dcsr-Omega

dscr-Omega

- Motivation
- methods
- dscr simulation

Motivation

In last decade, many authors were working on the problem of estimation of covariance matrix and its inverse, and the estimation of covariance matrix and its inverse become important in statistical analysis. For example, in multiple comparison, principle component analysis, linear discriminant analysis and graphical model, we need those estimation and apply them to our statistical analysis.

Motivation

In our project mvash, we want to deal with the problem of multiple comparison considering the covariance structure to gain more power.

The model is:

$$\hat{\beta} \sim N(\beta, \Sigma)$$

where the $\hat{\beta}$ is a P vector which is observed gene expression level, β is the true value and Σ is the covariance matrix under the Gaussian assumption. We need to estimate the $\Omega = \Sigma^{-1}$ from gene expression data.

Motivation

In the setting of P (dimension of variables) is larger than n (number of observations), the accuracy of estimation of covariance matrix and precision matrix and the computational cost are challenges in practical studies.

We deal with the data with P is around thousand or million and n is around hundred and thousand quite often, so the computational efficiency is a important criteria for our methods. Sometime we need consider the trade off between statistical accuracy and computational efficiency.

Difficulty

- In the setting P much larger than n, $\Sigma_n = \frac{1}{n} X^T X$ is singular and unstable. For the eigen values of Σ_n are underestimated when the eigen values of Σ are small and overestimated when the corresponding eigen values of Σ are large.
- When P is larger than n, Σ_n is not invertible. So we need to find some way to estimate the precision matrix Ω rather than inverse the sample covariance matrix Σ_n
- To estimate the Ω we have $\frac{p(P+1)}{2}$ parameters to estimate, while we only have n samples.

Methods

There are many methods to estimate the precision matrix Ω

Since the parameters are more than sample size. So we have to make some assumption about the estimation of precision matrix.

Reparameterization of covariance matrix and its inverse is still open problem. We want they are statistically interpretable and satisfying with the constraint themselves, for example, they need to be positive-definite.

One assumption we can make is that assume Ω is sparse.

Sparse Estimation

There are a lot of methods based on sparse assumption

Glasso: Graphical Lasso

Clime: Constrained L_1 Minimization

Tiger: Tuning-Insensitive Approach for Optimally Estimating Gaussian Graphical Models

PCS: Partial Correlation Screening

Sparse Estimation

- Glasso

$$min : logdet(\Omega) - trace(\Sigma_n \Omega) - \rho ||\Omega||_1$$

and then convert to a lasso problem for each row

- Clime

$$min : ||\Omega||_1$$

subject to

$$||\Sigma_n \Omega - I||_\infty$$

also use sparse estimation of each row and combine them together

Sparse Estimation

- PCS & Tiger

$$X_j = \sum_{i \neq j} \beta_i X_i + V_j$$

$$\beta_i = -\Omega_{ij}/\Omega_{jj};\ V_j \sim N(0, \frac{1}{\Omega_{jj}})$$

$i, j = 1 \cdots P$, X is the n by P data matrix.

PCS screening these partial correlations, Tiger minimize the sum square of prediction error with sparsity constraint.

problems

- If we want to make sparse assumption, we need to decide whether we want to make the covariance matrix sparse or the precision matrix. It is easy to understand that the inverse of a sparse matrix is no longer sparse.
- The marginal precision matrix is not sparse when the joint precision matrix is sparse

$$(\Sigma_O)^{-1} = \Omega_O - \Omega_{OH}\Omega_H^{-1}\Omega_{HO}$$

where O stand for observed, H means hidden, and the subscript means the corresponding partition of the Ω matrix.

Factor Model

Instead of starting from the precision matrix itself with sparsity assumption, we can start from the data and try to learn the data well, then turn to the precision matrix. We first put the data into a low-dimension space which might not be a subspace of what the data matrix expand (different from PCA) and try to capture all the variation information in this low-dimensional space. Hopefully we can recover most of the variation information in this lower dimensional subspace in order to estimate the covariance matrix and precision matrix.

$$X = LF + E$$

where Y is n by P matrix, L is n by K matrix, F is K by P matrix and $E \sim N(0, \Psi)$

Estimate Precision Matrix

- covariance matrix

$$\hat{\Sigma} = \Psi + L^T \Lambda_F L$$

- precision matrix

$$\hat{\Omega} = \Psi^{-1} - \Psi^{-1} L^T (\Lambda_F + L\Psi^{-1} L^T)^{-1} L\Psi^{-1}$$

Both $\hat{\Sigma}$ and $\hat{\Omega}$ are diagonal plus low rank.

