Package: GoodFitSBM (via r-universe)

August 22, 2024

Title Monte Carlo goodness-of-fit tests for Stochastic Blockmodels

Version 0.0.1

Description Performing goodness-of-fit tests for stochastic blockmodels used to fit network data. Among the three variants of SBMs discussed in <https://doi.org/10.1093/jrsssb/qkad084>, goodness-of-fit test has been performed for the Erdős-Rényi (ER) and Beta versions of SBMs.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL https://github.com/Roy-SR-007/GoodFitSBM

BugReports https://github.com/Roy-SR-007/GoodFitSBM/issues

Imports stats, igraph, utils, irlba

Depends R (>= 2.10)

LazyData true

Collate 'Bipartite_Walk.R' 'Estimation_BetaSBM.R' 'Estimation_Block.R' 'Estimation_ERSBM.R' 'Get_Directed_Piece.R' 'Get_Directed_Move_p1_ed.R' 'Get_Between_Blocks_Move_beta_SBM.R' 'as_arbitrary_directed.R' 'Get_Bidirected_Piece.R' 'Get_Bidirected_Move.R' 'Get_Induced_Subgraph.R' 'Get_Within_Blocks_beta_SBM.R' 'Get_Move_Beta_SBM.R' 'Get_Next_Network.R' 'Sampling_Graph_BetaSBM.R' 'TestStatistic_BetaSBM.R' 'GoFTest_BetaSBM.R' 'Sampling_Graph_ERSBM.R' 'TestStatistic_ERSBM.R' 'GoFTest_ERSBM.R' 'zachary.R'

Acknowledgement We would like to thank all the authors of <https://doi.org/10.1093/jrsssb/qkad084>.

Repository https://roy-sr-007.r-universe.dev

RemoteUrl https://github.com/roy-sr-007/goodfitsbm

RemoteRef HEAD

RemoteSha f5d20fae9d9a836c62e125c6fa65bd96230fc174

Contents

get_mle_BetaSBM . . . 2
get_mle_ERSBM . . . 3
goftest_BetaSBM . . . 5
goftest_ERSBM . . . 8
graphchi_BetaSBM . . . 12
graphchi_ERSBM . . . 14
sample_a_move_BetaSBM . . . 15
sample_a_move_ERSBM . . . 17
zachary . . . 18

Index **20**

`get_mle_BetaSBM` *Maximum Likelihood Estimation of edge probabilities between blocks of a graph, under beta-SBM*

Description

`get_mle_BetaSBM` obtains MLE for the probability of edges between blocks in a graph, used in calculating the goodness-of-fit test statistic for the beta-SBM (Karwa et al. (2023))

Usage

```
get_mle_BetaSBM(G, C)
```

Arguments

`G`	an igraph object which is an undirected graph with no self loop
`C`	a positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)

Value

A matrix of maximum likelihood estimates

`mleMatr`	a matrix containing the estimated edge probabilities between blocks in a graph

References

Karwa et al. (2023). "Monte Carlo goodness-of-fit tests for degree corrected and related stochastic blockmodels", *Journal of the Royal Statistical Society Series B: Statistical Methodology*, https://doi.org/10.1093/jrsssb/qkad084

See Also

`goftest_BetaSBM()` performs the goodness-of-fit test for the beta-SBM, where the MLE of the edge probabilities are required

Examples

```
RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# graph from the adjacency matrix
G = igraph::graph_from_adjacency_matrix(adjsymm, mode = "undirected", weighted = NULL)

# mle of the edge probabilities
get_mle_BetaSBM(G, class)
```

`get_mle_ERSBM` *Maximum Likelihood Estimation of edge probabilities between blocks of a graph, under ERSBM*

Description

`get_mle_ERSBM` obtains MLE for the probability of edges between blocks in a graph, used in calculating the goodness-of-fit test statistic for the ERSBM (Karwa et al. (2023))

Usage

```
get_mle_ERSBM(G, C)
```

Arguments

`G`	an igraph object which is an undirected graph with no self loop
`C`	a positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)

Value

A matrix of maximum likelihood estimates

`mleMatr`	a matrix containing the estimated edge probabilities between blocks in a graph

References

Karwa et al. (2023). "Monte Carlo goodness-of-fit tests for degree corrected and related stochastic blockmodels", *Journal of the Royal Statistical Society Series B: Statistical Methodology*, https://doi.org/10.1093/jrsssb/qkad084

