Package: JANE (via r-universe)

October 25, 2026

Title Just Another Latent Space Network Clustering Algorithm

Version 2.1.0

Description Fit latent space network cluster models using an expectation-maximization algorithm. Enables flexible modeling of unweighted or weighted network data (with or without noise edges), supporting both directed and undirected networks (with or without degree and strength heterogeneity). Designed to handle large networks efficiently, it allows users to explore network structure through latent space representations, identify clusters (i.e., community detection) within network data, and simulate networks with varying clustering, connectivity patterns, and noise edges. Methodology for the implementation is described in Arakkal and Sewell (2025) <doi:10.1016/j.csda.2025.108228>.

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 7.3.2

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

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

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

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

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

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

Date/Publication 2025-11-03 22:00:02 UTC

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

RemoteRef HEAD

RemoteSha 05bcdd17367fa46852637b8cc2602b1be9da6ee1

Contents

JANE 2
plot.JANE 9
sim_A 13
specify_initial_values 17
specify_priors 20
summary.JANE 25

Index **29**

JANE *Fit JANE*

Description

Fit a latent space cluster model, with or without noise edges, using an EM algorithm.

Usage

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

Arguments

A	A square matrix or sparse matrix of class 'Matrix' (from the Matrix package) representing the adjacency matrix of the 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).
family	A character string specifying the distribution of the edge weights. • 'bernoulli': Expects an **unweighted** network; utilizes a Bernoulli distribution with a logit link (default) • 'lognormal': Expects a **weighted** network with positive, non-zero, continuous edge weights; utilizes a log-normal distribution with an identity link • 'poisson': Expects a **weighted** network with edge weights representing non-zero counts; utilizes a zero-truncated Poisson distribution with a log link
noise_weights	A logical; if TRUE then a Hurdle model is used to account for noise weights, if FALSE simply utilizes the supplied network (converted to an unweighted binary network if a weighted network is supplied, i.e., (A > 0.0)*1.0) and fits a latent space cluster model (default is FALSE).
guess_noise_weights	Only applicable if noise_weights = TRUE. A numeric value specifying the best guess for the mean of the noise weight distribution for family %in% c('lognormal', 'poisson') (mean is on the log-scale for lognormal) **OR** proportion (i.e. in (0,1)) of all edges that are noise edges for family = 'bernoulli'. If NULL (i.e., default) and noise_weights = TRUE then the 1st percentile of the non-zero weights will be used for family %in% c('lognormal', 'poisson') and 1% will be used for family = 'bernoulli'.
model	A character string specifying the model to fit: • 'NDH': **undirected** network with no degree heterogeneity (or connection strength heterogeneity if working with weighted network) • 'RS': **undirected** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network) • 'RSR': **directed** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network)
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 S3 class "JANE.initial_values" representing initial values for the EM algorithm. See specify_initial_values for details on how to specify initial values
case_control	A logical; if TRUE then uses a case-control approximation approach (default is FALSE).

DA_type (Experimental) A character string to specify the type of deterministic annealing approach to use

- 'none': does not employ a deterministic annealing approach (default)
- 'cooling': (Experimental) employs a traditional deterministic annealing approach where temperature decreases
- 'heating': (Experimental) employs a deterministic anti-annealing approach where temperature increases
- 'hybrid': (Experimental) employs 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

Isolates are removed from the adjacency matrix A. 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 )`); only able to do so if `family = 'bernoulli'`. Additionally, if a weighted network is supplied and `noise_weights = FALSE`, then the network will be converted to an unweighted binary network (i.e., (A > 0.0)*1.0) and a latent space cluster model is fit.

