ERGM

wkp

2021/4/18

##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 指数随机图模型示例  
# 依赖包：  
# (1)network：导入数据  
# (2)sna:可视化  
# (3)ergm:建模  
# 也可以直接加载statnet包  
##\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# 设定路径  
setwd("E:\\Office Account\\指数随机图模型2")  
# 加载包  
library(network)

## network: Classes for Relational Data  
## Version 1.16.1 created on 2020-10-06.  
## copyright (c) 2005, Carter T. Butts, University of California-Irvine  
## Mark S. Handcock, University of California -- Los Angeles  
## David R. Hunter, Penn State University  
## Martina Morris, University of Washington  
## Skye Bender-deMoll, University of Washington  
## For citation information, type citation("network").  
## Type help("network-package") to get started.

library(sna)

## Loading required package: statnet.common

##   
## Attaching package: 'statnet.common'

## The following object is masked from 'package:base':  
##   
## order

## sna: Tools for Social Network Analysis  
## Version 2.6 created on 2020-10-5.  
## copyright (c) 2005, Carter T. Butts, University of California-Irvine  
## For citation information, type citation("sna").  
## Type help(package="sna") to get started.

library(ergm)

##   
## ergm: version 3.11.0, created on 2020-10-14  
## Copyright (c) 2020, Mark S. Handcock, University of California -- Los Angeles  
## David R. Hunter, Penn State University  
## Carter T. Butts, University of California -- Irvine  
## Steven M. Goodreau, University of Washington  
## Pavel N. Krivitsky, UNSW Sydney  
## Martina Morris, University of Washington  
## with contributions from  
## Li Wang  
## Kirk Li, University of Washington  
## Skye Bender-deMoll, University of Washington  
## Chad Klumb  
## Micha? Bojanowski, Kozminski University  
## Ben Bolker  
## Based on "statnet" project software (statnet.org).  
## For license and citation information see statnet.org/attribution  
## or type citation("ergm").

## NOTE: Versions before 3.6.1 had a bug in the implementation of the bd()  
## constraint which distorted the sampled distribution somewhat. In  
## addition, Sampson's Monks datasets had mislabeled vertices. See the  
## NEWS and the documentation for more details.

## NOTE: Some common term arguments pertaining to vertex attribute and  
## level selection have changed in 3.10.0. See terms help for more  
## details. Use 'options(ergm.term=list(version="3.9.4"))' to use old  
## behavior.

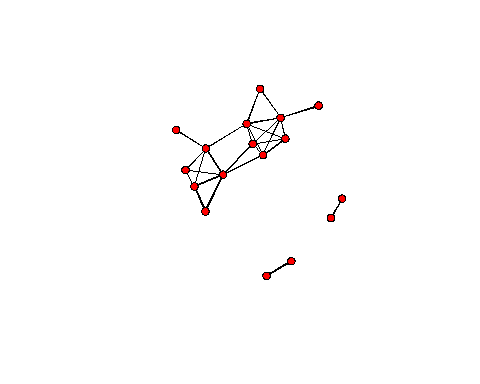
library(readxl)  
## Step1:导入数据  
data1 <- read\_xlsx("data1.xlsx")  
data1 <- as.matrix(data1)  
# 利用network包将原始数据转化为network对象  
net1 <- as.network(data1,directed = F)  
class(net1)

## [1] "network"

# 简单的描述统计：度数分布  
summary(net1 ~ degree(1) + degree(2) + degree(3))

## degree1 degree2 degree3   
## 6 2 1

# 利用sna包的gplot函数进行可视化，默认的可视化方式是fruchtermanreingold  
# 默认的可视化方式是fruchtermanreingold,可以调整mode参数进行更改  
# 另外：gmode参数指定--单向、无向,具体可以参照sna包gplot函数  
gplot(net1,gmode = "graph")



# 建模  
# (1)基准模型  
model1 <- ergm(net1 ~ edges,  
 control=control.ergm(MCMC.samplesize=100000,   
 MCMC.burnin=1000000,   
 MCMC.interval=1000,   
 seed = 567))

## Starting maximum pseudolikelihood estimation (MPLE):

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Stopping at the initial estimate.

## Evaluating log-likelihood at the estimate.

summary(model1)

## Call:  
## ergm(formula = net1 ~ edges, control = control.ergm(MCMC.samplesize = 1e+05,   
## MCMC.burnin = 1e+06, MCMC.interval = 1000, seed = 567))  
##   
## Iterations: 5 out of 20   
##   
## Monte Carlo MLE Results:  
## Estimate Std. Error MCMC % z value Pr(>|z|)   
## edges -1.395 0.215 0 -6.492 <1e-04 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Null Deviance: 188.5 on 136 degrees of freedom  
## Residual Deviance: 135.6 on 135 degrees of freedom  
##   
## AIC: 137.6 BIC: 140.5 (Smaller is better.)

# (2)加入内生结构项  
# 注：网络内生项可以根据模型调试加入，也可以通过模体分析再加入  
model2 <- ergm(net1 ~ edges + gwesp(0.1,T),  
 control=control.ergm(MCMC.samplesize=100,   
 MCMC.burnin=100,   
 MCMC.interval=100,   
 seed = 567))

## Starting maximum pseudolikelihood estimation (MPLE):  
## Evaluating the predictor and response matrix.  
## Maximizing the pseudolikelihood.  
## Finished MPLE.  
## Starting Monte Carlo maximum likelihood estimation (MCMLE):  
## Iteration 1 of at most 20:  
## Optimizing with step length 1.  
## The log-likelihood improved by 2.379.  
## Step length converged once. Increasing MCMC sample size.  
## Iteration 2 of at most 20:  
## Optimizing with step length 1.  
## The log-likelihood improved by 0.08643.  
## Step length converged twice. Stopping.  
## Finished MCMLE.  
## Evaluating log-likelihood at the estimate. Using 20 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 .  
## This model was fit using MCMC. To examine model diagnostics and check  
## for degeneracy, use the mcmc.diagnostics() function.

# (3)加入节点属性  
node\_attr <- read\_xlsx("patent.xlsx")  
net2 <- network(net1,vertex.attr = node\_attr,  
 vertex.attrnames = "patent",  
 directed = F)  
summary(net2)

## Network attributes:  
## vertices = 17  
## directed = FALSE  
## hyper = FALSE  
## loops = FALSE  
## multiple = FALSE  
## bipartite = FALSE  
## total edges = 27   
## missing edges = 0   
## non-missing edges = 27   
## density = 0.1985294   
##   
## Vertex attributes:  
##   
## patent:  
## numeric valued attribute  
## attribute summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 5.00 15.00 28.00 72.29 88.00 289.00   
## vertex.names:  
## character valued attribute  
## 17 valid vertex names  
##   
## No edge attributes  
##   
## Network edgelist matrix:  
## [,1] [,2]  
## [1,] 2 1  
## [2,] 3 1  
## [3,] 5 1  
## [4,] 7 1  
## [5,] 8 1  
## [6,] 9 1  
## [7,] 3 2  
## [8,] 5 2  
## [9,] 6 2  
## [10,] 8 2  
## [11,] 5 3  
## [12,] 6 3  
## [13,] 8 3  
## [14,] 7 4  
## [15,] 8 5  
## [16,] 9 5  
## [17,] 16 5  
## [18,] 7 6  
## [19,] 12 6  
## [20,] 13 6  
## [21,] 14 6  
## [22,] 13 7  
## [23,] 14 7  
## [24,] 15 10  
## [25,] 17 11  
## [26,] 13 12  
## [27,] 14 13

model3 <- ergm(net2 ~ edges + gwesp(0.1,T) + nodecov("patent"),  
 control=control.ergm(MCMC.samplesize=100,   
 MCMC.burnin=100,   
 MCMC.interval=100,  
 seed = 567))

## Starting maximum pseudolikelihood estimation (MPLE):  
## Evaluating the predictor and response matrix.  
## Maximizing the pseudolikelihood.  
## Finished MPLE.  
## Starting Monte Carlo maximum likelihood estimation (MCMLE):  
## Iteration 1 of at most 20:  
## Optimizing with step length 1.  
## The log-likelihood improved by 1.954.  
## Step length converged once. Increasing MCMC sample size.  
## Iteration 2 of at most 20:  
## Optimizing with step length 1.  
## The log-likelihood improved by 0.677.  
## Step length converged twice. Stopping.  
## Finished MCMLE.  
## Evaluating log-likelihood at the estimate. Using 20 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 .  
## This model was fit using MCMC. To examine model diagnostics and check  
## for degeneracy, use the mcmc.diagnostics() function.

summary(model3)

## Call:  
## ergm(formula = net2 ~ edges + gwesp(0.1, T) + nodecov("patent"),   
## control = control.ergm(MCMC.samplesize = 100, MCMC.burnin = 100,   
## MCMC.interval = 100, seed = 567))  
##   
## Iterations: 2 out of 20   
##   
## Monte Carlo MLE Results:  
## Estimate Std. Error MCMC % z value Pr(>|z|)   
## edges -3.232041 0.579206 0 -5.580 <1e-04 \*\*\*  
## gwesp.fixed.0.1 0.886165 0.422616 0 2.097 0.0360 \*   
## nodecov.patent 0.004648 0.001835 0 2.533 0.0113 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Null Deviance: 188.5 on 136 degrees of freedom  
## Residual Deviance: 117.0 on 133 degrees of freedom  
##   
## AIC: 123 BIC: 131.7 (Smaller is better.)