Inverse $\hat{\Sigma}$ is not computationally expensive.

Model is statistically interpretable.

understanding factor model for estimating precision matrix

In gene expression data, gene can not function alone, instead, genes tend to work together to manifest biological function.

Imagine we can observe the latent factor (unmeasured, unobserved, confounding factor) in gene expression data, such as batch effects, latent population structure, biological covariates and transcript factor.

Usually we impose sparsity on factor loadings to facilitate kind of clustering information: gene with zero loadings not belongs to the "gene cluster" corresponding to the factor.

In this way, we divide the covariance structure into variation common genes and variation specific to gene.

All for Fun, Fun for All

Start with the easiest case of 3 nodes and two latent factors. Each latent factor connect with two observed nodes.

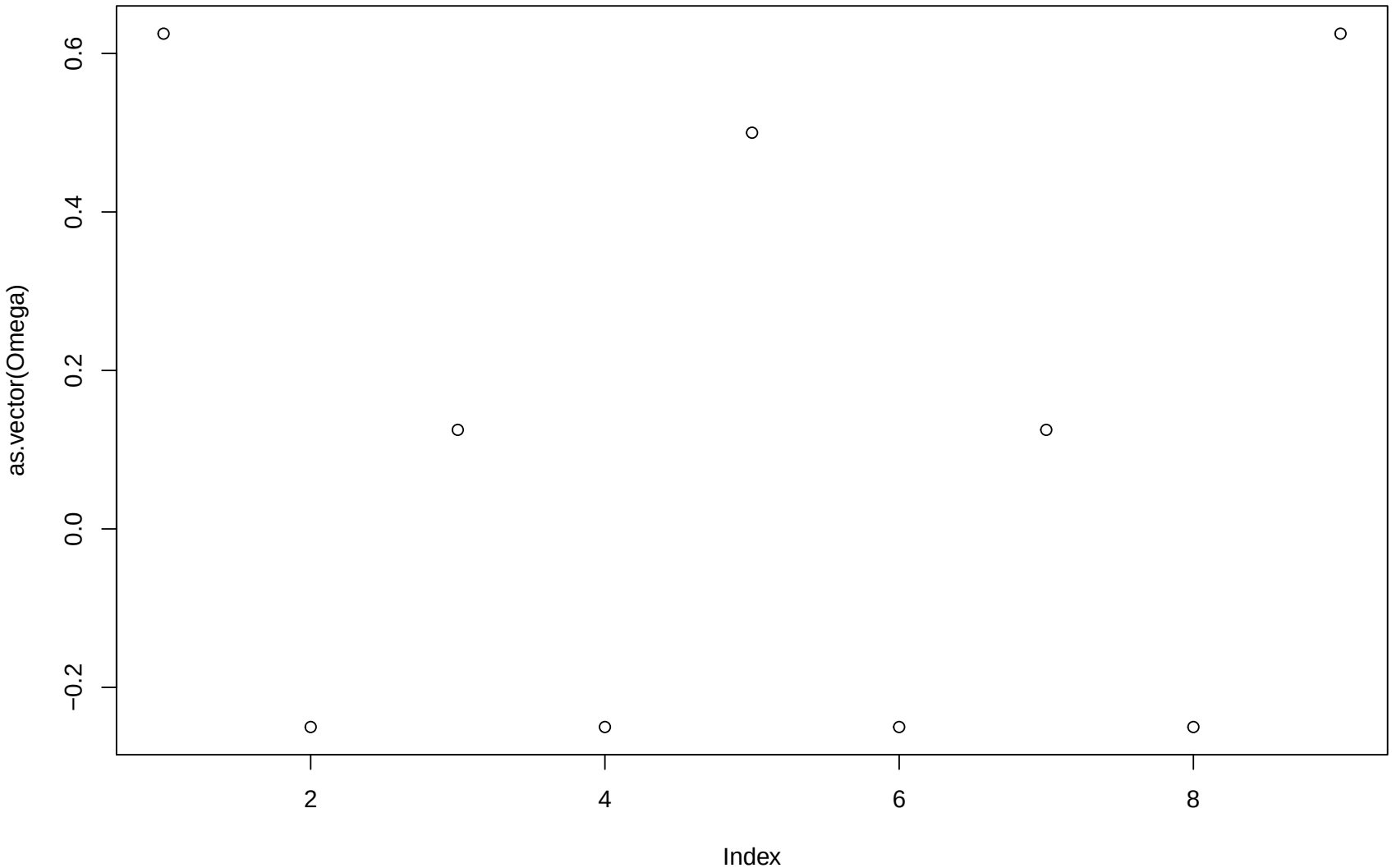
```
F
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    0
## [2,]    0    1    1
```

Set Λ_L and Ψ all to identity matrix.

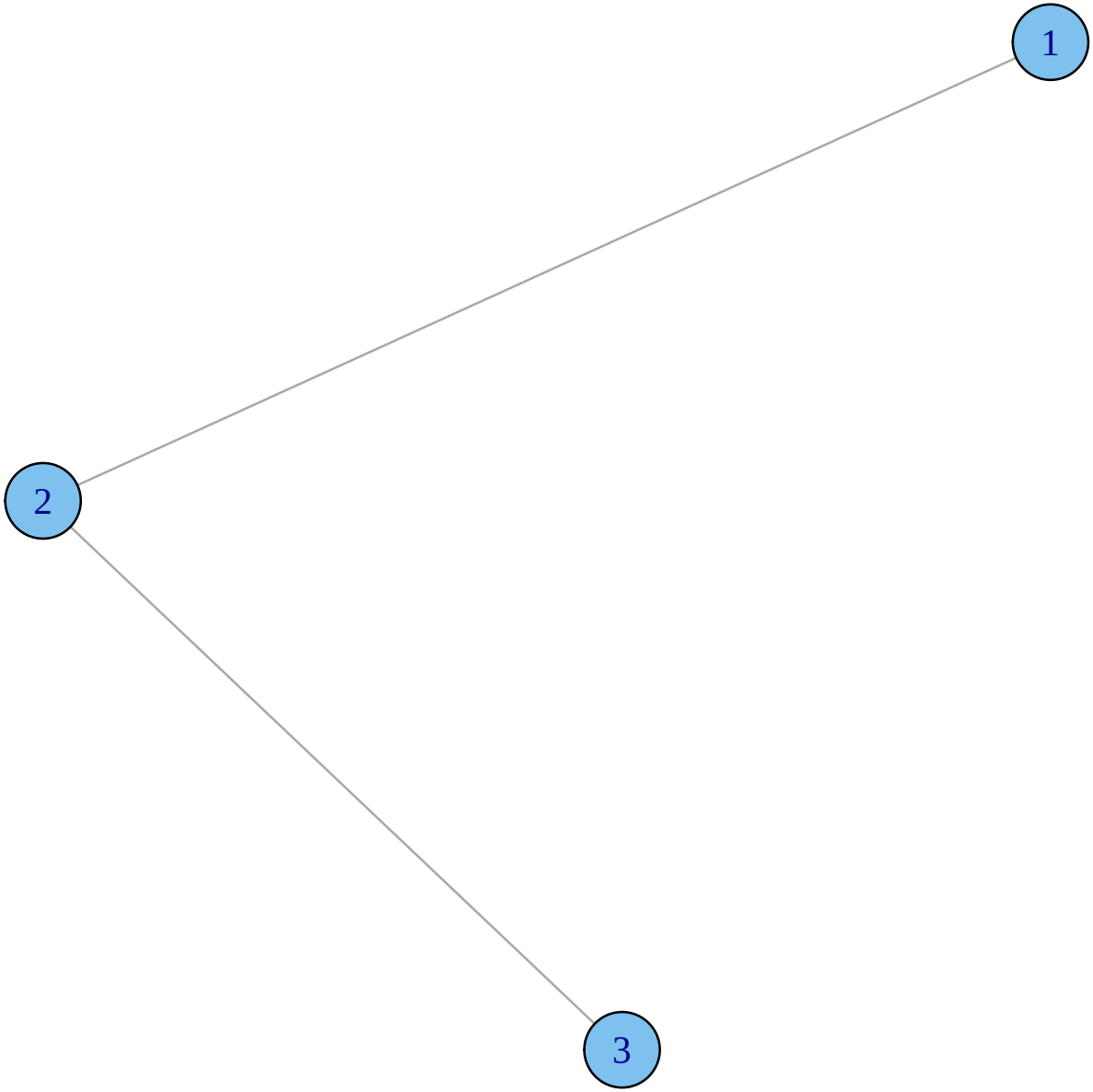
plot of 3 nodes case

Precision matrix calculate from factor model.



set a threshold cut the small values.

plot of 3 nodes case



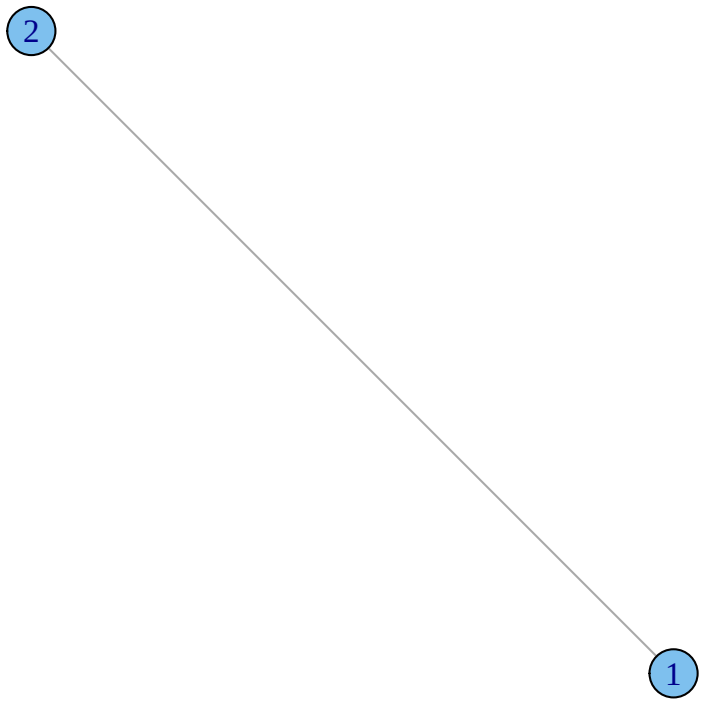
delete node 3 & delete node 2

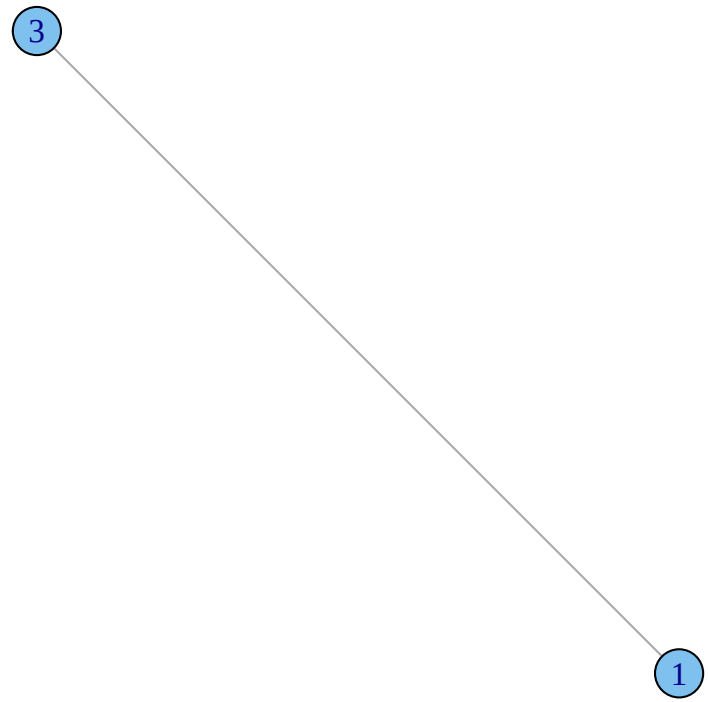
when we delete a observed node, it is equivalent to add a latent node into model. In our model, we just delete the corresponding column (3rd or 2nd) in the original F and add one more row as the loading for the new factor.