`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 S3 `class` `"JANE.priors"` representing prior hyperparameter specifications (default is `NULL`). See `specify_priors` for details 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 (and connection strength heterogeneity if working with weighted network) 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 $\hat{Z}^U$ (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 the change in cumulative averages (note: the change in the cumulative average of $\hat{U}$, 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 $\hat{U}$ (note: the change in the cumulative average of $\hat{U}$ 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 the 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 $\hat{Z}^U$ and $\hat{U}$ between subsequent iterations (default is 1, i.e., max).

beta_temp_schedule (Experimental) 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_model': BIC computed from logistic regression or Hurdle model component
- 'BIC_mbc': BIC computed from model based clustering component
- 'ICL_mbc': ICL computed from model based clustering component
- 'Total_BIC': sum of 'BIC_model' and 'BIC_mbc'
- 'Total_ICL': sum of 'BIC_model' 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 (smaller values are favored):

- 'BIC_model': BIC computed from logistic regression or Hurdle model 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_model' and 'BIC_mbc', this is the model selection criterion proposed by Handcock et al. (2007)
- 'Total_ICL': (default) sum of 'BIC_model' 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.

Based on simulation studies, Biernacki et al. (2000) recommends that when the interest in mixture modeling is cluster analysis, instead of density estimation, the ICL_{mbc} criterion should be favored over the BIC_{mbc} criterion, as the BIC_{mbc} criterion tends to overestimate the number of clusters. The BIC_{mbc} criterion is designed to choose the number of components in a mixture model rather than the number of 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, DA_type, model, family, and noise_weights 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 chosen optimal fit.
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 of S3 `class` "`JANE.initial_values`" 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,
                        params_LR = list(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 (NOT RUN)
# 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,
  remove_noise_edges = FALSE,
  ...
)
```

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.
remove_noise_edges	(only applicable if JANE was run with noise_weights = TRUE) A logical; if TRUE will remove noise edges based on hard clustering rule of $\{h\|\hat{Z}^W_{eh} = max(\hat{Z}^W_{e1}, \hat{Z}^W_{e2})\}$ for $e = 1, \ldots, \|E\|$, where $\hat{Z}^W_{e1}$ and $\hat{Z}^W_{e2}$ are the estimated conditional probabilities that the e^{th} edge is a non-noise and noise edge, respectively (default is FALSE).
...	Unused.

Details

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

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 $\hat{Z}^U$ (i.e., the $N \times K$ cluster membership probability matrix) between subsequent iterations and, if `noise_weights = TRUE`, the change in the absolute difference in $\hat{Z}^W$ (i.e., the $|E| \times 2$ edge weight 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 $\hat{Z}^U$ (and potentially $\hat{Z}^W$) 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 $\hat{Z}^U$ (and potentially $\hat{Z}^W$) and $\hat{U}$ (i.e., the $N \times D$ matrix of latent positions) across subsequent iterations (absolute change in $\hat{Z}^U$, $\hat{Z}^W$, and $\hat{U}$ are computed in an identical manner as described previously). 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 $\{h|\hat{Z}^U_{ih} = max_k \hat{Z}^U_{ik}\}$ 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 $\hat{Z}^U$ (and potentially $\hat{Z}^W$) 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 $\hat{U}$ 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 $\hat{U}$ 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.

Note

If an error interrupts the plotting process, the graphics device may be left in a state where par("new") = TRUE. This can cause subsequent plots to be overlaid. To reset the graphics state, call plot.new() or close and reopen the device with dev.off(); dev.new().

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,
                        params_LR = list(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_U, 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 or weighted networks, with or without noise edges, from latent space cluster models*

Description

Simulate an unweighted or weighted network, with or without noise edges, 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 a D-dimensional social space.

Usage

```
sim_A(
  N,
  mus,
  omegas,
  p,
  model = "NDH",
  family = "bernoulli",
  params_LR,
  params_weights = NULL,
  noise_weights_prob = 0,
  mean_noise_weights,
  precision_noise_weights,
  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 K D-variate normal distributions for the latent positions.
`omegas`	A numeric $D \times D \times K$ array specifying the precision matrices of the K D-variate normal distributions for the latent positions.
`p`	A numeric vector of length K specifying the mixture weights of the finite multivariate normal mixture distribution for the latent positions.
`model`	A character string specifying the type of model used to simulate the network: • 'NDH': generates an **undirected** network with no degree heterogeneity (or connection strength heterogeneity if working with weighted network) • 'RS': generates an **undirected** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network), specifically by including actor specific random sociality effects • 'RSR': generates a **directed** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network), specifically by including actor specific random sender and receiver effects

`family` A character string specifying the distribution of the edge weights.

