

Assignment 2 - Network Analysis 2022

Kyuri Park, 12183881

November 23, 2022

Contents

Conceptual Questions	2
Question 1 (3 points)	2
Partial Correlation Networks	2
Question 2 (2 points)	3
Question 3 (1 point)	3
Ising Model	5
Question 4 (2 points)	5
Network Inference	6
Question 5 (2 points)	6
Accuracy and stability (conceptual)	8
Question 6 (3 points)	8
Question 7 (2 points)	9
Question 8 (3 points)	14
Gaussian Graphical Models	15
Question 9 (3 points)	16
BONUS: Question 10 (1 point)	17

Conceptual Questions

Question 1 (3 points)

Are the following statements true or false (0.5 point per statement)? Explain why.

1. Suppose that nodes n_1 and n_2 in a GGM are not connected (edge weight = 0). We can thus infer that these nodes are conditionally independent.
TRUE: Yes, if two nodes are not connected, they are independent *conditional on all other nodes* in the network.
2. Suppose that model A is more complex than model B, and model B is nested in model A. If model A and model B fit the data about equally well, we would prefer model A.
FALSE: No, if we have two models fit equally well (performs equally well), then we prefer the simpler one (i.e., Occam's razor principle). So we would prefer model B not model A in this case.
3. Thresholding and pruning at the same α level should lead to the same estimated network parameters.
FALSE: No, they don't lead to the same estimated parameters. Because in pruning, we remove the edges that do not meet the criterion and *re-estimate* the model so that the parameters are estimated based on the final (pruned) model. Contrary to this, in thresholding, the edges that do not meet the criterion are set to zero so that they are not visualized in the network, yet the other non-zero edge weights are *not re-estimated*. Therefore, the parameter estimates would likely to differ.
4. Regularization penalizes the complexity of the parameters in the model.
TRUE: Yes, the statement is true. With regularization, parameters are estimated by optimizing *penalized* likelihood function, which is based on the complexity of the model parameters.
5. A stepwise model search algorithm is guaranteed to find a global optimum.
FALSE: No, It is guaranteed to find a *local* optimum but not a *global* optimum (i.e., we do not know if it is the *best* fitting model).
6. The null hypothesis when comparing networks via the NCT is that networks are different.
FALSE: No, the null hypothesis when comparing networks via the NCT is that the (true data generating) network structures are the same.

Partial Correlation Networks

```
# download the file NA_2020_data.csv from https://osf.io/45n6d/
# load the data into R
data <- read.csv("data/NA_2020_data.csv")

# select a subset of variables
include <- c(
  "Q10", # I try to keep a regular sleep pattern
  "Q13", # I am worried about my current sleeping behavior
  "Q14", # My sleep interferes with my daily functioning
  "Q68", # I am happy with my physical health.
  "Q70", # I feel optimistic about the future.
  "Q75", # I am very happy
  "Q77", # I often feel alone
  "Q80" # I am happy with my love life
```

```

)
# subset the data
data_subset <- data[,include]

# rename the variables
names(data_subset) <- c("regular_sleep",
"worried_sleep",
"sleep_interfere",
"happy_health",
"optimistic_future",
"very_happy",
"feel_alone",
"happy_love_life")

```

Question 2 (2 points)

Compute both the marginal correlation (using `cor()`) and the partial correlation (using `partial.r()`), conditioning on all remaining variables between the nodes `feel alone` and `optimistic future`. Interpret your results by comparing the marginal and partial correlation you computed.

As shown below, the marginal correlation between `feel alone` and `optimistic future` is around -0.26 , while the partial correlation is almost *zero* (-0.02). This indicates that the (linear) dependencies between `feel alone` and `optimistic future` are mainly due to the other variables. That is, once we remove (account for) the influence of the other variables, there is almost no dependencies left between `feel alone` and `optimistic future`. In other words, the dependencies between `feel alone` and `optimistic future` can be mostly explained by the other variables.

```

# marginal correlation
data_subset %>%
  # filter NAs
  na.omit() %>%
  select(feel_alone, optimistic_future) %>%
  cor() %>%
  round(2)

# partial correlation
partial.r(data = data_subset, x = c(5,7), y = c(1,2,3,4,6,8))

```

	feel_alone	optimistic_future
feel_alone	1.00	-0.26
optimistic_future	-0.26	1.00

partial correlations

	optimistic_future	feel_alone
optimistic_future	1.00	-0.02
feel_alone	-0.02	1.00

Question 3 (1 point)

```
# We can compute the sample variance-covariance matrix as follows:
covMat <- cov(data_subset, use = "pairwise.complete.obs")
round(covMat,2)
```

```
# Following, we invert a variance-covariance matrix using solve()
Kappa <- solve(covMat)
round(Kappa, 2)
```

This precision matrix K can then be standardized to obtain the partial correlation coefficient matrix P (the partial correlation between variable i and j after conditioning on all other variables in the data set):

$$p_{i,j} = \begin{cases} -\frac{K_{i,j}}{\sqrt{K_{i,i}}\sqrt{K_{j,j}}}, i \neq j \\ 1, i = j \end{cases} \quad (1)$$

Using the formula above, compute the partial correlation between `feel_alone` and `optimistic_future` and compare your result to the partial correlation obtained using `partial.r()`.

As shown below, the computed partial correlation between `feel_alone` and `optimistic_future` using the formula above leads to the same result (−0.02) as using `partial.r()` function.

```
# calculate the partial correlation matrix
par.corr <- matrix(nrow=nrow(Kappa), ncol=ncol(Kappa))
for(k in 1:nrow(par.corr)) {
  for(j in 1:ncol(par.corr)) {
    if(k == j){
      par.corr[j, k] <- 1
    } else {
      par.corr[j, k] <- -Kappa[j,k]/sqrt(Kappa[j,j]*Kappa[k,k])}
  }
}
# specify the dimension names
colnames(par.corr) <- rownames(par.corr) <- colnames(Kappa)
# show the partial correlation matrix
round(par.corr,2)
```

	regular_sleep	worried_sleep	sleep_interfere	happy_health
regular_sleep	1.00	-0.07	-0.01	0.13
worried_sleep	-0.07	1.00	0.65	-0.04
sleep_interfere	-0.01	0.65	1.00	-0.08
happy_health	0.13	-0.04	-0.08	1.00
optimistic_future	0.03	-0.08	0.03	0.19
very_happy	0.02	0.01	-0.05	0.26
feel_alone	0.12	0.12	0.10	0.11
happy_love_life	0.08	-0.04	0.07	0.01

	optimistic_future	very_happy	feel_alone	happy_love_life
regular_sleep	0.03	0.02	0.12	0.08
worried_sleep	-0.08	0.01	0.12	-0.04
sleep_interfere	0.03	-0.05	0.10	0.07
happy_health	0.19	0.26	0.11	0.01
optimistic_future	1.00	0.45	-0.02	0.00

very_happy	0.45	1.00	-0.27	0.24
feel_alone	-0.02	-0.27	1.00	-0.14
happy_love_life	0.00	0.24	-0.14	1.00

```
## extract partial corr. between optimistic_future and feel_alone
```

```
par.corr[5,7]
```

```
[1] -0.02208709
```

Ising Model

```
trueNetwork <- read.csv('http://sachaepskamp.com/files/weiadj.csv')[,-1]
trueNetwork <- as.matrix(trueNetwork)
Symptoms <- rownames(trueNetwork) <- colnames(trueNetwork)
Thresholds <- read.csv('http://sachaepskamp.com/files/thr.csv')[,-1]
graph <- qgraph(trueNetwork, labels = Symptoms, layout='spring', theme = "colorblind")
Layout <- graph$layout # to save the layout for visual comparison
```

```
sampleSize <- 1000
set.seed(2022)
newData <- IsingSampler(sampleSize, graph = trueNetwork, thresholds = Thresholds)
```

Question 4 (2 points)

Use the data we just sampled to estimate an Ising model, following the same procedures as the authors took in their original paper. Use the bootnet package and specify the right “default”. In estimating the network, make sure that you take notice of the default values that are set. Plot the resulting network and fix the layout of your estimated network to be the same as in the original network model. Visually compare the two networks to verify whether the networks match one another.

As shown below in Figure 1, the estimated network on the simulated data (right) mostly matches to the original network (left), as majority of the strong edges present in the original network also are observed in the estimated network. Yet, the estimated Ising model is much more sparse than the original network and that is partly due to the regularization that is applied (LASSO regularization based on EBIC with $\gamma = 0.25$ by default). In addition, there are a couple of edges that are newly introduced or the sign is flipped in the estimated network (i.e., happy-sad, sad-sleep), which could be again partly due to regularization process (e.g., after some of the edges are set to zero, the rest of non-zero edge weights are re-estimated and there the differences may come about). **ADD more explanation , book?**

```
## estimate network
est_net <- estimateNetwork(newData, default="IsingFit")
est_net$labels <- Symptoms
est_graph <- plot(est_net, labels=Symptoms, layout=Layout)

## compare networks visually
par(mar=c(0.1, 1, 1, 1))
layout(t(1:2))
```

```

plot(graph)
title("Original Network")
plot(est_graph)
title("Estimated Network")

```

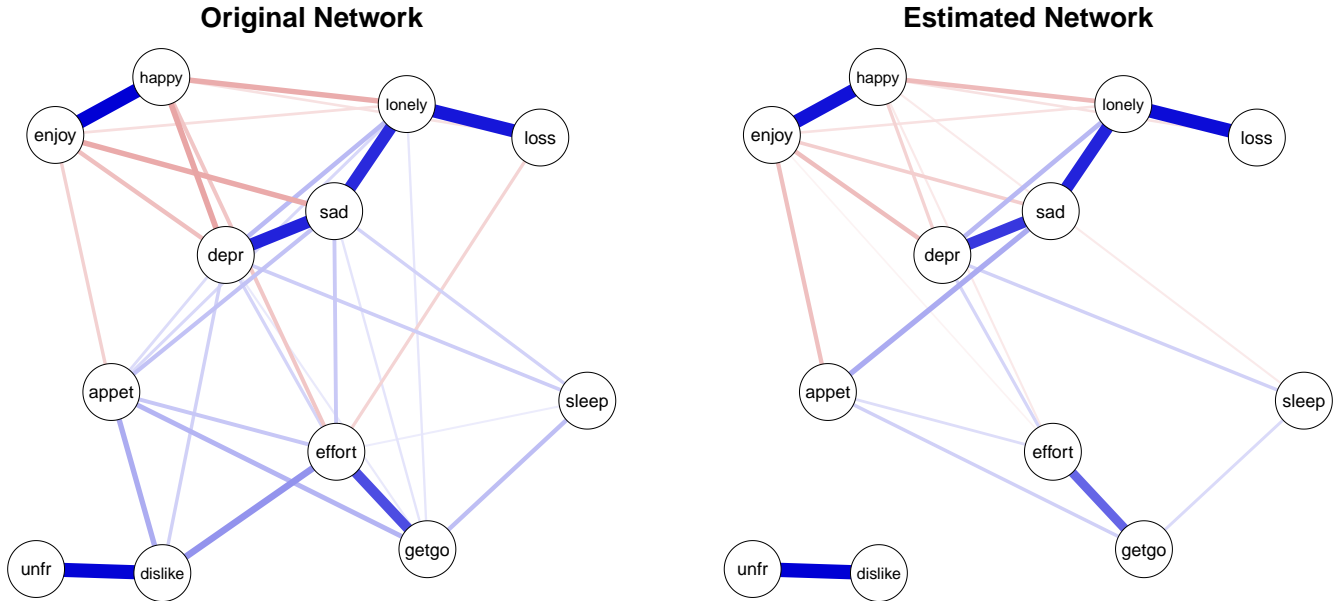


Figure 1: Comparison of Network Models

Network Inference

Question 5 (2 points)

Choose at least one metric to compute on the network you just estimated. Explain how to interpret the metric conceptually, why you choose this method for this estimated network, and interpret the results.

Out of various centrality indices, I wanted to focus on centrality metrics for *direct* connectivity, that is taking into account only the immediate connections of a node. My interest lies in investigating the importance of nodes based on their direct connections (i.e., most (strongly) connected nodes). For that, I choose to compute the *node degree* and *node strength*. *Degree* is one of the most general measures for direct centrality, but then considering that the estimated network is weighted network, it makes sense to also look at the weighted version of *degree*, which is *node strength*.

- **Node degree** (summation of the number of connected edges to a node):

$$Degree(i) = \sum_{j=1}^n a_{ij}$$

,where a_{ij} represents the element at row i and column j of the adjacency matrix A .

- **Node strength** (summation of the absolute edge weights connected to a node).

$$Strength(i) = \sum_{j=1}^n |w_{ij}|$$

,where w_{ij} represents the element at row i and column j of the weight matrix W .

If a node ranks high on these metrics of direct centrality, it indicates that it has many direct (strong) relations to other nodes in the network. Accordingly, it implies that the node is relatively more central than the other nodes, which could help inferring the importance of the node in terms of influencing the state of other nodes.