# (4)加入外生协变量  
tech\_proximity <- read\_xlsx("tech\_proximity.xlsx",col\_names = T)  
tech\_proximity <- as.matrix(tech\_proximity)  
net2 %e% "tech\_proximity" <- tech\_proximity  
summary(net2)

## Network attributes:  
## vertices = 17  
## directed = FALSE  
## hyper = FALSE  
## loops = FALSE  
## multiple = FALSE  
## bipartite = FALSE  
## total edges = 27   
## missing edges = 0   
## non-missing edges = 27   
## density = 0.1985294   
##   
## Vertex attributes:  
##   
## patent:  
## numeric valued attribute  
## attribute summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 5.00 15.00 28.00 72.29 88.00 289.00   
## vertex.names:  
## character valued attribute  
## 17 valid vertex names  
##   
## Edge attributes:  
##   
## tech\_proximity:  
## numeric valued attribute  
## attribute summary:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.7101 0.8946 0.9671 0.9301 0.9913 1.0000   
##   
## Network edgelist matrix:  
## [,1] [,2]  
## [1,] 2 1  
## [2,] 3 1  
## [3,] 5 1  
## [4,] 7 1  
## [5,] 8 1  
## [6,] 9 1  
## [7,] 3 2  
## [8,] 5 2  
## [9,] 6 2  
## [10,] 8 2  
## [11,] 5 3  
## [12,] 6 3  
## [13,] 8 3  
## [14,] 7 4  
## [15,] 8 5  
## [16,] 9 5  
## [17,] 16 5  
## [18,] 7 6  
## [19,] 12 6  
## [20,] 13 6  
## [21,] 14 6  
## [22,] 13 7  
## [23,] 14 7  
## [24,] 15 10  
## [25,] 17 11  
## [26,] 13 12  
## [27,] 14 13

model4 <- ergm(net2 ~ edges + gwesp(0.1,T) +   
 nodecov("patent") + edgecov(tech\_proximity),  
 control=control.ergm(MCMC.samplesize=100,   
 MCMC.burnin=100,   
 MCMC.interval=100,  
 seed = 567))

## Starting maximum pseudolikelihood estimation (MPLE):  
## Evaluating the predictor and response matrix.  
## Maximizing the pseudolikelihood.  
## Finished MPLE.  
## Starting Monte Carlo maximum likelihood estimation (MCMLE):  
## Iteration 1 of at most 20:  
## Optimizing with step length 0.996331240658116.  
## The log-likelihood improved by 1.727.  
## Iteration 2 of at most 20:  
## Optimizing with step length 1.  
## The log-likelihood improved by 0.2131.  
## Step length converged once. Increasing MCMC sample size.  
## Iteration 3 of at most 20:  
## Optimizing with step length 1.  
## The log-likelihood improved by 0.0567.  
## Step length converged twice. Stopping.  
## Finished MCMLE.  
## Evaluating log-likelihood at the estimate. Using 20 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 .  
## This model was fit using MCMC. To examine model diagnostics and check  
## for degeneracy, use the mcmc.diagnostics() function.

# 模型有效性检验  
# (1)拟合优度检验  
# 以model4为例  
model4\_gof <- gof(model4,GOF = ~ degree + espartners+ dspartners,  
 verbose = T, burnin = 10000,  
 interval = 10000)

## Starting GOF for the given ERGM formula.  
## Calculating observed network statistics.  
## Starting simulations.  
## Sim 1 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 2 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 3 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 4 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 5 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 6 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 7 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 8 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 9 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 10 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 11 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 12 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 13 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 14 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 15 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 16 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 17 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 18 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 19 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 20 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 21 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 22 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 23 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 24 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 25 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 26 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 27 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 28 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 29 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 30 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 31 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 32 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 33 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 34 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 35 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 36 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 37 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 38 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 39 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 40 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 41 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 42 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 43 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 44 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 45 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 46 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 47 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 48 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 49 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 50 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 51 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 52 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 53 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 54 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 55 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 56 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 57 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 58 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 59 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 60 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 61 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 62 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 63 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 64 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 65 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 66 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 67 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 68 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 69 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 70 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 71 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 72 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 73 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 74 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 75 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 76 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 77 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 78 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 79 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 80 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 81 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 82 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 83 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 84 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 85 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 86 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 87 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 88 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 89 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 90 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 91 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 92 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 93 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 94 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 95 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 96 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 97 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 98 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 99 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.  
## Sim 100 of 100: Starting MCMC iterations to generate 1 network   
## Finished simulation 1 of 1.

par(mfrow = c(2,3),mar = c(4,4,4,1),cex.main = .9,cex.lab = .9,cex.axis = .75)  
plot(model4\_gof,plotlogodds = T)  
# (2)MCMC检验  
mcmc.diagnostics(model4)

## Sample statistics summary:  
##   
## Iterations = 100:40000  
## Thinning interval = 100   
## Number of chains = 1   
## Sample size per chain = 400   
##   
## 1. Empirical mean and standard deviation for each variable,  
## plus standard error of the mean:  
##   
## Mean SD Naive SE Time-series SE  
## edges -0.7625 4.915 0.2458 0.3675  
## gwesp.fixed.0.1 -0.9887 6.328 0.3164 0.4769  
## nodecov.patent -95.8075 1069.827 53.4914 91.7708  
## edgecov.tech\_proximity -0.6507 4.627 0.2314 0.3456  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## edges -10.000 -4.000 -1.0000 3.000 9.00  
## gwesp.fixed.0.1 -12.551 -5.456 -1.0349 3.371 11.47  
## nodecov.patent -2373.775 -807.500 -95.0000 632.500 1865.32  
## edgecov.tech\_proximity -9.547 -3.759 -0.7425 2.665 8.28  
##   
##   
## Sample statistics cross-correlations:  
## edges gwesp.fixed.0.1 nodecov.patent  
## edges 1.0000000 0.9156925 0.8581256  
## gwesp.fixed.0.1 0.9156925 1.0000000 0.8592637  
## nodecov.patent 0.8581256 0.8592637 1.0000000  
## edgecov.tech\_proximity 0.9986704 0.9186672 0.8625615  
## edgecov.tech\_proximity  
## edges 0.9986704  
## gwesp.fixed.0.1 0.9186672  
## nodecov.patent 0.8625615  
## edgecov.tech\_proximity 1.0000000  
##   
## Sample statistics auto-correlation:  
## Chain 1   
## edges gwesp.fixed.0.1 nodecov.patent edgecov.tech\_proximity  
## Lag 0 1.00000000 1.000000000 1.00000000 1.000000000  
## Lag 100 0.38086120 0.387607683 0.49186625 0.380086923  
## Lag 200 0.17933830 0.156614512 0.23978834 0.182997023  
## Lag 300 0.01726771 0.055121556 0.08260131 0.022935906  
## Lag 400 0.01112754 0.049843230 0.07075560 0.021677080  
## Lag 500 -0.01526456 0.001730352 0.02821728 -0.006095341  
##   
## Sample statistics burn-in diagnostic (Geweke):  
## Chain 1   
##   
## Fraction in 1st window = 0.1  
## Fraction in 2nd window = 0.5   
##   
## edges gwesp.fixed.0.1 nodecov.patent   
## 0.06720 -0.09705 -0.26060   
## edgecov.tech\_proximity   
## 0.09809   
##   
## Individual P-values (lower = worse):  
## edges gwesp.fixed.0.1 nodecov.patent   
## 0.9464262 0.9226863 0.7943993   
## edgecov.tech\_proximity   
## 0.9218593   
## Joint P-value (lower = worse): 0.8111692 .

##   
## MCMC diagnostics shown here are from the last round of simulation, prior to computation of final parameter estimates. Because the final estimates are refinements of those used for this simulation run, these diagnostics may understate model performance. To directly assess the performance of the final model on in-model statistics, please use the GOF command: gof(ergmFitObject, GOF=~model).

## 至此指数随机图模型的三类变量都依次加入模型中  
# 模型整理  
library(texreg)

## Version: 1.37.5  
## Date: 2020-06-17  
## Author: Philip Leifeld (University of Essex)  
##   
## Consider submitting praise using the praise or praise\_interactive functions.  
## Please cite the JSS article in your publications -- see citation("texreg").

screenreg(list(model1,model2,model3,model4),digits = 3)

##   
## ========================================================================  
## Model 1 Model 2 Model 3 Model 4   
## ------------------------------------------------------------------------  
## edges -1.396 \*\*\* -2.719 \*\*\* -3.232 \*\*\* -7.346   
## (0.215) (0.562) (0.579) (3.992)   
## gwesp.fixed.0.1 1.084 \* 0.886 \* 0.861 \*  
## (0.473) (0.423) (0.425)   
## nodecov.patent 0.005 \* 0.004 \*  
## (0.002) (0.002)   
## edgecov.tech\_proximity 4.578   
## (4.341)   
## ------------------------------------------------------------------------  
## AIC 137.553 129.627 122.992 122.382   
## BIC 140.466 135.452 131.730 134.032   
## Log Likelihood -67.777 -62.813 -58.496 -57.191   
## ========================================================================  
## \*\*\* p < 0.001; \*\* p < 0.01; \* p < 0.05