- 'bernoulli': generates an **unweighted** network from a latent space cluster model
- 'lognormal': generates a **weighted** network by first simulating an unweighted network using a latent space cluster model, and then assigning edge weights based on a log-normal GLM utilizing an identity link
- 'poisson': generates a **weighted** network by first simulating an unweighted network using a latent space cluster model, and then assigning edge weights based on a zero-truncated Poisson GLM utilizing a log link

`params_LR` A list containing the parameters of the logistic regression model to simulate the unweighted network, including:

- 'beta0': a numeric value specifying the intercept parameter for the logistic regression model
- 'precision_R_effects': precision parameters for random degree heterogeneity effects, specific to the logistic regression model:
  - '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)

`params_weights` Only relevant when `family %in% c('lognormal', 'poisson')`. A list containing the parameters of the GLMs for the edge weights, including:

- 'beta0': a numeric value specifying the intercept parameter for the zero-truncated Poisson or log-normal GLM
- 'precision_R_effects': precision parameters for random connection strength heterogeneity effects, specific to the zero-truncated Poisson or log-normal GLM:
  - '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)
- 'precision_weights': a positive, non-zero, numeric representing the precision (on the log scale) of the log-normal weight distribution. Only relevant when `family = 'lognormal'`

`noise_weights_prob`
A numeric in [0,1] representing the proportion of all edges in the simulated network that are noise edges (default is 0.0).

`mean_noise_weights`
A numeric representing the mean of the noise weight distribution. Only relevant when `family %in% c('lognormal', 'poisson')` and `noise_weights_prob>0.0`.

For family = 'poisson' value has to be > 0.0, for family = "lognormal" the mean is on the log scale.

`precision_noise_weights`
: A positive, non-zero, numeric representing the precision (on the log scale) of the log-normal noise weight distribution. Only relevant when `family = 'lognormal'` and `noise_weights_prob>0.0`.

`remove_isolates`
: A logical; if `TRUE` then isolates from the network are removed (default is `TRUE`).

Details

The returned scalar `q_prob` represents the proportion of non-edges in the simulated network to be converted to noise edges, computed as $\frac{p_{noise} \times D_A}{(1-D_A) \times (1-p_{noise})}$, where D_A is the density of the simulated network without noise and p_{noise} is the inputted `noise_weights_prob`.

Value

A list containing the following components:

`A`
: A sparse adjacency matrix of class 'dgCMatrix' representing the "true" underlying unweighted network with no noise edges.

`W`
: A sparse adjacency matrix of class 'dgCMatrix' representing the unweighted or weighted network, with or without noise. Note, if `family = 'bernoulli'` and `noise_weights_prob = 0`, then `A = W`.

`q_prob`
: A numeric scalar representing the proportion of non-edges in the "true" underlying network converted to noise edges. See 'Details' for how this value is computed.

`Z_U`
: 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).

`Z_W`
: A numeric $|E| \times 4$ edge weight cluster assignment matrix, with $|E|$ representing the total number of edges in the network (for undirected networks, only the upper diagonal edges are retained). The first two columns (i.e., 'i' and 'j') contains the specific indices of the edge between the i^{th} and j^{th} actors, the third column (i.e., 'weight') contains the specific edge weight, and the fourth column (i.e., 'Z_W') represents a noise-cluster label, where 1 denotes a non-noise edge and 2 denotes a noise edge. Will be `NULL` if `noise_weights_prob = 0`.

`U`
: A numeric $N \times D$ matrix with rows representing an actor's position in a D-dimensional social space.

`mus`
: The inputted numeric $K \times D$ `mus` matrix.

`omegas`
: The inputted numeric $D \times D \times K$ `omegas` array.

`p`
: The inputted numeric vector `p` of length K.