The resulting centrality metrics on the estimated network is shown in Figure 2. Based on the pure *degree*, it can be seen that **happy**, **enjoy**, and **depre** have the highest number of edges (6 edges in total). However, when taking into account for the weights of edges (based on *strength*), **lonely**, **sad**, and **depre** are the most central nodes in this corresponding order. Note that, however, without the knowledge on the accuracy and stability of these centrality estimates, I cannot surely conclude that for example, **lonely** is the most central node in this estimated network. We will actually check the stability of centrality metric in the following section (Question 8).

```
## compute the degree
degree <- apply(est_net$graph, 2, function(x) sum(x != 0)) %>%
  as.data.frame() %>%
  rename("degree" = ".") %>%
  mutate(vars = est_net$labels, title= "degree") %>%
  ggplot(aes(x=reorder(vars, degree), y = degree)) +
  geom_col(width=.07) + geom_point()+
  coord_flip() + theme_bw() +
  labs(x="", y="") + facet_grid(. ~title)

## compute strength
strength <- centralityPlot(est_graph, include = "Strength", scale = "raw", orderBy = "Strength")
# combine two plots
ggpubr::ggarrange(degree, strength, ncol=2)
```

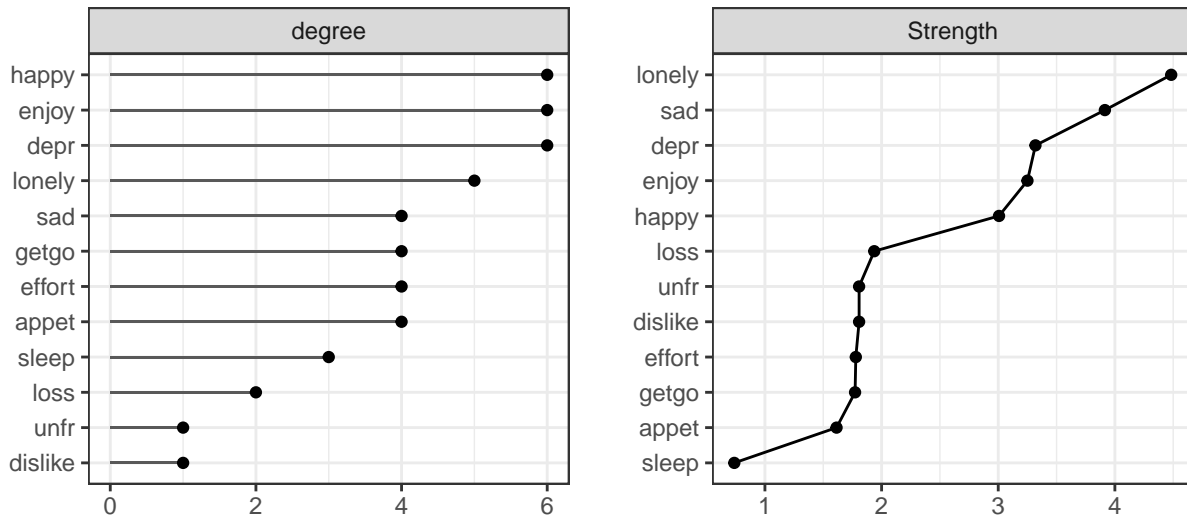


Figure 2: Centrality metrics

Accuracy and stability (conceptual)

Question 6 (3 points)

```
# estimated network weight matrix
estEdges <- est_net$graph
dimnames(estEdges) <- list(est_net$labels, est_net$labels)
# extract lower triangles
trueEdges <- trueNetwork[lower.tri(trueNetwork,diag=FALSE)]
estEdges <- estEdges[lower.tri(estEdges, diag=FALSE)]

## true positive (TP)
tp <- sum(trueEdges != 0 & estEdges != 0)
## true positive (TN)
tn <- sum(trueEdges == 0 & estEdges == 0)
## false positive (FP)
fp <- sum(trueEdges == 0 & estEdges != 0)
## false negative (FN)
fn <- sum(trueEdges != 0 & estEdges == 0)
```

6-1. Compute the sensitivity, the specificity, and the edge weight correlation for the network model you estimated from the simulated data. Which conclusions can you draw from your calculations? (2 points)

As shown in Table 1, the *specificity* is relatively high around 0.94, while the *sensitivity* is low as 0.64. This tells us that the estimated network contains very few *false positive* edges and quite many *false negative* edges. It perhaps indicates that the regularization might have been too conservative such that it has put too many edges to zero. In addition, the fairly high *edge weight correlation* of 0.94 implies that the estimated edge weights overall align quite well with the true edge weights. The low *sensitivity* and high *edge weight correlation* together reflect the fact that most of the weak edges are set to zero, yet the important connections are mostly correctly identified.

```
## sensitivity
sensitivity <- tp / (tp + fn)
## specificity
specificity <- tn / (tn + fp)
## edge weight correlation
edge_corr <- cor(trueEdges, estEdges)
```

Table 1: Evaluation criteria

Sensitivity	Specificity	Edge.weight.correlation
0.636	0.939	0.944

6-2. As we have also discussed during the lecture, there is a trade-off between the sensitivity, specificity, and the estimation parameters that we have set. What do you expect to happen to the sensitivity and specificity when you would lower the tuning hyper parameter? (1 points)

If we lower the tuning parameter (γ), which controls the strength of extra penalty, it would lead to a greater number of estimated edges (i.e., less edges will be put to zero). Accordingly, the sensitivity will become higher, but at the

cost of specificity. Therefore, lowering gamma (γ) would increase sensitivity and decrease specificity.

Question 7 (2 points)

Perform a non-parametric bootstrap to investigate the accuracy of the Ising model that we estimated on the $N = 1000$ simulated data set. Plot the bootstrapped confidence intervals. What do you notice with the confidence intervals? Investigate different options (e.g., `split0`).

The red dot indicates the sample values, gray dots are bootstrapped mean values, and the gray lines are the 95% bootstrapped CIs. Each horizontal line represents one edge of the network, ordered from the edge with the highest edge-weight to the edge with the lowest edge-weight.

Figure 3 shows the resulting bootstrapped CIs (BCIs) around the estimated edge-weights of Ising model. One thing to note is that as these BCIs are obtained from the regularized edge weights, the interpretation of BCIs is limited to only show the variability in parameter estimates (not the typical interpretation of 95% analytic CIs). Overall most of the CIs are not too wide, indicating that the stability of edge-weights estimates are decent. However, there is a sign of instability in one of the edges (the one on the top), which shows a much larger BCI. It turns out to be the edge between V6 and V10 and this enormous BCI could be due to the violation of assumptions. The generally not too narrow BCIs imply that we need to be cautious when we interpret the order of the edges in our estimated network. There is a quite a big chunk of edges that are estimated to be zero, but again, since the edges are regularized, we need to keep in mind that all edge estimates are biased towards zero (observing that an edge is not set to zero already indicates that the edge is sufficiently strong to be included in the model).

Figure 4 shows the *split-0* BCIs, where the BCIs are drawn only using the bootstrapped samples of non-zero estimates and coupled with a proportion of times that the parameter was put to zero. Accordingly, it shows how often a parameter was included and the obtained values of parameters separately such that the BCIs are faded when the edges are not often included (also shown in proportion). As expected, if the edges that are consistently not included (faded ones) the *split-0* BCIs tend to be wider, as it is based on only the few times the parameters included as non-zero. The edges that are consistently included have smaller BCIs and compared to the BCIs shown in Figure 3, the consistently-included edges have smaller BCIs again as they are based only on the times they were not zero.

Figure 5 and Figure 6 shows the bootstrapped difference tests ($\alpha = 0.05$) between edge-weights that were non-zero in the estimated network and node strength of the symptoms, respectively. Gray boxes indicate nodes or edges that do not differ significantly from one-another and black boxes represent nodes or edges that do differ significantly from one-another. Colored boxes in the edge-weight plot correspond to the color of the edge in the estimated network (see Figure 1), and white boxes in the centrality plot show the value of node strength.

From Figure 5, it can be seen that quite a portion of edges (almost half) are shown to not significantly differ from one another. And Figure 6 shows that again about a half of node strengths cannot be shown to significantly differ from each other. Note that *no correction* for multiple testing was applied in both plots.

- **INTERPRETATION:** overlapping sample mean and bootstrapped mean are overlapping \rightarrow the sample mean are probably accurate?

```
# run bootstrapping
boots_ising <- bootnet(est_net, nBoots = 3000, nCores = 8)
# save results
save(est_net, boots_ising, file="data/Ising_Bootstrap.RData")
```

```
# load bootstrapping result
load("data/Ising_Bootstrap.RData")
## plot the bootstrapped confidence interval
plot(boots_ising, order = "sample", plot = "interval")
```

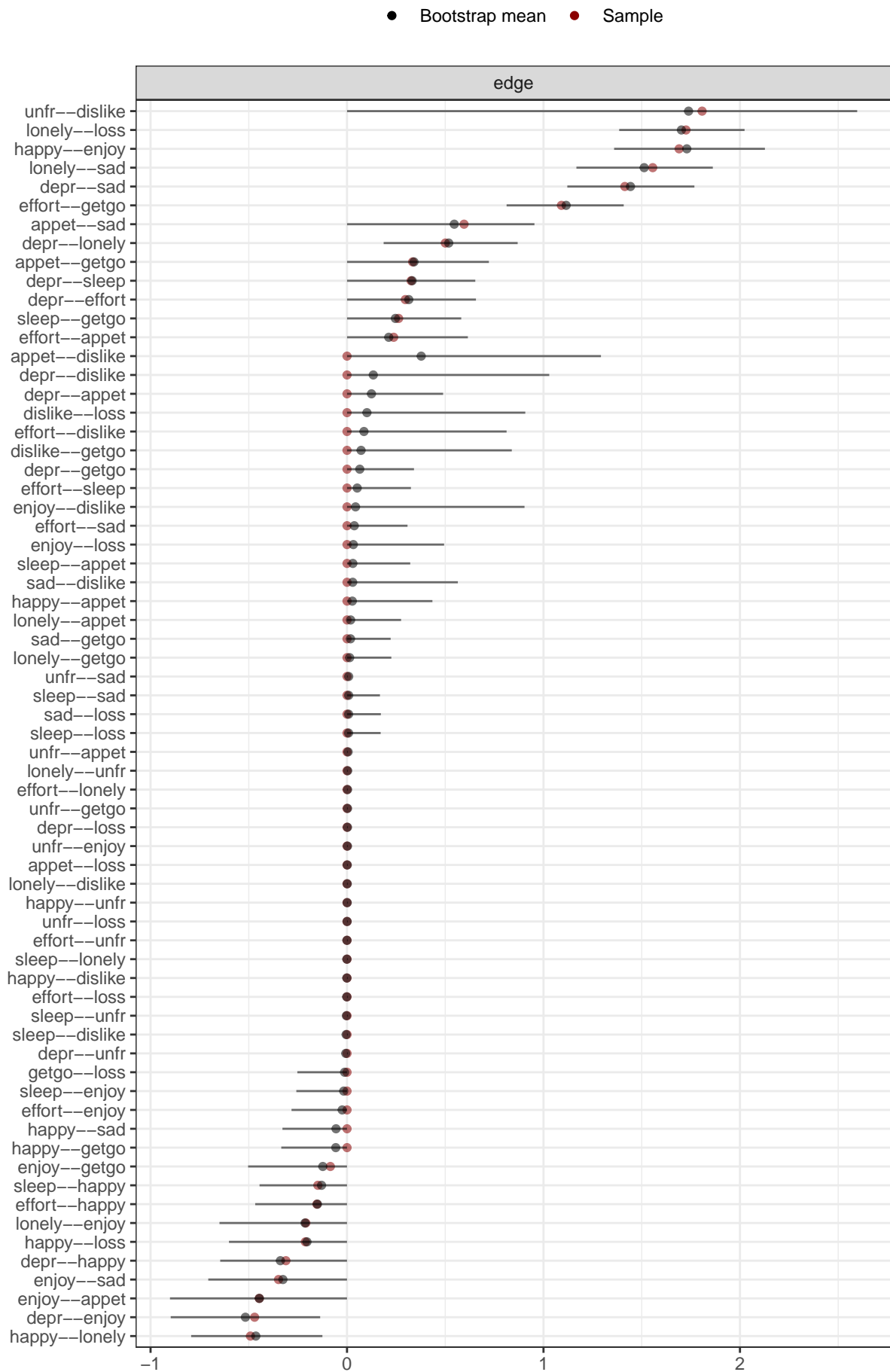


Figure 3: Bootstrapped CIs

```
## plot split-0 BCIs
```

```
plot(boots_ising, order = "sample", plot = "interval", split0 = TRUE)
```

```
## plot significant differences (alpha = 0.05) of edges
```

```
plot(boots_ising, "edge", plot = "difference", onlyNonZero = TRUE, order = "sample") + My_Theme
```

```
## plot significant differences (alpha = 0.05) of node strength
```

```
plot(boots_ising, statistics = "strength", plot = "difference") + My_Theme
```

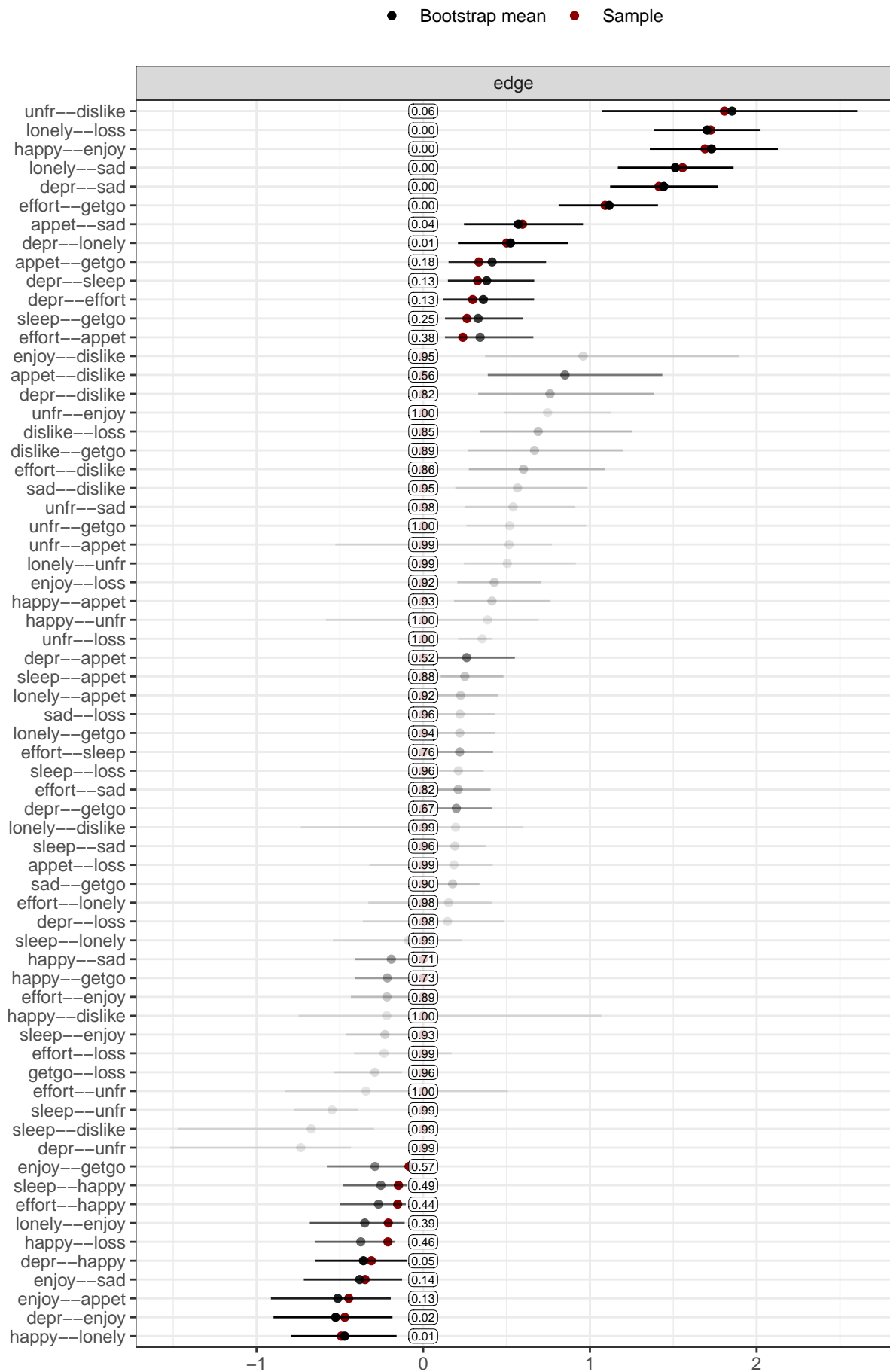


Figure 4: split0 Bootstrapped CIs

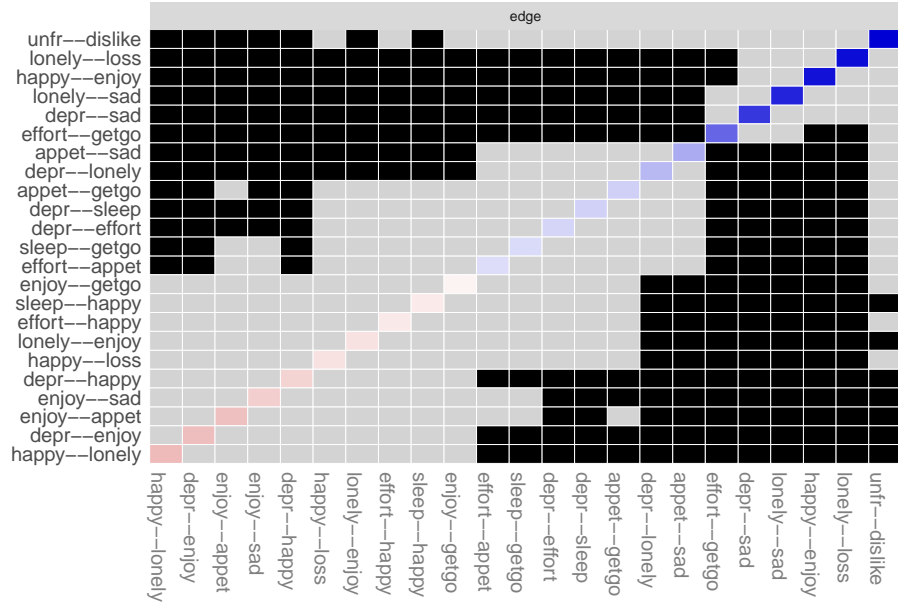


Figure 5: Bootstrapped difference tests between edge weights

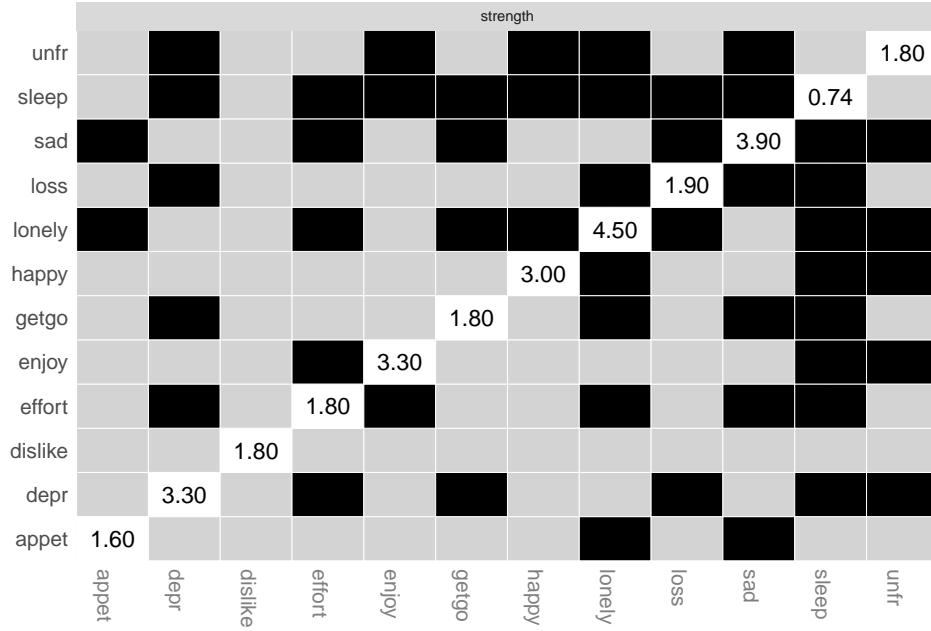


Figure 6: Bootstrapped difference tests between node strength

Question 8 (3 points)

Perform a case-dropping bootstrap to investigate the stability of the centrality metric you choose in Question 5. Generate the stability plot as discussed in the lecture and report the CS-coefficient. Interpret the results of the case-dropping bootstrap.

Figure 7 (a) shows that the correlation of node strength over various size of samples. The *line* indicates the mean strength of correlation, and the *area* indicates the range from 2.5th quantile to the 97.5th quantile. As you can see, the mean strength drops slightly but mainly remain stable across different sampled cases. It means that the order of node strength remains more or less the same after re-estimating the network with smaller subsetting cases. The area on the other hand becomes obviously wider as the sampled cases become smaller, which is not surprising as there are higher variabilities with the smaller samples. Figure 7 (b) shows the node strength of individual nodes across case-dropping bootstrapped samples. It shows that the strength steadily decreases in most of the nodes as sampled cases become smaller, but the changes are not dramatic but rather stable. All in all, we could conclude that the stability of *node strength* is fine based on the stability plot (Figure 7).

The CS-coefficient of node strength is 0.672, which is above the recommended cut-off 0.5. Given this, it could be concluded that we can substantively interpret the order of node strength. However, note that the cut-off value of 0.5 is rather chosen arbitrarily (just based on one simulation study: Epskamp et al., 2018), hence we still need to be careful when we make inference on node strength.

```
## perform case-dropping bootstrapping
boots_casedrop <- bootnet(est_net, nBoots = 3000, nCores = 8, type = "case", statistics = "strength")
# save results
save(est_net, boots_casedrop, file="data/Casedrop_Bootstrap.RData")

# load case-dropping bootstrapping result
load("data/Casedrop_Bootstrap.RData")
## generate the stability plot
str_stab <- plot(boots_casedrop) + labs(title="(a) Overall Node Strength") +
  theme(legend.position = "none", plot.title = element_text(hjust = 0.5)) + my_Theme
## node strength estimates of individual nodes
str_pernode <- plot(boots_casedrop, perNode = TRUE) + labs(title="(b) Strength of Individual Node") +
  theme(legend.position = "none", plot.title = element_text(hjust = 0.5)) + my_Theme
# combine plots
ggpubr::ggarrange(str_stab, str_pernode, ncol=2)

## Centrality stability coefficient
corStability(boots_casedrop)
```

=== Correlation Stability Analysis ===

Sampling levels tested:

	nPerson	Drop%	n
1	250	75.0	278
2	328	67.2	294
3	406	59.4	298
4	483	51.7	302
5	561	43.9	322
6	639	36.1	301
7	717	28.3	286

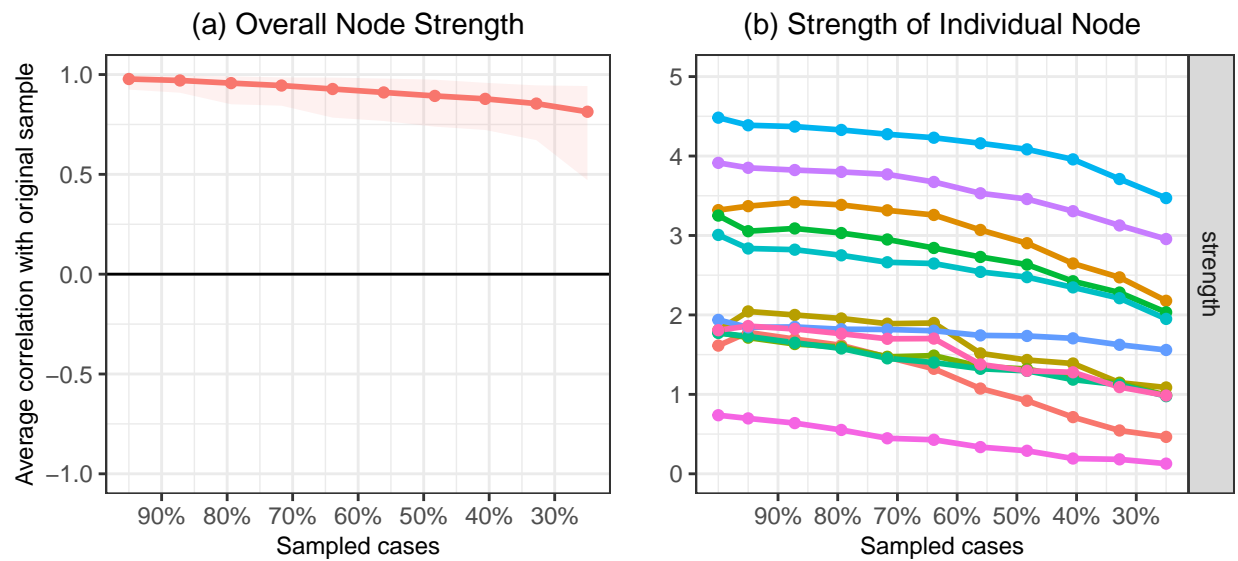


Figure 7: Node strength between case-dropping bootstrapped samples

8	794	20.6	309
9	872	12.8	288
10	950	5.0	322

Maximum drop proportions to retain correlation of 0.7 in at least 95% of the samples:

intercept: 0.75 (CS-coefficient is highest level tested)

- For more accuracy, run `bootnet(..., caseMin = 0.672, caseMax = 1)`

strength: 0.672

- For more accuracy, run `bootnet(..., caseMin = 0.594, caseMax = 0.75)`

Accuracy can also be increased by increasing both 'nBoots' and 'caseN'.

```
## run case-drop bootstrapping
boots_ising2 <- bootnet(est_net, nBoots = 3000, nCores = 8, type = "case")

# save results
save(est_net, boots_ising2, file="data/Ising_CasedropBootstrap.RData")

# load bootstrapping result
load("data/Ising_CasedropBootstrap.RData")

## plot the bootstrapped confidence interval
#plot(boots_ising2, order = "sample", plot = "interval")
```

Gaussian Graphical Models

```
# download the network.csv file from https://osf.io/vufj4/ & load into R
data <- read.csv("data/network.csv")
```

```

# We will only look at depression symptoms
data_dep <- data %>% select(D.Anhedonia:D.Suicide)
# Rename:
names(data_dep) <- gsub("D\\.", "", names(data_dep))
#Show the data:
head(data_dep)

```

Question 9 (3 points)

With the **bootnet** package estimate a GGM using a (1) a pruning method, (2) a regularization method, and (3) a stepwise model search method. Plot all networks. Are there strong differences between the estimated network models? Please explain why (or why not).

The layout is set the same across the different networks using `AverageLayout` and the `maximum` argument is used to make the edges in the networks comparable.

Overall, there seems to be not much of strong differences in the structure of estimated network models. It could be due to the sufficient sample size (in this case $n = 625$), which make the different methods to converge.

The regularization method presents relatively more edges than the other two methods, which is as expected, since it tends to be less conservative than the other non-regularized methods (Isvoranu & Epskamp, 2021).

When you look at overall structure, there is not really difference

No specific minimum/maximum/cut values have been used. Edge weights in the separated network ranged from 0.001 (mood – unfriendly) to 0.320 (lonely – separation). Edge weights in the widowed network ranged from 0.002 (sad – getgo) to 0.300 (lonely – widowed).

page.125 book (#7) : high sample, the three methods give the same

```

## (1) pruning
# use false discovery rate (FDR) adjustment
# alpha level = 0.05
# recursive = TRUE (repeat the pruning process)
prunde_mod <- ggm(data_dep) %>% runmodel %>%
  prune(alpha=0.05, adjust = "fdr", recursive = TRUE)
pruned_net <- getmatrix(prunde_mod, "omega")

## (2) regularization
# tuning = 0.5 by default
reg_net <- estimateNetwork(data_dep, default="EBICglasso")

## (3) stepwise model search
sw_net <- estimateNetwork(data_dep, default="ggmModSelect", stepwise=TRUE)

## plotting the graphs
layout(t(1:3))
# get the average layout
avelayout <- averageLayout(pruned_net, reg_net, sw_net)
# set the same max value
Max <- max(abs(c(getWmat(pruned_net), getWmat(reg_net), getWmat(sw_net))))

```



```
# get labels
labelss <- c("Anhd", "Sad", "Sleep", "Energy", "Appe", "Guilt", "Concen", "Motor", "Suicide")
pruned_graph <- qgraph(pruned_net, layout = avelayout, theme = "colorblind",
  title = "Pruning (sig = 0.05)", labels=labelss, vsize=15, title.cex = 1.5,
  maximum=Max, details=T)
reg_graph <- plot(reg_net, title = "Regularization", layout=avelayout, vsize=15, labels=labelss,
  title.cex = 1.5, maximum=Max, details=T)
sw_graph <- plot(sw_net, layout = avelayout, title = "Step-wise", vsize=15, labels=labelss,
  title.cex = 1.5, maximum=Max, details=T)
```

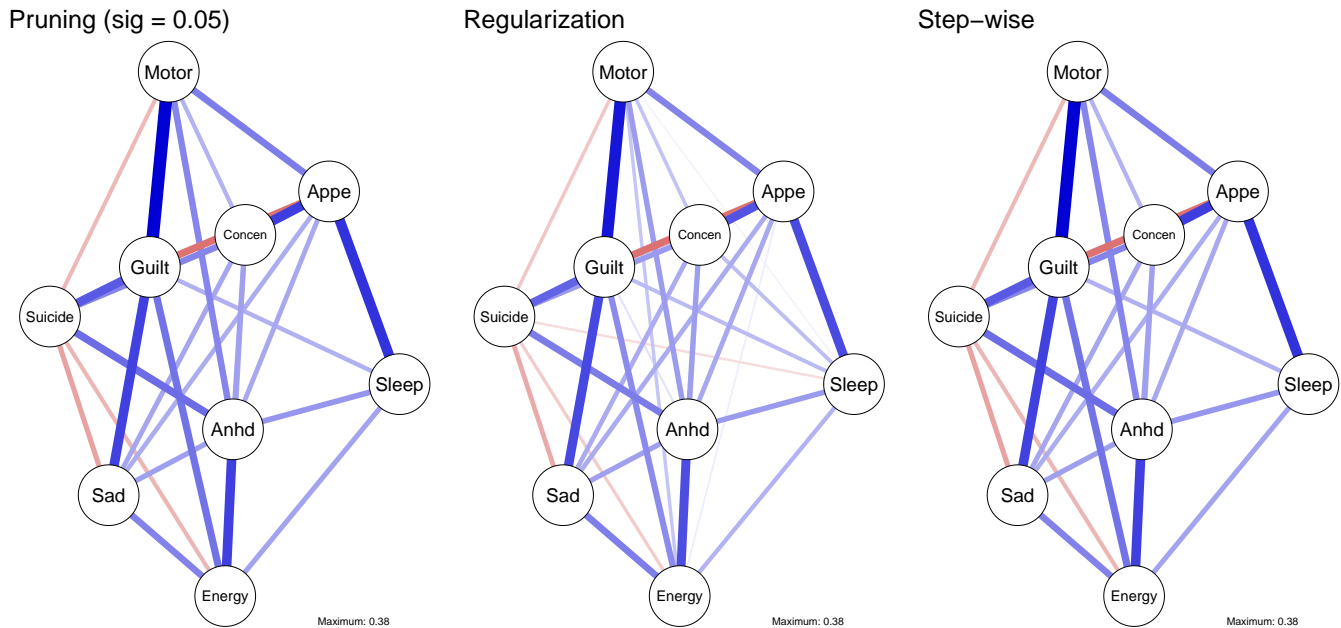


Figure 8: Comparison of model search algorithms

BONUS: Question 10 (1 point)

(a) Estimate separate networks for both sexes with your preferred algorithm from above.

```
## data of male groups
data_males <- data %>%
  select(sex, D.Anhedonia:D.Suicide) %>%
  filter(sex == 1) %>%
  rename_all(~stringr::str_remove_all(., "[.D]")) %>%
  select(-sex)

## data of female groups
data_females <- data %>%
  select(sex, D.Anhedonia:D.Suicide) %>%
  filter(sex == 2) %>%
  rename_all(~stringr::str_remove_all(., "[.D]")) %>%
  select(-sex)

## estimate network with EBICglasso regularization
network_males <- estimateNetwork(data_males,
```

```

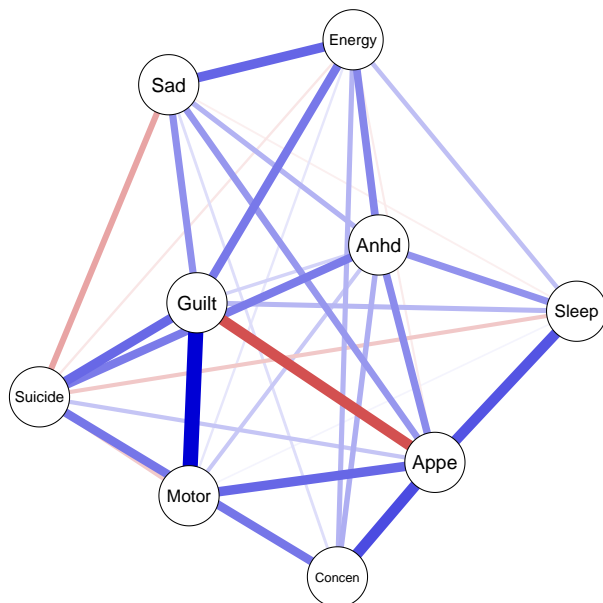
default = "EBICglasso",
corMethod = "spearman")

network_females <- estimateNetwork(data_females,
default = "EBICglasso",
corMethod = "spearman")

## plot the network for both groups
L <- averageLayout(network_males, network_females)
Max <- max(abs(c(getWmat(network_males), getWmat(network_females))))
layout(t(1:2))
plot(network_males, layout = L, title = "Males", maximum = Max, labels=labelss)
plot(network_females, layout = L, title = "Females", maximum = Max, labels=labelss)