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    0    1
## [3,]    0    1
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
## [3,]    1    1
```

delete node 3 and delete node 2





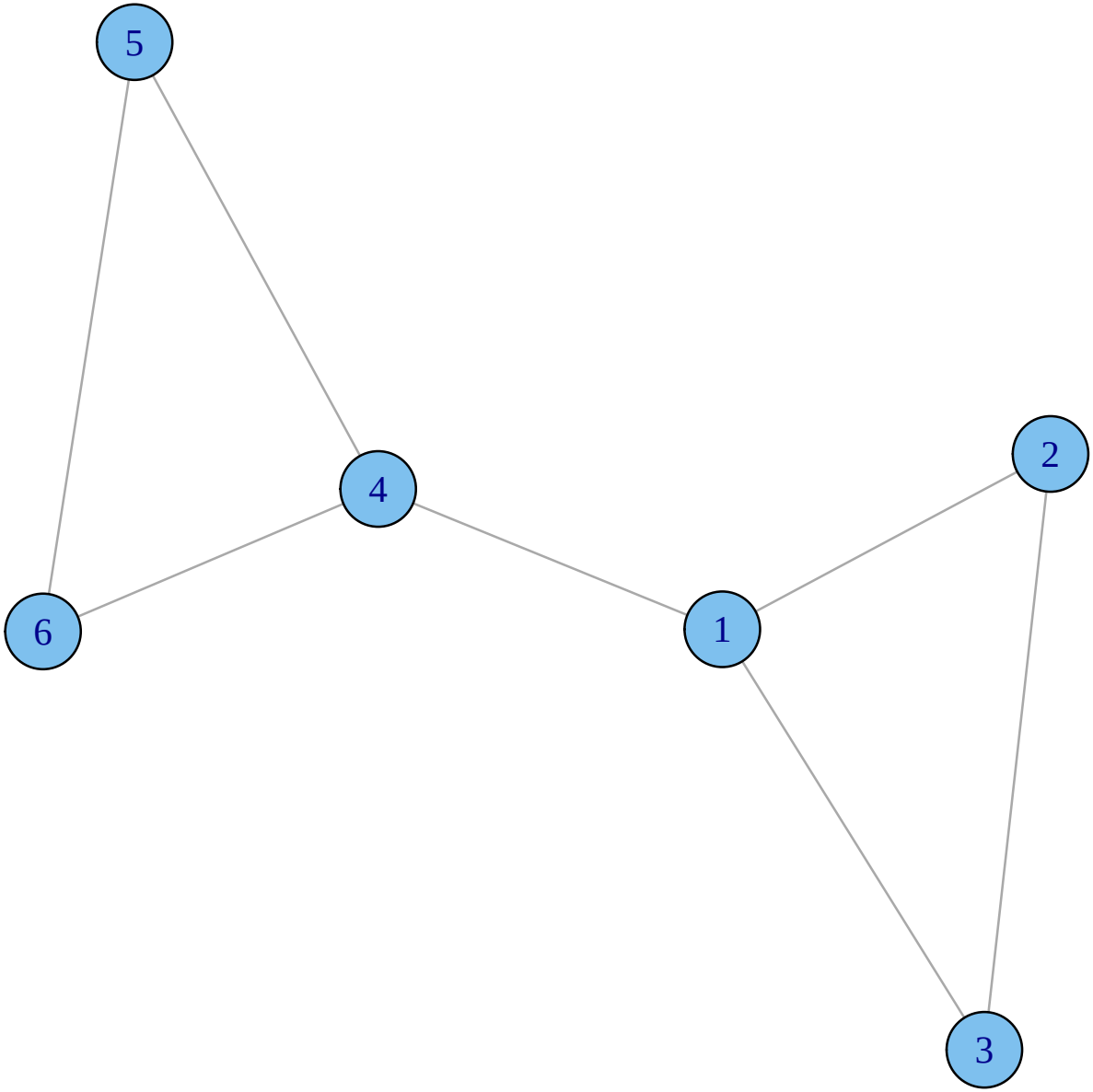
6 nodes case

By now we can see that the factor model is "consistent" applying to estimate precision matrix. We can try more examples.

Here is a example with 6 nodes and 3 latent factors.

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    0    0    0
## [2,]    0    0    0    1    1    1
## [3,]    1    0    0    1    0    0
```

6 nodes case



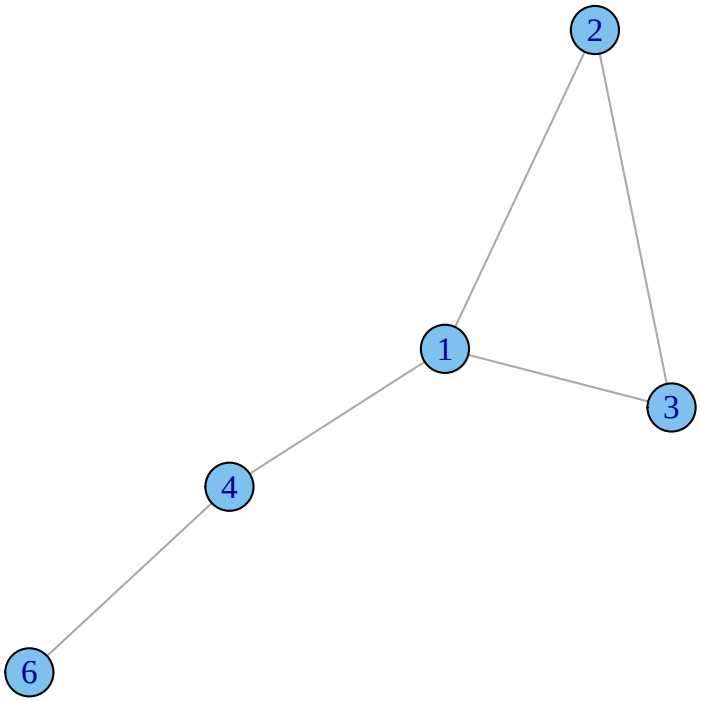
delete node 5 & delete node 4

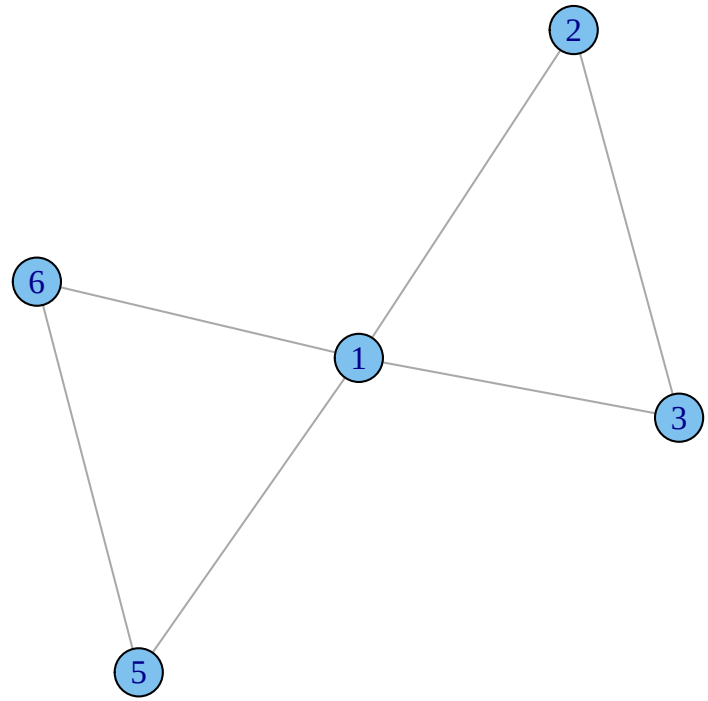
similarly, we delete one observed node, at the same time we have one more latent factor. n our model, we just delete the corresponding column (5th or 4th) in the original F and add one more row as the loading for the new factor.