`noise_weights_prob`
: The inputted numeric scalar `noise_weights_prob`.

`mean_noise_weights`
: The inputted numeric scalar `mean_noise_weights`. Will be `NULL` if `noise_weights_prob = 0`.

precision_noise_weights
: The inputted numeric scalar `precision_noise_weights`. Will be `NULL` if `noise_weights_prob = 0`.

model
: The inputted `model` character string.

family
: The inputted `family` character string.

params_LR
: The inputted `params_LR` list. If `model != "NDH"`, will have an additional element "RE" containing 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).

params_weights
: The inputted `params_weights` list. If `model != "NDH"`, will have an additional element "RE" containing 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).

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

# Simulate an undirected, unweighted network, with no noise and no degree heterogeneity
JANE::sim_A(N = 100L,
            model = "NDH",
            mus = mus,
            omegas = omegas,
            p = p,
            params_LR = list(beta0 = beta0),
            remove_isolates = TRUE)

# Simulate a directed, weighted network, with degree and strength heterogeneity but no noise
JANE::sim_A(N = 100L,
            model = "RSR",
            family = "lognormal",
            mus = mus,
            omegas = omegas,
            p = p,
            params_LR = list(beta0 = beta0),
            params_weights = list(beta0 = 2,
                                  precision_weights = 1),
            remove_isolates = TRUE)

# Simulate an undirected, weighted network, with noise and degree and strength heterogeneity
JANE::sim_A(N = 100L,
```

```
model = "RS",
family = "poisson",
mus = mus,
omegas = omegas,
p = p,
params_LR = list(beta0 = beta0),
params_weights = list(beta0 = 2),
noise_weights_prob = 0.1,
mean_noise_weights = 1,
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,
  family = "bernoulli",
  noise_weights = FALSE,
  n_interior_knots = NULL,
  U,
  omegas,
  mus,
  p,
  Z,
  beta,
  beta2,
  precision_weights,
  precision_noise_weights
)
```

Arguments

A	A square matrix or sparse matrix of class 'dgCMatrix' representing the adjacency matrix of the 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 (or connection strength heterogeneity if working with weighted network)
- 'RS': **undirected** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network)
- 'RSR': **directed** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network)

`family` A character string specifying the distribution of the edge weights.

- 'bernoulli': for **unweighted** networks; utilizes a Bernoulli distribution with a logit link (default)
- 'lognormal': for **weighted** networks with positive, non-zero, continuous edge weights; utilizes a log-normal distribution with an identity link
- 'poisson': for **weighted** networks with edge weights representing non-zero counts; utilizes a zero-truncated Poisson distribution with a log link

`noise_weights` A logical; if TRUE then a Hurdle model is used to account for noise weights, if FALSE simply utilizes the supplied network (converted to an unweighted binary network if a weighted network is supplied, i.e., (A > 0.0)*1.0) and fits a latent space cluster model (default is FALSE).

`n_interior_knots`
An integer specifying the number of interior knots used in fitting a natural cubic spline for degree heterogeneity (and connection strength heterogeneity if working with weighted network) 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 K D-variate normal distributions for the latent positions.

`mus` A numeric $K \times D$ matrix specifying the mean vectors of the K D-variate normal distributions for the latent positions.

`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)`.

`beta2` A numeric vector specifying the regression coefficients for the zero-truncated Poisson or log-normal GLM. Specifically, a vector of length
`1 + (model =="RS")*(n_interior_knots + 1) +`
`(model =="RSR")*2*(n_interior_knots + 1)`.
Only relevant when `noise_weights = TRUE & family != 'bernoulli'`.

precision_weights
: A positive numeric scalar specifying the precision (on the log scale) of the log-normal weight distribution. Only relevant when `noise_weights = TRUE & family = 'lognormal'`.

precision_noise_weights
: A positive numeric scalar specifying the precision (on the log scale) of the log-normal noise weight distribution. Only relevant when `noise_weights = TRUE & family = 'lognormal'`.

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 )`). Additionally, if a weighted network is supplied and `noise_weights = FALSE`, then the network will be converted to an unweighted binary network (i.e., (A > 0.0)*1.0).

Value

A list of S3 class "JANE.initial_values" representing starting values for the EM algorithm, 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,
                        params_LR = list(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 = 2,
  K = 2,
  model,
  family = "bernoulli",
  noise_weights = FALSE,
  n_interior_knots = NULL,
  a,
  b,
  c,
  G,
  nu,
```

```
  e,
  f,
  h,
  l,
  e_2,
  f_2,
  m_1,
  o_1,
  m_2,
  o_2
)
```

Arguments

D	An integer specifying the dimension of the latent positions (default is 2).
K	An integer specifying the total number of clusters (default is 2).
model	A character string specifying the model: • 'NDH': **undirected** network with no degree heterogeneity (or connection strength heterogeneity if working with weighted network) • 'RS': **undirected** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network) • 'RSR': **directed** network with degree heterogeneity (and connection strength heterogeneity if working with weighted network)
family	A character string specifying the distribution of the edge weights. • 'bernoulli': for **unweighted** networks; utilizes a Bernoulli distribution with a logit link (default) • 'lognormal': for **weighted** networks with positive, non-zero, continuous edge weights; utilizes a log-normal distribution with an identity link • 'poisson': for **weighted** networks with edge weights representing non-zero counts; utilizes a zero-truncated Poisson distribution with a log link
noise_weights	A logical; if TRUE then a Hurdle model is used to account for noise weights, if FALSE simply utilizes the supplied network (converted to an unweighted binary network if a weighted network is supplied, i.e., (A > 0.0)*1.0) and fits a latent space cluster model (default is FALSE).
n_interior_knots	An integer specifying the number of interior knots used in fitting a natural cubic spline for degree heterogeneity (and connection strength heterogeneity if working with weighted network) 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 strictly positive numeric scalar 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 positive numeric scalar $> D$ 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 positive definite $D \times D$ 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 positive numeric vector of length K with elements ≥ 1 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_{LR}, where β_{LR} represents the coefficients of the logistic regression model.

f — A numeric positive definite 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_{LR}, where β_{LR} represents the coefficients of the logistic regression model.

h — A positive numeric scalar ≥ 1 specifying the first shape parameter for the Beta prior on q, where q is the proportion of non-edges in the "true" underlying network converted to noise edges. Only relevant when `noise_weights = TRUE`.

l — A strictly positive numeric scalar specifying the second shape parameter for the Beta prior on q, where q is the proportion of non-edges in the "true" underlying network converted to noise edges. Only relevant when `noise_weights = TRUE`.

e_2 — 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_{GLM}, where β_{GLM} represents the coefficients of the zero-truncated Poisson or log-normal GLM. Only relevant when `noise_weights = TRUE & family != 'bernoulli'`.

f_2 — A numeric positive definite 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_{GLM}, where β_{GLM} represents the coefficients of the zero-truncated Poisson or log-normal GLM. Only relevant when `noise_weights = TRUE & family != 'bernoulli'`.

m_1 — A positive numeric scalar > 1 specifying the shape parameter for the Gamma prior on $\tau^2_{weights}$, where $\tau^2_{weights}$ is the precision (on the log scale) of the log-normal weight distribution. Note, this value is scaled by 0.5, see 'Details'. Only relevant when `noise_weights = TRUE & family = 'lognormal'`.

o_1 — A positive numeric scalar > 0 specifying the rate parameter for the Gamma prior on $\tau^2_{weights}$, where $\tau^2_{weights}$ is the precision (on the log scale) of the log-normal weight distribution. Note, this value is scaled by 0.5, see 'Details'. Only relevant when `noise_weights = TRUE & family = 'lognormal'`.

m_2 — A positive numeric scalar ≥ 2 specifying the shape parameter for the Gamma prior on $\tau^2_{noise\ weights}$, where $\tau^2_{noise\ weights}$ is the precision (on the log scale)

of the log-normal noise weight distribution. Note, this value is scaled by 0.5, see 'Details'. Only relevant when `noise_weights = TRUE & family = 'lognormal'`.