```

Males



Females

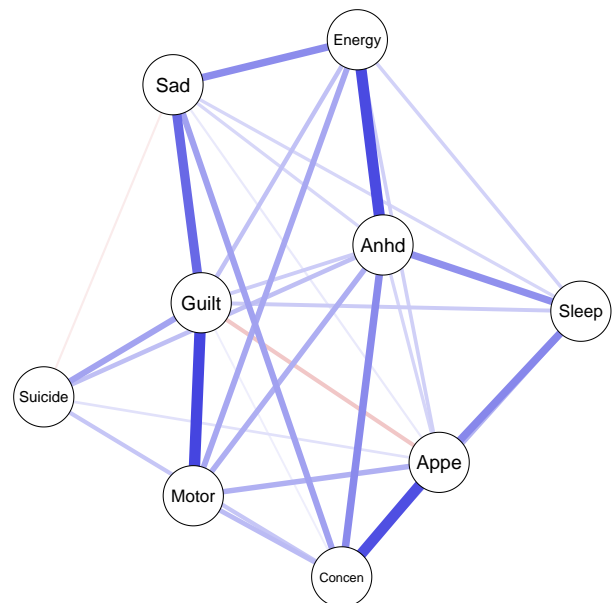


Figure 9: Comparison of network models between gender

b) Use the `NetworkComparisonTest` (NCT) to perform compare the global characteristics of both networks by focusing on (1) the global strength test evaluating the global strength invariance, and (2) the omnibus test evaluating the network structure invariance. What do you conclude about differences between these two samples?

`NetworkComparisonTest` (NCT) is a permutation based hypothesis test, suited for gaussian and binary data, assesses the difference between two networks based on several invariance measures (network structure invariance, global strength invariance, edge invariance). Network structures are estimated with *l1-regularized partial correlations* (gaussian data) or with *l1-regularized logistic regression* (eLasso, binary data).

Conclusion:

- (1) The global strength test result is shown below in the *summary* as well as in the left graph in Figure 10. The null hypothesis states that the overall level of connectivity is the same across sub-populations. As the resulting p-value is 0.0538 ($p > 0.05$), we don't reject the null hypothesis and conclude that the global strength does not differ significantly in the two sample networks.

- (2) Regarding the network structure, again we can check the *summary* and the right graph in Figure 10. The null hypothesis states that all edges are equal. Given that the p-value is 0.2762 ($p > 0.05$), again we don't reject the null hypothesis and conclude that the network structures are not significantly different.

```
## network comparison test
# set the seed for reproducibility
set.seed(123)
# number of iterations: 5000 (minimum = 1000 for reliable result)
# Testing the two aspects that are validated (network invariance, global strength)
NCT <- NCT(network_males, network_females, it=10000)
# save the nct results
save(NCT, file="data/nct.RData")
```

```
# load nct results
load("data/nct.RData")
# summary of nct results
summary(NCT)
```

```
NETWORK INVARIANCE TEST
  Test statistic M:  0.1960128
p-value 0.2762
```

```
GLOBAL STRENGTH INVARIANCE TEST
  Global strength per group:  5.083901 3.903436
Test statistic S:  1.180466
p-value 0.0538
```

```
EDGE INVARIANCE TEST
  Edges tested:
Test statistic E:
p-value
```

```
CENTRALITY INVARIANCE TEST
  Nodes tested:
Centralities tested:
Test statistic C:
p-value
```

```
## Plotting of NCT results
layout(t(1:2))
# global strength invariance test
plot(NCT, what="strength")
# network structure invariance test
plot(NCT, what="network")
```

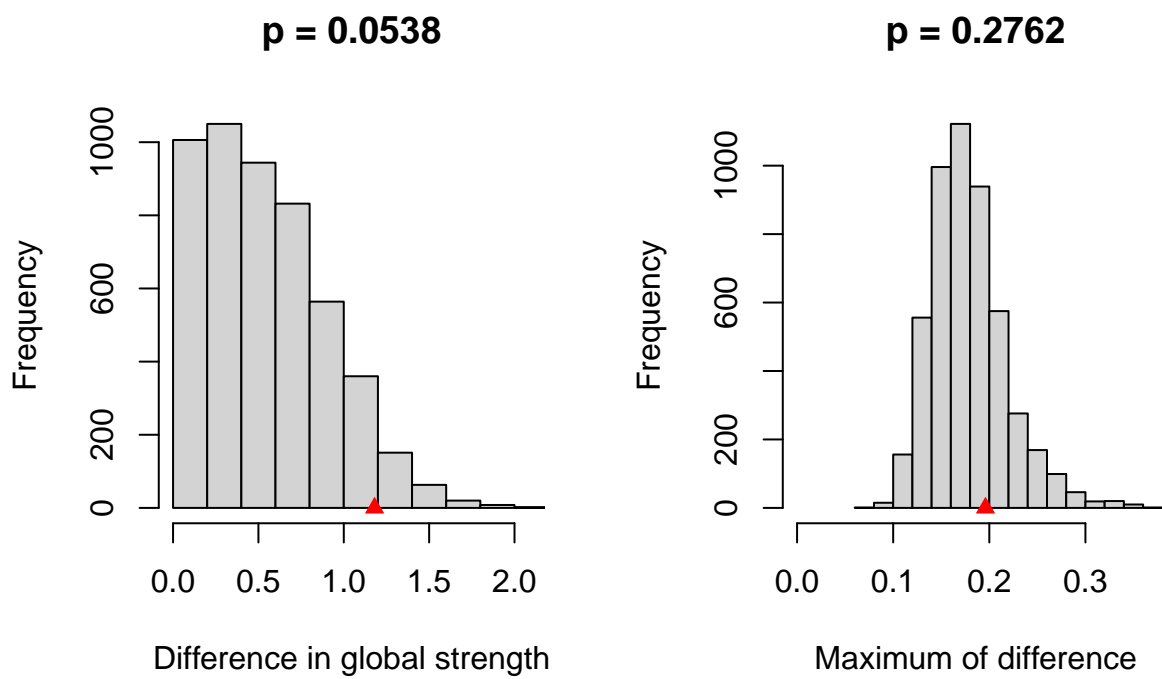


Figure 10: Reference distributions for network comparison test