Gaussian Graphical Model Pre

HeYing

Dec 1st, 2018

高斯图模型的模拟实现

Tasks:

- 分别使用三种数据生成模式模拟实现 GGM
- 对每一种数据模式，分别使用罚项对数似然函数和线性回归模型的方式进行求解
- 罚项系数 λ 或 ρ 的值通过 10-folds 交叉验证确定，模型评价标准使用 Frobenius 范数

三种数据生成模式（精度矩阵 c 的形式不同）：

- $c_{i,i} = \frac{1}{i}$, $c_{i,j} = 0$，即仅对角元素非零；
- $c_{i,i} = 1$, $c_{i,i-1} = c_{i-1,i} = 0.3$, $others = 0$, 对角线及其外沿非零；
- $c = b + \rho \times I$, 其中 b 为二项分布变量，ρ 为概率，I 为单位对角阵，其中 ρ 必须使得矩阵是正定的。

数据准备

这里设置了两个维度，一个低维的 $(n = 20, p = 50)$，以及一个高维的 $(n = 250, p = 500)$，这样做是因为维度过高时，绘制出的无向图变量过于密集，容易掩盖变量间的连线，不够美观，较低维度的无向图呈现比较清晰。

高维的情况则是使用了 space package 中的 spaceSimu 数据集

```
rm(list = ls())


# install.packages("glasso")
# install.packages("igraph")
# install.packages("qgraph")
library(MASS)
library(glasso) # Graphical Lasso
library(igraph) # 画图
#library(qgraph) # 画图

# 给定维度
```

```
n = 20; p = 40
# mu = rep(0, p)

n2 = 200; p2 = 400
# mu2 = rep(0, p2)
```

k-fold 交叉验证

分配样本点到 k 个数据子集

进行 k 折交叉验证时，需要随机将每个样本点分配到 k 个数据子集里，编写函数 assignData(n, k)

```
assignData <- function (n, k) {
    small_set <- floor(n/k)
    group_assign <- NULL
    if (n%%k == 0) { # n能被k整除时
        group_assign <- rep(seq(1, k), n/k)
    }
    else { # n不能被k整除
        remainder <- n%%k
        for (i in 1:remainder) {
            group_assign <- c(group_assign, rep(i, small_set +
                1))
        }
        group_assign <- c(group_assign, rep(seq((i + 1), k),
            small_set))
    }
    sample(group_assign)
}
```

交叉验证的函数 my.glasso_cv()

```
my.glasso_cv <- function(x=NULL, k=10, n, p, lambdaList=NULL,
                         solveType = "likelihood",
                        dataMode = "Diag"){

  # parameters:
  #            x: 样本数据集相关信息,这一参数当且仅当dataMode
  #               取值为"Actual"时会用到, x是一个list, x[[1]]
  #               是样本数据集, 是一个n*p维的矩阵, x[[2]]是
  #               x[[1]]对应的真实协方差矩阵;
  #            k: 交叉验证fold数, 默认为10折交叉验证;
```

```
#            n: 样本数
#            p: 变量个数
#   lambdaList: 待遍历的lambda值列表, 若为空, 会给出一个合适的范围;
#    solveType: 求解方式, "likelihood"表示对数罚项似然,
#               "linear"代表线性回归方法求解, 默认"likelihood";
#     dataMode: 数据生成模式, "Diag"表示仅对角元素非零,
#               "DiagExtend"表示对角线以及次对角线元素非零,
#               "Binomial"表示按照二项分布概率随机指定非零元素,
#               "Actual"表示已给定一个实际数据集,
#                       包括该数据集的真实方差矩阵,
#                       这一情况下,不需要经过数据生成过程.

# 检查参数solveType是否符合要求
if(all(solveType != c("likelihood", "linear"))){
  stop("The solveType should be either likelihood or linear.")
}

# 检查参数dataMode是否符合要求
if(all(dataMode!=c("Diag","DiagExtend","Binomial","Actual"))){
  stop("The dataMode value is not supported.")
}

# 根据dataMode, 生成对应的数据样本方差矩阵s和精度矩阵c
if(dataMode == "Diag"){
  mu <- rep(0,p)
  sigma <- diag(seq(p))
  c <- solve(sigma)
  x <- mvrnorm(n,mu,sigma)
  s <- var(x)
}
if(dataMode == "DiagExtend"){
  c <- diag(p)
  for (i in (2:p)) {
    c[i,i-1] <- c[i-1,i] <- .3
  }
  mu <- rep(0,p)
  sigma = solve(c)
  x <- mvrnorm(n,mu,sigma)
  s <- var(x)
}
```

```r
if(dataMode == "Binomial"){
  c <- matrix(rbinom(p*p,1,0.2),p)
  c[lower.tri(c)] <- 0
  diag(c) <- 0
  c <- c + t(c) + 10*diag(p)  # 常数系数使得矩阵是正定的
  mu <- rep(0,p)
  sigma <- solve(c)
  x <- mvrnorm(n,mu,sigma)
  s <- var(x)
}
if(dataMode == "Actual"){
  if (is.null(x)) {
    stop("The Actual data mode requires a input data list.")
  }
  c <- solve(x[[2]])
  x <- x[[1]]
}


# 载入所依赖的包
library(MASS) # 生成多元正态分布数据
library(glasso) # Graphical Lasso Model
#library(igraph) # 画图
#library(qgraph) # 画图

# 若lambdaList为空，按照以下方法为其赋值:
if(is.null(lambdaList)){
  lambda_max = max(abs(t(x)%*%x))
  lambda_min = lambda_max*1e-4
  lambdaList = seq(log(lambda_max),log(lambda_min),length.out = 100)
  lambdaList = exp(lambdaList)
}

# Cross Validation主体
# 数据准备
# k-折交叉验证，随机为每个样本点分配一个数据子集，
# 使用函数assignData(n,k).
foldsid <- assignData(n,k=10)

FrobNorm <- rep(0,length(lambdaList))
for (i in 1:length(lambdaList)) {
```

```
  # 遍历每一个lambda
  FrobNorm_tmp <- rep(0,k)
  for (j in seq(k)) {
    # 一个lambda值下进行10-folds CV
    xtrain <- x[foldsid != j,]
    s <- cov(xtrain)
    if (solveType == "likelihood") {
      r <- glasso(s,lambdaList[i])$wi
      FrobNorm_tmp[j] <- sqrt(sum((r-c)*(r-c)))
    }
    if (solveType == "linear") {
      weight <- glasso(s,lambdaList[i],approx = TRUE)$wi
      diag(weight) <- 1
      zero <- which(weight==0,arr.ind = TRUE)
      r <- glasso(s,0,zero)$wi
      FrobNorm_tmp[j] <- sqrt(sum((r-c)*(r-c)))
    }
  }
  # 取10次的均值作为该lambda值下的结果
  FrobNorm[i] <- mean(FrobNorm_tmp)
}

lambda_bestID <- which.min(FrobNorm)

# 为了防止矩阵过于稀疏，当Frobenius Norm指标的提升小于某个阈值时，
# 最优的lambda值不应该大于此阈值
FNrange <- max(FrobNorm)-min(FrobNorm)
for (f in 1:(length(lambdaList)-1)) {
  boost = FrobNorm[f+1]-FrobNorm[f]
  if (((dataMode=="Diag")|(dataMode=="DiagExtend"))&&(boost>0.3)){
    lambda_bestID = f+1
    break
  }
  if ((dataMode=="Binomial") && (boost>6)&&(solveType=="linear")) {
    lambda_bestID = f+1
    break
  }
}
```

```
  return(list(bestID = lambda_bestID,
              lambdaBest = lambdaList[lambda_bestID],
              lambdaList = lambdaList,
              FrobeniusNorm = FrobNorm,
              data = x, c =c))
  # 返回结果列表:
  #        bestID: 使得模型综合结果最优的lambda值对应的下标
  #    lambdaBest: 使得模型综合结果最优的lambda值
  #    lambdaList: 交叉验证时lambda遍历的取值列表
  # FrobeniusNorm: 不同lambda取值下Frobenius Norm指标的取值列表
  #          data: 用到的样本数据, n*p维
  #             c: 精度矩阵, p*p维
}
```

数据模拟及其结果

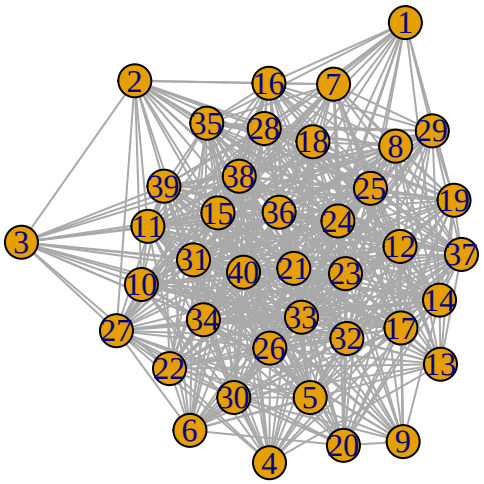
总共有四种数据模式，两种求解方法，共 $4 \times 2 = 8$ 种模拟及结果。

仅对角线元素非零，即

$c_{i,i} = \frac{1}{i}$, $c_{i,j} = 0 (i \neq j)$
采用罚项似然函数的求解方法 (Yuan and Lin, 2007),
低维: $n = 20$, $p = 40$

```
# library(igraph)
# library(qgraph)

Gresult <- my.glasso_cv(n=n, p=p, solveType = "likelihood",
                        dataMode = "Diag")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
s <- var(x)
r <- glasso(s,lambda_star)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Low dimensions: dataMode=Diag, solveType=likelihood")
```



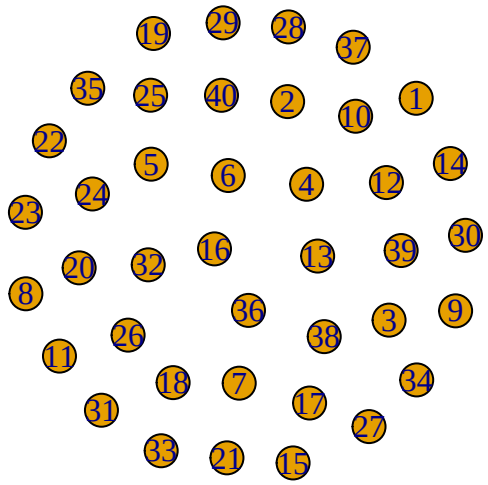

```
# 显示Fobenius Norm指标的值
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

```
## The value of Frobenius Norm is:
##  2.973323 ,
##  and the value of lambda is:
##  0.335759
```

采用线性回归的求解方法 (Meinshausen N., 2006),
低维: $n = 20$, $p = 40$

```
Gresult <- my.glasso_cv(n=n, p=p, solveType = "linear",
                        dataMode = "Diag")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
s <- var(x)
```

```
weight <- glasso(s,lambda_star)$wi
diag(weight) <- 1
zero <- which(weight==0,arr.ind = TRUE)
r <- glasso(s,1e-5,zero)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Low dimensions: dataMode=Diag, solveType=linear")
```




```
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

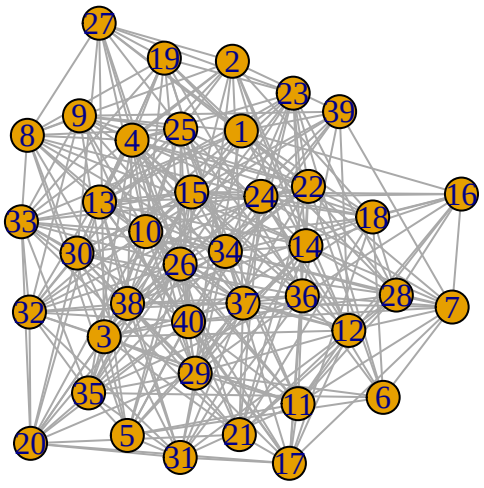
```
## The value of Frobenius Norm is:
##  0.346618 ,
##  and the value of lambda is:
##  901.204424
```

