Package 'tip'

November 14, 2022

Type Package

Title Bayesian Clustering Using the Table Invitation Prior (TIP)

Version 0.1.0

Description Cluster data without specifying the number of clusters using the Table Invitation Prior (TIP) introduced in the paper ``Clustering Gene Expression Using the Table Invitation Prior'' by Charles W. Harrison, Qing He, and Hsin-Hsiung Huang (2022) <doi:10.3390/genes13112036>. TIP is a Bayesian prior that uses pairwise distance and similarity information to cluster vectors, matrices, or tensors.

License MIT + file LICENSE

Encoding UTF-8

Imports rlang, igraph, network, ggplot2, GGally, LaplacesDemon, changepoint, parallel, doParallel, foreach, methods, mniw

RoxygenNote 7.2.1

Suggests knitr, sna, mcclust, SMFilter, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Language en-US

Config/testthat/edition 3

NeedsCompilation no

Author Charles W. Harrison [cre, aut, cph], Qing He [aut, cph], Hsin-Hsiung Huang [aut, cph]

Maintainer Charles W. Harrison <charleswharrison@knights.ucf.edu>

Repository CRAN

Date/Publication 2022-11-14 17:30:02 UTC

R topics documented:

`bcm-class` *Bayesian Clustering Model (bcm) S4 class.*

Description

An S4 class to store the results of the Gibbs sampler.

Value

An object of class bcm.

Slots

`n` Positive integer: the sample size (i.e., the number of subjects).

`burn` Non-negative integer: the number of burn-in iterations in the Gibbs sampler.

`samples` Positive integer: the number of sampling iterations in the Gibbs sampler.

`posterior_assignments` List of vectors of positive integers: a list of vectors of cluster assignments (i.e., positive integers) for each sampling iteration in the Gibbs sampler.

`posterior_similarity_matrix` Matrix: a matrix where the (i,j)th element is the posterior probability that subject i and subject j belong to the same cluster.

`posterior_number_of_clusters` Vector of positive integers: each vector element is the number of clusters after posterior sampling for each sampling iteration in the Gibbs sampler.

`prior_name` Character: the name of the prior used.

`likelihood_name` Character: the name of the likelihood used.

get_cpt_neighbors *Estimate the number of similar subjects*

Description

Estimate the number of similar subjects using univariate multiple change point detection (i.e., binary segmentation in the changepoint package).

Usage

```
get_cpt_neighbors(.distance_matrix)
```

Arguments

`.distance_matrix`
: Matrix: a symmetric n x n matrix of distance values.

Value

Vector of positive integers: a vector of positive integers where the (i)th integer corresponds to the number of subjects (observations) that are similar to the (i)th subject.

Examples

```
# Import the tip library
library(tip)

# Choose an arbitrary random seed
set.seed(007)

# Generate some data (i.e., 20 subjects described by a 5 x 1 vector)
X <- matrix(rnorm(10*10),nrow=20,ncol=5)

# Compute the pairwise distances between the subjects
distance_matrix <- data.matrix(dist(X))

# For each subject, find the estimate for the number of similar subjects
get_cpt_neighbors(.distance_matrix = distance_matrix)
```

ggnet2_network_plot *Visualize the posterior similarity matrix (i.e., posterior probability matrix)*

Description

A function that produces a ggnet2 network plot to visualize the posterior similarity matrix (i.e., the matrix of posterior probabilities).

Usage

```
ggnet2_network_plot(
  .matrix_graph,
  .subject_names = vector(),
  .subject_class_names = vector(),
  .class_colors,
  .class_shapes,
  .random_seed = 7,
  .node_size = 6,
  .add_node_labels = TRUE
)
```

Arguments

`.matrix_graph`	Matrix: a matrix M where each element Mij corresponds to the posterior probability that the (i)th subject and the (j)th subject are in the same cluster.
`.subject_names`	Vector of characters: an optional vector of subject names that will appear in the graph plot.
`.subject_class_names`	Vector of characters: an optional vector of class names corresponding to each subject (i.e. vertex in the graph) which influences each vertex's color and shape. For example, the subject class names can be the true label (for the purpose of research) or it can be any other label that analyst chooses.
`.class_colors`	Named vector of characters: an optional named vector of colors that correspond to each unique value in `.subject_class_names`. The vector names are required to be the unique .subject_class_names whereas the vector values are required to be the colors.
`.class_shapes`	Named vector of integers: an optional named vector of shapes that correspond to each unique value in the `.subject_class_names`. The vector names are required to be the unique `.subject_class_names` whereas the vector values are required to be positive integers (i.e., pch values like 15, 16, 17, and so on).
`.random_seed`	Numeric: the plot uses the random layout, so set a seed for reproducibility.
`.node_size`	Positive integer: the size of each node (i.e., vertex) in the graph plot.
`.add_node_labels`	Boolean (i.e., TRUE or FALSE): should individual node labels be added to each node (i.e., vertex) in the graph plot?

Value

ggnet2 network plot: a network plot with respect to the undirected network given by .matrix_graph. This is used to visualize the posterior similarity matrix.

Examples

```
# Import the tip library
```

```
library(tip)

# Choose an arbitrary random seed to generate the data
set.seed(4*8*15*16*23*42)

# Generate a symmetric posterior probability matrix
# Each element is the probability that the two subjects belong
# to the same cluster
n1 <- 10
posterior_prob_matrix <- matrix(NA, nrow = n1, ncol = n1)
for(i in 1:n1){
  for(j in i:n1){
    if(i != j){
      posterior_prob_matrix[i,j] <- runif(n=1,min=0,max=1)
      posterior_prob_matrix[j,i] <- posterior_prob_matrix[i,j]
    }else{
      posterior_prob_matrix[i,j] <- 1.0
    }
  }
}

# --- BEGIN GRAPH PLOT 1: NO COLORS OR NODE LABELS ---

# Set an arbitrary random seed
random_seed <- 815

# Set add_node_labels to FALSE
add_node_labels <- FALSE

# Set the node size
node_size <- 6

# Construct the graph plot
ggnet2_network_plot(.matrix_graph = posterior_prob_matrix,
                    .subject_names = vector(),
                    .subject_class_names = vector(),
                    .class_colors = vector(),
                    .class_shapes = vector(),
                    .random_seed = random_seed,
                    .node_size = node_size,
                    .add_node_labels = add_node_labels)

# --- END GRAPH PLOT 1: NO COLORS OR NODE LABELS ---

# --- BEGIN GRAPH PLOT 2: NO COLORS, BUT ADD NODE LABELS ---

# Set an arbitrary random seed
random_seed <- 815

# Add node labels to the plot
add_node_labels <- TRUE

# Set graph nodes (i.e. subject identifier) a larger size
```

```
node_size <- 6

# Add subject names to the plot
subject_names <- paste("S", 1:n1, sep = "")

# Construct the graph plot
ggnet2_network_plot(.matrix_graph = posterior_prob_matrix,
                    .subject_names = subject_names,
                    .subject_class_names = vector(),
                    .class_colors = vector(),
                    .class_shapes = vector(),
                    .random_seed = random_seed,
                    .node_size = 6,
                    .add_node_labels = TRUE)

# --- END GRAPH PLOT 2: NO COLORS, BUT ADD NODE LABELS ---


# --- BEGIN GRAPH PLOT 3: ADD COLORS AND NODE LABELS ---

# Set an arbitrary random seed
random_seed <- 815

# Add node labels to the plot
add_node_labels <- TRUE

# Set graph nodes (i.e. subject identifier) a larger size
node_size <- 10

# Add subject names to the plot
subject_names <- paste("S", 1:n1, sep = "")

# Create a vector of class labels
subject_class_names <- c("Class2","Class2","Class1","Class2","Class1",
                             "Class1","Class2","Class1","Class1","Class2")

# Assign a color to each class; this can be a character value or a hex value
class_colors <- c("Class1" = "skyblue", "Class2" = "#FF9999")

# Assign a pch integer value to each class
class_shapes <- c("Class1" = 16, "Class2" = 17)

# Generate the plot
ggnet2_network_plot(.matrix_graph = posterior_prob_matrix,
                    .subject_names = subject_names,
                    .subject_class_names = subject_class_names,
                    .class_colors = class_colors,
                    .class_shapes = class_shapes,
                    .random_seed = random_seed,
                    .node_size = node_size,
                    .add_node_labels = add_node_labels)

# --- END GRAPH PLOT 3: ADD COLORS AND NODE LABELS ---
```

`ggplot_line_point` *Plot connected points using ggplot2*

Description

A function to that produces a ggplot2 plot of .y versus .x where points are added via geom_point() and the points are connected via geom_line().

Usage

```
ggplot_line_point(.x, .y, .xlab = "", .ylab = "")
```

Arguments

`.x`	The variable on the horizontal axis.
`.y`	The variable on the vertical axis.
`.xlab`	Character: the label on the horizontal axis.
`.ylab`	Character: the label on the vertical axis.

Value

ggplot2 geom_line + geom_point plot: a ggplot2 plot of .y versus .x with a label .xlab on the horizontal axis and label .ylab on the vertical axis.

Examples

```
# Import the tip library
library(tip)

# Create the variable that appears on the horizontal axis
x <- 1:10

# Create the variable that appears on the vertical axis
y <- rnorm(n = length(x), mean = 3, sd = 1)

# Create a label that appears on the horizontal axis
xlab <- "x"

# Create a label that appears on the vertical axis
ylab <- "y"

# Create the plot of y versus x with
ggplot_line_point(.x = x, .y = y, .xlab = xlab, .ylab = ylab)
```

`ggplot_number_of_clusters_hist` *Plot the posterior distribution of the number of clusters.*

Description

A function that produces a ggplot bar chart (i.e., geom_bar) that corresponds to the posterior number of clusters. The vertical axis is normalized so that it displays the posterior probability.

Usage

```
ggplot_number_of_clusters_hist(.posterior_number_of_clusters)
```

Arguments

`.posterior_number_of_clusters`
: Vector of positive integers: each integer corresponds to the number of clusters after posterior sampling for a given sampling iteration in the Gibbs sampler.

Value

ggplot2 geom_bar plot: a plot of the distribution of the posterior number of clusters computed after each sampling iteration in the Gibbs sampler.

Examples

```
# Import the tip library
library(tip)

# Generate a vector of positive integers
# Example: the posterior number of clusters computed after posterior
# sampling in each sampling iteration of the Gibbs sampler.
num_clusters <- c(1,2,2,2,2,3,3,1,2,3,3,3,1,3)

# Generate the plot of the posterior number of clusters versus the
# sampling iteration number in the Gibbs sampler.
ggplot_number_of_clusters_hist(.posterior_number_of_clusters = num_clusters)
```

`ggplot_number_of_clusters_trace` *Plot the trace plot of the posterior number of clusters*

Description

A function that produces a ggplot2 trace plot (i.e., geom_line) with respect to the posterior number of clusters.

Usage

```
ggplot_number_of_clusters_trace(.posterior_number_of_clusters)
```

Arguments

.posterior_number_of_clusters	Vector of positive integers: each (s)th element denotes the number of clusters after posterior sampling for each iteration s = 1, 2, ..., samples + burn in the Gibbs sampler.

Value

ggplot2 geom_line plot: a plot of the posterior number of clusters in each Gibbs sampling iteration versus the Gibbs sampling iteration number.

Examples

```
# Import the tip library
library(tip)

# Generate a vector of positive integers
# Example: the posterior number of clusters computed after posterior
# sampling in each sampling iteration of the Gibbs sampler.
num_clusters <- c(1,2,2,2,2,3,3,1,2,3,3,3,1,3)

# Generate the plot of the posterior number of clusters versus the
# sampling iteration number in the Gibbs sampler.
ggplot_number_of_clusters_trace(.posterior_number_of_clusters = num_clusters)
```

---

partition_undirected_graph

Partition an undirected graph

Description

A function that iteratively applies the transformation max(0, .graph_matrix - cutoff) until there are <.num_components> graph components where cutoff = cutoff + .step_size. This is used to generate the one-cluster graph and plot.

Usage

```
partition_undirected_graph(.graph_matrix, .num_components, .step_size)
```

Arguments

.graph_matrix	Matrix: a symmetric matrix that the analyst wishes to decompose into <.num_components> components.
.num_components	Positive integer: the number of components that the analyst wishes to decompose <.graph_matrix> into.
.step_size	Positive numeric: the size of the update for the cutoff in the transformation max(0, .graph_matrix - cutoff) where cutoff = cutoff + .step_size.

Value

List with three elements:

graph_component_members	Vector. A vector of positive integers: the (i)th element is the graph component assignment for the (i)th subject.
cutoff	Numeric. The value max(0, g_i,j - cutoff) so that there are <.num_components> components in the graph.
partitioned_graph_matrix	Matrix. The graph with <.num_components> components (parts).

Examples

```
# Import the tip library
library(tip)

# Choose an arbitrary random seed to generate the data
set.seed(4*8*15*16*23*42)

# Generate a symmetric posterior probability matrix
# Each element is the probability that the two subjects belong
# to the same cluster
n1 <- 10
posterior_prob_matrix <- matrix(NA, nrow = n1, ncol = n1)
for(i in 1:n1){
  for(j in i:n1){
    if(i != j){
      posterior_prob_matrix[i,j] <- runif(n=1,min=0,max=1)
      posterior_prob_matrix[j,i] <- posterior_prob_matrix[i,j]
    }else{
      posterior_prob_matrix[i,j] <- 1.0
    }
  }
}

# Generate a one-cluster graph (i.e., partitioned_graph_matrix)
partition_undirected_graph(.graph_matrix = posterior_prob_matrix,
                           .num_components = 1,
                           .step_size = 0.001)
```

plot,bcm,missing-method *Generate plots from a Bayesian Clustering Model (bcm) object*

Description

Generate plots from a Bayesian Clustering Model (bcm) object

Usage

```
## S4 method for signature 'bcm,missing'
plot(x, y, ...)
```

Arguments

x	bcm object: a Bayesian Clustering Model (bcm) object fit to a dataset
y	Not used.
...	Not used.

Value

List: a list of two ggplot2 plots that are constructed using the information contained in an object of class bcm (Bayesian Clustering Model). A bcm object contains the clustering results from a clustering model that uses the TIP prior.

`trace_plot_posterior_number_of_clusters`
: ggplot2 Plot: a plot of the posterior number of clusters (sampling iterations only) versus the corresponding sampling iteration number from the Gibbs sampler.

`histogram_posterior_number_of_clusters`
: ggplot2 Plot: a bar plot of the posterior number of clusters (sampling iterations only) from the Gibbs sampler.

Examples

```
# Import the tip library
library(tip)

# Import the iris dataset
data(iris)

# The first 4 columns are the data whereas
# the 5th column refers to the true labels
X <- data.matrix(iris[,c("Sepal.Length",
                         "Sepal.Width",
                         "Petal.Length",
                         "Petal.Width")])
```

```
# Extract the true labels (optional)
# True labels are only necessary for constructing network
# graphs that incorporate the true labels; this is often
# for research.
true_labels <- iris[,"Species"]

# Compute the distance matrix
distance_matrix <- data.matrix(dist(X))

# Compute the temperature parameter estiamte
temperature <- 1/median(distance_matrix[upper.tri(distance_matrix)])

# For each subject, compute the point estimate for the number of similar
# subjects using  univariate multiple change point detection (i.e.)
init_num_neighbors = get_cpt_neighbors(.distance_matrix = distance_matrix)

# Set the number of burn-in iterations in the Gibbs samlper
# RECOMMENDATION: burn >= 1000
burn <- 1000

# Set the number of sampling iterations in the Gibbs sampler
# RECOMMENDATION: burn >= 1000
samples <- 1000

# Set the subject names
names_subjects <- paste(1:dim(iris)[1])

# Run TIP clustering using only the prior
# --> That is, the likelihood function is constant
tip1 <- tip(.data = data.matrix(X),
            .burn = burn,
            .samples = samples,
            .similarity_matrix = exp(-1.0*temperature*distance_matrix),
            .init_num_neighbors = init_num_neighbors,
            .likelihood_model = "NIW",
            .subject_names = names_subjects,
            .num_cores = 1)

# Produce plots for the Bayesian Clustering Model
tip_plots <- plot(tip1)
```

`tip` *Bayesian Clustering with the Table Invitation Prior*

Description

Bayesian clustering with the Table Invitation Prior (TIP) and optional likelihood functions.

