "Normalized distance", ylab = "Decay")
lines(d, fitted_decay[, 1], col = "firebrick", lwd = 2, lty = 2)
lines(d, sigmoid(d, beta[2]), col = "navy", lwd = 2)
lines(d, fitted_decay[, 2], col = "navy", lwd = 2, lty = 2)
legend("bottomright",
       c("Component 1: sigmoid", "Component 1: I-spline",
         "Component 2: sigmoid", "Component 2: I-spline"),
```

```
          col = c("firebrick", "firebrick", "navy", "navy"),
          lty = c(1, 2, 1, 2), lwd = 2, bty = "n")
```

Index

spatmix, 2