SMCB Project 1

February 24, 2023

Problem 1: Conditional independence and BNs

(a)

$A \perp B|C$ holds, but $A \perp B$ does not hold.

Proof for $A \perp B|C$

From the BN, we know that

$$P(A, B, C) = P(A|C)P(B|C)P(C)$$

To prove that $A \perp B|C$, we need to show that $P(A, B|C) = P(A|C)P(B|C)$.

$$\begin{aligned} P(A, B|C) &= \frac{P(A, B, C)}{P(C)} \\ &= \frac{P(A|C)P(B|C)P(C)}{P(C)} \\ &= P(A|C)P(B|C) \end{aligned}$$

Disproof for $A \perp B$

Consider $P(A, B) = \sum_C P(A, B, C)$. From the BN, we have:

$$\begin{aligned} P(A, B) &= \sum_C P(A, B, C) \\ &= \sum_C P(A|C)P(B|C)P(C) \\ &= \sum_C \frac{P(C|A)P(A)}{P(C)} \frac{P(C|B)P(B)}{P(C)} P(C) \\ &= P(A)P(B) \sum_C \frac{P(C|A)P(C|B)}{P(C)} \\ &\neq P(A)P(B) && \text{(usually)} \end{aligned}$$

Hence, in general $A \perp B$ does not hold for this Bayesian network.

(b)

$A \perp B|C$ does not hold, but $A \perp B$ holds

Disproof for $A \perp B|C$

Consider $A \perp B|C$, we'll have $P(A,B|C) = P(A|C)P(B|C)$.

From the BN and by applying Bayes theorem we can derive that:

$$\begin{aligned}
P(A,B|C) &= \frac{P(A,B,C)}{P(C)} \\
&= \frac{P(A)P(B)P(C|A,B)}{P(C)} \\
&= \frac{P(A)P(B)P(C|A,B)}{P(C)} \frac{P(C|A)}{P(C|A)} \frac{P(C|B)}{P(C|B)} \\
&= \frac{P(C|A)P(A)}{P(C)} \frac{P(C|B)P(B)}{P(C)} \frac{P(C)P(C|A,B)}{P(C|A)P(C|B)} \\
&= P(A|C)P(B|C) \frac{P(C)P(C|A,B)}{P(C|A)P(C|B)} \\
&\neq P(A|C)P(B|C) && \text{(usually)}
\end{aligned}$$

Proof for $A \perp B$

From the BN, we know that

$$P(A,B,C) = P(A)P(B)P(C|A,B)$$

We know from Bayes theorem that $P(A,B,C) = P(C|A,B)P(A,B)$.

Hence we have:

$$P(C|A,B)P(A)P(B) = P(C|A,B)P(A,B)$$

i.e., $P(A,B) = P(A)P(B)$. We have consequently proven that $A \perp B$.

Problem 2: Markov blanket

The Markov blanket $MB(D)$ is $\{B,C,E,F,G\}$, where B and F are the parents of D, C and G are the children of D, and E is the co-parent of D.

To prove that the conditional probability of $P(X_k|X_{n \neq k})$ is equivalent to $P(X_k|MB(X_k))$, we can prove that $\forall X_j$ where $j \in [1,n], j \neq k$, if $X_j \notin MB(X_k)$, then $X_j \perp X_k|MB(X_k)$.

For this specific question, we need to prove that $A \perp D|MB(D)$. It is obvious that $A \perp D|MB(D)$ because A and D are d-separated given $MB(D)$ since $E \in MB(D)$ is on the (only) path from A to D and E is in a cascade structure $A \rightarrow E \rightarrow G$.

Problem 3

Package and Dataset Preparation

```r
set.seed(2023)
```

```r
# download dataset
df <- read.csv(file="https://raw.githubusercontent.com/felixleopoldo/benchpress/master/resources/data/my
head(df)
```

```
##           Akt        Erk        Jnk         Mek         P38       PIP2
## 1 -0.63433612 -0.1117883 -0.3707515 -0.58558428 -0.06458972  0.6818205
## 2 -3.04091029 -2.5379116  1.0548648 -0.08291055 -0.10231212  1.6658269
## 3 -0.10795269 -0.7494918  0.7096003  0.86363654 -0.23355736 -1.1057101
## 4 -0.05846518  0.3253933  1.1038411 -1.69765652 -0.18079485  1.3026594
## 5  0.32095996  0.5678002  0.6809554 -0.22157981 -0.11327037  0.1823161
## 6 -0.25409240 -0.2603138 -1.6322745 -1.61663169  0.92638128 -1.3078990
##          PIP3         PKA        PKC       Plcg        Raf
## 1 -0.32402294 -0.04326735 -0.6878319 -0.3955337 -0.5148379
## 2  1.18130472 -4.07209170  0.2993658  0.6777917 -0.1101130
## 3  0.09555701 -0.43897960 -0.1565200 -0.2206535  0.6457071
## 4  1.36106920 -0.12200029  0.1600811  0.9960279 -1.6614249
## 5 -0.08623616 -0.74481674 -0.2974304 -0.7794014 -0.3258126
## 6  0.23050612 -0.11188567 -0.6522629 -0.9392948 -0.8496813
```

(a)

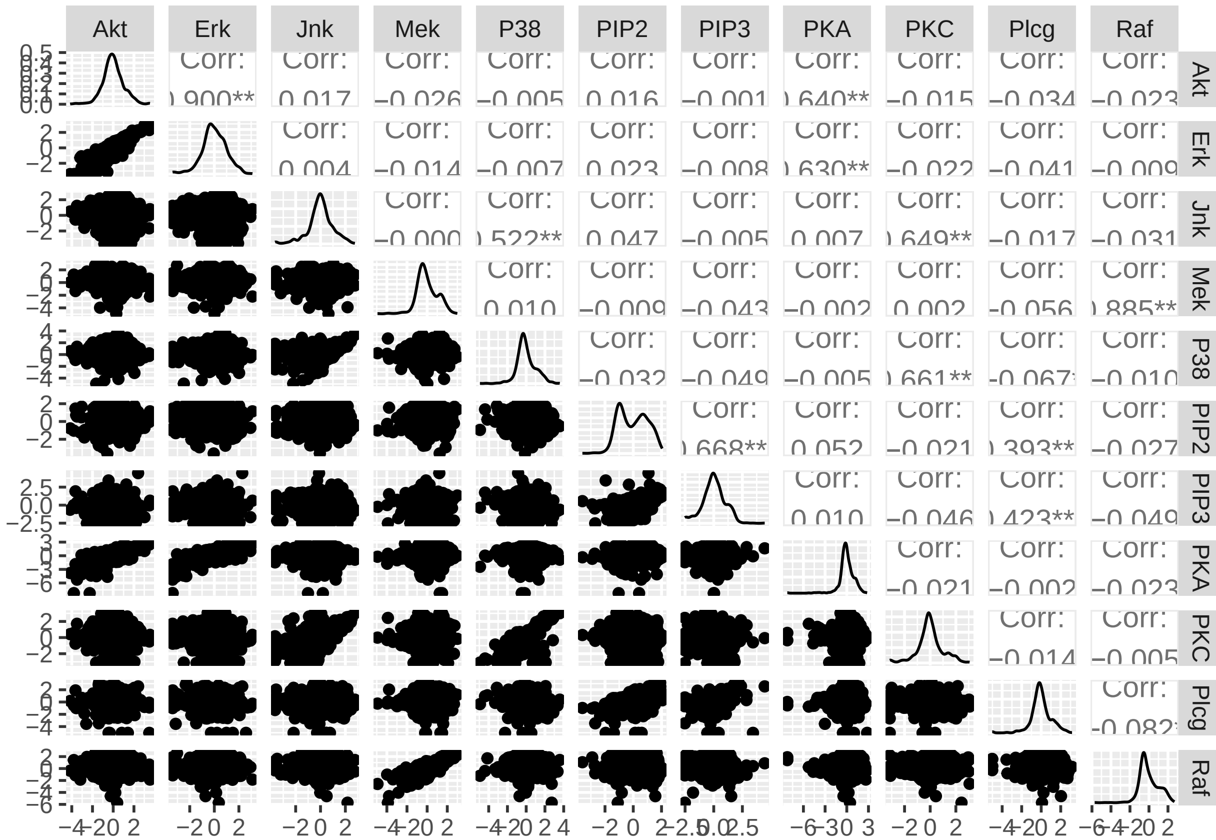
Number of variables (n): 11 Number of observations (N): 902

```r
dim(df)
```

```
## [1] 902  11
```

Visualization of the transformed data using `ggpairs` function.

```r
ggpairs(df)
```



Randomly split the data into 80% traning data and 20% test data.

```
train_data_size <- floor(0.8*nrow(df))
picked <- sample(seq_len(nrow(df)), size=train_data_size)
train_data <- df[picked, ]
test_data  <- df[-picked,]
```

Initialize the parameters using the function `BiDAG::scoreparameters` with the training data and the Bayesian Gaussian equivalent (BGe) score.

```
train_scorepar <- BiDAG::scoreparameters(scoretype="bge", train_data)
test_scorepar <- BiDAG::scoreparameters(scoretype="bge", test_data)
```

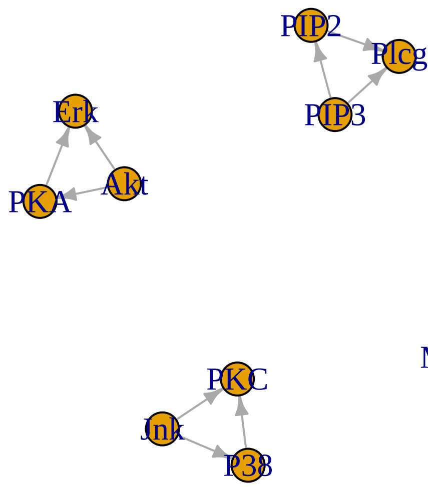
(b)

Learn a Bayesian network using the `BiDAG::iterativeMCMC` function.

```
BN <- iterativeMCMC(scorepar=train_scorepar, verbose=FALSE)
```

Plot the DAG.

```
DAG <- getDAG(BN, amat=TRUE)
g <- graph.adjacency(DAG, mode="directed")
plot.igraph(g, edge.arrow.size=.5)
```



Evaluate the log score of the test data against the estimated DAG using `BiDAG::scoreagainstDAG`.

```
log_score <- scoreagainstDAG(scorepar=test_scorepar, incidence=DAG)
mean(log_score)
```

```
## [1] -12.42144
```

(c)

The following code blocks run 100 iterations for each experiments.

```
res <- data.frame(matrix(ncol=5, nrow=2))
colnames(res) <- c(1, 2, 3, 4, 5)
rownames(res) <- c("ecount", "logscore")
```

```
procedure <- function(am) {
    picked <- sample(seq_len(nrow(df)), size=train_data_size)
    train_data <- df[picked, ]
    test_data  <- df[-picked,]
    train_scorepar <- BiDAG::scoreparameters(
        scoretype="bge",
        train_data,
        bgepar=list(am=am, aw=NULL)
    )
```

```
    BN <- BiDAG::iterativeMCMC(scorepar=train_scorepar)
    return(BN)
}
```

```
num_cores <- detectCores()
registerDoParallel(num_cores)
index <- 1
foreach (am=c(1e-3, 1e-1, 1e0, 1e1, 1e2)) %do% {
    set.seed(2023)
    v <- foreach (i=1:100, .combine=rbind) %dorng% {
        picked <- sample(seq_len(nrow(df)), size=train_data_size)
        train_data <- df[picked, ]
        test_data  <- df[-picked,]
        train_scorepar <- BiDAG::scoreparameters(
            scoretype="bge", train_data,
            bgepar=list(am=am, aw=NULL)
        )
        test_scorepar <- BiDAG::scoreparameters(
            scoretype="bge", test_data,
            bgepar=list(am=am, aw=NULL)
        )
        BN <- BiDAG::iterativeMCMC(scorepar=train_scorepar)
        DAG <- BiDAG::getDAG(BN)
        g <- igraph::graph.adjacency(DAG, mode="directed")
        ecount <- igraph::ecount(g)
        log_score <- BiDAG::scoreagainstDAG(
            scorepar=test_scorepar,
            incidence=DAG
        )
        avg_log_score <- mean(log_score)
        return(c(ecount=ecount, log_score=avg_log_score))
    }
    avg <- colMeans(v)
    res[1, index] <- avg[1]
    res[2, index] <- avg[2]
    index <- index + 1
}
stopImplicitCluster()
```

```
colnames(res) <- c(1e-3, 1e-1, 1, 10, 1e2)
print(res)
```

```
##               0.001       0.1         1        10       100
## ecount     7.02000   9.36000  10.12000  12.76000  15.62000
## logscore -12.63433 -12.59783 -12.57238 -12.63054 -13.50398
```

It seems that as `am` increase, the average number of edges also increases. The log score seems to have an optimum at `aw=1` for the parameters we tested.

The final graph from the whole dataset is visualized in the following code block.

```
scorepar <- BiDAG::scoreparameters(scoretype="bge", df, bgepar=list(am=1, aw=NULL))
BN <- BiDAG::iterativeMCMC(scorepar=scorepar, verbose=FALSE)
DAG <- BiDAG::getDAG(BN, amat=TRUE)
g <- graph.adjacency(DAG, mode="directed")
plot.igraph(g, edge.arrow.size=.5)
```