Package 'JANE'

January 20, 2025

Title Just Another Latent Space Network Clustering Algorithm

Version 0.2.1

Description Fit and simulate latent space network cluster models using an expectation-maximization algorithm.
Enables flexible modeling of unweighted network data, supporting both directed and undirected networks,
with or without degree heterogeneity. Designed to handle large networks efficiently, it allows users to explore
network structure through latent space representations, identify clusters within network data, and simulate models
with varying clustering and connectivity patterns.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp (>= 1.0.10), Matrix, extraDistr, mclust, scales, aricode,
stringdist, utils, splines, rlang, future.apply, future,
progressr, progress, igraph, methods

RoxygenNote 7.2.3

URL https://github.com/a1arakkal/JANE

BugReports https://github.com/a1arakkal/JANE/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Alan Arakkal [aut, cre, cph] (<https://orcid.org/0000-0002-7001-493X>),
Daniel Sewell [aut] (<https://orcid.org/0000-0002-9238-4026>)

Maintainer Alan Arakkal <alan-arakkal@uiowa.edu>

Repository CRAN

Date/Publication 2024-12-19 17:00:06 UTC

Contents

JANE 2
plot.JANE 8
sim_A 11
specify_initial_values 13
specify_priors 16
summary.JANE 18

Index **22**

JANE *Fit JANE*

Description

Fit the latent space cluster model using an EM algorithm.

Usage

```
JANE(
  A,
  D = 2,
  K = 2,
  model,
  initialization = "GNN",
  case_control = FALSE,
  DA_type = "none",
  seed = NULL,
  control = list()
)
```

Arguments

A	A square matrix or sparse matrix of class 'dgCMatrix' representing the adjacency matrix of the **unweighted** network of interest.
D	Integer (scalar or vector) specifying the dimension of the latent space (default is 2).
K	Integer (scalar or vector) specifying the number of clusters to consider (default is 2).
model	A character string specifying the model to fit: • 'NDH': **undirected** network with no degree heterogeneity • 'RS': **undirected** network with degree heterogeneity • 'RSR': **directed** network with degree heterogeneity
initialization	A character string or a list to specify the initial values for the EM algorithm: • 'GNN': uses a type of graphical neural network approach to generate initial values (default)

- 'random': uses random initial values
- A user supplied list of initial values. See specify_initial_values on how to specify initial values

`case_control` A logical; if TRUE then uses a case-control approximation approach (default is FALSE).

`DA_type` A character string to specify the type of deterministic annealing approach to use

- 'none': does not employ a deterministic annealing approach (default)
- 'cooling': employes a traditional deterministic annealing approach where temperature decreases
- 'heating': employes a deterministic anti-annealing approach where temperature increases
- 'hybrid': employes a combination of the 'cooling' and 'heating' approach

`seed` (optional) An integer value to specify the seed for reproducibility.

`control` A list of control parameters. See 'Details'.

Details

If an unsymmetric adjacency matrix A is supplied for `model %in% c('NDH', 'RS')` the user will be asked if they would like to proceed with converting A to a symmetric matrix (i.e., `A <- 1.0 * ( (A + t(A)) > 0.0 )`).

`control`:

The `control` argument is a named list that the user can supply containing the following components:

`verbose` A logical; if TRUE causes additional information to be printed out about the progress of the EM algorithm (default is FALSE).

`max_its` An integer specifying the maximum number of iterations for the EM algorithm (default is 1e3).

`min_its` An integer specifying the minimum number of iterations for the EM algorithm (default is 10).

`priors` A list of prior hyperparameters (default is NULL). See specify_priors on how to specify the hyperparameters.

`n_interior_knots` (only relevant for `model %in% c('RS', 'RSR')`) An integer specifying the number of interior knots used in fitting a natural cubic spline for degree heterogeneity models (default is 5).