Usage

```
tip(
  .data = list(),
  .burn = 1000,
  .samples = 1000,
  .similarity_matrix,
  .init_num_neighbors,
  .likelihood_model = "CONSTANT",
  .subject_names = vector(),
  .num_cores = 1,
  .step_size = 0.001
)
```

Arguments

.data	Data frame (vectors comprise a row in a data frame; NIW only) or a list of matrices (MNIW only) that the analyst wishes to cluster. Note: if .likelihood_model = "CONSTANT", then the .data argument has no effect.
.burn	Non-negative integer: the number of burn-in iterations in the Gibbs sampler.
.samples	Positive integer: the number of sampling iterations in the Gibbs sampler.
.similarity_matrix	Matrix: an n x n matrix of similarity values.
.init_num_neighbors	Vector of positive integers: each (i)th positive integer corresponds to the estimate of the number of subjects that are similar to the (i)th subject.
.likelihood_model	Character: the name of the likelihood model used to compute the posterior probabilities. Options: "NIW" (vectors; .data is a dataframe), "MNIW" (matrices; .data is a list of matrices), or "CONSTANT" (vector, matrices, and tensors; .data is an empty list)
.subject_names	Vector of characters: an optional vector of names for the individual subjects. This is useful for the plotting function.
.num_cores	Positive integer: the number of cores to use.
.step_size	Positive numeric: A parameter used to ensure matrices are invertible. A small number is iteratively added to a matrix diagonal (if necessary) until the matrix is invertible.

Value

Object of class bcm: bcm denotes a "Bayesian Clustering Model" object that contains the results from a clustering model that uses the TIP prior.

n	Positive integer: the sample size or number of subjects.
burn	Non-negative integer: the number of burn-in iterations in the Gibbs sampler.
samples	Positive integer: the number of sampling iterations in the Gibbs sampler.

posterior_assignments
: List: a list of <samples> vectors where the (i)th element of each vector is the posterior cluster assignment for the (i)th subject.

posterior_similarity_matrix
: Matrix: an n x n matrix where the (i,j)th element is the posterior probability that subject i and subject j are in the same cluster.

posterior_number_of_clusters
: Vector of positive integers: a vector where the jth element is the number of clusters after posterior sampling (i.e., the posterior number of clusters).

prior_name
: Character: the name of the prior used.

likelihood_name
: Character: the name of the likelihood used.

Examples

```
##### BEGIN EXAMPLE 1: Clustering the Iris Dataset (vector clustering) #####
##### Prior Distribution: Table Invitation Prior (TIP)
##### Likelihood Model: Normal Inverse Wishart (NIW)

# Import the tip library
library(tip)

# Import the iris dataset
data(iris)

# The first 4 columns are the data whereas
# the 5th column refers to the true labels
X <- data.matrix(iris[,c("Sepal.Length",
                         "Sepal.Width",
                         "Petal.Length",
                         "Petal.Width")])

# Extract the true labels (optional)
# True labels are only necessary for constructing network
# graphs that incorporate the true labels; this is often
# for research.
true_labels <- iris[,"Species"]

# Compute the distance matrix
distance_matrix <- data.matrix(dist(X))

# Compute the temperature parameter estiamte
temperature <- 1/median(distance_matrix[upper.tri(distance_matrix)])

# For each subject, compute the point estimate for the number of similar
# subjects using  univariate multiple change point detection (i.e.)
init_num_neighbors = get_cpt_neighbors(.distance_matrix = distance_matrix)

# Set the number of burn-in iterations in the Gibbs samlper
```

```
# RECOMMENDATION: burn >= 1000
burn <- 1000

# Set the number of sampling iterations in the Gibbs sampler
# RECOMMENDATION: samples >= 1000
samples <- 1000

# Set the subject names
names_subjects <- paste(1:dim(iris)[1])

# Run TIP clustering using only the prior
# --> That is, the likelihood function is constant
tip1 <- tip(.data = data.matrix(X),
            .burn = burn,
            .samples = samples,
            .similarity_matrix = exp(-1.0*temperature*distance_matrix),
            .init_num_neighbors = init_num_neighbors,
            .likelihood_model = "NIW",
            .subject_names = names_subjects,
            .num_cores = 1)

# Produce plots for the Bayesian Clustering Model
tip_plots <- plot(tip1)

# View the posterior distribution of the number of clusters
tip_plots$histogram_posterior_number_of_clusters

# View the trace plot with respect to the posterior number of clusters
tip_plots$trace_plot_posterior_number_of_clusters

# Extract posterior cluster assignments using the Posterior Expected Adjusted Rand (PEAR) index
cluster_assignments <- mcclust::maxpear(psm = tip1@posterior_similarity_matrix)$cl

# If the true labels are available, then show the cluster result via a contingency table
table(data.frame(true_label = true_labels,
                 cluster_assignment = cluster_assignments))

# Create the one component graph with minimum entropy
partition_list <- partition_undirected_graph(.graph_matrix = tip1@posterior_similarity_matrix,
                                             .num_components = 1,
                                             .step_size = 0.001)

# Associate class labels and colors for the plot
class_palette_colors <- c("setosa" = "blue",
                          "versicolor" = 'green',
                          "virginica" = "orange")

# Associate class labels and shapes for the plot
class_palette_shapes <- c("setosa" = 19,
                          "versicolor" = 18,
                          "virginica" = 17)

# Visualize the posterior similarity matrix by constructing a graph plot of
```

```
# the one-cluster graph. The true labels are used here (below they are not).
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = NA,
                    .subject_class_names = true_labels,
                    .class_colors = class_palette_colors,
                    .class_shapes = class_palette_shapes,
                    .node_size = 2,
                    .add_node_labels = FALSE)

# If true labels are not available, then construct a network plot
# of the one-cluster graph without any class labels.
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
                    .add_node_labels = TRUE)

# If true labels are not available, then construct a network plot
# of the one-cluster graph without any class labels. Also, suppress
# the subject labels.
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
                    .add_node_labels = FALSE)
##### END EXAMPLE 1: Vector Clustering (NIW) #####



##### BEGIN EXAMPLE 2: Clustering the US Arrests Dataset (vector clustering) #####
##### Prior Distribution: Table Invitation Prior (TIP)
##### Likelihood Model: Normal Inverse Wishart (NIW)

# Import the TIP library
library(tip)

# Import the US Arrests dataset
data(USArrests)

# Convert the data to a matrix
X <- data.matrix(USArrests)

# Compute the distance matrix
distance_matrix <- data.matrix(dist(X))

# Compute the temperature parameter estiamte
temperature <- 1/median(distance_matrix[upper.tri(distance_matrix)])

# For each subject, compute the point estimate for the number of similar
# subjects using  univariate multiple change point detection (i.e.)
init_num_neighbors = get_cpt_neighbors(.distance_matrix = distance_matrix)

# Set the number of burn-in iterations in the Gibbs samlper
# RECOMENDATION: *** burn >= 1000 ***
burn <- 1000
```

```
# Set the number of sampling iterations in the Gibbs sampler
# RECOMENDATION: *** samples >= 1000 ***
samples <- 1000

# Extract the state names
names_subjects <- rownames(USArrests)

# Run TIP clustering using only the prior
# --> That is, the likelihood function is constant
tip1 <- tip(.data = data.matrix(X),
            .burn = burn,
            .samples = samples,
            .similarity_matrix = exp(-1.0*temperature*distance_matrix),
            .init_num_neighbors = init_num_neighbors,
            .likelihood_model = "NIW",
            .subject_names = names_subjects,
            .num_cores = 1)

# Produce plots for the Bayesian Clustering Model
tip_plots <- plot(tip1)

# View the posterior distribution of the number of clusters
tip_plots$trace_plot_posterior_number_of_clusters

# View the trace plot with respect to the posterior number of clusters
tip_plots$trace_plot_posterior_number_of_clusters

# Extract posterior cluster assignments using the Posterior Expected Adjusted Rand (PEAR) index
cluster_assignments <- mcclust::maxpear(psm = tip1@posterior_similarity_matrix)$cl

# Create a list where each element stores the cluster assignments
cluster_assignment_list <- list()
for(k in 1:length(unique(cluster_assignments))){
  cluster_assignment_list[[k]] <- names_subjects[which(cluster_assignments == k)]
}
cluster_assignment_list

# Create the one component graph with minimum entropy
partition_list <- partition_undirected_graph(.graph_matrix = tip1@posterior_similarity_matrix,
                                             .num_components = 1,
                                             .step_size = 0.001)