对角线元素以及次对角线元素非零

$c_{i,i} = 1$, $c_{i,i-1} = c_{i-1,i} = 0.3$, $others = 0$
采用罚项似然函数的求解方法 (Yuan and Lin, 2007),
低维: $n = 20$, $p = 40$

```
# library(igraph)
# library(qgraph)

Gresult <- my.glasso_cv(n=n, p=p, solveType = "likelihood",
                        dataMode = "DiagExtend")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
s <- var(x)
r <- glasso(s,lambda_star)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Low dimensions: dataMode=DiagExtend, solveType=likelihood")
```

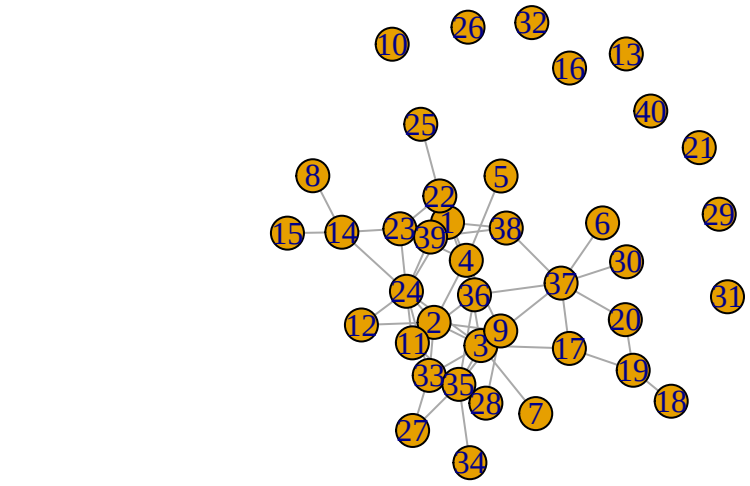


```
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

```
## The value of Frobenius Norm is:
##  4.125937 ,
##  and the value of lambda is:
##  0.167303
```

采用线性回归的求解方法 (Meinshausen N., 2006),
低维: $n = 20$, $p = 40$

```
Gresult <- my.glasso_cv(n=n, p=p, solveType = "linear",
                        dataMode = "DiagExtend")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
FNlist <- Gresult$FrobeniusNorm
s <- var(x)
weight <- glasso(s,lambda_star)$wi
diag(weight) <- 1
zero <- which(weight==0,arr.ind = TRUE)
r <- glasso(s,1e-5,zero)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Low dimensions: dataMode=DiagExtend, solveType=linear")
```




```
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

```
## The value of Frobenius Norm is:
##  6.397487 ,
##  and the value of lambda is:
##  0.533165
```

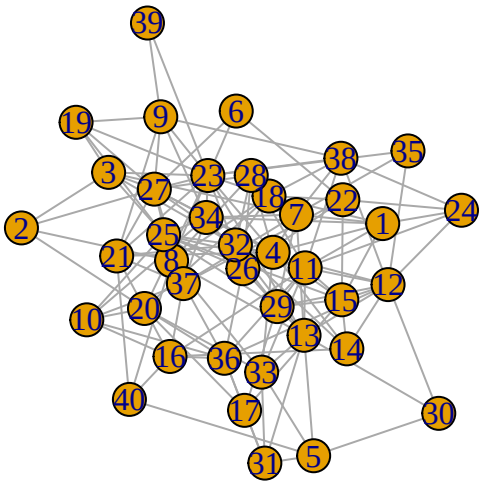
```
# FNlist
```

按照二项分布随机设定矩阵中的非零元

$c = b + \rho \times I$,
其中 b 为二项分布变量，ρ 为概率，I 为单位对角阵，其中 ρ 必须使得矩阵是正定的。
采用罚项似然函数的求解方法 (Yuan and Lin, 2007),
低维: $n = 20$, $p = 40$

```
# library(igraph)
# library(qgraph)

Gresult <- my.glasso_cv(n=n, p=p, solveType = "likelihood",
                        dataMode = "Binomial")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
s <- var(x)
r <- glasso(s,lambda_star)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Low dimensions: dataMode=Binomial, solveType=likelihood")
```

