GoodFitSBM: Monte Carlo goodness-of-fit tests for Stochastic Blockmodels (SBMs)

NEWS

GoodFitSBM (Version 0.0.1): `GoodFitSBM` comprises functionality that performs the *goodness-of-fit* test for a **beta-SBM** (one of the three variants of SBMs as discussed in Karwa et al. (2023), used for modelling network data).

Note

The math rendering of `README.md` is not ideal, as a result of which most of the math notations and equations remains un-rendered, hence `knit` the `README.Rmd`, or refer to the PDF attached as `README.pdf`.

Overview

Stochastic blockmodels (SBMs) contributed to the theoretical and algorithmic developments for analyzing *network* data, which has been facilitated by the availability of data in diverse fields like *social sciences*, *web recommender systems*, *protein networks*, *genomics*, and *neuroscience*. SBMs being the generalization of the Erdős-Rényi model given by Erdős and Rényi (1960) was proposed originally in context of *social sciences* for directed and undirected graphs, whereas now it has been vastly extended and utilized in *latent blocks in undirected graphs*, *latent space models*, *variable degree distribution*, *dynamically evolving networks*, etc., becoming one of the more popular approaches to model network data in computer science, statistics and machine learning.

Karwa et al. (2023) addresses a very important aspect of *model fitting* by constructing (exact) goodness-of-fit tests (under finite-sample setting) for three variants of SBMs respectively used for modeling network data (where model adequacy procedures are somewhat elusive in general), viz., *Erdős-Rényi SBM* (ER-SBM), *Additive SBM*, and *beta-SBM*, where the main idea revolves around a *frequentist conditional goodness-of-fit test* conditioned on a sufficient statistic (Karwa et al. (2023) also illustrates its Bayesian counterpart).

Intended Use of the package `GoodFitSBM`

Out of the three variants of the SBMs in (Karwa et al. (2023)) to model network data, `GoodfitSBM` addresses goodness-of-fit test under the framework of a **beta-SBM**. With a focus on *simple undirected*, and *unweighted* networks (graphs) having *no self-loop*, the package comprises of four functions viz., `get_mle()`, `goftest()`, `graphchi()`, and `sample_a_move()` - among which `goftest()` performs the major functionality of performing the goodness-of-fit test, in process, obtains the value of the chi-square test statistic and the corresponding $p-$value (using a Monte Carlo approach), after proper sampling of the graph (a Markov move or basis) - done by the function `sample_a_move()` under the beta-SBM framework. Computation of the chi-square test statistic value under the beta-SBM framework is done by `graphchi()`, which in turn requires the estimates (maximum likelihood estimation (MLE) in our case) of the edge probabilities q_{ij} between block B_i and block B_j as, $\widehat{q}_{ij}$, which is done by the function `get_mle()`.

The corresponding $p-$value obtained from `goftest()` are further analysed to examine the extent of fit of the beta-SBM to the given network (graph) (usual rule applies to reject the null of a good fit when, $p \leq \alpha$ at level $0 < \alpha < 1$). There are other functions collated, each performing the required functionality in the process, see the `R` files in GoodFitSBM Github Repo for details.

The beta-SBM and Related Goodness-of-Fit (GoF) Framework

In this section, we outline the theoretical part on which the entire package `GoodFitSBM` is based.

Stochastic Blockmodels (SBMs) Consider G to be a graph on n nodes, g being its realization. It is assumed that, all graphs are *unweighted* and *undirected*, and there are *no self-loops*. Therefore, the graph G

can also be referred to by its $n \times n$ adjacency matrix, $\mathbf{g}$, where $g_{uv} = 1$, if there is an edge between node u and v in the graph $\mathbf{g}$, and 0 otherwise.