`termination_rule` A character string to specify the termination rule to determine the convergence of the EM algorithm:

- 'prob_mat': uses change in the absolute difference in $\hat{Z}$ (i.e., the $N \times K$ cluster membership probability matrix) between subsequent iterations (default)
- 'Q': uses change in the absolute difference in the objective function of the E-step evaluated using parameters from subsequent iterations
- 'ARI': comparing the classifications between subsequent iterations using adjusted Rand index

- 'NMI': comparing the classifications between subsequent iterations using normalized mutual information
- 'CER': comparing the classifications between subsequent iterations using classification error rate

tolerance A numeric specifying the tolerance used for termination_rule %in% c('Q', 'prob_mat') (default is 1e-3).

tolerance_ARI A numeric specifying the tolerance used for termination_rule = 'ARI' (default is 0.999).

tolerance_NMI A numeric specifying the tolerance used for termination_rule = 'NMI' (default is 0.999).

tolerance_CER A numeric specifying the tolerance used for termination_rule = 'CER' (default is 0.01).

n_its_start_CA An integer specifying what iteration to start computing cumulative averages (note: cumulative average of U, the latent position matrix, is not tracked when termination_rule = 'Q') (default is 20).

tolerance_diff_CA A numeric specifying the tolerance used for the change in cumulative average of termination_rule metric and U (note: cumulative average of U is not tracked when termination_rule = 'Q') (default is 1e-3).

consecutive_diff_CA An integer specifying the tolerance for the number of consecutive instances where change in cumulative average is less than tolerance_diff_CA (default is 5).

quantile_diff A numeric in [0,1] specifying the quantile used in computing the change in the absolute difference of Z and U between subsequent iterations (default is 1, i.e., max).

beta_temp_schedule A numeric vector specifying the temperature schedule for deterministic annealing (default is 1, i.e., deterministic annealing not utilized).

n_control An integer specifying the fixed number of controls (i.e., non-links) sampled for each actor; only relevant when case_control = TRUE (default is 100 when case_control = TRUE and NULL when case_control = FALSE).

n_start An integer specifying the maximum number of starts for the EM algorithm (default is 5).

max_retry An integer specifying the maximum number of re-attempts if starting values cause issues with EM algorithm (default is 5).

IC_selection A character string to specify the information criteria used to select the optimal fit based on the combinations of K, D, and n_start considered:

- 'BIC_logit': BIC computed from logistic regression component
- 'BIC_mbc': BIC computed from model based clustering component
- 'ICL_mbc': ICL computed from model based clustering component
- 'Total_BIC': sum of 'BIC_logit' and 'BIC_mbc'
- 'Total_ICL': sum of 'BIC_logit' and 'ICL_mbc' (default)

sd_random_U_GNN (only relevant when initialization = 'GNN') A positive numeric value specifying the standard deviation for the random draws from a normal distribution to initialize U (default is 1).

max_retry_GNN (only relevant when initialization = 'GNN') An integer specifying the maximum number of re-attempts for the GNN approach before switching to random starting values (default is 10).

n_its_GNN (only relevant when initialization = 'GNN') An integer specifying the maximum number of iterations for the GNN approach (default is 10).

downsampling_GNN (only relevant when initialization = 'GNN') A logical; if TRUE employs downsampling s.t. the number of links and non-links are balanced for the GNN approach (default is TRUE).

Running JANE in parallel:

JANE integrates the **future** and **future.apply** packages to fit the various combinations of K, D, and n_start in parallel. The 'Examples' section below provides an example of how to run JANE in parallel. See plan and future.apply for more details.

Choosing the number of clusters:

JANE allows for the following model selection criteria to choose the number of clusters:

- 'BIC_logit': BIC computed from logistic regression component
- 'BIC_mbc': BIC computed from model based clustering component
- 'ICL_mbc': ICL (Biernacki et al. (2000)) computed from model based clustering component
- 'Total_BIC': Sum of 'BIC_logit' and 'BIC_mbc', this is the model selection criterion proposed by Handcock et al. (2007)
- 'Total_ICL': (default) sum of 'BIC_logit' and 'ICL_mbc', this criterion is similar to 'Total_BIC', but uses ICL for the model based clustering component which tends to favor more well-separated clusters.

Warning: It is not certain whether it is appropriate to use the model selection criterion above to select D.

Value

A list of S3 class "JANE" containing the following components:

input_params — A list containing the input parameters for IC_selection, case_control, and DA_type used in the function call.

A — The square sparse adjacency matrix of class 'dgCMatrix' used in fitting the latent space cluster model. This matrix can be different than the input A matrix as isolates are removed.

IC_out — A matrix containing the relevant information criteria for all combinations of K, D, and n_start considered. The 'selected' column indicates the optimal fit chosen.

all_convergence_ind — A matrix containing the convergence information (i.e., 1 = converged, 0 = did not converge) and number of iterations for all combinations of K, D, n_start, and beta_temperature considered.

optimal_res — A list containing the estimated parameters of interest based on the optimal fit selected. It is recommended to use summary() to extract the parameters of interest. See summary.JANE for more details.

optimal_starting — A list containing the starting parameters used in the EM algorithm that resulted in the optimal fit selected. It is recommended to use summary() to extract the parameters of interest. See summary.JANE for more details.

References

Biernacki, C., Celeux, G., Govaert, G., 2000. Assessing a mixture model for clustering with the integrated completed likelihood. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 719–725.

Handcock, M.S., Raftery, A.E., Tantrum, J.M., 2007. Model-based clustering for social networks. Journal of the Royal Statistical Society Series A: Statistics in Society 170, 301–354.

Examples

```
# Simulate network
mus <- matrix(c(-1,-1,1,-1,1,1),
              nrow = 3,
              ncol = 2,
              byrow = TRUE)
omegas <- array(c(diag(rep(7,2)),
                  diag(rep(7,2)),
                  diag(rep(7,2))),
                  dim = c(2,2,3))
p <- rep(1/3, 3)
beta0 <- 1.0
sim_data <- JANE::sim_A(N = 100L,
                        model = "NDH",
                        mus = mus,
                        omegas = omegas,
                        p = p,
                        beta0 = beta0,
                        remove_isolates = TRUE)

# Run JANE on simulated data
res <- JANE::JANE(A = sim_data$A,
                  D = 2L,
                  K = 3L,
                  initialization = "GNN",
                  model = "NDH",
                  case_control = FALSE,
                  DA_type = "none")

# Run JANE on simulated data - consider multiple D and K
res <- JANE::JANE(A = sim_data$A,
                  D = 2:5,
                  K = 2:10,
                  initialization = "GNN",
                  model = "NDH",
                  case_control = FALSE,
                  DA_type = "none")

# Run JANE on simulated data - parallel with 5 cores
future::plan(future::multisession, workers = 5)
res <- JANE::JANE(A = sim_data$A,
                  D = 2L,
                  K = 3L,
```

```
                  initialization = "GNN",
                  model = "NDH",
                  case_control = FALSE,
                  DA_type = "none")
future::plan(future::sequential)

# Run JANE on simulated data - case/control approach with 20 controls sampled for each actor
res <- JANE::JANE(A = sim_data$A,
                  D = 2L,
                  K = 3L,
                  initialization = "GNN",
                  model = "NDH",
                  case_control = TRUE,
                  DA_type = "none",
                  control = list(n_control = 20))

# Reproducibility
res1 <- JANE::JANE(A = sim_data$A,
                   D = 2L,
                   K = 3L,
                   initialization = "GNN",
                   seed = 1234,
                   model = "NDH",
                   case_control = FALSE,
                   DA_type = "none")

res2 <- JANE::JANE(A = sim_data$A,
                   D = 2L,
                   K = 3L,
                   initialization = "GNN",
                   seed = 1234,
                   model = "NDH",
                   case_control = FALSE,
                   DA_type = "none")

## Check if results match
all.equal(res1, res2)

# Another reproducibility example where the seed was not set.
# It is possible to replicate the results using the starting values due to
# the nature of EM algorithms
res3 <- JANE::JANE(A = sim_data$A,
                   D = 2L,
                   K = 3L,
                   initialization = "GNN",
                   model = "NDH",
                   case_control = FALSE,
                   DA_type = "none")
## Extract starting values
start_vals <- res3$optimal_start

## Run JANE using extracted starting values, no need to specify D and K
## below as function will determine those values from start_vals
```

```
res4 <- JANE::JANE(A = sim_data$A,
                   initialization = start_vals,
                   model = "NDH",
                   case_control = FALSE,
                   DA_type = "none")

## Check if optimal_res are identical
all.equal(res3$optimal_res, res4$optimal_res)
```

`plot.JANE` *Plot JANE fits*

Description

S3 plot method for object of class "JANE".

Usage

```
## S3 method for class 'JANE'
plot(
  x,
  type = "lsnc",
  true_labels,
  initial_values = FALSE,
  zoom = 100,
  density_type = "contour",
  rotation_angle = 0,
  alpha_edge = 0.1,
  alpha_node = 1,
  swap_axes = FALSE,
  main,
  xlab,
  ylab,
  cluster_cols,
  ...
)
```

Arguments

x	An object of S3 class "JANE", a result of a call to JANE.
type	A character string to select the type of plot: • 'lsnc': plot the network using the estimated latent positions and color-code actors by cluster (default) • 'misclassified': (can only be used if true_labels is !NULL) similar to 'lsnc', but will color misclassified actors in black

- 'uncertainty': similar to 'lsnc', but here the color gradient applied represents the actor-specific classification uncertainty
- 'trace_plot': presents various trace plots across the iterations of the EM algorithm

`true_labels` (optional) A numeric, character, or factor vector of known true cluster labels. Must have the same length as number of actors in the fitted network. Need to account for potential isolates removed.