See Also

`goftest_ERSBM()` performs the goodness-of-fit test for the ERSBM, where the MLE of the edge probabilities are required

Examples

```
RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
```

```
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# graph from the adjacency matrix
G = igraph::graph_from_adjacency_matrix(adjsymm, mode = "undirected", weighted = NULL)

# mle of the edge probabilities
get_mle_ERSBM(G, class)
```

`goftest_BetaSBM` *Monte Carlo goodness-of-fit test for a beta stochastic blockmodel (beta-SBM)*

Description

`goftest_BetaSBM` performs chi square goodness-of-fit test for network data considering the model as beta-SBM (Karwa et al. (2023))

Usage

```
goftest_BetaSBM(A, K = NULL, C = NULL, numGraphs = 100)
```

Arguments

`A`	n by n binary symmetric adjacency matrix representing an undirected graph where n is the number of nodes in the graph
`K`	positive integer scalar representing the number of blocks; K>1
`C`	positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)
`numGraphs`	number of graphs to be sampled; default value is 100

Value

A list with the elements

`statistic`	the values of the chi-square test statistics on each sampled graph
`p.value`	the p-value for the test

References

Karwa et al. (2023). "Monte Carlo goodness-of-fit tests for degree corrected and related stochastic blockmodels", *Journal of the Royal Statistical Society Series B: Statistical Methodology*, https://doi.org/10.1093/jrsssb/qkad084

Examples

```
# Example 1

RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)
```

```
# When class assignment is known
out = goftest_BetaSBM(adjsymm, C = class, numGraphs = 100)

chi_sq_seq = out$statistic
pvalue = out$p.value
print(pvalue)

# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL)
abline(v = chi_sq_seq[1], col = "red", lwd = 5) # adding test statistic on the observed network
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))

# Example 2

#' RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 30
n2 = 20
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# When class assignment is known
out = goftest_BetaSBM(adjsymm, C = class, numGraphs = 100)
```

```
chi_sq_seq = out$statistic
pvalue = out$p.value
print(pvalue)

# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL)
abline(v = chi_sq_seq[1], col = "red", lwd = 5) # adding test statistic on the observed network
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))

# Application on real dataset: Testing on the Zachary's Karate Club Data

set.seed(100000)

data("zachary")

d = zachary # the Zachary's Karate Club data set

# the adjacency matrix
A_zachary = as.matrix(d[1:34, ])
colnames(A_zachary) = 1:34

# obtaining the graph from the adjacency matrix above
g_zachary = igraph::graph_from_adjacency_matrix(A_zachary, mode = "undirected", weighted = NULL)

# plotting the graph (network) obtained
plot(g_zachary,
main = "Network (Graph) for the Zachary's Karate Club data set; reference clustering")

# block assignments
K = 2 # no. of blocks

n1 = 10
n2 = 24
n = n1 + n2

# known class assignments
class = rep(c(1, 2), c(n1, n2))
# goodness-of-fit tests for the Zachary's Karate Club data set
out_zachary = goftest_BetaSBM(A_zachary, C = class, numGraphs = 100)

chi_sq_seq = out_zachary$statistic
pvalue = out_zachary$p.value
print(pvalue)

# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL)
abline(v = chi_sq_seq[1], col = "red", lwd = 5) # adding test statistic on the observed network
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))
```

goftest_ERSBM *Monte Carlo goodness-of-fit test for an Erdős-Rényi stochastic block-model (ERSBM)*

Description

goftest_ERSBM performs chi square goodness-of-fit test for network data considering the model as ERSBM (Karwa et al. (2023))

Usage

```
goftest_ERSBM(A, K = NULL, C = NULL, numGraphs = 100)
```

Arguments

A	n by n binary symmetric adjacency matrix representing an undirected graph where n is the number of nodes in the graph
K	positive integer scalar representing the number of blocks; K>1
C	positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)
numGraphs	number of graphs to be sampled; default value is 100

Value

A list with the elements

statistic	the values of the chi-square test statistics on each sampled graph
p.value	the p-value for the test

References

Karwa et al. (2023). "Monte Carlo goodness-of-fit tests for degree corrected and related stochastic blockmodels", *Journal of the Royal Statistical Society Series B: Statistical Methodology*, https://doi.org/10.1093/jrsssb/qkad084