An *exponential family random graph model (ERGM)* assumes that the probability of observing a graph $\mathbf{g}$ depends only on a vector of sufficient statistics, $T(\mathbf{g})$, i.e., the probability of observing a given network $G = \mathbf{g}$ is,

$$\mathbb{P}_\theta(G = \mathbf{g}) = \frac{exp\langle T(\mathbf{g}), \theta\rangle}{\psi(\theta)}$$

where $\psi(\theta) = \sum_{\mathbf{g}} exp(\langle T(\mathbf{g}), \theta\rangle)$ is the normalizing constant, $\theta \in \Theta$ is a vector of natural parameters, and $T(\mathbf{g})$ is the vector of minimal sufficient statistics for the underlying model. Let p_{uv} be the probability of an edge between node u and node v, where it is assumed that, $g_{uv} \sim \text{Bernoulli}(p_{uv})$.

A *stochastic blockmodel (SBM)* postulates that the nodes are partitioned into k blocks and the probability p_{uv} depends on their block membership, where (Karwa et al. (2023)) considers three different *log-linear* parametrizations of p_{uv}, one of which yields the beta-SBM.

The beta-SBM Beta-SBM is exponential family version of the degree-corrected stochastic blockmodels suggested by Karrer and Newman (2011). The log-linear model for the edge probabilities is parametrized by $\binom{k+1}{2}$ block parameters $\alpha_{z(u)z(v)}$ and n node-specific parameters β_u for $1 \le u \le n$, with the log-odds of the probability of an edge $\{u, v\}$ given by,

$$\log\left(\frac{p_{uv}}{1 - p_{uv}}\right) = \alpha_{z(u)z(v)} + \beta_u + \beta_v$$

When $\mathcal{Z}$ (the block assignments) are known, the beta-SBM is an exponential family model with natural parameter vector $\theta = (\beta, \alpha)$, where $\beta = (\beta_1, \ldots, \beta_n)$ and α is the vector of the upper diagonal elements of the $k \times k$ matrix $((\alpha_{i,j}))$. The natural parameter space is $\Theta \equiv \mathbb{R}^n \times \mathbb{R}^{\binom{k+1}{2}}$. The vector of sufficient statistics T_β contains the degree of each node $i \in [n]$ and the number of edges between each pair of blocks B_i and B_j for $1 \le i < j \le k$.

Goodness-of-Fit Test for the beta-SBM Let us define the goodness-of-fit test as,

$$H_0 : G \sim \mathbb{P}_\theta(G|Z)$$

against general alternatives, where $\mathbb{P}_\theta(G|Z)$ is a variant of the SBM with a generic parameter vector θ and block assignment Z (it is assumed that the number of blocks k is fixed and known).

Considering $T(\mathbf{g})$ to be the vector of sufficient statistics in $\mathbb{P}_\theta(G|Z)$, define a subset of the sample space as, $F_u := \{\mathbf{g} : T(\mathbf{g}) = u\}$, which is the known as the *fiber* of u under the given exponential family model. Now the usual chi-square statistic (for the goodness-of-fit test) is not constant on the above defined fibers of the beta-SBM variant. Thus, analogous to the beta-model in Petrovic et al. (2010), we use the Pearson's chi-square statistic,

$$GoF_Z(\mathbf{g}) = \chi^2_{\text{beta-SBM}}(\mathbf{g}, Z) = \sum_{1 \le u < v \le n} \frac{(g_{uv} - \widehat{g}_{uv})^2}{\widehat{g}_{uv}}$$

where, $\widehat{g}_{uv} = exp(\widehat{\alpha}_{z(u)z(v)} + \widehat{\beta}_u + \widehat{\beta}_v)/(exp(\widehat{\alpha}_{z(u)z(v)} + \widehat{\beta}_u + \widehat{\beta}_v))$, where the MLEs $\widehat{\alpha}$ and $\widehat{\beta}$ are associated with the MLEs of the edge probabilities q_{uv}, as obtained in the `get_mle()` function. Large values of $\chi^2_{\text{beta-SBM}}$ correspond to departure from the null (H_0) as stated above.

- For the algorithm of sampling of the graph (as done by `sample_a_move()`) - or a Markov move (basis) - and other details, see Section 5.3 of (Karwa et al. (2023)).

Use and Related Functionalities

In this section, we highlight the installation and different functionalities included in `GoodFitSBM` along with its implementation on some test cases and a real-life dataset.

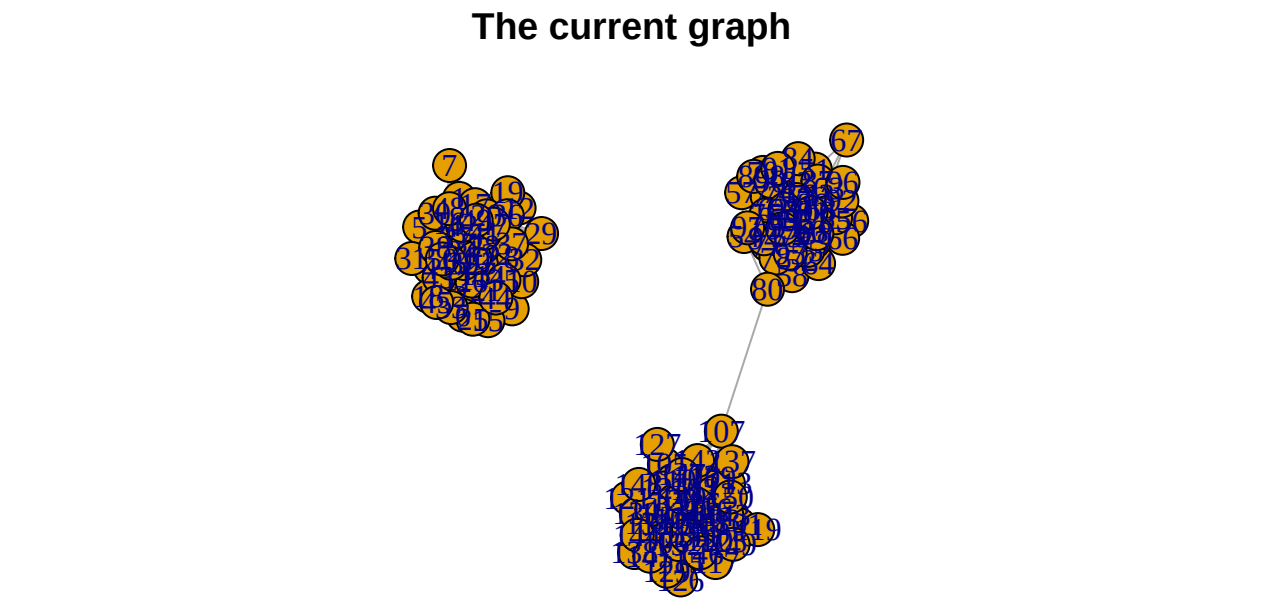
Installation To install and load-up the (development version 0.0.1) package `GoodFitSBM` from GoodFitSBM Github Repo, run the following commands.

```
# install.packages("devtools")
# install.packages("remotes")
remotes::install_github("Roy-SR-007/GoodFitSBM")

library(GoodFitSBM)
```