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    1    0    0
## [2,]    0    0    0    1    1
## [3,]    1    0    0    1    0
## [4,]    0    0    0    1    1
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    1    0    0
## [2,]    0    0    0    1    1
## [3,]    1    0    0    0    0
## [4,]    1    0    0    1    1
```

delete node 5 & delete node 4





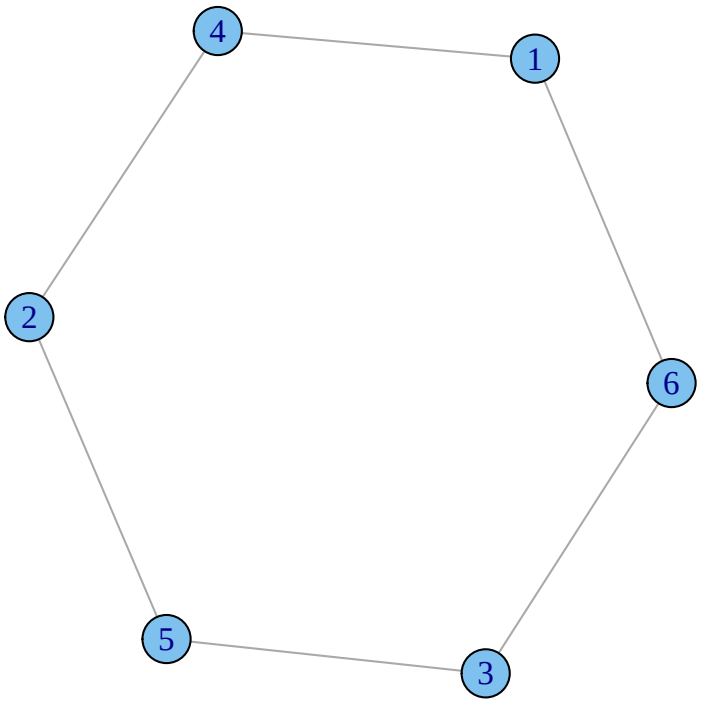
3 nodes case for stability

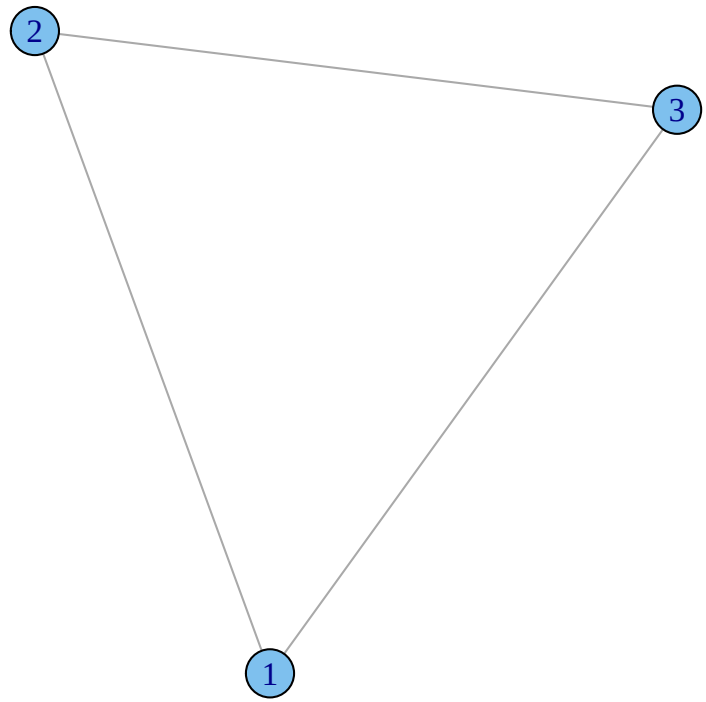
We also want the factor model also to be "stable" for estimating the precision matrix, which mean in different latent structures providing the same correlation within the observed nodes, the observed structure won't change too much. For example:

```
##      [,1] [,2] [,3]
## [1,]    1    1    0
## [2,]    1    0    1
## [3,]    0    1    1
```

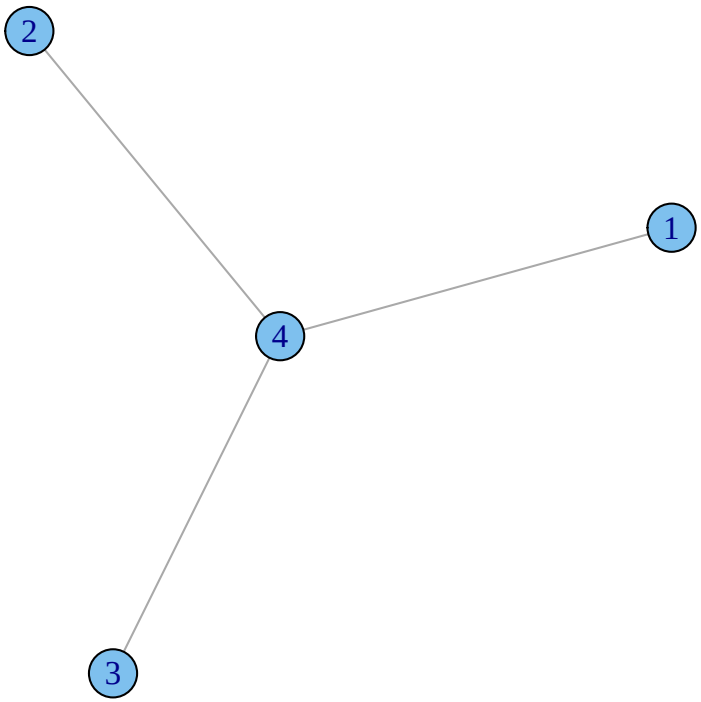
```
##      [,1] [,2] [,3]
## [1,]    1    1    1
```

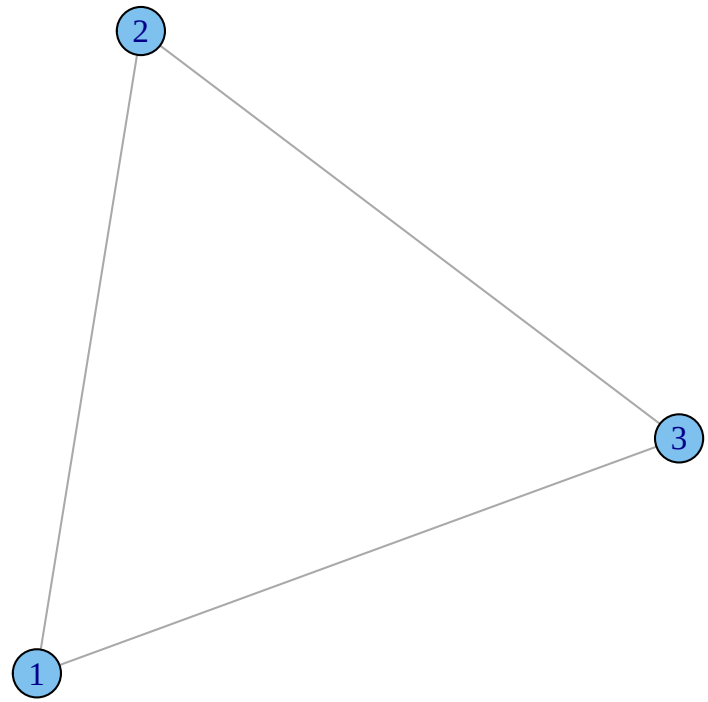
3 nodes case





3 nodes case





DSCR package (Dynamic Statistical Comparisons)

- Good artist copy, great artist steal. So, the following are from Matthew's github page for dscr

Dynamic statistical comparisons (DSC) are an attempt to change the way that researchers perform statistical comparisons of methods. When a new statistical method is developed, it is almost inevitable that it will be useful to compare it to other methods for tackling the same problem. However, the way these comparisons are currently (usually) done is suboptimal in so many ways. First, comparisons are usually performed by the research group that developed one of the methods, which almost inevitably favors that method. Furthermore, performing these kinds of comparisons is incredibly time-consuming, requiring careful familiarization with software implementing the methods, and the creation of pipelines and scripts for running and comparing them.

DSCR package (Dynamic Statistical Comparisons)

And in fast-moving fields new methods or software updates appear so frequently that comparisons are out of date before they even appear. In summary, the current system results in a large amount of wasted effort, with multiple groups performing redundant and sub-optimal comparisons. A DSC is a public Internet repository that allows methods to be compared with one another in a reproducible and easily-extensible way. . . . We envisage DSC as complementing, rather than replacing, the standard publication model: papers introducing a new method will "deposit"" the method in the DSC repo, in the same way that scientific data are deposited in data repositories

DSCR

- scenarios: generate different scenarios for comparison
- data maker: take scenario names and provide input and meta
- methods: take the input and provide output
- score: take output and provide scores as criterion for comparison
- application: addscenario(), addmethod(), addscore(), run_dsc()

DSCR-Omega

In our application:

- scenarios: identity, diagonal, toeplitz, real date.
- methods: all mentioned
- scores: likelihood, prediction, F-norm of error

Scores

All the scores we use are not depend on the true value.

Likelihood:

$$logdet(\Omega) - trace(\Sigma_n \Omega) = -E_p log(p/q) + C$$

F-norm of error:

$$\frac{\partial}{\partial \Omega} E_p ||\nabla logp - \nabla logq||_2^2 = \frac{1}{2}\Sigma_n \Omega + \frac{1}{2}\Omega \Sigma_n - I$$

prediction error

$$\sum_j (X_j - (\sum_{i \neq j} -\Omega_{ij}/\Omega_{jj} X_i))^2$$

this is for one observation, we have n observation, so just add them together.

DSCR (money in the bank)

In our application of dscr to estimate the precision matrix, all kinds methods have their own assumption, either sparse or low rank, which just result in we don't have enough sample size. When someone want to apply some methods to get the precision matrix, whether the assumption is satisfied should be taken into consideration. But we will never know the truth.

In DSCR package, we don't need to worry about that, just add the scenario one want to deal with, and the data will tell which one is perfect to the goal (based on the score chosen). If you want to develop new method and test its performance, DSCR has already "deposit" lots of previous well established methods and their performance on different scores. All you need is just add you method and choose the scenario you want to deal with.

Futher work

- Some time, if the co-regulate gene is around 80% of all the genes we consider, it is hard to tell whether it should be sparse or dense. We need a more flexible shrinkage method to estimate the factor loadings.
- In some case, only a subgroup of observation have certain structure and some structure exist in all observations, which means we also need a flexible shrinkage method on factors to capture the structure in samples.
- The solution of that mentioned above comes to the upcoming ASH-SFA.
- Not to be continued.