# View the state names
# names_subjects

# Create a vector of true region labels to see if there is a pattern
true_region <- c("Southeast", "West", "Southwest", "Southeast", "West", "West",
                 "Northeast", "Northeast", "Southeast", "Southeast", "West", "West",
                 "Midwest", "Midwest", "Midwest", "Midwest", "Southeast", "Southeast",
              "Northeast", "Northeast", "Northeast", "Midwest", "Midwest", "Southeast",
                 "Midwest", "West", "Midwest", "West", "Northeast", "Northeast",
              "Southwest", "Northeast", "Southeast", "Midwest", "Midwest", "Southwest",
```

```
                "West", "Northeast", "Northeast", "Southeast", "Midwest", "Southeast",
                 "Southwest", "West", "Northeast", "Southeast", "West", "Southeast",
                 "Midwest", "West")

names_subjects

# Associate class labels and colors for the plot
class_palette_colors <- c("Northeast" = "blue",
                          "Southeast" = "red",
                          "Midwest" = "purple",
                          "Southwest" = "orange",
                          "West" = "green")

# Associate class labels and shapes for the plot
class_palette_shapes <- c("Northeast" = 15,
                          "Southeast" = 16,
                          "Midwest" = 17,
                          "Southwest" = 18,
                          "West" = 19)

# Visualize the posterior similarity matrix by constructing a graph plot of
# the one-cluster graph. The true labels are used here (below they are not).
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .subject_class_names = true_region,
                    .class_colors = class_palette_colors,
                    .class_shapes = class_palette_shapes,
                    .node_size = 2,
                    .add_node_labels = TRUE)


# Visualize the posterior similarity matrix by constructing a graph plot of
# the one-cluster graph. The true labels are used here (below they are not).
# Remove the subject names with .add_node_labels = FALSE
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .subject_class_names = true_region,
                    .class_colors = class_palette_colors,
                    .class_shapes = class_palette_shapes,
                    .node_size = 2,
                    .add_node_labels = FALSE)

# Construct a network plot without class labels
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
                    .add_node_labels = TRUE)

# Construct a network plot without class labels. Also, suppress
# the subject labels.
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
```

```
                      .add_node_labels = FALSE)
  ##### END EXAMPLE 2: Clustering the US Arrests Dataset (vector clustering) #####


## Not run:

  ##### BEGIN EXAMPLE 3: Clustering gene expression data (vector clustering) #####
  ##### Prior Distribution: Table Invitation Prior (TIP)
  ##### Likelihood Model: Normal Inverse Wishart (NIW)

  # Import the TIP library
  library(tip)

  # ----- Dataset information -----
  # The data were accessed from the UCI machine learning repository
 # Original link: https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq
  # Source: Samuele Fiorini, samuele.fiorini '@' dibris.unige.it,
  # University of Genoa, redistributed under Creative Commons license
  # (http://creativecommons.org/licenses/by/3.0/legalcode)
  # from https://www.synapse.org/#!Synapse:syn4301332.
  # Data Set Information: Samples (instances) are stored row-wise. Variables
  # (attributes) of each sample are RNA-Seq gene expression levels measured by
  # illumina HiSeq platform.
  # Relevant Papers: Weinstein, John N., et al. 'The cancer genome atlas pan-cancer
  # analysis project.' Nature genetics 45.10 (2013): 1113-1120.
  # -------------------------------

  # Import the data (see the provided link above)
  X <- read.csv("data.csv")
  true_labels <- read.csv("labels.csv")

  # Extract the true indices
  subject_names <- true_labels$X

  # Extract the true classes
  true_labels <- true_labels$Class

  # Convert the dataset into a matrix
  X <- data.matrix(X)

  ##### BEGIN: Apply PCA to the dataset #####

  # Step 1: perform Prinicpal Components Analysis (PCA) on the dataset
  pca1 <- prcomp(X)

  # Step 2: compute summary information
  summary1 <- summary(pca1)

  # Step 3: plot the cumulative percentage of the variance
  # explained against the number of principal components
  tip::ggplot_line_point(.x = 1:length(summary1$importance['Cumulative Proportion',]),
                         .y = summary1$importance['Cumulative Proportion',],
                         .xlab = "Number of Principal Components",
```

```
                        .ylab = "Cumulative Percentage of the Variance Explained")

# The number of principal components chosen is 7, and
# the 7 principal components explain roughly 80% of
# the variance
num_principal_components <- which(summary1$importance['Cumulative Proportion',] <= 0.8)

# The clustering is applied to the principal component dataset
X <- pca1$x[,as.numeric(num_principal_components)]
##### END: Apply PCA to the dataset #####

# Compute the distance matrix
distance_matrix <- data.matrix(dist(X))

# Compute the temperature parameter estiamte
temperature <- 1/median(distance_matrix[upper.tri(distance_matrix)])

# For each subject, compute the point estimate for the number of similar
# subjects using  univariate multiple change point detection (i.e.)
init_num_neighbors = get_cpt_neighbors(.distance_matrix = distance_matrix)

# Set the number of burn-in iterations in the Gibbs samlper
# RECOMENDATION: *** burn >= 1000 ***
burn <- 1000

# Set the number of sampling iterations in the Gibbs sampler
# RECOMENDATION: *** samples >= 1000 ***
samples <- 1000

# Run TIP clustering using only the prior
# --> That is, the likelihood function is constant
tip1 <- tip(.data = data.matrix(X),
            .burn = burn,
            .samples = samples,
            .similarity_matrix = exp(-1.0*temperature*distance_matrix),
            .init_num_neighbors = init_num_neighbors,
            .likelihood_model = "NIW",
            .subject_names = c(),
            .num_cores = 1)

# Produce plots for the Bayesian Clustering Model
tip_plots <- plot(tip1)

# View the posterior distribution of the number of clusters
tip_plots$trace_plot_posterior_number_of_clusters

# View the trace plot with respect to the posterior number of clusters
tip_plots$trace_plot_posterior_number_of_clusters

# Extract posterior cluster assignments using the Posterior Expected Adjusted Rand (PEAR) index
cluster_assignments <- mcclust::maxpear(psm = tip1@posterior_similarity_matrix)$cl

# Create a list where each element stores the cluster assignments
```

```
cluster_assignment_list <- list()