`initial_values` A logical; if `TRUE` then plots fit using the starting parameters used in the EM algorithm (default is `FALSE`, i.e., the results after the EM algorithm is run are plotted).

`zoom` A numeric value > 0 that controls the % magnification of the plot (default is 100%).

`density_type` Choose from one of the following three options: 'contour' (default), 'hdr', 'image', and 'persp' indicating the density plot type.

`rotation_angle` A numeric value that rotates the estimated latent positions and contours of the multivariate normal distributions clockwise (or counterclockwise if `swap_axes = TRUE`) through the specified angle about the origin (default is 0 degrees). Only relevant when `D` (i.e., dimension of the latent space) `>= 2` and `type != 'trace_plot'`.

`alpha_edge` A numeric value in `[0,1]` that controls the transparency of the network edges (default is 0.1).

`alpha_node` A numeric value in `[0,1]` that controls the transparency of the actors in the network (default is 1).

`swap_axes` A logical; if `TRUE` will swap the x and y axes (default is `FALSE`).

`main` An optional overall title for the plot.

`xlab` An optional title for the x axis.

`ylab` An optional title for the y axis.

`cluster_cols` An optional vector of colors for the clusters. Must have a length of at least K.

`...` Unused.

Details

The classification of actors into specific clusters is based on a hard clustering rule of $\{h|Z_{ih} = max_k Z_{ik}\}$. Additionally, the actor-specific classification uncertainty is derived as 1 - $max_k Z_{ik}$.

The trace plot contains up to five unique plots tracking various metrics across the iterations of the EM algorithm, depending on the JANE control parameter `termination_rule`:

- `termination_rule = 'prob_mat'`: Five plots will be presented. Specifically, in the top panel, the plot on the left presents the change in the absolute difference in $\hat{Z}$ (i.e., the $N \times K$ cluster membership probability matrix) between subsequent iterations. The exact quantile of the absolute difference plotted are presented in parentheses and determined by the JANE control parameter `quantile_diff`. For example, the default control parameter `quantile_diff = 1`, so the values being plotted are the max absolute difference in $\hat{Z}$ between subsequent iterations. The plot on the right of the top panel presents the absolute difference in the cumulative average of the absolute change in $\hat{Z}$ and U (i.e., the $N \times D$ matrix of latent positions) across subsequent iterations (absolute change in $\hat{Z}$ and U computed in an identical manner as described above).

This metric is only tracked beginning at an iteration determined by the `n_its_start_CA` control parameter in JANE. Note, this plot may be empty if the EM algorithm converges before the `n_its_start_CA`-th iteration. Finally, the bottom panel presents ARI, NMI, and CER values comparing the classifications between subsequent iterations, respectively. Specifically, at a given iteration we determine the classification of actors in clusters based on a hard clustering rule of $\{h|Z_{ih} = max_k Z_{ik}\}$ and given these labels from two subsequent iterations, we compute and plot the ARI, NMI and CER.

- `termination_rule = 'Q'`: Plots generated are similar to those described in the previous bullet point. However, instead of tracking the change in $\hat{Z}$ over iterations, here the absolute difference in the objective function of the E-step evaluated using parameters from subsequent iterations is tracked. Furthermore, the cumulative average of the absolute change in U is no longer tracked.
- `termination_rule %in% c('ARI', 'NMI', 'CER')`: Four plots will be presented. Specifically, the top left panel presents a plot of the absolute difference in the cumulative average of the absolute change in the specific `termination_rule` employed and U across iterations. As previously mentioned, if the EM algorithm converges before the `n_its_start_CA`-th iteration then this will be an empty plot. Furthermore, the other three plots present ARI, NMI, and CER values comparing the classifications between subsequent iterations, respectively.

Value

A plot of the network or trace plot of the EM run.

See Also

surfacePlot, adjustedRandIndex, classError, NMI

Examples

```
# Simulate network
mus <- matrix(c(-1,-1,1,-1,1,1),
              nrow = 3,
              ncol = 2,
              byrow = TRUE)
omegas <- array(c(diag(rep(7,2)),
                  diag(rep(7,2)),
                  diag(rep(7,2))),
                  dim = c(2,2,3))
p <- rep(1/3, 3)
beta0 <- 1.0
sim_data <- JANE::sim_A(N = 100L,
                        model = "NDH",
                        mus = mus,
                        omegas = omegas,
                        p = p,
                        beta0 = beta0,
                        remove_isolates = TRUE)

# Run JANE on simulated data
res <- JANE::JANE(A = sim_data$A,
```

```
                    D = 2L,
                    K = 3L,
                    initialization = "GNN",
                    model = "NDH",
                    case_control = FALSE,
                    DA_type = "none")

# plot trace plot
plot(res, type = "trace_plot")

# plot network
plot(res)

# plot network - misclassified
plot(res, type = "misclassified", true_labels = apply(sim_data$Z, 1, which.max))

# plot network - uncertainty and swap axes
plot(res, type = "uncertainty", swap_axes = TRUE)

# plot network - but only show contours of MVNs
plot(res, swap_axes = TRUE, alpha_edge = 0, alpha_node = 0)

# plot using starting values of EM algorithm
plot(res, initial_values = TRUE)
```

---

sim_A *Simulate unweighted networks from latent space cluster models*

Description

Simulate an unweighted network from a D-dimensional latent space cluster model with K clusters and N actors. The *squared* euclidean distance is used (i.e., $dist(U_i, U_j)^2$), where U_i and U_j are the respective actor's positions in an unobserved social space.

Usage

```
sim_A(
  N,
  mus,
  omegas,
  p,
  beta0,
  model,
  precision_R_effects,
  remove_isolates = TRUE
)
```

Arguments

N	An integer specifying the number of actors in the network.
mus	A numeric $K \times D$ matrix specifying the mean vectors of the multivariate normal distribution for the latent positions of the K clusters.
omegas	A numeric $D \times D \times K$ array specifying the precision matrices of the multivariate normal distribution for the latent positions of the K clusters.
p	A numeric vector of length K specifying the mixture weights of the finite multivariate normal mixture distribution for the latent positions.
beta0	A numeric value specifying the intercept parameter for the logistic regression model.
model	A character string to specify the model to simulate the network from: • 'NDH': generates an **undirected** network with no degree heterogeneity • 'RS': generates an **undirected** network with degree heterogeneity, specifically by including actor specific random sociality effects • 'RSR': generates a **directed** network with degree heterogeneity, specifically by including actor specific random sender and receiver effects
precision_R_effects	Precision parameters for random degree heterogeneity effects: • 'NDH': does not apply, can leave as missing • 'RS': a numeric value specifying the precision parameter of the normal distribution of the random sociality effect, if missing will generate from a gamma(shape = 1, rate = 1) • 'RSR': a numeric matrix specifying the precision matrix of the multivariate normal distribution of the random sender and receiver effects, if missing will generate from a Wishart(df = 3, Sigma = I_2)
remove_isolates	A logical; if TRUE then isolates from the network are removed (default is TRUE).

Value

A list containing the following components:

A	A sparse adjacency matrix of class 'dgCMatrix' representing the simulated network.
Z	A numeric $N \times K$ cluster assignment matrix with rows representing the cluster an actor belongs to (i.e. indicated by a value of 1.0).
U	A numeric $N \times D$ matrix with rows representing an actor's position in a D-dimensional social space.
RE	A numeric $N \times 1$ matrix representing the actor specific random sociality effect (i.e., s) OR a $N \times 2$ matrix representing the actor specific random sender and receiver effects (i.e., s and r, respectively).
precision_R_effects	The specific precision_R_effects used to simulate RE.
model	A character string representing the specific model used to simulate the network.

Examples

```
mus <- matrix(c(-1,-1,1,-1,1,1),
              nrow = 3,
              ncol = 2,
              byrow = TRUE)
omegas <- array(c(diag(rep(7,2)),
                  diag(rep(7,2)),
                  diag(rep(7,2))),
                  dim = c(2,2,3))
p <- rep(1/3, 3)
beta0 <- 1.0
JANE::sim_A(N = 100L,
            model = "NDH",
            mus = mus,
            omegas = omegas,
            p = p,
            beta0 = beta0,
            remove_isolates = TRUE)
```

`specify_initial_values` *Specify starting values for EM algorithm*

Description

A function that allows the user to specify starting values for the EM algorithm in a structure accepted by `JANE`.

Usage

```
specify_initial_values(
  A,
  D,
  K,
  model,
  n_interior_knots = NULL,
  U,
  omegas,
  mus,
  p,
  Z,
  beta
)
```

Arguments

A	A square matrix or sparse matrix of class 'dgCMatrix' representing the adjacency matrix of the unweighted network of interest.
D	An integer specifying the dimension of the latent positions.
K	An integer specifying the total number of clusters.
model	A character string specifying the model: • 'NDH': **undirected** network with no degree heterogeneity • 'RS': **undirected** network with degree heterogeneity • 'RSR': **directed** network with degree heterogeneity
n_interior_knots	An integer specifying the number of interior knots used in fitting a natural cubic spline for degree heterogeneity models (i.e., 'RS' and 'RSR' only; default is NULL).
U	A numeric $N \times D$ matrix with rows specifying an actor's position in a D-dimensional social space.
omegas	A numeric $D \times D \times K$ array specifying the precision matrices of the multivariate normal distribution for the latent positions of the K clusters.
mus	A numeric $K \times D$ matrix specifying the mean vectors of the multivariate normal distribution for the latent positions of the K clusters.
p	A numeric vector of length K specifying the mixture weights of the finite multivariate normal mixture distribution for the latent positions.
Z	A numeric $N \times K$ matrix with rows representing the conditional probability that an actor belongs to the cluster $K = k$ for $k = 1, \ldots, K$.
beta	A numeric vector specifying the regression coefficients for the logistic regression model. Specifically, a vector of length `1 + (model =="RS")*(n_interior_knots + 1) + (model =="RSR")*2*(n_interior_knots + 1)`.

Details

To match JANE, this function will remove isolates from the adjacency matrix A and determine the total number of actors after excluding isolates. If this is not done, errors with respect to incorrect dimensions in the starting values will be generated when executing JANE.

Similarly to match JANE, if an unsymmetric adjacency matrix A is supplied for `model %in% c('NDH', 'RS')` the user will be asked if they would like to proceed with converting A to a symmetric matrix (i.e., `A <- 1.0 * ( (A + t(A)) > 0.0 )`).

Value

A list of starting values for the EM algorithm generated from the input values in a structure accepted by JANE.

Examples

```
# Simulate network
mus <- matrix(c(-1,-1,1,-1,1,1),
              nrow = 3,
              ncol = 2,
              byrow = TRUE)
omegas <- array(c(diag(rep(7,2)),
                  diag(rep(7,2)),
                  diag(rep(7,2))),
                dim = c(2,2,3))
p <- rep(1/3, 3)
beta0 <- -1
sim_data <- JANE::sim_A(N = 100L,
                        model = "RSR",
                        mus = mus,
                        omegas = omegas,
                        p = p,
                        beta0 = beta0,
                        remove_isolates = TRUE)

# Specify starting values
D <- 3L
K <- 5L
N <- nrow(sim_data$A)
n_interior_knots <- 5L

U <- matrix(stats::rnorm(N*D), nrow = N, ncol = D)
omegas <- stats::rWishart(n = K, df = D+1, Sigma = diag(D))
mus <- matrix(stats::rnorm(K*D), nrow = K, ncol = D)
p <- extraDistr::rdirichlet(n = 1, rep(3,K))[1,]
Z <-  extraDistr::rdirichlet(n = N, alpha = rep(1, K))
beta <- stats::rnorm(n = 1 + 2*(1 + n_interior_knots))

my_starting_values <- JANE::specify_initial_values(A = sim_data$A,
                                                   D = D,
                                                   K = K,
                                                   model = "RSR",
                                                   n_interior_knots = n_interior_knots,
                                                   U = U,
                                                   omegas = omegas,
                                                   mus = mus,
                                                   p = p,
                                                   Z = Z,
                                                   beta = beta)

# Run JANE using my_starting_values (no need to specify D and K as function will
# determine those values from my_starting_values)
res <- JANE::JANE(A = sim_data$A,
                  initialization = my_starting_values,
                  model = "RSR")
```

`specify_priors` *Specify prior hyperparameters for EM algorithm*

Description

A function that allows the user to specify the prior hyperparameters for the EM algorithm in a structure accepted by `JANE`.

Usage

```
specify_priors(D, K, model, n_interior_knots = NULL, a, b, c, G, nu, e, f)
```

Arguments

`D`	An integer specifying the dimension of the latent positions.
`K`	An integer specifying the total number of clusters.
`model`	A character string specifying the model: • 'NDH': **undirected** network with no degree heterogeneity • 'RS': **undirected** network with degree heterogeneity • 'RSR': **directed** network with degree heterogeneity
`n_interior_knots`	An integer specifying the number of interior knots used in fitting a natural cubic spline for degree heterogeneity models (i.e., 'RS' and 'RSR' only; default is `NULL`).
`a`	A numeric vector of length D specifying the mean of the multivariate normal prior on μ_k for $k = 1, \ldots, K$, where μ_k represents the mean of the multivariate normal distribution for the latent positions of the k^{th} cluster.
`b`	A numeric value specifying the scaling factor on the precision of the multivariate normal prior on μ_k for $k = 1, \ldots, K$, where μ_k represents the mean of the multivariate normal distribution for the latent positions of the k^{th} cluster.
`c`	A numeric value specifying the degrees of freedom of the Wishart prior on Ω_k for $k = 1, \ldots, K$, where Ω_k represents the precision of the multivariate normal distribution for the latent positions of the k^{th} cluster.
`G`	A numeric $D \times D$ matrix specifying the inverse of the scale matrix of the Wishart prior on Ω_k for $k = 1, \ldots, K$, where Ω_k represents the precision of the multivariate normal distribution for the latent positions of the k^{th} cluster.
`nu`	A numeric vector of length K specifying the concentration parameters of the Dirichlet prior on p, where p represents the mixture weights of the finite multivariate normal mixture distribution for the latent positions.
`e`	A numeric vector of length `1 + (model =='RS')*(n_interior_knots + 1) + (model =='RSR')*2*(n_interior_knots + 1)` specifying the mean of the multivariate normal prior on β, where β represents the coefficients of the logistic regression model.

`f` A numeric square matrix of dimension `1 + (model =='RS')*(n_interior_knots + 1) + (model =='RSR')*2*(n_interior_knots + 1)` specifying the precision of the multivariate normal prior on β, where β represents the coefficients of the logistic regression model.

Details

Prior on μ_k and Ω_k (note: the same prior is used for $k = 1, \ldots, K$) :
$\pi(\mu_k, \Omega_k) = \pi(\mu_k|\Omega_k)\pi(\Omega_k)$, thus

$$\mu_k|\Omega_k \sim MVN(a, (b\Omega_k)^{-1})$$

$$\Omega_k \sim Wishart(c, G^{-1})$$

Prior on p:

For the current implementation we require that all elements of the nu vector be >= 1 to prevent against negative mixture weights for empty clusters.

$$p \sim Dirichlet(\nu_1, \ldots, \nu_K)$$

Prior on β:

$$\beta \sim MVN(e, f^{-1})$$

Value

A list of prior hyperparameters for the EM algorithm generated from the input values in a structure accepted by `JANE`.

Examples

```
# Simulate network
mus <- matrix(c(-1,-1,1,-1,1,1),
              nrow = 3,
              ncol = 2,
              byrow = TRUE)
omegas <- array(c(diag(rep(7,2)),
                  diag(rep(7,2)),
                  diag(rep(7,2))),
                  dim = c(2,2,3))
p <- rep(1/3, 3)
beta0 <- 1.0
sim_data <- JANE::sim_A(N = 100L,
                        model = "RS",
                        mus = mus,
                        omegas = omegas,
                        p = p,
                        beta0 = beta0,
                        remove_isolates = TRUE)


# Specify prior hyperparameters
```

```
D <- 3L
K <- 5L
n_interior_knots <- 5L

a <- rep(1, D)
b <- 3
c <- 4
G <- 10*diag(D)
nu <- rep(2, K)
e <- rep(0.5, 1 + (n_interior_knots + 1))
f <- diag(c(0.1, rep(0.5, n_interior_knots + 1)))

my_prior_hyperparameters <- specify_priors(D = D,
                                           K = K,
                                           model = "RS",
                                           n_interior_knots = n_interior_knots,
                                           a = a,
                                           b = b,
                                           c = c,
                                           G = G,
                                           nu = nu,
                                           e = e,
                                           f = f)

# Run JANE on simulated data using supplied prior hyperparameters
res <- JANE::JANE(A = sim_data$A,
                  D = D,
                  K = K,
                  initialization = "GNN",
                  model = "RS",
                  case_control = FALSE,
                  DA_type = "none",
                  control = list(priors = my_prior_hyperparameters))
```

`summary.JANE` *Summarizing JANE fits*

Description

S3 summary method for object of class "JANE".

Usage

```
## S3 method for class 'JANE'
summary(object, true_labels = NULL, initial_values = FALSE, ...)
```

Arguments

object	An object of S3 class "JANE", a result of a call to JANE.
true_labels	(optional) A numeric, character, or factor vector of known true cluster labels. Must have the same length as number of actors in the fitted network (default is NULL).
initial_values	A logical; if TRUE then summarize fit using the starting parameters used in the EM algorithm (default is FALSE, i.e., the results after the EM algorithm is run are summarized).
...	Unused.

Value

A list of S3 class "summary.JANE" containing the following components (Note: N is the number of actors in the network, K is the number of clusters, and D is the dimension of the latent space):

coefficients	A numeric vector representing the estimated coefficients from the logistic regression model.
p	A numeric vector of length K representing the estimated mixture weights of the finite multivariate normal mixture distribution for the latent positions.
U	A numeric $N \times D$ matrix with rows representing an actor's estimated latent position in a D-dimensional social space.
mus	A numeric $K \times D$ matrix representing the estimated mean vectors of the multivariate normal distributions for the latent positions of the K clusters.
omegas	A numeric $D \times D \times K$ array representing the estimated precision matrices of the multivariate normal distributions for the latent positions of the K clusters.
Z	A numeric $N \times K$ matrix with rows representing the estimated conditional probability that an actor belongs to the cluster $K = k$ for $k = 1, \ldots, K$.
uncertainty	A numeric vector of length N representing the uncertainty of the i^{th} actor's classification, derived as 1 - $max_k Z_{ik}$.
cluster_labels	A numeric vector of length N representing the cluster assignment of each actor based on a hard clustering rule of $\{h \vert Z_{ih} = max_k Z_{ik}\}$.
input_params	A list with the following components: • model: A character string representing the specific model used (i.e., 'NDH', 'RS', or 'RSR') • IC_selection: A character string representing the specific information criteria used to select the optimal fit (i.e., 'BIC_logit', 'BIC_mbc', 'ICL_mbc', 'Total_BIC', or 'Total_ICL') • case_control: A logical; if TRUE then the case/control approach was utilized • DA_type: A character string representing the specific deterministic annealing approach utilized (i.e., 'none', 'cooling', 'heating', or 'hybrid') • priors: A list of the prior hyperparameters used. See specify_priors for definitions.
clustering_performance	(only if true_labels is !NULL) A list with the following components:

- CER: A list with two components: (i) misclassified: The indexes of the misclassified data points in a minimum error mapping between the cluster labels and the known true cluster labels (i.e., true_labels) and (ii) errorRate: The error rate corresponding to a minimum error mapping between the cluster labels and the known true cluster labels (see classError for details)
- ARI: A numeric value representing the adjusted Rand index comparing the cluster labels and the known true cluster labels (see adjustedRandIndex for details)
- NMI: A numeric value representing the normalized mutual information comparing the cluster labels and the known true cluster labels (see NMI for details)
- confusion_matrix: A numeric table representing the confusion matrix comparing the cluster labels and the known true cluster labels.

Examples

```
# Simulate network
mus <- matrix(c(-1,-1,1,-1,1,1),
              nrow = 3,
              ncol = 2,
              byrow = TRUE)
omegas <- array(c(diag(rep(7,2)),
                  diag(rep(7,2)),
                  diag(rep(7,2))),
                  dim = c(2,2,3))
p <- rep(1/3, 3)
beta0 <- 1.0
sim_data <- JANE::sim_A(N = 100L,
                        model = "NDH",
                        mus = mus,
                        omegas = omegas,
                        p = p,
                        beta0 = beta0,
                        remove_isolates = TRUE)

# Run JANE on simulated data
res <- JANE::JANE(A = sim_data$A,
                  D = 2L,
                  K = 3L,
                  initialization = "GNN",
                  model = "NDH",
                  case_control = FALSE,
                  DA_type = "none")

# Summarize fit
summary(res)

# Summarize fit and compare to true cluster labels
summary(res, true_labels = apply(sim_data$Z, 1, which.max))
```

```
# Summarize fit using starting values of EM algorithm
summary(res, initial_values = TRUE)
```

Index

adjustedRandIndex, *10*, *20*

class, *5*, *8*, *19*
classError, *10*, *20*

future.apply, *5*

JANE, 2, *9*, *10*, *14*

NMI, *10*, *20*

plan, *5*
plot.JANE, 8

sim_A, 11
specify_initial_values, *3*, 13
specify_priors, *3*, 16, *19*
summary.JANE, *5*, 18
surfacePlot, *10*