Examples

```
# Example 1

RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
```

```
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
   p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# When class assignment is known
out = goftest_ERSBM(adjsymm, C = class, numGraphs = 100)

chi_sq_seq = out$statistic
pvalue = out$p.value
print(pvalue)

# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL)
abline(v = chi_sq_seq[1], col = "red", lwd = 5) # adding test statistic on the observed network
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))

# Example 2

#' RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 30
n2 = 20
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
```

```
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# When class assignment is known
out = goftest_ERSBM(adjsymm, C = class, numGraphs = 100)

chi_sq_seq = out$statistic
pvalue = out$p.value
print(pvalue)

# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL)
abline(v = chi_sq_seq[1], col = "red", lwd = 5) # adding test statistic on the observed network
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))

# Application on real dataset: Testing on the Zachary's Karate Club Data

set.seed(100000)

data("zachary")

d = zachary # the Zachary's Karate Club data set

# the adjacency matrix
A_zachary = as.matrix(d[1:34, ])
colnames(A_zachary) = 1:34

# obtaining the graph from the adjacency matrix above
g_zachary = igraph::graph_from_adjacency_matrix(A_zachary, mode = "undirected", weighted = NULL)

# plotting the graph (network) obtained
plot(g_zachary,
main = "Network (Graph) for the Zachary's Karate Club data set; reference clustering")
```

```
# block assignments
K = 2 # no. of blocks

n1 = 10
n2 = 24
n = n1 + n2

# known class assignments
class = rep(c(1, 2), c(n1, n2))
# goodness-of-fit tests for the Zachary's Karate Club data set
out_zachary = goftest_ERSBM(A_zachary, C = class, numGraphs = 100)

chi_sq_seq = out_zachary$statistic
pvalue = out_zachary$p.value
print(pvalue)

# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL)
abline(v = chi_sq_seq[1], col = "red", lwd = 5) # adding test statistic on the observed network
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))
```

`graphchi_BetaSBM` *Computation of the chi-square test statistic for goodness-of-fit, under beta-SBM*

Description

`graphchi_BetaSBM` obtains the value of the chi-square test statistic required for the goodness-of-fit of a beta-SBM (Karwa et al. (2023))

Usage

```
graphchi_BetaSBM(G, C, p_mle)
```

Arguments

`G`	an igraph object which is an undirected graph with no self loop
`C`	a positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)
`p_mle`	a matrix with the MLE estimates of the edge probabilities

Value

A numeric value

`teststat_val`	The value of the chi-square test statistic

See Also

`goftest_BetaSBM()` performs the goodness-of-fit test for the beta-SBM, where the values of the chi-square test statistics are required

Examples

```
RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# graph from the adjacency matrix
G = igraph::graph_from_adjacency_matrix(adjsymm, mode = "undirected", weighted = NULL)

# mle of the edge probabilities
p.hat = get_mle_BetaSBM (G, class)

# chi-square test statistic values
graphchi_BetaSBM(G, class, p.hat)
```

graphchi_ERSBM *Computation of the chi-square test statistic for goodness-of-fit, under ERSBM*

Description

`graphchi_ERSBM` obtains the value of the chi-square test statistic required for the goodness-of-fit of a ERSBM (Karwa et al. (2023))

Usage

```
graphchi_ERSBM(G, C, p_mle)
```

Arguments

`G`	an igraph object which is an undirected graph with no self loop
`C`	a positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)
`p_mle`	a matrix with the MLE estimates of the edge probabilities

Value

A numeric value

`teststat_val`	The value of the chi-square test statistic

See Also

`goftest_ERSBM()` performs the goodness-of-fit test for the ERSBM, where the values of the chi-square test statistics are required

Examples

```
RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
```

```
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# graph from the adjacency matrix
G = igraph::graph_from_adjacency_matrix(adjsymm, mode = "undirected", weighted = NULL)

# mle of the edge probabilities
p.hat = get_mle_ERSBM(G, class)

# chi-square test statistic values
graphchi_ERSBM(G, class, p.hat)
```

---

sample_a_move_BetaSBM *Sampling a graph through a Markov move (basis) for beta-SBM*

Description

`sample_a_move_BetaSBM` to sample a graph in the same fiber; sampling according to the beta-SBM (Karwa et al. (2023))

Usage

```
sample_a_move_BetaSBM(C, G_current)
```

Arguments

`C`	a positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)
`G_current`	an igraph object which is an undirected graph with no self loop

Value

A graph

`sampled graph` the sampled graph after one move as per the beta-SBM

References

Karwa et al. (2023). "Monte Carlo goodness-of-fit tests for degree corrected and related stochastic blockmodels", *Journal of the Royal Statistical Society Series B: Statistical Methodology*, https://doi.org/10.1093/jrsssb/qkad084

See Also

`goftest_BetaSBM()` performs the goodness-of-fit test for the beta-SBM, where graphs are being sampled

Examples

```
RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}
```

```
adjsymm = adj + t(adj)

# graph from the adjacency matrix
G = igraph::graph_from_adjacency_matrix(adjsymm, mode = "undirected", weighted = NULL)

# sampling a Markov move for the beta-SBM
G_sample = sample_a_move_BetaSBM(class, G)

# plotting the sampled graph
plot(G_sample, main = "The sampled graph after one Markov move for beta-SBM")
```

`sample_a_move_ERSBM` *Sampling a graph through a Markov move (basis) for ERSBM*

Description

`sample_a_move_ERSBM` to sample a graph in the same fiber; sampling according to the ERSBM (Karwa et al. (2023))

Usage

```
sample_a_move_ERSBM(C, G_current)
```

Arguments

`C`	a positive integer vector of size n for block assignments of each node; from 1 to K (no of blocks)
`G_current`	an igraph object which is an undirected graph with no self loop

Value

A graph

`sampled graph`	the sampled graph after one move as per the ERSBM

References

Karwa et al. (2023). "Monte Carlo goodness-of-fit tests for degree corrected and related stochastic blockmodels", *Journal of the Royal Statistical Society Series B: Statistical Methodology*, https://doi.org/10.1093/jrsssb/qkad084

See Also

`goftest_ERSBM()` performs the goodness-of-fit test for the ERSBM, where graphs are being sampled

Examples

```
RNGkind(sample.kind = "Rounding")
set.seed(1729)

# We model a network with 3 even classes
n1 = 50
n2 = 50
n3 = 50

# Generating block assignments for each of the nodes
n = n1 + n2 + n3
class = rep(c(1, 2, 3), c(n1, n2, n3))

# Generating the adjacency matrix of the network
# Generate the matrix of connection probabilities
cmat = matrix(
  c(
    30, 0.05, 0.05,
    0.05, 30, 0.05,
    0.05, 0.05, 30
  ),
  ncol = 3,
  byrow = TRUE
)
pmat = cmat / n

# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

# graph from the adjacency matrix
G = igraph::graph_from_adjacency_matrix(adjsymm, mode = "undirected", weighted = NULL)

# sampling a Markov move for the ERSBM
G_sample = sample_a_move_ERSBM(class, G)

# plotting the sampled graph
plot(G_sample, main = "The sampled graph after one Markov move for ERSBM")
```

---

zachary *Zachary Karate Club Data*

Description

Zachary's Karate club data is a classic, well-studied social network of friendships between 34 members of a Karate club at a US university, collected by Wayne Zachary in 1977. Each node represents a member of the club, and each edge represents a tie between two members of the club. The network is undirected. An often discussed problem using this dataset is to find the two groups of people into which the karate club split after an argument between two teachers.

Usage

```
zachary
```

Format

Two 34 by 34 matrices:

ZACHE symmetric, binary 34 by 34 adjacency matrix.

ZACHC symmetric, valued 34 by 34 matrix, indicating the relative strength of the associations

Source

(Zachary, 1977), `http://vlado.fmf.uni-lj.si/pub/networks/data/Ucinet/UciData.htm#zachary`.

Index

∗ **datasets**
    zachary, 18

get_mle_BetaSBM, 2
get_mle_ERSBM, 3
goftest_BetaSBM, 5
goftest_BetaSBM(), *2*, *13*, *16*
goftest_ERSBM, 8
goftest_ERSBM(), *4*, *14*, *17*
graphchi_BetaSBM, 12
graphchi_ERSBM, 14

sample_a_move_BetaSBM, 15
sample_a_move_ERSBM, 17

zachary, 18