for(k in 1:length(unique(cluster_assignments))){
  cluster_assignment_list[[k]] <- true_labels[which(cluster_assignments == k)]
}
cluster_assignment_list

# Create the one component graph with minimum entropy
partition_list <- partition_undirected_graph(.graph_matrix = tip1@posterior_similarity_matrix,
                                             .num_components = 1,
                                             .step_size = 0.001)

# Associate class labels and shapes for the plot
class_palette_shapes <- c("PRAD" = 19,
                          "BRCA" = 18,
                          "KIRC" = 17,
                          "LUAD" = 16,
                          "COAD" = 15)

# Associate class labels and colors for the plot
class_palette_colors <- c("PRAD" = "blue",
                          "BRCA" = "red",
                          "KIRC" = "black",
                          "LUAD" = "green",
                          "COAD" = "orange")

# Visualize the posterior similarity matrix by constructing a graph plot of
# the one-cluster graph. The true labels are used here (below they are not).
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = subject_names,
                    .subject_class_names = true_labels,
                    .class_colors = class_palette_colors,
                    .class_shapes = class_palette_shapes,
                    .node_size = 2,
                    .add_node_labels = TRUE)

# Visualize the posterior similarity matrix by constructing a graph plot of
# the one-cluster graph. The true labels are used here (below they are not).
# Remove the subject names with .add_node_labels = FALSE
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = subject_names,
                    .subject_class_names = true_labels,
                    .class_colors = class_palette_colors,
                    .class_shapes = class_palette_shapes,
                    .node_size = 2,
                    .add_node_labels = FALSE)

# Construct a network plot without class labels but with subject labels
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = subject_names,
                    .node_size = 2,
                    .add_node_labels = TRUE)

# Construct a network plot without class labels and subject labels
```

```
  ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                      .subject_names = subject_names,
                      .node_size = 2,
                      .add_node_labels = FALSE)
  ##### END EXAMPLE 3: Clustering gene expression data (vector clustering) #####

## End(Not run)



  ##### BEGIN EXAMPLE 4: Matrix Clustering (MNIW) #####
  ##### Prior Distribution: Table Invitation Prior
  ##### Likelihood Model: Matrix Normal Inverse Wishart (MNIW)

  # Import the tip library
  library(tip)

  # A function to generate random matrices from a matrix normal distribution
  random_mat_normal <- function(mu, num_rows, num_cols){
    LaplacesDemon::rmatrixnorm(M = matrix(mu,
                                          nrow = num_rows,
                                          ncol = num_cols),
                               U = diag(num_rows),
                               V = diag(num_cols))
  }

  # Generate 3 clusters of matrices
  p <- 5
  m <- 3
  c1 <- lapply(1:20, function(x) random_mat_normal(mu = 0, num_rows = m, num_cols = p))
  c2 <- lapply(1:25, function(x) random_mat_normal(mu = 5, num_rows = m, num_cols = p))
  c3 <- lapply(1:30, function(x) random_mat_normal(mu = -5, num_rows = m, num_cols = p))

  # Put all the data into a list
  data_list <- c(c1,c2,c3)

  # Create a vector of true labels. True labels are only necessary
  # for constructing network graphs that incorporate the true labels;
  # this is often useful for research.
  true_labels <- c(rep("Cluster 1", 20),
                   rep("Cluster 2", 25),
                   rep("Cluster 3", 30))

  distance_matrix <- matrix(NA,
                            nrow = length(true_labels),
                            ncol = length(true_labels))
  # Distance matrix
  for(i in 1:length(true_labels)){
    for(j in i:length(true_labels)){
      distance_matrix[i,j] <- SMFilter::FDist2(mX = data_list[[i]],
                                               mY = data_list[[j]])
      distance_matrix[j,i] <- distance_matrix[i,j]
    }
```

```
}

# Compute the temperature parameter estiamte
temperature <- 1/median(distance_matrix[upper.tri(distance_matrix)])

# For each subject, compute the point estimate for the number of similar
# subjects using  univariate multiple change point detection (i.e.)
init_num_neighbors = get_cpt_neighbors(.distance_matrix = distance_matrix)

# Set the number of burn-in iterations in the Gibbs samlper
# RECOMMENDATION: burn >= 1000
burn <- 1000

# Set the number of sampling iterations in the Gibbs sampler
# RECOMMENDATION: samples >= 1000
samples <- 1000

# Set the subject names
names_subjects <- paste(1:dim(distance_matrix)[1])

# Run TIP clustering using only the prior
# --> That is, the likelihood function is constant
tip1 <- tip(.data = data_list,
            .burn = burn,
            .samples = samples,
            .similarity_matrix = exp(-1.0*temperature*distance_matrix),
            .init_num_neighbors = init_num_neighbors,
            .likelihood_model = "MNIW",
            .subject_names = names_subjects,
            .num_cores = 1)

# Produce plots for the Bayesian Clustering Model
tip_plots <- plot(tip1)

# View the posterior distribution of the number of clusters
tip_plots$histogram_posterior_number_of_clusters

# View the trace plot with respect to the posterior number of clusters
tip_plots$trace_plot_posterior_number_of_clusters

# Extract posterior cluster assignments using the Posterior Expected Adjusted Rand (PEAR) index
cluster_assignments <- mcclust::maxpear(psm = tip1@posterior_similarity_matrix)$cl

# If the true labels are available, then show the cluster result via a contigency table
table(data.frame(true_label = true_labels,
                 cluster_assignment = cluster_assignments))

# Create the one component graph with minimum entropy
partition_list <- partition_undirected_graph(.graph_matrix = tip1@posterior_similarity_matrix,
                                             .num_components = 1,
                                             .step_size = 0.001)

# Associate class labels and colors for the plot
```

```
class_palette_colors <- c("Cluster 1" = "blue",
                          "Cluster 2" = 'green',
                          "Cluster 3" = "red")

# Associate class labels and shapes for the plot
class_palette_shapes <- c("Cluster 1" = 19,
                          "Cluster 2" = 18,
                          "Cluster 3" = 17)

# Visualize the posterior similarity matrix by constructing a graph plot of
# the one-cluster graph. The true labels are used here (below they are not).
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = NA,
                    .subject_class_names = true_labels,
                    .class_colors = class_palette_colors,
                    .class_shapes = class_palette_shapes,
                    .node_size = 2,
                    .add_node_labels = FALSE)

# If true labels are not available, then construct a network plot
# of the one-cluster graph without any class labels.
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
                    .add_node_labels = TRUE)

# If true labels are not available, then construct a network plot
# of the one-cluster graph without any class labels. Also, suppress the
# subject labels.
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
                    .add_node_labels = FALSE)

##### END EXAMPLE 4: Matrix Clustering (MNIW) #####

##### BEGIN EXAMPLE 5: Tensor Clustering #####
##### Prior Distribution: Table Invitation Prior
##### Likelihood Model: CONSTANT

# Import the tip library
library(tip)

# ################## NOTE ##################
# Order 3 Tensor dimension: c(d1,d2,d3)
# d1: number of rows
# d2: number of columns
# d3: number of slices
##########################################

# Set a random seed for reproducibility
set.seed(007)
```

```
# A function to generate an order-3 tensor
generate_gaussian_tensor <- function(.tensor_dimension, .mean = 0, .sd = 1){
  array(data = c(rnorm(n = prod(.tensor_dimension),
                       mean = .mean,
                       sd = .sd)),
        dim = .tensor_dimension)
}

# Define the tensor dimension
tensor_dimension <- c(256,256,3)

# Generate clusters of tensors
c1 <- lapply(1:30, function(x) generate_gaussian_tensor(.tensor_dimension = tensor_dimension,
                                                        .mean = 0,
                                                        .sd = 1))


# Generate clusters of tensors
c2 <- lapply(1:40, function(x) generate_gaussian_tensor(.tensor_dimension = tensor_dimension,
                                                        .mean = 5,
                                                        .sd = 1))


# Generate clusters of tensors
c3 <- lapply(1:50, function(x) generate_gaussian_tensor(.tensor_dimension = tensor_dimension,
                                                        .mean = -5,
                                                        .sd = 1))

# Make a list of tensors
X <- c(c1, c2, c3)

# Compute the number of subjects for each cluster
n1 <- length(c1)
n2 <- length(c2)
n3 <- length(c3)

# Create a vector of true labels. True labels are only necessary
# for constructing network graphs that incorporate the true labels;
# this is often useful for research.
true_labels <- c(rep("Cluster 1", n1),
                 rep("Cluster 2", n2),
                 rep("Cluster 3", n3))

# Compute the total number of subjects
n <- length(X)

# Construct the distance matrix
distance_matrix <- matrix(data = NA, nrow = n, ncol = n)
for(i in 1:n){
  for(j in i:n){
    distance_matrix[i,j] <- sqrt(sum((X[[i]] - X[[j]])^2))
    distance_matrix[j,i] <- distance_matrix[i,j]
  }
```

```
}

# Compute the temperature parameter estiamte
temperature <- 1/median(distance_matrix[upper.tri(distance_matrix)])

# For each subject, compute the point estimate for the number of similar
# subjects using  univariate multiple change point detection (i.e.)
init_num_neighbors = get_cpt_neighbors(.distance_matrix = distance_matrix)

# Set the number of burn-in iterations in the Gibbs samlper
# RECOMMENDATION: burn >= 1000
burn <- 1000

# Set the number of sampling iterations in the Gibbs sampler
# RECOMMENDATION: samples >= 1000
samples <- 1000

# Set the subject names
names_subjects <- paste(1:n)

# Run TIP clustering using only the prior
# --> That is, the likelihood function is constant
tip1 <- tip(.data = list(),
            .burn = burn,
            .samples = samples,
            .similarity_matrix = exp(-1.0*temperature*distance_matrix),
            .init_num_neighbors = init_num_neighbors,
            .likelihood_model = "CONSTANT",
            .subject_names = names_subjects,
            .num_cores = 1)

# Produce plots for the Bayesian Clustering Model
tip_plots <- plot(tip1)

# View the posterior distribution of the number of clusters
tip_plots$histogram_posterior_number_of_clusters

# View the trace plot with respect to the posterior number of clusters
tip_plots$trace_plot_posterior_number_of_clusters

# Extract posterior cluster assignments using the Posterior Expected Adjusted Rand (PEAR) index
cluster_assignments <- mcclust::maxpear(psm = tip1@posterior_similarity_matrix)$cl

# If the true labels are available, then show the cluster result via a contigency table
table(data.frame(true_label = true_labels,
                 cluster_assignment = cluster_assignments))

# Create the one component graph with minimum entropy
partition_list <- partition_undirected_graph(.graph_matrix = tip1@posterior_similarity_matrix,
                                             .num_components = 1,
                                             .step_size = 0.001)

# Associate class labels and colors for the plot
```

```
class_palette_colors <- c("Cluster 1" = "blue",
                          "Cluster 2" = 'green',
                          "Cluster 3" = "red")

# Associate class labels and shapes for the plot
class_palette_shapes <- c("Cluster 1" = 19,
                          "Cluster 2" = 18,
                          "Cluster 3" = 17)

# Visualize the posterior similarity matrix by constructing a graph plot of
# the one-cluster graph. The true labels are used here (below they are not).
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = NA,
                    .subject_class_names = true_labels,
                    .class_colors = class_palette_colors,
                    .class_shapes = class_palette_shapes,
                    .node_size = 2,
                    .add_node_labels = FALSE)

# If true labels are not available, then construct a network plot
# of the one-cluster graph without any class labels.
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
                    .add_node_labels = TRUE)

# If true labels are not available, then construct a network plot
# of the one-cluster graph without any class labels. Also, suppress
# the subject labels.
ggnet2_network_plot(.matrix_graph = partition_list$partitioned_graph_matrix,
                    .subject_names = names_subjects,
                    .node_size = 2,
                    .add_node_labels = FALSE)

##### END EXAMPLE 5: Tensor Clustering #####
```

Index