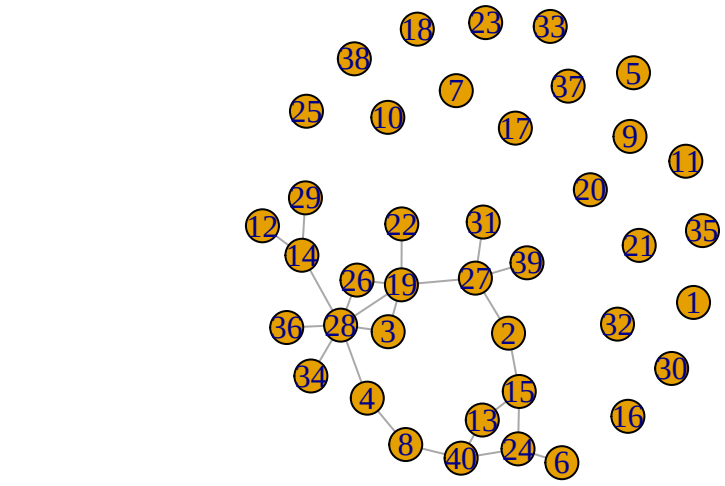



```
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

```
## The value of Frobenius Norm is:
##  27.190167 ,
##  and the value of lambda is:
##  0.032099
```

采用线性回归的求解方法 (Meinshausen N., 2006),
低维: $n = 20$, $p = 40$

```
Gresult <- my.glasso_cv(n=n, p=p, solveType = "linear",
                        dataMode = "Binomial")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
FNlist <- Gresult$FrobeniusNorm
s <- var(x)
weight <- glasso(s,lambda_star)$wi
diag(weight) <- 1
zero <- which(weight==0,arr.ind = TRUE)
r <- glasso(s,1e-8,zero)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Low dimensions: dataMode=Binomial, solveType=linear")
```




```
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

```
## The value of Frobenius Norm is:
##  57.10502 ,
##  and the value of lambda is:
##  0.055203
```

使用给定数据集

这里使用的是 space package 里的 spaceSimu 数据集
采用罚项似然函数的求解方法 (Yuan and Lin, 2007),

```
library(space)
data("spaceSimu")
np = dim(spaceSimu[[1]])
n = np[1]
p = np[2]
```

