Assignment_1

Jieran Sun, Hui Jeong (HJ) Jung, Gumdmundur Björgvin Magnusson

2023-02-27

Question 1

For the network models a) and b), case a) holds the statement of $A \perp B|C$, whereas b) holds the statement of $A \perp B$.

For case a):

$$P(A, B \mid C) = \frac{P(A, B, C)}{P(C)} \tag{1}$$

$$= \frac{P(A \mid C)P(B \mid C)P(C)}{P(C)} \tag{2}$$

$$= P(A \mid C)P(B \mid C) \tag{3}$$

Hence the conditional independence.

For case b):

$$P(C \mid A, B) = \frac{P(A, B, C)}{P(A, B)} \tag{4}$$

$$= \frac{P(C \mid A, B)P(B)P(A)}{P(A, B)} \tag{5}$$

Moving the denominator to the left, we end up with the equality

$$P(A, B) = P(A)P(B) \tag{6}$$

Question 2

Markov blanket $MB(D)$ is the set of nodes composed of the parents, co-parents and children of D. In this case, $MB(D) = \{B, F, C, G, E\}$. Given that

$$P(D \mid MB(D), A) = \frac{P(D, MB(D), A)}{P(A, MB(D))} \tag{7}$$

$$= \frac{P(A \mid D, MB(D))P(D, MB(D))}{P(A \mid MB(D))P(MB(D))} \tag{8}$$

Because A is independent to D and $MB(D)$, we can simplify the expression $P(A|D, MB(D))$ to $P(A)$.

$$\frac{P(A)P(D, MB(D))}{P(A)P(MB(D))} = \frac{P(D, MB(D))}{P(MB(D))} \tag{9}$$

$$= P(D \mid MB(D)) \tag{10}$$

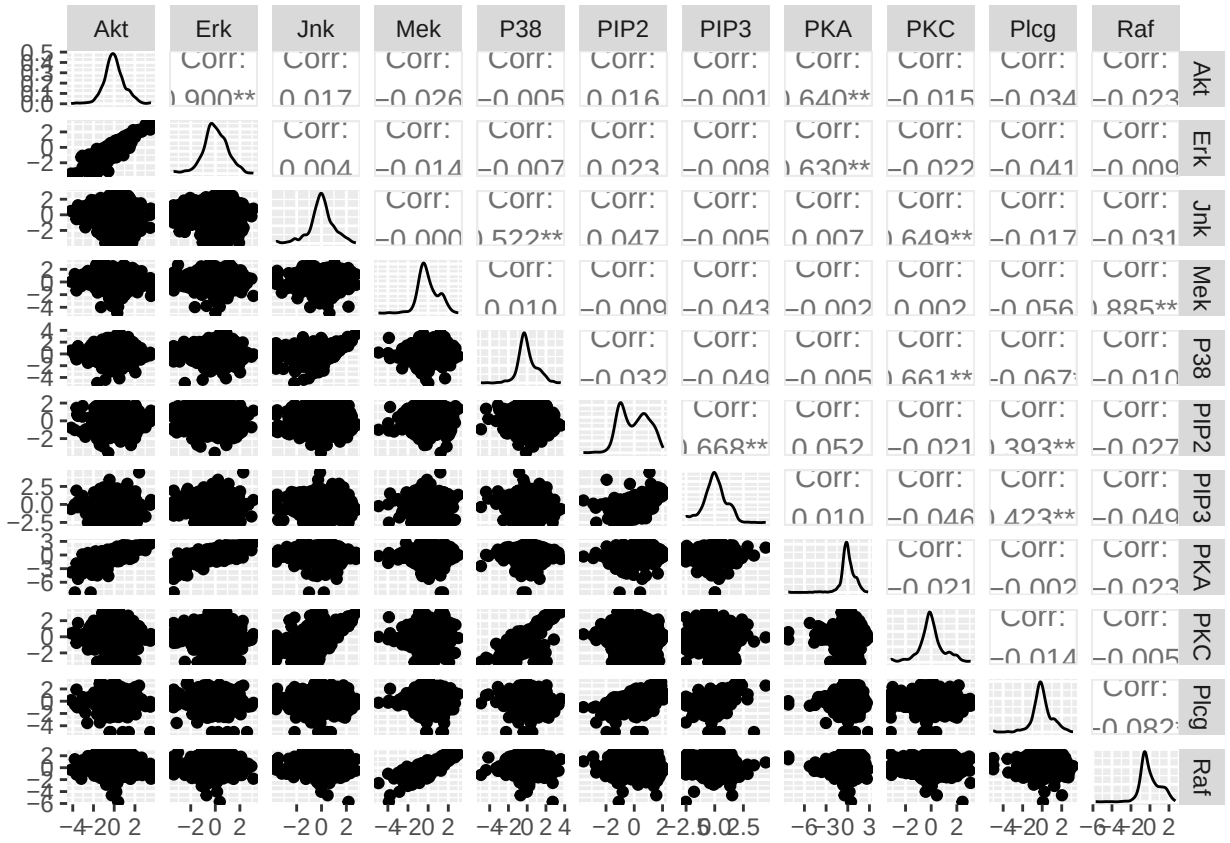
Question 3

a

```
data <- read.csv("https://raw.githubusercontent.com/felixleopoldo/benchpress/master/resources/data/myda
```

The number of observations N is equal to 902, the number of variables n is equal to 11.

```
GGally::ggpairs(data)
```




```
set.seed(2023)
ind <- sample(1:nrow(data), as.integer(0.8*nrow(data)), replace = FALSE)
train_data <- data[ind,]
test_data <- data[-ind,]
```

```
init_param <- BiDAG::scoreparameters(scoretype= "bge", data= train_data)
```

b

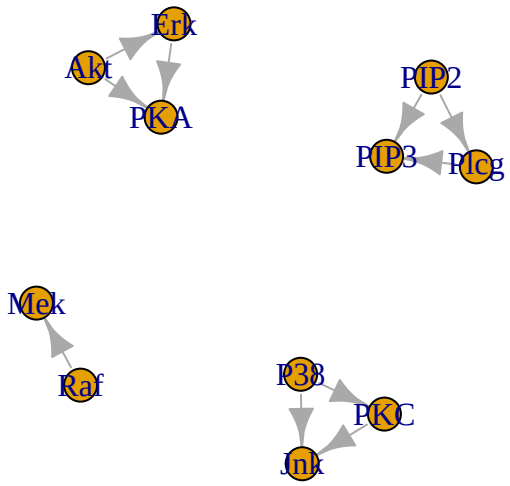
```
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
mcmc <- iterativeMCMC(init_param)
```

```
## maximum parent set size is 2
## core space defined, score table are being computed
## score tables completed, iterative MCMC is running
```

```
adj <- getDAG(mcmc)

ig <- graph_from_adjacency_matrix(adj, mode= "directed")
plot.igraph(ig)
```



c

To compare over different am values, we learned the DAG 100 times for each am values and obtained the average number of edges that the different DAGs had for 100 different datasets, as well as the average sum of the log scores and the average mean of the log scores.

```
set.seed(2023)
# use when utilizing mclapply
RNGkind("L'Ecuyer-CMRG")

# different am values to compare which performs best
am <- c(10^-3, 10^-1, 1, 10, 10^2)

results <- parallel::mclapply(am, mc.cores= 5, function(am_index){
    sum <- 0
    mean <- 0
    edge <- 0
    iter <- 100

    for (c in 1:iter) {
      ind <- sample(1:nrow(data), as.integer(0.8*nrow(data)), replace = FALSE)
      train_data_c <- data[ind,]
      test_data_c <- data[-ind,]

      scorepar_train <- scoreparameters(scoretype= "bge", data= train_data_c,
```

```
                    bgepar= list(am= am_index))
      mcmc_train <- iterativeMCMC(scorepar_train)

      num_edges <- sum(mcmc_train$DAG)
      log_score <- scoreagainstDAG(scorepar= scorepar_train, incidence= mcmc_train$DAG,
                          datatoscore= test_data_c)
      sum <- sum + sum(log_score)
      mean <- mean + mean(log_score)
      edge <- edge + num_edges
    }
    log_score_sum  <- sum / iter
    log_score_mean <- mean / iter
    num_edges <- edge / iter

    c(num_edges, log_score_sum, log_score_mean)

}) %>% as.data.frame() %>% set_colnames(am) %>%
  set_rownames(c('Avg Number of Edges', 'Avg Sum of Log Scores', 'Avg Mean of Log Scores'))

knitr::kable(results)
```

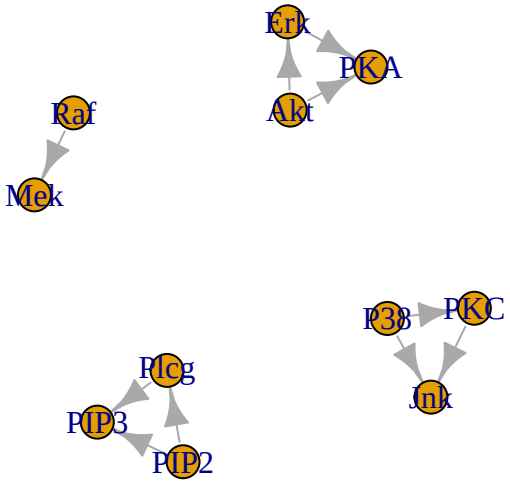
	0.001	0.1	1	10	100
Avg Number of Edges	7.03000	9.34000	10.11000	12.82000	15.87000
Avg Sum of Log Scores	-2328.01211	-2332.19842	-2323.03818	-2323.11963	-2367.03003
Avg Mean of Log Scores	-12.86195	-12.88507	-12.83447	-12.83492	-13.07751

From the above analysis, we can assume that the best am value is 1. Based on that we generated the optimal DAG

```
final_param <- BiDAG::scoreparameters(scoretype= "bge", data= data)
mcmc_final <- iterativeMCMC(final_param)
```

```
## maximum parent set size is 3
## core space defined, score table are being computed
## score tables completed, iterative MCMC is running
```

```
adj_final <- getDAG(mcmc_final)

ig_final <- graph_from_adjacency_matrix(adj_final, mode= "directed")
plot.igraph(ig)
```


Erk
PKA
Akt
Raf
Mek
P38
PKC
Jnk
Plcg
PIP3
PIP2