

モデル選択によるネットワーク構造の推定

7 章の練習問題

テッサ・F・ブランケン¹,
アデラ＝マリア・イスヴォラヌ¹, &
サシャ・エプスカンプ^{1,2}

1. アムステルダム大学心理学科
2. アムステルダム大学都市精神健康センター

本書の特設サイトから PTSD_MAGNA.csv のファイルをダウンロードし、以下のようにして R に読み込もう。

```
true_network <- as.matrix(read.csv("PTSD_MAGNA.csv", row.names = 1))
```

こうすることで、第 7 章の本文でも紹介した心的外傷後ストレス症状のメタ分析的ガウシアン・グラフィカルモデルの重み行列が手に入る。なお、以下のようにすれば、ひとつのベクトルのなかにエッジを集めておくこともできる。

```
true_edges <- true_network[lower.tri(true_network)]
```

そして、以下のコードを実行すれば、理論上は 136 本のエッジが存在し得るなかで、非ゼロのエッジが 78 本があるということがわかる。

```
# 理論上存在し得るエッジの総数：
length(true_edges)

## [1] 136

# 上記の計算結果は、以下の計算結果と一致する：
17 * 16 / 2

## [1] 136

# 非ゼロのエッジの本数：
sum(true_edges!=0)

## [1] 78
```

このネットワークを用いて、 $N = 1,000$ 名分の観測値を含んだ新しいデータセットを生成することができる。そのために、まずは以下のようにして、bootnet パッケージのシミュレータ生成関数 ggmGenerator() を使い、データをシミュレートする際に活用可能な関数を作り出す。

```
library("bootnet")
simulator <- ggmGenerator()
```

続けて、以下のようにすればデータセットを生成できる。

```
simData <- simulator(1000, true_network)
```

Question 1

シミュレーション用データセット (simData) に対して EBICglasso アルゴリズムを適用し、GGM を推定しよう。なお、重み行列を est_network という名称のオブジェクトに保存したうえで、それとは別に、エッジの重みをひとつのベクトル (重み行列の下三角行列) として est_edges という名称のオブジェクトに保存しよう。真のネットワークと推定されたネットワークの両方を、同一のレイアウトかつ maximum 引数を同一の値に設定したうえでプロットしよう。

続けて、以下のようにすれば**真陽性 (true positive)** のエッジの本数を計算することができる。真陽性のエッジとは、非ゼロと推定されたエッジの重みのうち、もともとの (真の) ネットワークでも非ゼロであるもののことを指す。

```
sum(true_edges != 0 & est_edges != 0)
```

これと同じ要領で、演算子 ‘!=’ を ‘==’ に変更すれば、エッジがゼロに等しいかどうかを検定することができる。そうした検定を実施すれば、以下の情報を得ることが可能になる。

- **偽陽性 (false positive)** のエッジの本数：非ゼロと推定されたエッジの重みのうち、もともとの (真の) ネットワークではゼロであったもの。
- **真陰性 (true negative)** のエッジの本数：ゼロと推定されたエッジの重みのうち、もともとの (真の) ネットワークでも重みがゼロであったもの。
- **偽陰性 (false negative)** のエッジの本数：ゼロと推定されたエッジの重みのうち、もともとの (真の) ネットワークでは非ゼロであったもの。

なお、「陽性 (positive)」という用語は非ゼロのエッジを表しているのであって、エッジの重みが正であるとは限らないということに注意してほしい。負の重みをもったエッジというのも非ゼロであるため、これもまた「陽性 (positive)」だということになる。

感度 (sensitivity) とは、真陽性率 (true positive rate) とも呼ばれている。以下のように、「真のモデルにおけるエッジの総数」に対する「推定において検出された真のエッジの本数」の比率を表している。

$$\text{感度} = \frac{\text{真陽性のエッジの本数}}{\text{真陽性のエッジの本数} + \text{偽陰性のエッジの本数}}$$

特異度 (specificity) とは、真陰性率 (true negative rate) とも呼ばれている。以下のように、「真のモデルには含まれなかったエッジの総数」に対する「推定において検出された、真の欠損エッジ (true missing edges) エッジの本数」の比率を表している。

$$\text{特異度} = \frac{\text{真陰性のエッジの本数}}{\text{真陰性のエッジの本数} + \text{偽陽性のエッジの本数}}$$

研究者に洞察をもたらす可能性がある指標を最後にもうひとつ挙げておくと、「真のモデルにおけるエッジの重み」と「推定されたモデルにおけるエッジの重み」の間のピアソン相関というものもある。

Exercise 2 (1 point)

シミュレーションデータから推定したネットワークモデルについて、感度と特異度に加え、エッジの重みの相関を計算しよう。

EBICglasso アルゴリズムは、第 7 章で議論した複数の推定法のうちひとつに過ぎない。他に広く用いられているアルゴリズムとしては、*ggmModSelect* アルゴリズムがある。これは、非正則化推定を用いて拡張的なモデル探索を実行し、局所最適なネットワークモデルを見つけ出すというものである。

Exercise 3

ggmModSelect を用いてネットワークを推定しよう。そして、この新しいネットワークについて上記の問題 2 をもう一度解いてみよう。感度、特異度、エッジの相関のそれぞれについて、どちらのアルゴリズムがより良いパフォーマンスを発揮するだろうか。

上記のプロセスは、*bootnet* パッケージの `netSimulator` 関数を用いれば何度も自動で繰り返すことができる。この関数は、以下のように用いることができる。

```
# シミュレータを実行する：
simulation <- netSimulator(

  input = ..., # データの生成元となるネットワーク（上述したところの「真のネットワーク」）

  nCases = ..., # シミュレーションのなかで用いるケースの数を表したベクトル

  nReps = ..., # 各条件を何回繰り返すか

  dataGenerator = ..., # データ生成関数。たとえば、ggmGenerator() 関数を実行して得られたもの
  #（上述したところの「シミュレータ」）

  nCores = ..., # コンピュータのスレッド数。スレッド数は、分析に用いるコンピュータによりけりである
  #（多くのコンピュータでは、8以上となっている）
  ... # estimateNetwork 関数に対する引数を必要な数だけ指定。たとえば、引数 'default' など
)

# シミュレーション結果の描画：
plot(simulation)
```

Exercise 4

EBICglasso アルゴリズムを用い、「生成構造として `true_network` を用いた場合の *EBICglasso* のパフォーマンス」についてのシミュレーション研究を実施しよう。その際に、サンプルサイズを 300, 1,000, 5,000 と変えていき、各条件を 100 回ずつ反復すること。そして、シミュレーションの結果をプロットして解釈しよう。さらに、あなたが高性能のコンピュータを持っている場合には、*ggmModSelect* アルゴリズムを用いてこのシミュレーション研究を再度実施してみよう（このアルゴリズムを用いた場合には、所要時間がずっと長くなるだろう）。 ■