`o_2` A positive numeric scalar > 0 specifying the rate parameter for the Gamma prior on $\tau^2_{noise\ weights}$, where $\tau^2_{noise\ weights}$ is the precision (on the log scale) of the log-normal noise weight distribution. Note, this value is scaled by 0.5, see 'Details'. Only relevant when `noise_weights = TRUE & family = 'lognormal'`.

Details

Prior on $\boldsymbol{\mu}_k$ and $\boldsymbol{\Omega}_k$ (note: the same prior is used for $k = 1, \ldots, K$) :

$$\boldsymbol{\Omega}_k \sim Wishart(c, \boldsymbol{G}^{-1})$$
$$\boldsymbol{\mu}_k | \boldsymbol{\Omega}_k \sim MVN(\boldsymbol{a}, (b\boldsymbol{\Omega}_k)^{-1})$$

Prior on $\boldsymbol{p}$:

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

$$\boldsymbol{p} \sim Dirichlet(\nu_1, \ldots, \nu_K)$$

Prior on $\boldsymbol{\beta}_{LR}$:

$$\boldsymbol{\beta}_{LR} \sim MVN(\boldsymbol{e}, \boldsymbol{F}^{-1})$$

Prior on q:

$$q \sim Beta(h, l)$$

Zero-truncated Poisson

Prior on $\boldsymbol{\beta}_{GLM}$:

$$\boldsymbol{\beta}_{GLM} \sim MVN(\boldsymbol{e}_2, \boldsymbol{F}_2^{-1})$$

Log-normal

Prior on $\tau^2_{weights}$:

$$\tau^2_{weights} \sim Gamma(\frac{m_1}{2}, \frac{o_1}{2})$$

Prior on $\boldsymbol{\beta}_{GLM}$:

$$\boldsymbol{\beta}_{GLM} | \tau^2_{weights} \sim MVN(\boldsymbol{e}_2, (\tau^2_{weights}\boldsymbol{F}_2)^{-1})$$

Prior on $\tau^2_{noise\ weights}$:

$$\tau^2_{noise\ weights} \sim Gamma(\frac{m_2}{2}, \frac{o_2}{2})$$

Unevaluated calls can be supplied as values for specific hyperparameters. This is particularly useful when running `JANE` for multiple combinations of `K` and `D`. See 'examples' section below for implementation examples.

Value

A list of S3 `class` `"JANE.priors"` representing prior hyperparameters for the EM algorithm, 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,
                        params_LR = list(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))

# Specify prior hyperparameters as unevaluated calls
n_interior_knots <- 5L
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(model = "RS",
                                           n_interior_knots = n_interior_knots,
                                           a = quote(rep(1, D)),
                                           b = b,
                                           c = quote(D + 1),
                                           G = quote(10*diag(D)),
                                           nu = quote(rep(2, K)),
                                           e = e,
                                           f = f)

# # Run JANE on simulated data using supplied prior hyperparameters (NOT RUN)
# future::plan(future::multisession, workers = 5)
# res <- JANE::JANE(A = sim_data$A,
#                   D = 2:5,
#                   K = 2:10,
#                   initialization = "GNN",
#                   model = "RS",
#                   case_control = FALSE,
#                   DA_type = "none",
#                   control = list(priors = my_prior_hyperparameters))
# future::plan(future::sequential)
```

`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. Need to account for potential isolates removed (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 list containing the estimated coefficients from the logistic regression model (i.e., 'beta_LR') and, if relevant, the estimated coefficients from the zero- truncated Poisson or log-normal GLM (i.e., 'beta_GLM').

`U` A numeric $N \times D$ matrix with rows containing an actor's estimated latent position in a D-dimensional social space.

`p` A numeric vector of length K containing the estimated mixture weights of the finite multivariate normal mixture distribution for the latent positions.

`mus` A numeric $K \times D$ matrix containing the estimated mean vectors of the K D-variate normal distributions for the latent positions.

`omegas` A numeric $D \times D \times K$ array containing the estimated precision matrices of the K D-variate normal distributions for the latent positions.

`Z_U` A numeric $N \times K$ matrix with rows containing 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 containing the uncertainty of the i^{th} actor's classification, derived as 1 - $max_k \hat{Z}^U_{ik}$.

`cluster_labels` A numeric vector of length N containing the cluster assignment of each actor based on a hard clustering rule of $\{h|\hat{Z}^U_{ih} = max_k \hat{Z}^U_{ik}\}$.

`Z_W` A numeric $|E| \times 6$ matrix, with $|E|$ representing the total number of edges in the network (for undirected networks, only the upper diagonal edges are retained). The first two columns (i.e., 'i' and 'j') contains the specific indices of the edge between the i^{th} and j^{th} actors, the third column (i.e., 'weight') contains the specific edge weight, the fourth column (i.e., 'hat_zij1') contains the estimated conditional probability that the specific edge is a non-noise edge, the fifth column (i.e., 'hat_zij2') contains the estimated conditional probability that the specific edge is a noise edge, and the sixth column (i.e., 'noise_edge_cluster_labels') contains the noise-edge cluster assignment of each edge based on a hard clustering rule of $\{h|\hat{Z}^W_{eh} = max(\hat{Z}^W_{e1}, \hat{Z}^W_{e2})\}$ for $e = 1, \ldots, |E|$, where $\hat{Z}^W_{e1}$ and $\hat{Z}^W_{e2}$ are the estimated conditional probabilities that the e^{th} edge is a non-noise and noise edge, respectively (labels defined as, 1: non-noise edge and 2: noise edge). Will be `NULL` if `noise_weights` = `FALSE` or `initial_values` = `TRUE`.

`q_prob` A numeric scalar representing the estimated proportion of non-edges in the "true" unobserved network that were converted to noise edges.

precision_weights
: A numeric scalar representing the estimated precision (on the log scale) of the log-normal weight distribution. Only relevant for family = 'lognormal' & noise_weights = TRUE.

precision_noise_weights
: A numeric scalar representing the estimated precision (on the log scale) of the log-normal noise weight distribution. Only relevant for family = 'lognormal' & noise_weights = TRUE.

IC
: Information criteria values of the optimal fit selected, including

- 'BIC_model': BIC computed from logistic regression or Hurdle model component
- 'BIC_mbc': BIC computed from model based clustering component
- 'ICL_mbc': ICL computed from model based clustering component
- 'Total_BIC': sum of 'BIC_model' and 'BIC_mbc'
- 'Total_ICL': sum of 'BIC_model' and 'ICL_mbc'

input_params
: A list with the following components:

- model: A character string containing the specific model used (i.e., 'NDH', 'RS', or 'RSR')
- family: A character string containing the specific family used (i.e., 'bernoulli', 'poisson', or 'lognormal')
- noise_weights: A logical; if TRUE then the approach utilizing a Hurdle model accounting for noise edges was utilized
- IC_selection: A character string containing the specific information criteria used to select the optimal fit (i.e., 'BIC_model', '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 containing 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 indices of the misclassified actors 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 containing the adjusted Rand index comparing the cluster labels and the known true cluster labels (see adjustedRandIndex for details)
- NMI: A numeric value containing the normalized mutual information comparing the cluster labels and the known true cluster labels (see NMI for details)
- confusion_matrix: A numeric table containing 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,
                        params_LR = list(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_U, 1, which.max))

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

Index

adjustedRandIndex, *12*, *27*

class, *3, 4*, *6, 7*, *10*, *19*, *23*, *25, 26*
classError, *12*, *27*

future.apply, *6*

JANE, 2, *10, 11*, *17*, *19, 20*, *25*

NMI, *12*, *27*

plan, *6*
plot.JANE, 9

sim_A, 13
specify_initial_values, *3*, 17
specify_priors, *4*, 20, *27*
summary.JANE, *7*, 25
surfacePlot, *12*