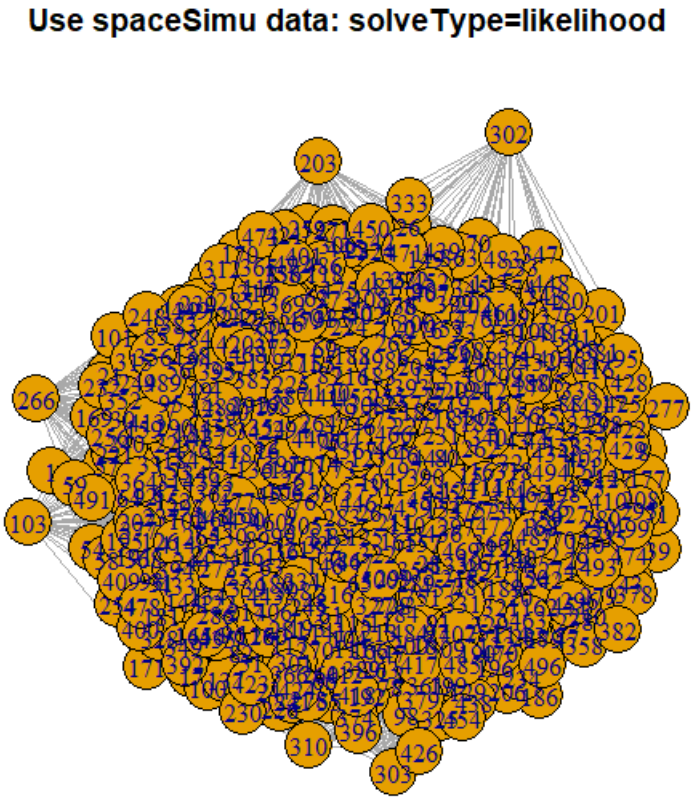


图 1:

```
Gresult <- my.glasso_cv(x=spaceSimu,n=n, p=p, solveType = "likelihood",
                        dataMode = "Actual")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
s <- var(x)
r <- glasso(s,lambda_star)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Use spaceSimu data: solveType=likelihood")
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

The value of Frobenius Norm is:

19.222894,
and the value of lambda is:
0.076041
采用线性回归的求解方法 (Meinshausen N., 2006),

```
# library(space)
# data("spaceSimu")
np = dim(spaceSimu[[1]])
n = np[1]
p = np[2]
Gresult <- my.glasso_cv(x=spaceSimu,n=n, p=p, solveType = "linear",
                        dataMode = "Actual")
x <- Gresult$data
c <- Gresult$c
lambda_star <- Gresult$lambdaBest
s <- var(x)
weight <- glasso(s,lambda_star)$wi
diag(weight) <- 1
zero <- which(weight==0,arr.ind = TRUE)
r <- glasso(s,1e-5,zero)$wi
adjacency <- abs(r) > 1e-4; diag(adjacency) <- 0
adjacency.plot <- graph.adjacency(adjacency, mode='undirected')
plot(adjacency.plot,
     main="Use spaceSimu data: solveType=linear")
FrobNorm <- Gresult$FrobeniusNorm[Gresult$bestID]
cat("The value of Frobenius Norm is: \n",
    as.character(round(FrobNorm,6)),
    ", \n","and the value of lambda is: \n",
    as.character(round(lambda_star,6)))
```

The value of Frobenius Norm is:
15.413151,
and the value of lambda is:
0.192792

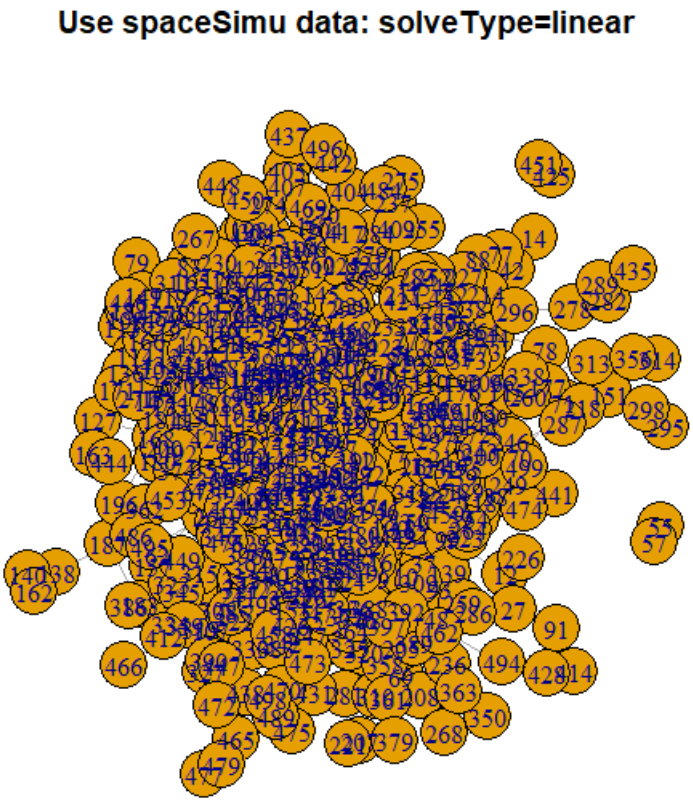


图 2: