SMCB Project 1

February 24, 2023

Problem 1: Conditional independence and BNs

$A \perp B|C$ holds for **(a)** and $A \perp B$ holds for **(b)**.

(a)

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

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

(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 balnket $MB(D)$ 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 $A \to E \to G$.

Problem 3

Package and Dataset Preparation

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   `force = TRUE` to re-install: 'graph' 'Rgraphviz' 'RBGL'
```

```
list.of.packages <- c("GGally", "BiDAG", "igraph")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)
```

```
library("GGally", "BiDAG", "igraph")
```

```
## Warning: package 'GGally' was built under R version 4.2.2

## Warning in library("GGally", "BiDAG", "igraph"): 'igraph' not found on search
## path, using pos = 2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.2

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

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

```
df <- read.csv(file="dataset.csv")
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

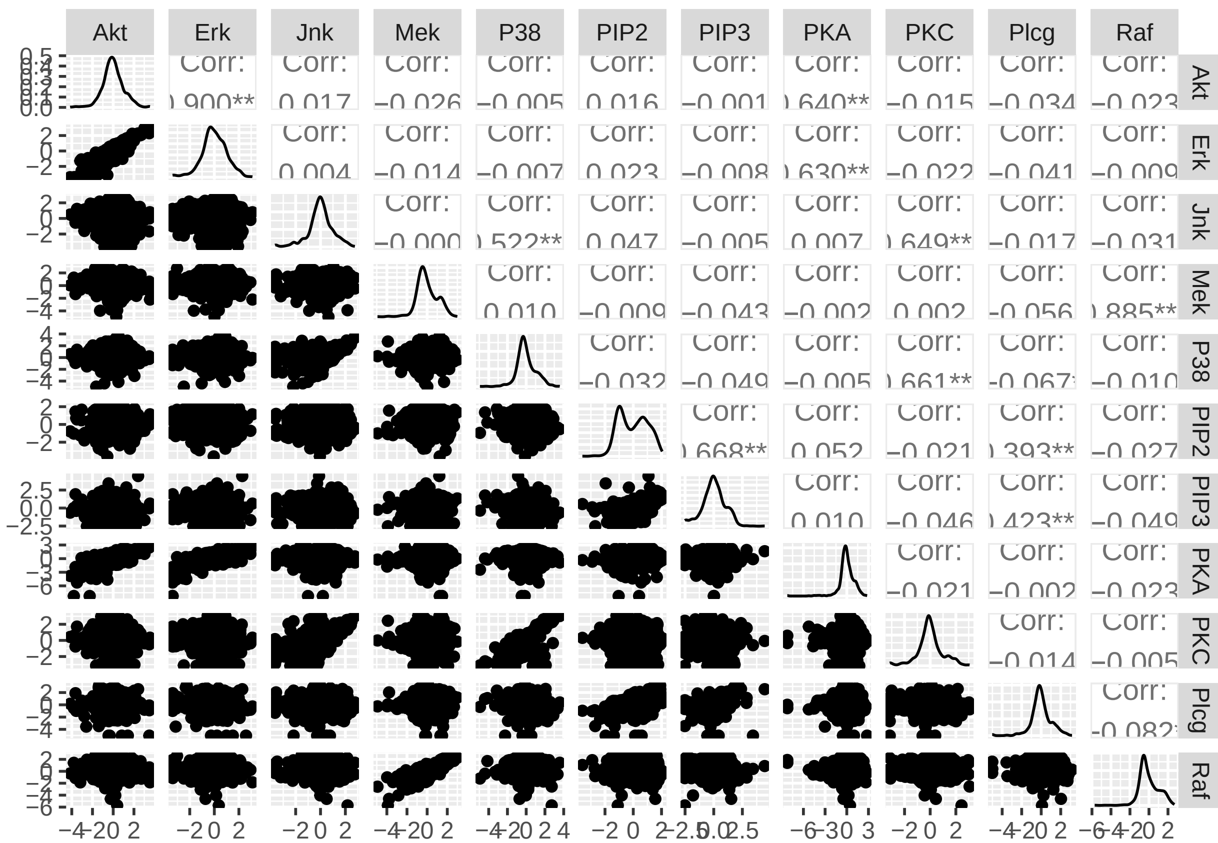
```
str(df)
```

```
## 'data.frame':    902 obs. of  11 variables:
##  $ Akt : num  -0.6343 -3.0409 -0.108 -0.0585 0.321 ...
##  $ Erk : num  -0.112 -2.538 -0.749 0.325 0.568 ...
##  $ Jnk : num  -0.371 1.055 0.71 1.104 0.681 ...
##  $ Mek : num  -0.5856 -0.0829 0.8636 -1.6977 -0.2216 ...
##  $ P38 : num  -0.0646 -0.1023 -0.2336 -0.1808 -0.1133 ...
##  $ PIP2: num  0.682 1.666 -1.106 1.303 0.182 ...
##  $ PIP3: num  -0.324 1.1813 0.0956 1.3611 -0.0862 ...
```

```
##  $ PKA : num  -0.0433 -4.0721 -0.439 -0.122 -0.7448 ...
##  $ PKC : num  -0.688 0.299 -0.157 0.16 -0.297 ...
##  $ Plcg: num  -0.396 0.678 -0.221 0.996 -0.779 ...
##  $ Raf : num  -0.515 -0.11 0.646 -1.661 -0.326 ...
```

Visualization of the transformed data using `ggpairs` function.

```r
ggpairs(df)
```



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

```r
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.

```r
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.

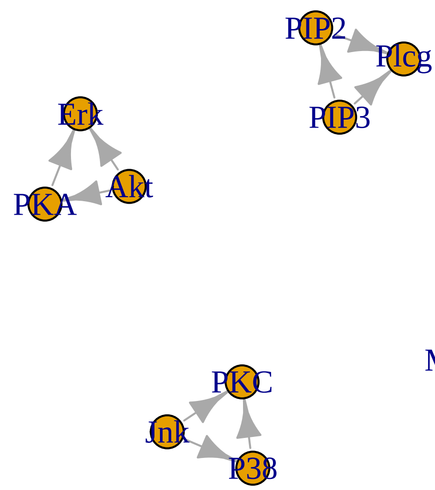
```r
BN <- BiDAG::iterativeMCMC(scorepar=train_scorepar)
```

```
## maximum parent set size is 2
```

```
## core space defined, score table are being computed
## score tables completed, iterative MCMC is running
```

Plot the DAG.

```
DAG <- BiDAG::getDAG(BN)
igraph::plot.igraph(igraph::graph.adjacency(DAG, mode="directed"))
```



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

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

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

(c)

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

```
procedure <- function(am, index=1) {
    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,
```

```r
        bgepar=list(am=am, aw=NULL)
    )
    BN <- BiDAG::iterativeMCMC(scorepar=train_scorepar)
    return(BN)
}
```

```r
list.of.packages <- c("foreach", "doParallel", "doRNG")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)
```

```r
library(foreach)
```

```
## Warning: package 'foreach' was built under R version 4.2.2
```

```r
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 4.2.2

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.2.2

## Loading required package: parallel
```

```r
library(doRNG)
```

```
## Warning: package 'doRNG' was built under R version 4.2.2

## Loading required package: rngtools

## Warning: package 'rngtools' was built under R version 4.2.2
```

```r
library(parallel)
```

```r
num_cores <- detectCores()
registerDoParallel(num_cores)
index <- 1
foreach (am=c(1e-3, 1e-1, 1e0, 1e1, 1e2)) %do% {
    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)
        test_scorepar <- BiDAG::scoreparameters(scoretype="bge", test_data)
        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]
```

```r
    index <- index + 1
}
```

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

```r
stopImplicitCluster()
print(res)
```

```
##                  1         2         3        4         5
## ecount    10.17000  10.17000  10.09000  10.0800  10.16000
## logscore -12.58023 -12.57828 -12.62868 -12.6935 -12.66024
```

Parameter `am`	10^{-3}	10^{-1}	1	10	10^{2}
Average number of edges	10.17	10.17	10.09	10.08	10.16
Average log score of the test data	-12.58023	-12.57828	-12.62828	-12.6935	-12.66024