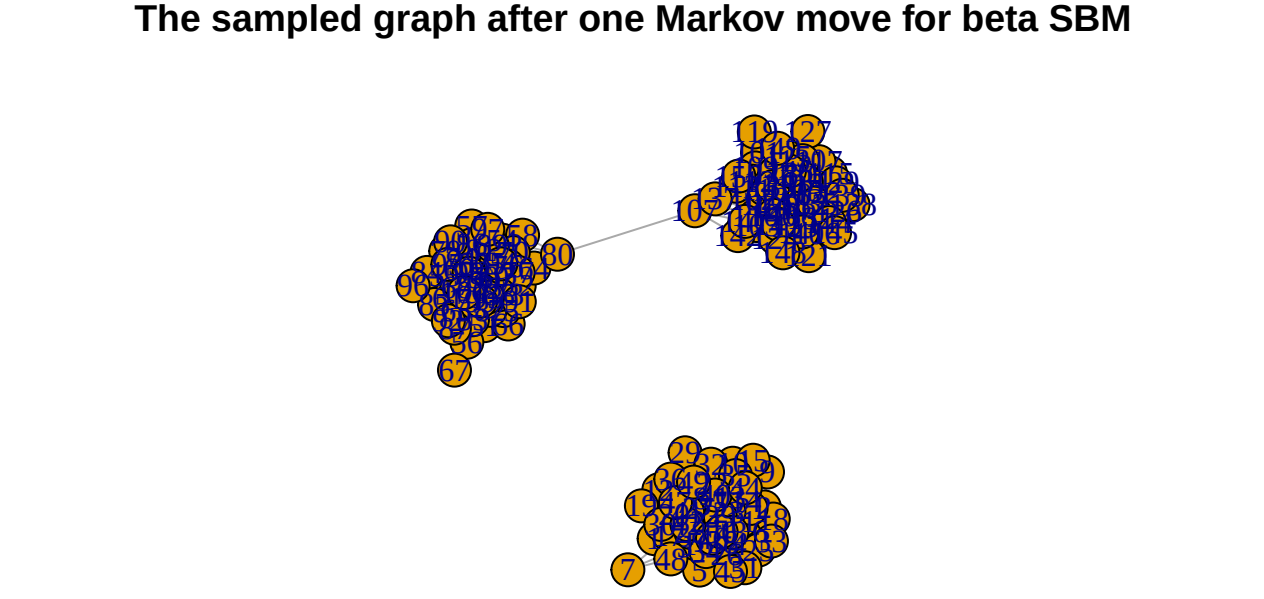
Example 1: Sampling a Graph using `sample_a_move()` Here we consider sampling (Markov move) a graph (under the beta-SBM framework) with a total of $n = 150$ nodes, with $k = 3$ blocks of sizes 50 each.

```
library(igraph)
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)

# plotting the current graph
plot(G, main = "The current graph")
```




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

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



Example 2: Estimating Edge Probabilities using `get_mle()` Here we determine the MLEs of the edge probabilities ($\widehat{q}_{uv}$) for a graph with $k = 3$ blocks each of size 2, and having a total of $n = 6$ nodes.

```
library(igraph)
RNGkind(sample.kind = "Rounding")
```

```r
set.seed(1729)

# We model a network with 3 even classes
n1 = 2
n2 = 2
n3 = 2

# 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, 0.50, 0.50,
    0.50, 30, 0.50,
    0.50, 0.50, 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(G, class)
```

```
#> 2 iterations: deviation 5.551115e-16
```

```
#>            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
#> [1,] 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333
#> [2,] 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333
#> [3,] 0.08333333 0.08333333 0.00000000 0.00000000 0.08333333 0.08333333
#> [4,] 0.08333333 0.08333333 0.00000000 0.00000000 0.08333333 0.08333333
#> [5,] 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333
#> [6,] 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333
```

Example 3: Computing the chi-square test statistic value using `graphchi()` Computing the chi-square test statistic value for a network with a total of $n = 9$ nodes, and $k = 3$ blocks of size 3 each.

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

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

# 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, 0.5, 0.5,
    0.5, 30, 0.5,
    0.5, 0.5, 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(G, class)
```

```
#> 2 iterations: deviation 6.661338e-16
```

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

```
#> [1] 16.66667
```

Main Example: `goftest()` in action

- **Instance 1**: Here we consider testing for the goodness-of-fit to the beta-SBM modeling a graph (network) with $k = 3$ blocks of size 50 each, and a total of $n = 150$ nodes.

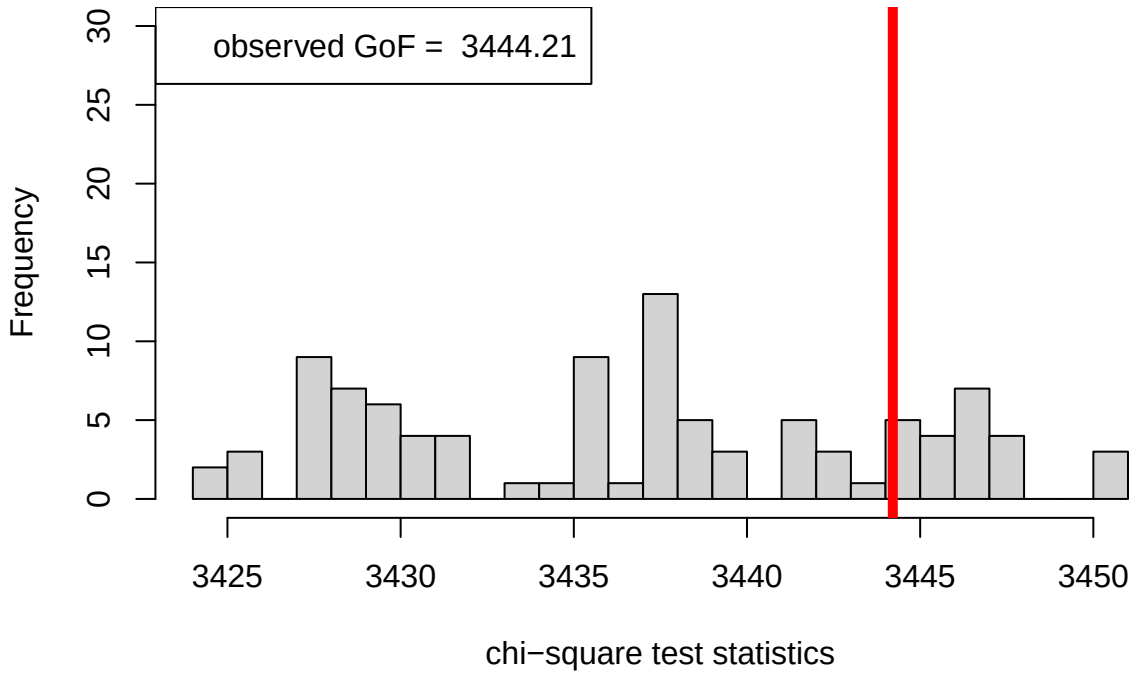
```r
library(igraph)
RNGkind(sample.kind = "Rounding")
set.seed(1729)

# 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(adjsymm, C = class, numGraphs = 100)
```

```
#> 2 iterations: deviation 1.580815e-10
```

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

```
#> [1] 0.2
```

```r
# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL, ylim = c(0, 30))
# adding test statistic on the observed network
abline(v = chi_sq_seq[1], col = "red", lwd = 5)
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))
```



From the $p-$value obtained viz., $p = 0.20 >> \alpha = 0.05$, we fail to the reject the null of a good fit of the given (observed) network to the beta-SBM, at level $\alpha = 0.05$.

- **Instance 2**: Here we consider testing for the goodness-of-fit to the beta-SBM modeling a graph (network) with $k = 3$ blocks with sizes 30, 20, and 50, i.e., with a total of $n = 100$ nodes.

```
library(igraph)
library(igraph)
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
```

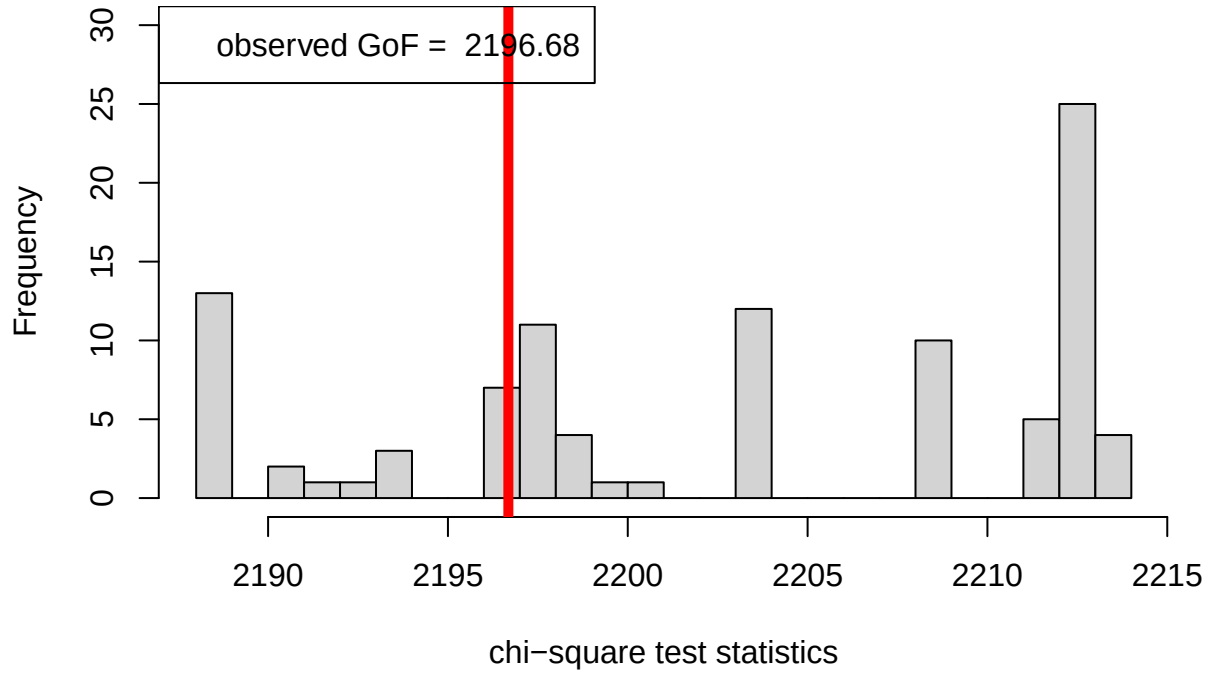
```r
# Creating the n x n adjacency matrix
adj <- matrix(0, n, n)
for (i in 2:n) {
  for (j in 1:(i - 1)) {
    p = pmat[class[i], class[j]] # We find the probability of connection with the weights
    adj[i, j] = rbinom(1, 1, p) # We include the edge with probability p
  }
}

adjsymm = adj + t(adj)

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

```
#> 2 iterations: deviation 3.353762e-11
```

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

```
#> [1] 0.73
```

```r
# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL, ylim = c(0, 30))
# adding test statistic on the observed network
abline(v = chi_sq_seq[1], col = "red", lwd = 5)
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))
```



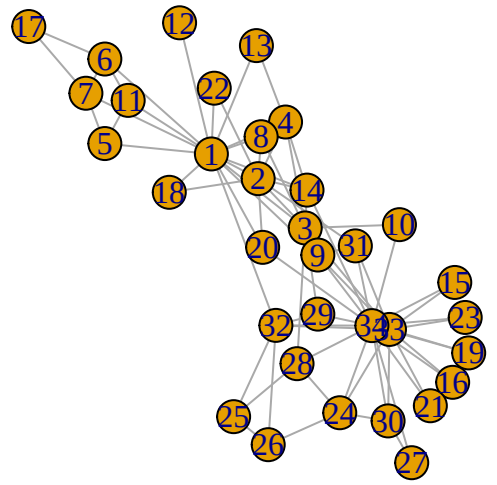
Observe that, the $p-$value obtained is much higher as compared to the previous instance viz., $p = 0.73 >>> \alpha = 0.05$, hence we fail to the reject the null of a good fit of the given (observed) network to the beta-SBM, at level $\alpha = 0.05$.

Application to Real Datasets: Zachary's Karate Club Data *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 (Zachary (1977)).

Here, we consider fitting the beta-SBM model with $k = 2$ blocks, each with sizes 10 and 24 respectively, i.e., a total of $n = 34$ nodes.

```
library(igraph)

set.seed(334003213)

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")
```



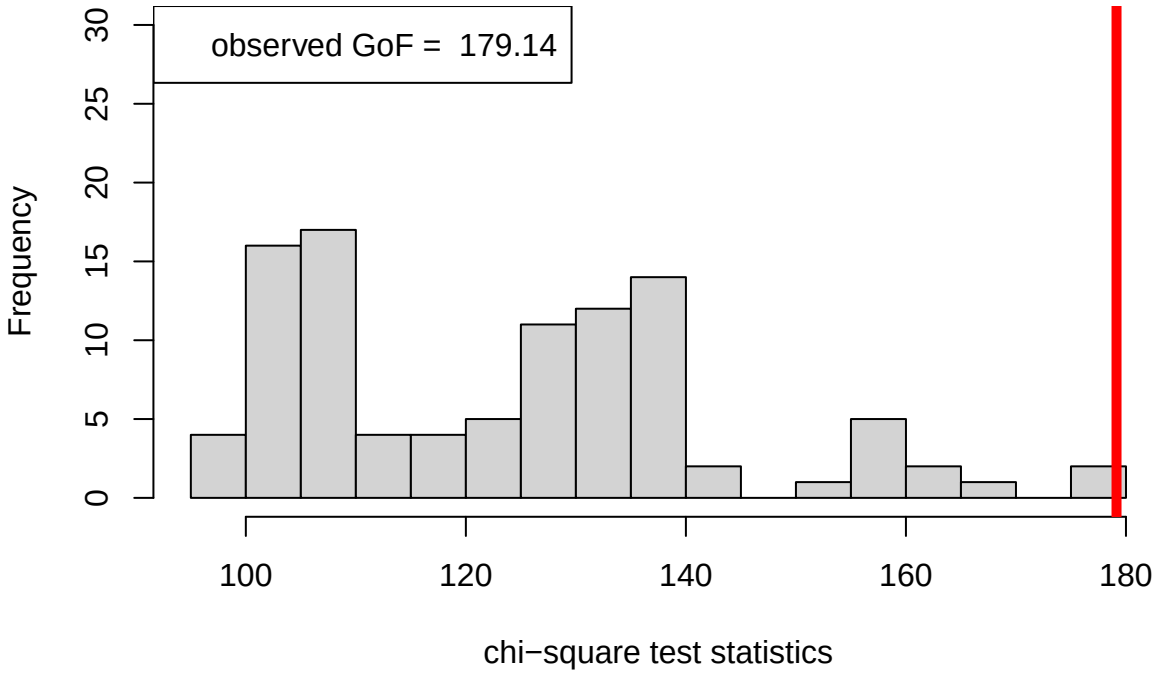

```
# block assignments
K = 2 # no. of blocks

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(A_zachary, C = class, numGraphs = 100)
```

```
#> 2 iterations: deviation 3.410605e-13
```

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

```
#> [1] 0
```

```
# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq, 20, xlab = "chi-square test statistics", main = NULL, ylim = c(0, 30))
# adding test statistic on the observed network
abline(v = chi_sq_seq[1], col = "red", lwd = 5)
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq[1]))
```



Note that, from the $p-$value obtained as well as the observed value of the chi-square test statistic (plotted histogram), we reject the null of a good fit, i.e., the data (network/graph) does not fit the beta-SBM at all; quite similar to the lines of conclusion as claimed by (Karwa et al. (2023)) under the framework of an ER-SBM, see Section 7.1.

An Addendum to Unknown Block Assignments All of the above examples along with the instances worked-out with the lines of code, deals with the situation where the *block assignments are known*. On the contrary, if the block assignments are *unknown* (with the number of blocks always fixed and known), we refer to (Qin and Rohe (2013)) for estimating the block assignments through *regularized spectral clustering*. See the following lines of code.

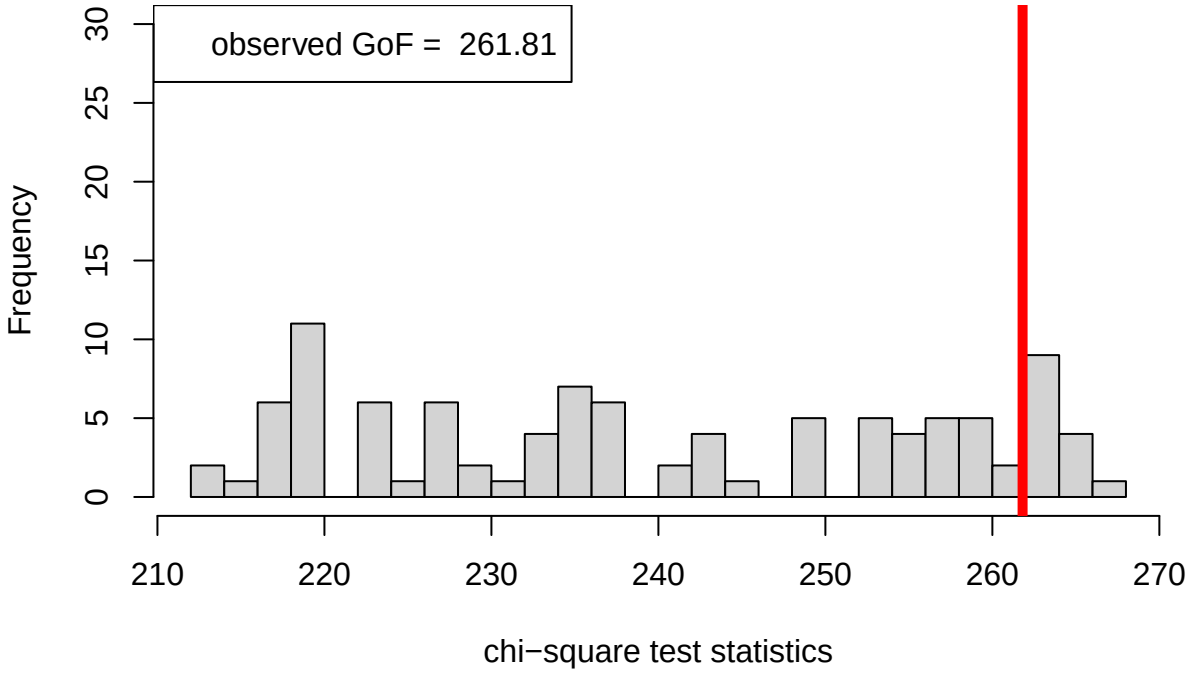
```
library(igraph)

set.seed(100)
# goodness-of-fit tests for the Zachary's Karate Club data set;
# with unknown block assignments
out_zachary_unknown = goftest(A_zachary, K = 2, C = NULL, numGraphs = 100)
```

```
#> 2 iterations: deviation 2.984279e-13
```

```r
chi_sq_seq_unknown = out_zachary_unknown$statistic
pvalue_unknown = out_zachary_unknown$p.value
print(pvalue_unknown)
```

```
#> [1] 0.14
```

```r
# Plotting histogram of the sequence of the test statistics
hist(chi_sq_seq_unknown, 20, xlab = "chi-square test statistics", main = NULL, ylim = c(0, 30))
# adding test statistic on the observed network
abline(v = chi_sq_seq_unknown[1], col = "red", lwd = 5)
legend("topleft", legend = paste("observed GoF = ", chi_sq_seq_unknown[1]))
```



With estimation of the unknown block assignments, the $p-$value obtained as well as the observed value of the chi-square test statistic (plotted histogram), similar to the case as shown earlier for known block assignments, we reject the null of a good fit, i.e., the data (network/graph) does not fit the beta-SBM at all.

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.
- Qin, T., and Rohe, K. (2013). "Regularized spectral clustering under the degree-corrected stochastic blockmodel", *Advances in neural information processing systems*, https://proceedings.neurips.cc/paper_files/paper/2013/file/0ed9422357395a0d4879191c66f4faa2-Paper.pdf.
- Lei, J., & Rinaldo, A. (2015). "Consistency of spectral clustering in stochastic block models", *The Annals of Statistics*, https://doi.org/10.1214/14-AOS1274.
- Li T, Levina E, & Zhu J (2021). "randnet: Random Network Model Estimation, Selection and Parameter Tuning", *R packageversion 0.3*, https://CRAN.R-project.org/package=randnet.
- Ghosh, S (2022). "MCGoSBM: Monte Carlo Goodness of Fit Tests for Stochastic Block Models", *Github Repository*, https://github.com/GhoshSoham/MCGoFSBM/tree/main.