BCB570 Assignment 4

Ashish Jain

Question 4:

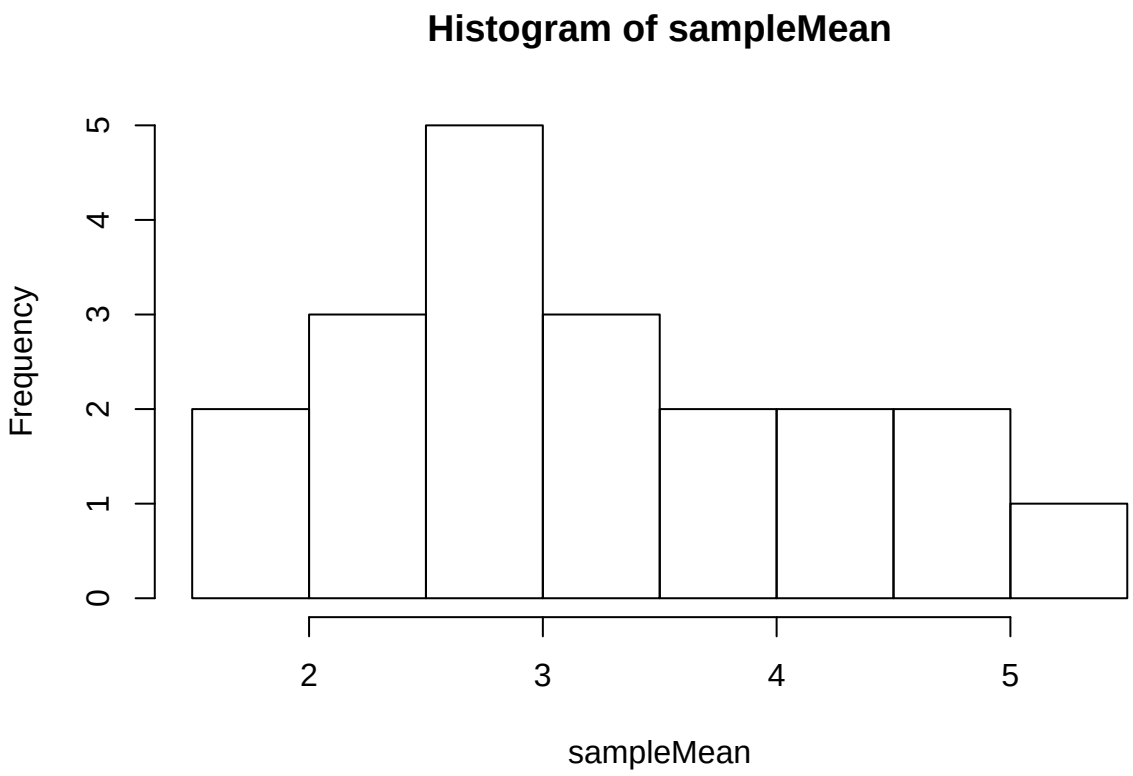
a) Bagging

Bootstrap aggregation is called as Bagging. Before explaining bagging first I would like to explain what bootstrap is. Bootstrap is random sampling with replacement which has been used to estimate the sample distribution. Now, bagging is a type of Ensembl method in which bootstrapping is carried out a number of times and the final result is being calculated by taking the average of the all the bootstrap samples. For example, if we have a training set D of size n, we can generate k new training set D_j of size n, by sampling the original training set D uniformly and with replacement. These training sets can have duplicate observations as the sampling is carried out with replacement. From definition, each sampling is called as bootstrapping and the generation of k new training set is called as bagging. Finally, the generated k models generated are combined by taking the average (regression) or through voting (classification).

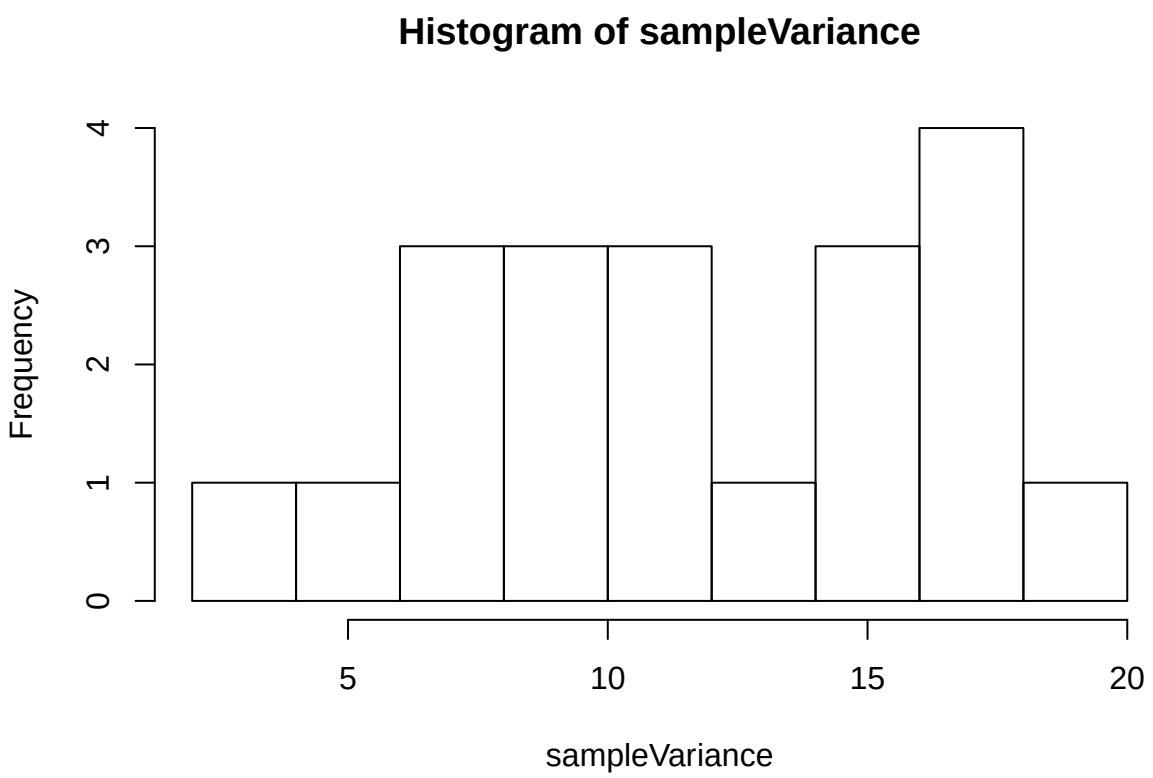
b)

```
normalData <- rnorm(100, mean = 2.5, sd = sqrt(10))
baggedData <- matrix(nrow = 20, ncol = 10)
for (i in 1:20) {
    baggedData[i, ] <- sample(normalData, 10, replace = TRUE)
}

sampleMean <- apply(baggedData, 1, mean)
sampleVariance <- (apply(baggedData, 1, var))
hist(sampleMean)
```




```
hist(sampleVariance)
```



```
samMean <- mean(sampleMean)
samVariance <- mean(sampleVariance)
```

The estimated value of the mean after bagging is 3.2161824 and the variance is 11.830491.

Question 5:

a).

WGCNA

In WGCNA, we first calculated the soft threshold for calculating the the power of the adjacency matrix. In this, we took 0.7 as the threshold for the R^2 value and took the corresponding power as the soft threshold for the adjacency matrix. After that, we took the top quartile edge weight value as the threshold to filter the edges.

GENIE3

In GENIE3, we first normalized the expression values as suggested in the GENIE3 paper. After that, we used 100 decision tress to predict the edges between the genes. After that, we took the top quartile value as the threshold to filter the edges.

ARACNE2

In ARACNE2, we took the we took the top quartile mutual information value as the threshold to filter the edges.

b).

Below is the R code which has been used to run the different tools on the dataset and then filter the edges based on their respective scores. As, we mentioned we filtered edges based on top quartile value. Below is the code and the commands we have used to predict the GRNs using GENIE3 which is based on the random forest algorithm. In our case, we have used a total of 100 decision trees to predict the GRN.

```
#Code for GENIE3
source("GENIE3.R")
setwd("./GENIE3_R_C_wrapper/GENIE3_R_C_wrapper/")
filePath<-"/home/jain/BCB570/Size_10/Size_10/DREAM4_training_data/insilico_size10_5/"
expr.matrix <- read.expr.matrix(paste0(filePath,"concatenateddata1.tsv"), form="rows.are.samples")
scaleData<-t(apply(expr.matrix,1,function(x){return((x-mean(x))/sd(x))}))
weight.matrix1 <- GENIE3(scaleData,ncores = 8, K="all",ntrees = 100)
link.list <- get.link.list(weight.matrix1)
write.table(link.list,paste0(filePath,"GENIE3-allLinks.txt"),row.names = F)
write.table(link.list[link.list$weight >= quantile(link.list$weight,0.75),],
paste0(filePath,"GENIE3-filterLinks.txt"), row.names = F,col.names = F,sep = "\t")
```

WGCNA is weighted gene co-expression network analysis package in R which predicts the GRNs based on the co-expression analysis. In our case, we have used spearson correlation method to calculate the co-expression

values. We have tried to make the network scale free but R^2 value for very high threshold is coming out to be only 0.4 which is not very good. The corresponding degree is also coming out to be very small. Due to these problems, we have taken power as 1 to calculate the adjacency matrix.

```
#Code for WGCNA
library(WGCNA)
library(igraph)
options(stringsAsFactors = FALSE);
filePath<-"/home/jain/BCB570/Size_10/Size_10/DREAM4_training_data/insilico_size10_5/"
datExpr <- read.table(paste0(filePath,"concatenateddata1.tsv"),header = T);
#softPower = 30
datAdj= adjacency(datExpr,
                  type = "unsigned",
                  power = 1,
                  corFnc = "cor", corOptions = "use = 'p', method = 'spearman'")

diag(datAdj) <- 0
threshold<-quantile(datAdj,0.75)
ig <- graph.adjacency(datAdj, mode="directed", weighted=TRUE)
edges<-get.edgelist(ig)
weights<-edge_attr(ig,"weight")
edgesWithWT<-data.frame(cbind(edges,weights))
write.table(edgesWithWT[edgesWithWT$weights >= quantile(weights,0.75),],
paste0(filePath,"WGCNA-filterLinks.txt"), row.names = F,col.names = F,sep = "\t")
```

ARACNE2 tool is based on the mutual information. It calculates and predicts the weights of the edges based on mutual information. In our study, we have used a java executable which takes gene expression files as an input. In this case, the gene expression file should contain gene as rows and samples as column. The results from this tool is not in a very good format, so we wrote a java code which give us the output as a list of edges with their weight.

```
#Code for ARACNE2
java -jar aracne2.jar -i ./HW4_Yeast1-1/concatenateddata2.tsv -o ./HW4_Yeast1-1/ARCANE-adjMat.txt

#Code to convert the Results
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;

public class CreateEdgeListFromARACNE {

    public static void main(String[] args) throws IOException{
        for(int f=1;f<=5;f++)
        {
            String filePath = "Size_10/Size_10/DREAM4_training_data/insilico_size10_"+f+"/";;
            BufferedReader br = new BufferedReader(new FileReader(filePath+"ARCANE-adjMat.txt"));
            String line = br.readLine();
            PrintWriter pw = new PrintWriter(filePath+"ARCANEedge.txt");
            while(line!=null)
            {
                if(!line.startsWith("<"))
                {
```

```
                    String lineData[] = line.split("\t");
                    String gene = lineData[0];
                    for(int i=1;i<lineData.length-1;i=i+2)
                    {
                        pw.println(gene+"\t"+lineData[i]+"\t"+lineData[i+1]);
                    }
                }
                line = br.readLine();
            }
            br.close();
            pw.close();
        }
    }
}
```

```
##R Code to Filter the size
filePath<-"/home/jain/BCB570/Size_10/Size_10/DREAM4_training_data/insilico_size10_5/"
edgesWithWT <-read.table(paste0(filePath,"ARCANEedge.txt"),header = F);
write.table(edgesWithWT[edgesWithWT$V3 >= quantile(edgesWithWT$V3,0.75),],
paste0(filePath,"ARCANE-filterLinks.txt"),row.names = F,col.names = F,sep = "\t")
```

The gold standards given to us are directed but the results that we get from our tools are undirected. So, in order to make the Precision-Recall (PR) curves, we have made the gold standards undirected making both the directions for a particular edge as positive. After that, we have wrote a java code to calculate the PR values, using which a PR curve is made in R.

```
#PR Java Code
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

public class PrecisionRecallCurveValues {

    public static void main(String[] args) throws IOException{
        for(int f=1;f<=5;f++)
        {
            String filePath = "./DREAM4_training_data/insilico_size10_"+f+"/";
            BufferedReader br = new BufferedReader(new FileReader(
            "./DREAM4_gold_standards/insilico_size10_" +f+"_goldstandard.tsv"));
            String line = br.readLine();
            Map<String, Integer> goldStandard = new HashMap<>();
            while(line!=null)
```

```
{
    String lineData[] = line.split("\t");
    if(Integer.parseInt(lineData[2]) == 1)
    {
        goldStandard.put(lineData[0]+"-"+lineData[1], Integer.parseInt(lineData[2]));
        goldStandard.put(lineData[1]+"-"+lineData[0], Integer.parseInt(lineData[2]));
    }
    line = br.readLine();
}
br.close();
List<String> toolList = Arrays.asList("WGCNA","GENIE3","ARCANE");
for(String s:toolList)
{
    String tool = s;
    Map<String, Float> predictData = new HashMap<>();
    br = new BufferedReader(new FileReader(filePath+tool+"-filterLinks.txt"));
    line = br.readLine();
    while(line!=null)
    {
        String lineData[] = line.split("\t");
        predictData.put(lineData[0]+"-"+lineData[1], Float.parseFloat(lineData[2]));
        line = br.readLine();
    }
    PrintWriter pw = new PrintWriter(filePath+tool+"-PrecisionRecallValues.txt");
    //Sort the map
    int postive = goldStandard.size();
    Map<String, Float> sortedPredictData = sortByComparatorValue(predictData);
    for(int i=1;i<=predictData.size();i++)
    {
        int j=0;
        float TP = 0;
        int FP = 0;
        float FN = 0;
        for(Entry<String, Float> entry : sortedPredictData.entrySet())
        {
            if(j<i)
            {
                if(goldStandard.containsKey(entry.getKey()))
                {
                    TP = TP + 1;
                }else
                {
                    FP = FP + 1;
                }
                j++;
            }else
            {
                break;
            }
        }
        FN = postive - TP;
        pw.println((TP/(TP+FP))+"\t"+(TP/(TP+FN)));
    }
    pw.close();
```

```
            }
        }
    }

    public static Map<String, Float> sortByComparatorValue(Map<String, Float> unsortMap) {

        // Convert Map to List
        List<Map.Entry<String, Float>> list =
            new LinkedList<Map.Entry<String, Float>>(unsortMap.entrySet());

        // Sort list with comparator, to compare the Map values
        Collections.sort(list, new Comparator<Map.Entry<String, Float>>() {
            public int compare(Map.Entry<String, Float> o1,
                                        Map.Entry<String, Float> o2) {
                return (o2.getValue()).compareTo(o1.getValue());
            }
        });

        // Convert sorted map back to a Map
        Map<String, Float> sortedMap = new LinkedHashMap<String, Float>();
        for (Iterator<Map.Entry<String, Float>> it = list.iterator(); it.hasNext();) {
            Map.Entry<String, Float> entry = it.next();
            sortedMap.put(entry.getKey(), entry.getValue());
        }
        return sortedMap;
    }
}
```

```
## Code to plot the PR curve install.packages('Bolstad2')
githubPage <- "https://raw.githubusercontent.com/ashishjain1988/BCB570/master/HW4/data/"
trainingData <- "training_data/insilico_size10_"
goldstandard <- "training_data/gold_standard/insilico_size10_"
library(Bolstad2)
AUCWGCNA <- c()
AUCGEN <- c()
AUCARCNE <- c()
AccWGCNA <- c()
AccGEN <- c()
AccARCNE <- c()
for (i in 1:5) {
    filePath <- paste0(githubPage, trainingData, i, "/")
    WGCNA <- read.table(paste0(filePath, "WGCNA-PrecisionRecallValues.txt"),
        header = F)
    GEN <- read.table(paste0(filePath, "GENIE3-PrecisionRecallValues.txt"),
        header = F)
    ARACNE <- read.table(paste0(filePath, "ARCANE-PrecisionRecallValues.txt"),
        header = F)
    AUCWGCNA <- c(AUCWGCNA, sintegral(WGCNA$V2, WGCNA$V1)$int)
    AUCGEN <- c(AUCGEN, sintegral(GEN$V2, GEN$V1)$int)
    AUCARCNE <- c(AUCARCNE, sintegral(ARACNE$V2, ARACNE$V1)$int)
    # png(filename=paste0(filePath,'PR-Curve.png'), width = 1200,
    # height = 800)
    ggplot(data.frame(WGCNA, ARACNE, GEN), aes(x = WGCNA$V2,
        y = WGCNA$V1, color = "WGCNA")) + ggtitle("PR Curve Network 5") +
```

```
        xlab("Recall") + ylab("Precision") + geom_line() + geom_line(aes(x = GEN$V2,
        y = GEN$V1, color = "GENIE3")) + geom_line(aes(x = ARACNE$V2,
        y = ARACNE$V1, color = "ARCNE"))
    # dev.off() Accuracy
    goldStd <- read.table(paste0(githubPage, goldstandard, i,
        "_goldstandard.tsv")) %>% mutate(edge = paste0(V1, "-",
        V2))
    aracaneEdges <- read.table(paste0(filePath, "ARCANE-filterLinks.txt"),
        header = F) %>% mutate(edge = paste0(V1, "-", V2))
    WGCNAEdges <- read.table(paste0(filePath, "WGCNA-filterLinks.txt"),
        header = F) %>% mutate(edge = paste0(V1, "-", V2))
    GENEdges <- read.table(paste0(filePath, "GENIE3-filterLinks.txt"),
        header = F) %>% mutate(edge = paste0(V1, "-", V2))

    AccARCNE <- c(AccARCNE, length(intersect(goldStd$edge, aracaneEdges$edge))/length(goldStd$edge) *
        100)
    AccGEN <- c(AccGEN, length(intersect(goldStd$edge, GENEdges$edge))/length(goldStd$edge) *
        100)
    AccWGCNA <- c(AccWGCNA, length(intersect(goldStd$edge, WGCNAEdges$edge))/length(goldStd$edge) *
        100)
}
details <- c("Network 1", "Network 2", "Network 3", "Network 4",
    "Network 5")
table <- data.frame(Network = details, ARACNE2 = AUCARCNE, GENIE3 = AUCGEN,
    WGCNA = AUCWGCNA)
table %>% knitr::kable(caption = "Area Under Precision-Recall Curve")
```

Table 1: Area Under Precision-Recall Curve

Network	ARACNE2	GENIE3	WGCNA
Network 1	0.4956888	0.4507886	0.2194936
Network 2	0.3693244	0.3172771	0.2757575
Network 3	0.4185679	0.3905613	0.4079854
Network 4	0.4074148	0.4712294	0.4537204
Network 5	0.5000249	0.5390574	0.4790988

```
table1 <- data.frame(Network = details, ARACNE2 = AccARCNE, GENIE3 = AccGEN,
    WGCNA = AccWGCNA)
table1 %>% knitr::kable(caption = "Accuracy in Percentage")
```

Table 2: Accuracy in Percentage

Network	ARACNE2	GENIE3	WGCNA
Network 1	60.00000	80.00000	33.33333
Network 2	56.25000	50.00000	43.75000
Network 3	60.00000	46.66667	53.33333
Network 4	46.15385	61.53846	53.84615
Network 5	66.66667	75.00000	58.33333

In the table1, the area under the PR curve and accuarcy is shown in table 1 and 2. It is seen from the results

that ARACNE2 and GENIE3 performs similarly in all the networks.For part c we used both ARACNE2 and GENIE3 for creating the GRN.

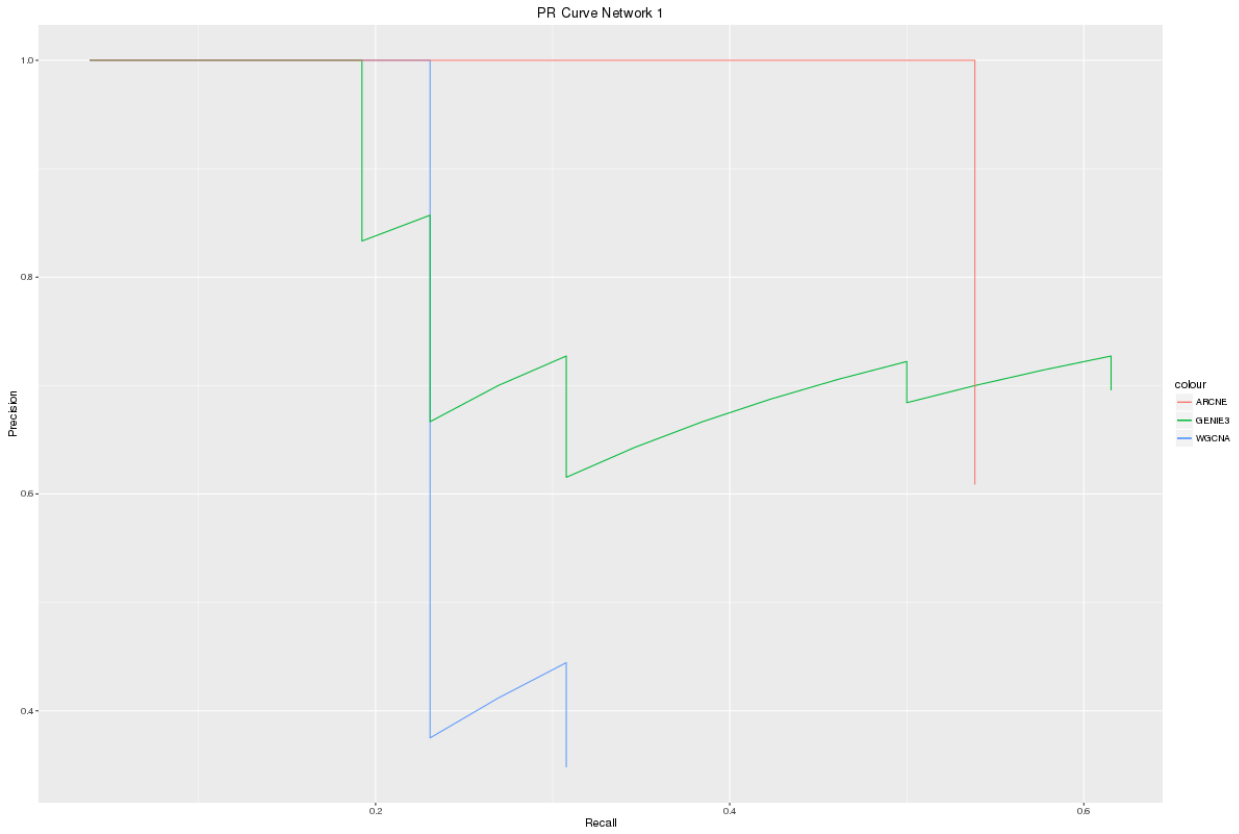


Figure 1: PR Curve for Network 1

c).

Based on the results from part b we used both ARACNE2 and GENIE3 for predicting the GRNs. As mentioned earlier, we have used top quartile value as the threshold for filtreing the edges. The results for both of the yeast networks are attached in the zip file under **"./results/Yeast-GRNs/"** folder.

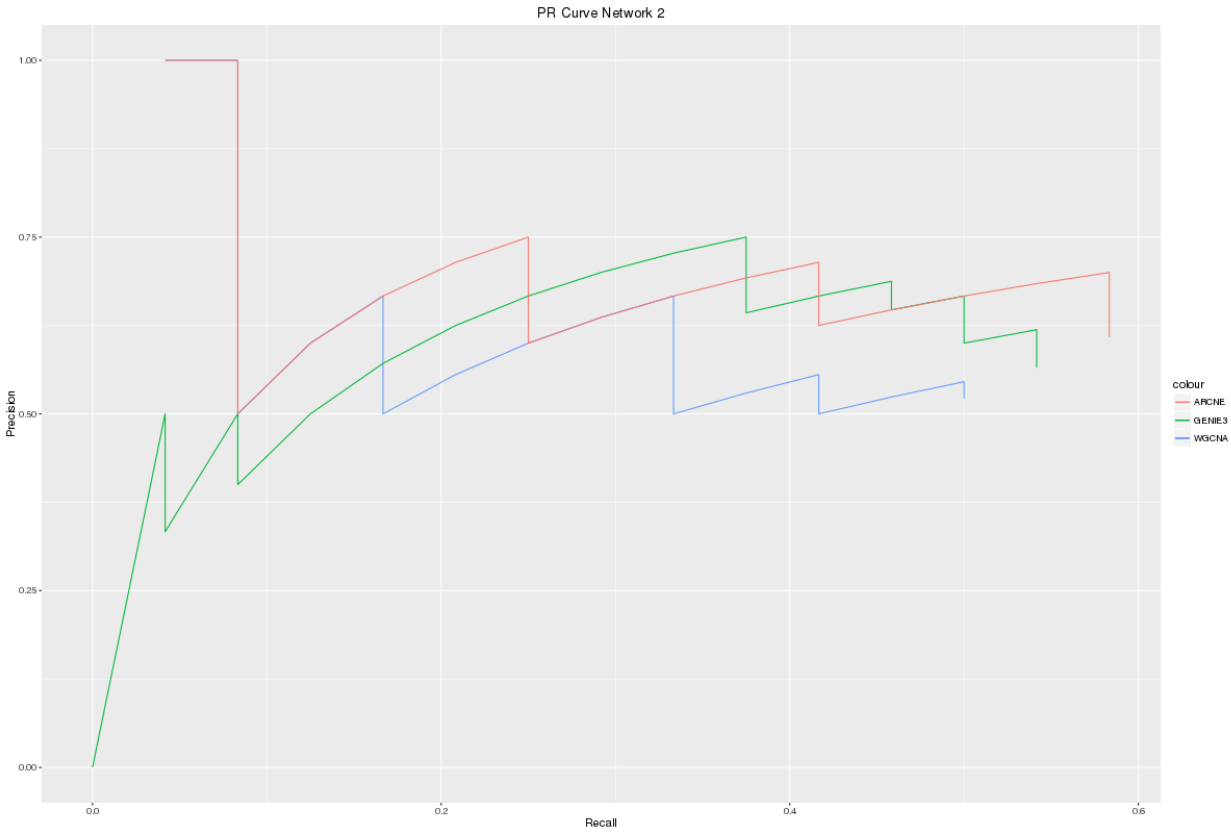


Figure 2: PR Curve for Network 2

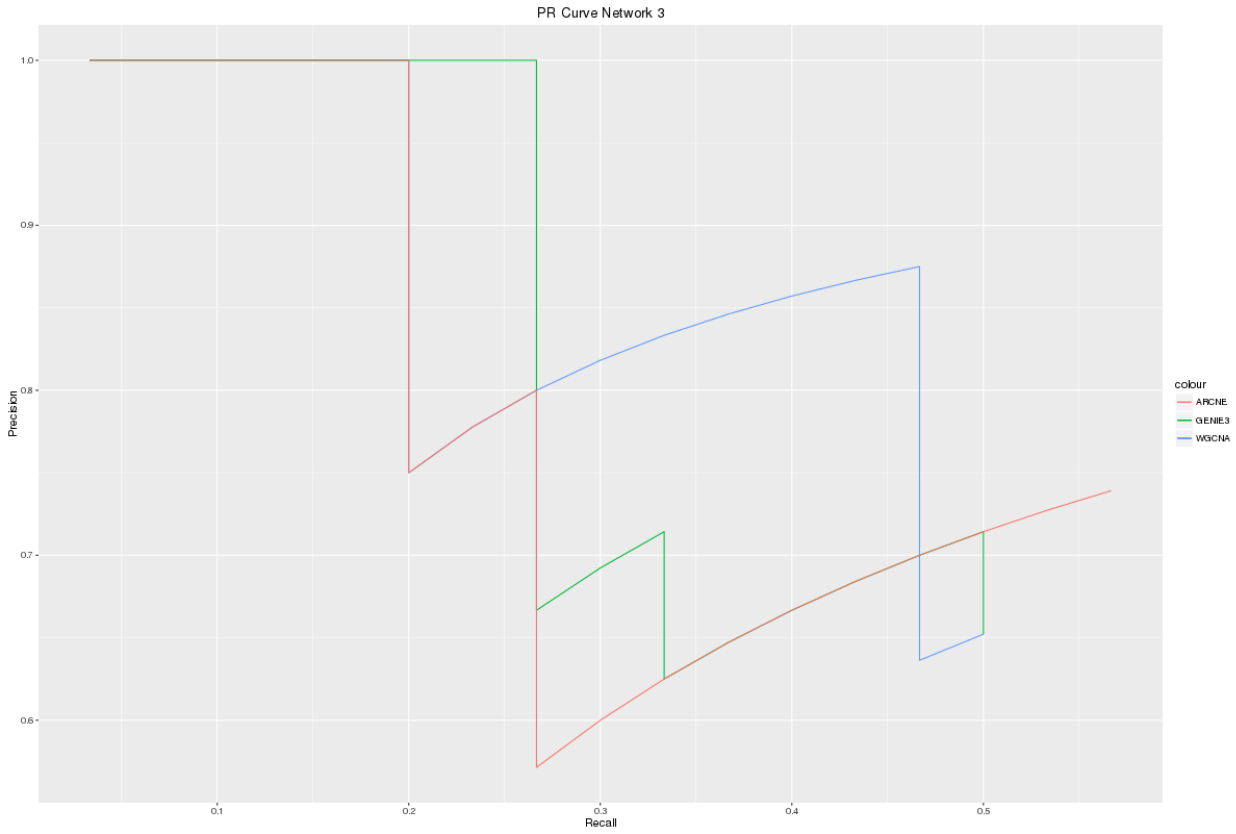


Figure 3: PR Curve for Network 3

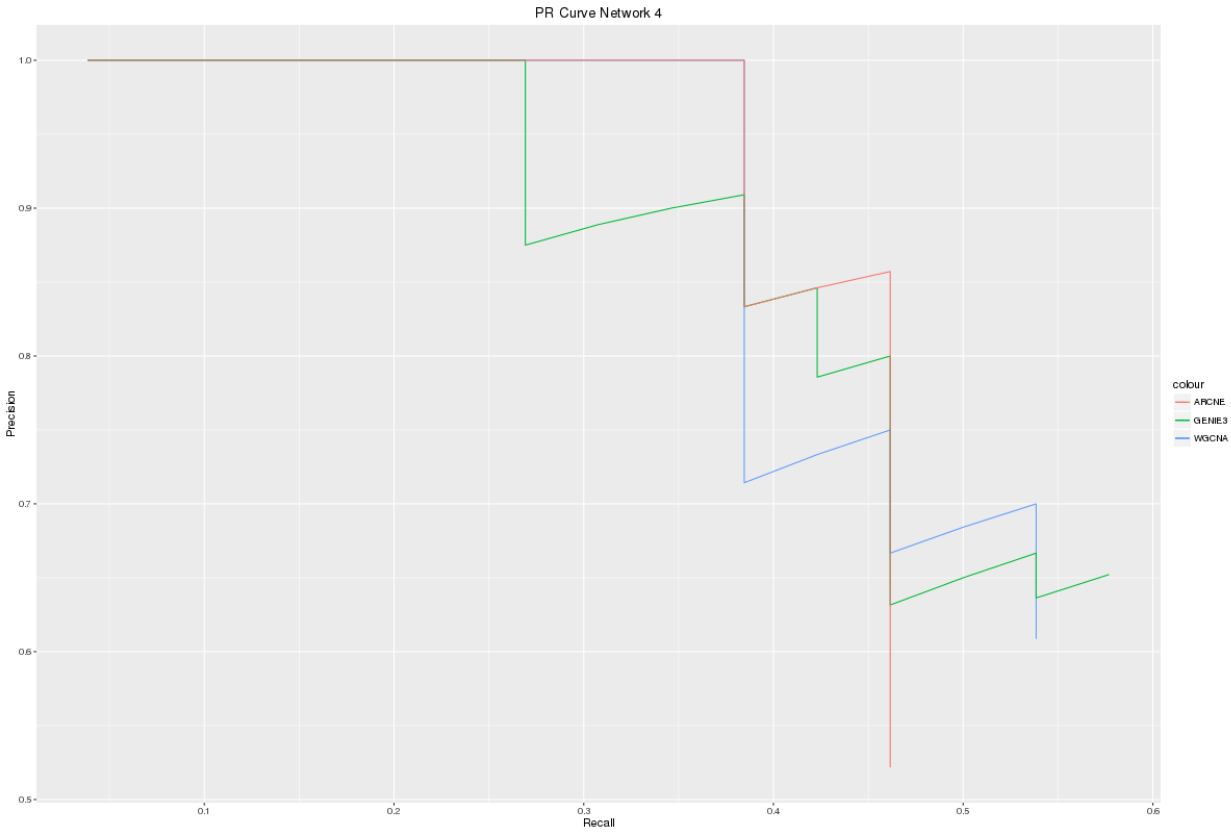


Figure 4: PR Curve for Network 4

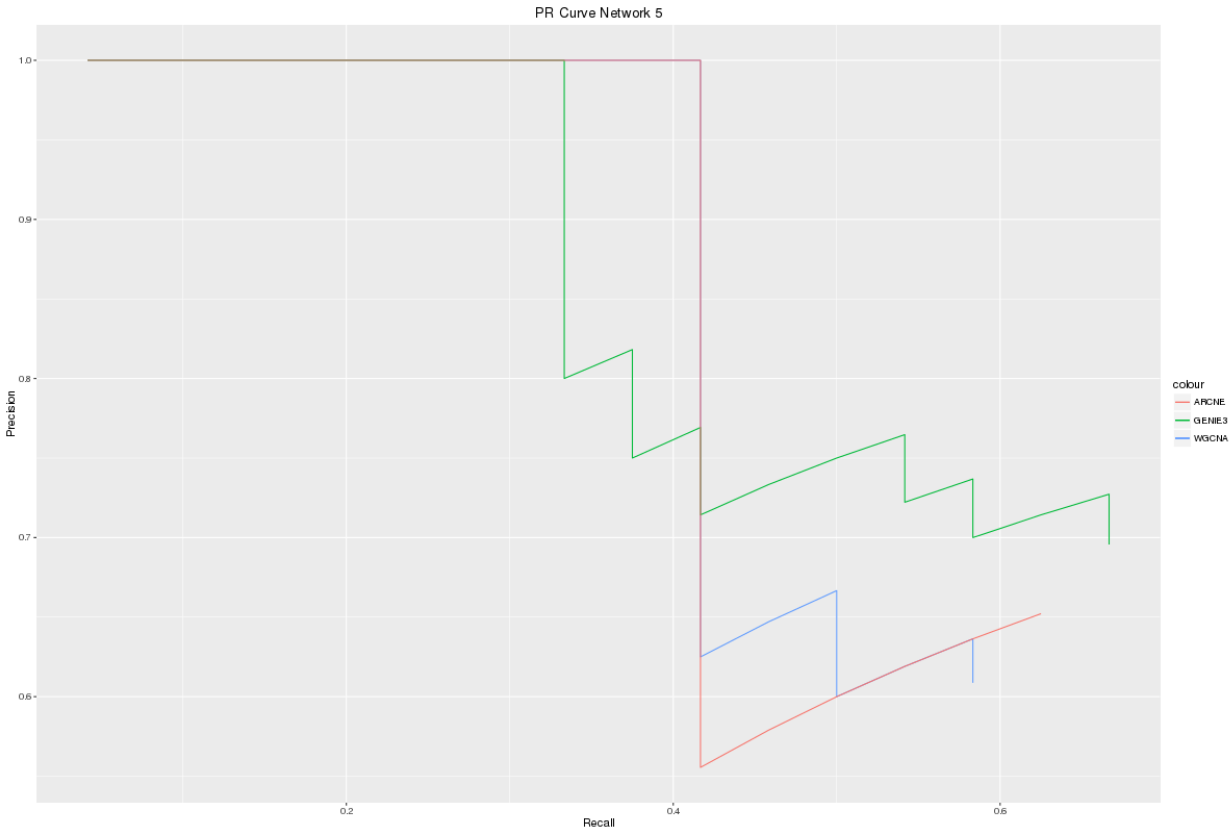


Figure 5: PR Curve for Network 5

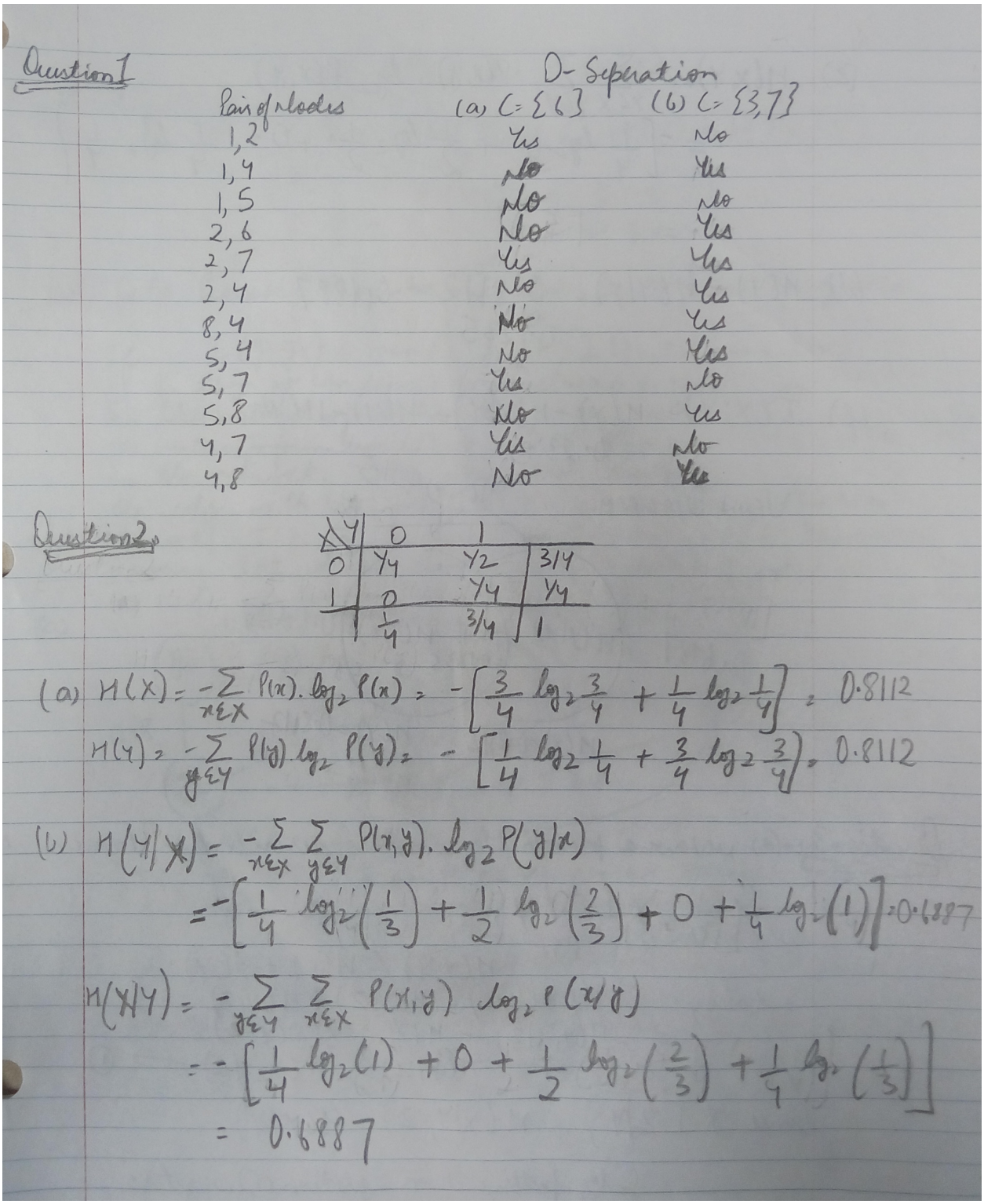

Question 1
D-Seperation
Pair of Nodes
(a) C = {6}
(b) C = {3,7}
1,2 Yes No
1,4 No Yes
1,5 No No
2,6 No Yes
2,7 Yes Yes
2,4 No Yes
8,4 No Yes
5,4 No Yes
5,7 Yes No
5,8 No Yes
4,7 Yes No
4,8 No Yes
Question 2
X \ Y 0 1
0 1/4 1/2 3/4
1 0 1/4 1/4
1/4 3/4 1
(a) H(X) = -Σ P(x). log2 P(x) = -[3/4 log2 3/4 + 1/4 log2 1/4] = 0.8112
H(Y) = -Σ P(y). log2 P(y) = -[1/4 log2 1/4 + 3/4 log2 3/4] = 0.8112
(b) H(Y/X) = -Σ Σ P(x,y). log2 P(y/x)
= -[1/4 log2(1/3) + 1/2 log2(2/3) + 0 + 1/4 log2(1)] = 0.6887
H(X/Y) = -Σ Σ P(x,y) log2 P(x/y)
= -[1/4 log2(1) + 0 + 1/2 log2(2/3) + 1/4 log2(1/3)]
= 0.6887

Figure 6:

(c) $H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} P(x,y)\cdot \log_2 P(x,y)$

$= -\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{2}\log_2\frac{1}{2} + 0 + \frac{1}{4}\log_2\frac{1}{4}\right]$

$= 1.5$

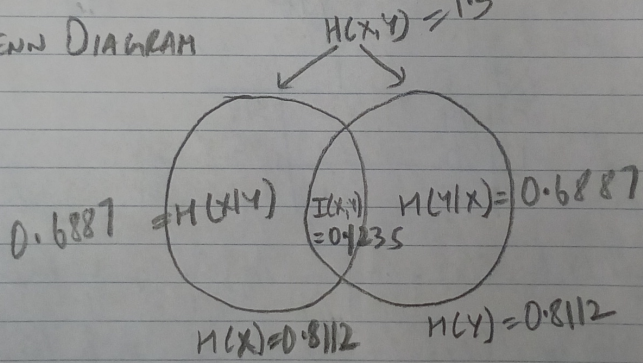
(d) $H(Y) - H(Y|X) = 0.8112 - 0.1887$

$= 0.1235$

(e) $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

$= 0.12345$

VENN DIAGRAM



Question 3:- (a) We have a probability model described as Markov chain,

$X \to Y \to Z$

where $X \perp Z | Y$,

We know that for Entropy,

$H(X,Y) \leq H(X) + H(Y)$

and also, $H(X,Y|Z) \leq H(X|Z) + H(Y|Z)$

$\Rightarrow H(X,Y|Z) - H(Y|Z) \leq H(X|Z)$

or $H(X|Z) \geq H(X,Y|Z) \longrightarrow (1)$

Since $X \perp Z|Y$, $H(X|Y,Z) = H(X|Y)$

So by putting it in equation (1) we get as

Figure 7:

Subtracting $H(X)$ from both sides

$$H(X|Z) \geq H(X|Y)$$

$$H(X) - H(X|Z) \leq H(X) - H(X|Y)$$

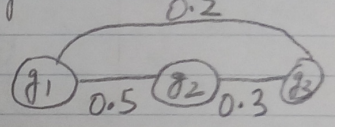
$$I(X;Z) \leq I(X;Z)$$

Hence proved.

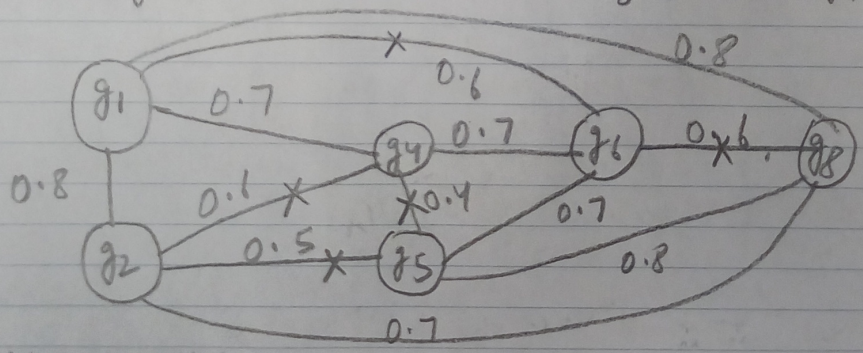
(b) Data Processing Inequality (DPI) is given by the following equation,

If $I(g_1, g_3) \leq \min\left(I(g_1, g_2), I(g_2, g_3)\right)$ for network then we can remove link between g_1 & g_3.

In the mutual information network, check the Mutual Information (I) for all the triangles in the network. After doing that, remove the edge with the smallest I. For example, in the above network, $I(g_1, g_3)$ is less than $I(g_1, g_2)$ & $I(g_2, g_3)$ so by using DPI we can remove edge between g_1 & g_3.



(c)



(i) Triangle $g_1 g_2 g_4$ - we can remove edge b/w g_2 & g_4

(ii) Triangle $g_2 g_5 g_8$ - we can remove edge b/w g_2 & g_5

(iii) Triangle $g_4 g_6 g_5$ - we can remove edge b/w g_4 & g_5

(iv) Triangle $g_1 g_4 g_6$ - we can remove edge b/w g_1 & g_6

(v) Triangle $g_5 g_6 g_8$ - we can remove edge b/w g_1 & g_8

Figure 8:

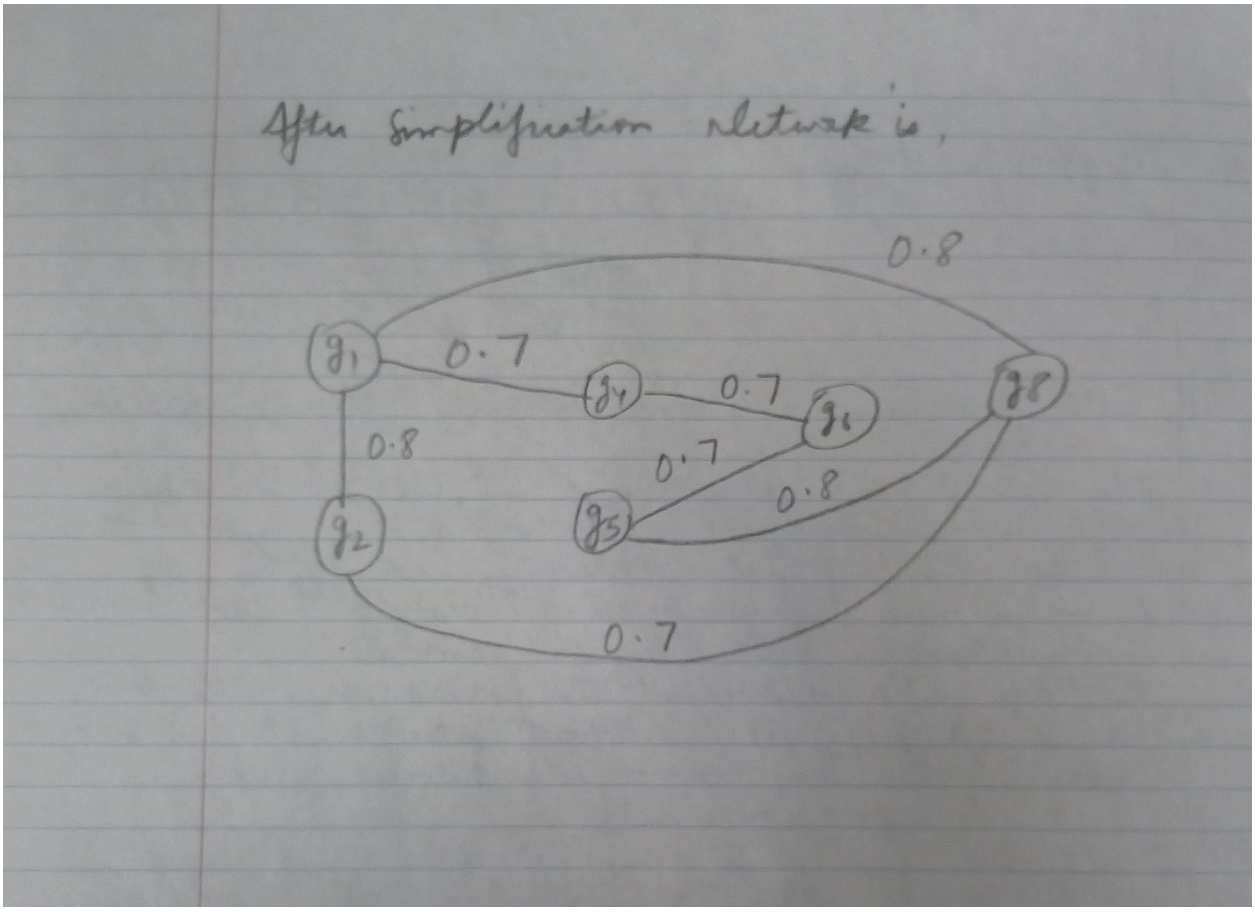


Figure 9: