STA 546 - Homework 3

Abbas Rizvi

April 8, 2016

Problem 1

The data sets "Les Miserables" and "Dolphins" networks were accessed from the Nexus repository using the `igraph::nexus.get` function in `R`.

```
library(igraph)
dolphins <- nexus.get("dolphins")
miserables <- nexus.get("miserables")
```

`dolphins` is an `igraph` object containing a dataset pertianing to an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand. `miserables` is also an `igraph` object containing a coappearance weighted network of characters in the French novel *Les Miserables* by Victor Hugo. Two characters are connected if they appear in the same scene. The weight of the connection is the number of common appearances.

Hierarchical random graphs (HRG) were used on `dolphins` and `miserables` to perform the tasks subsequent subsections of section 1.

1.1 Reveal Communities and Construct MCMC-based dendrogram

`igraphdemo('hrg')` was used as a tutorial/guide for subsection 1.1. The same pipeline was used for both the `dolphins` and `miserables` for this analysis.

The `dolphins` network was organized into a community structure using the `igraph::optimal.community` function. `optimal.community` calculates community structure of a graph by maximizing the modulatrity measure over all possible partitions. The communities for `dolphins` and `miserables` can be seen in Figure 1 and 3, respectively. We can see the constructed communitie did a good job identifying differing groups.

```
load("dolphins_optcom.Rdata") #optimal community
V(dolphins)$comm <- membership(dolphins.optcom) #membership gives vertices
plot(dolphins.optcom, dolphins) #revelation of communities
```

We now want to fit our `igraph` objects (`dolphins`/`miserables`) to a HRG. HRG conducts MCMC steps to perform the fitting until convergence. Dendrograms were constructed using the `hrg` objects and we can see that for both hrg-fitted `dolphin` and `miserables` datasets, that the dendrograms were able to represent the clusters in a much cleaner fashion than the community plots (Figure 2 and 4).

```
hrg <- hrg.fit(dolphins) #MCMC sampling
library(ape)
# plot it as dendrogram, looks better if the 'ape' package is installed
plot_dendrogram(hrg, mode="phylo", edge.color="black", cex=0.8)
```

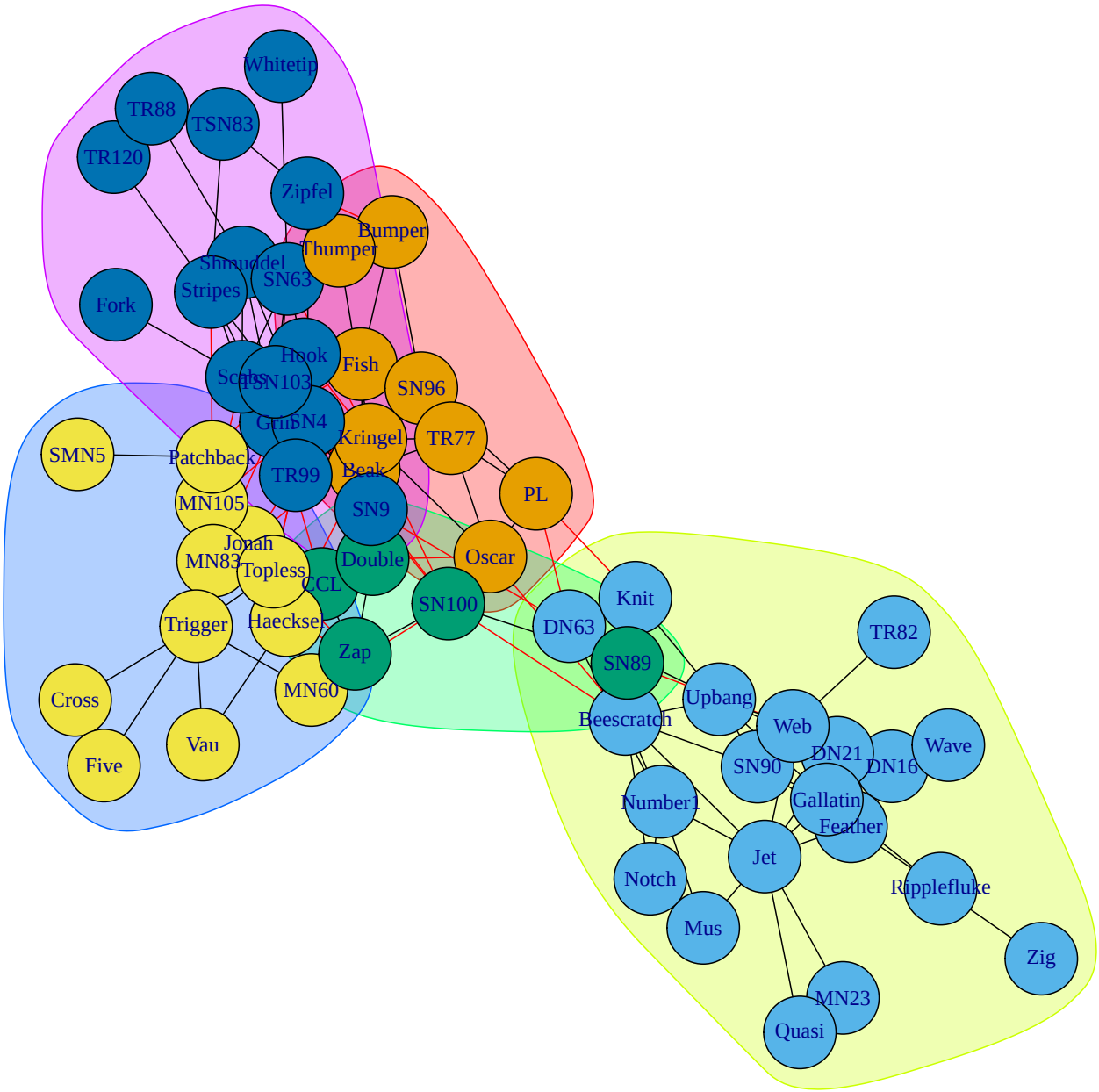

Figure 1: Clustered Dolphin Communities

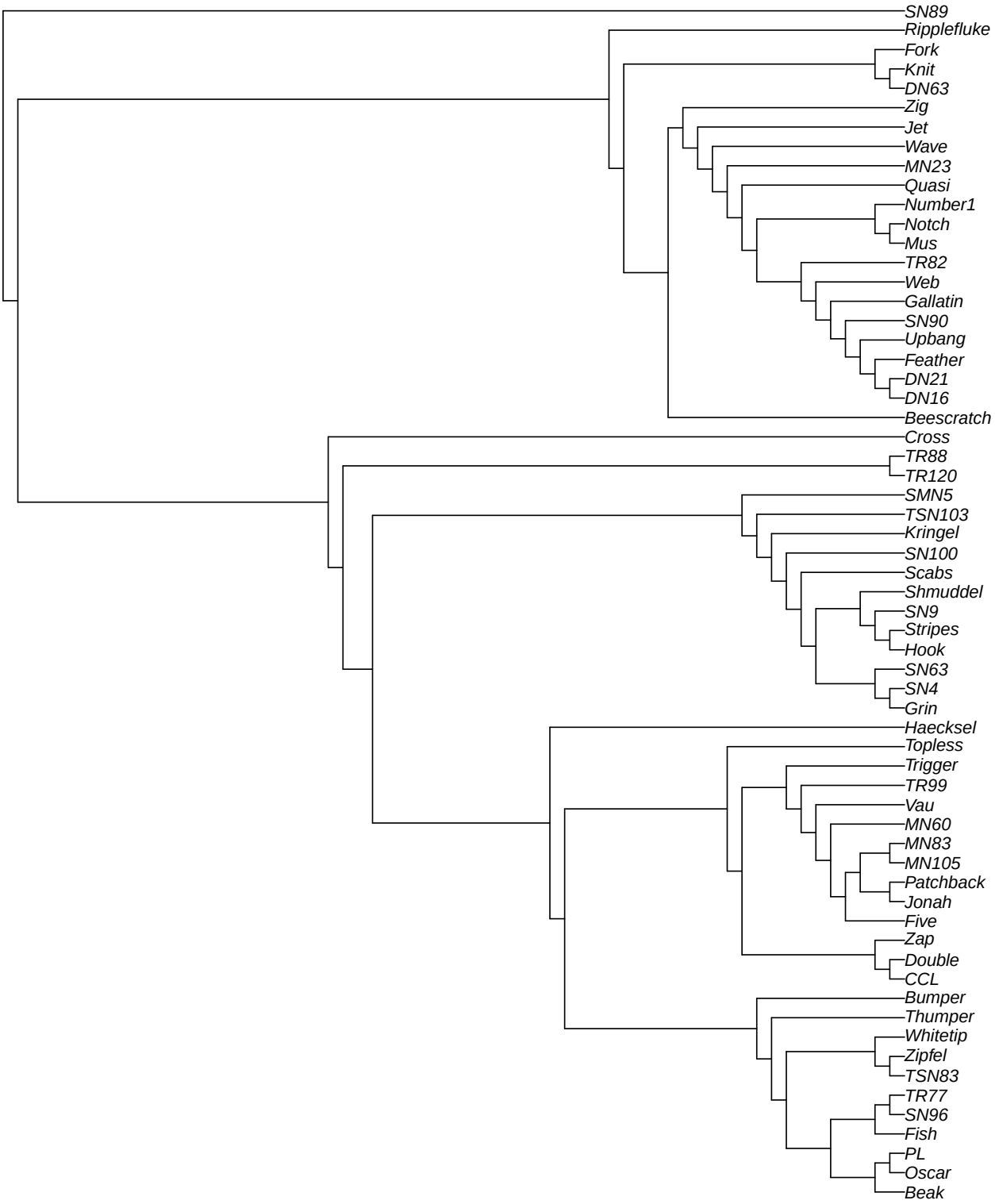

Figure 2: Dendrogram of MCMC-based Sampling of Dolphin Network

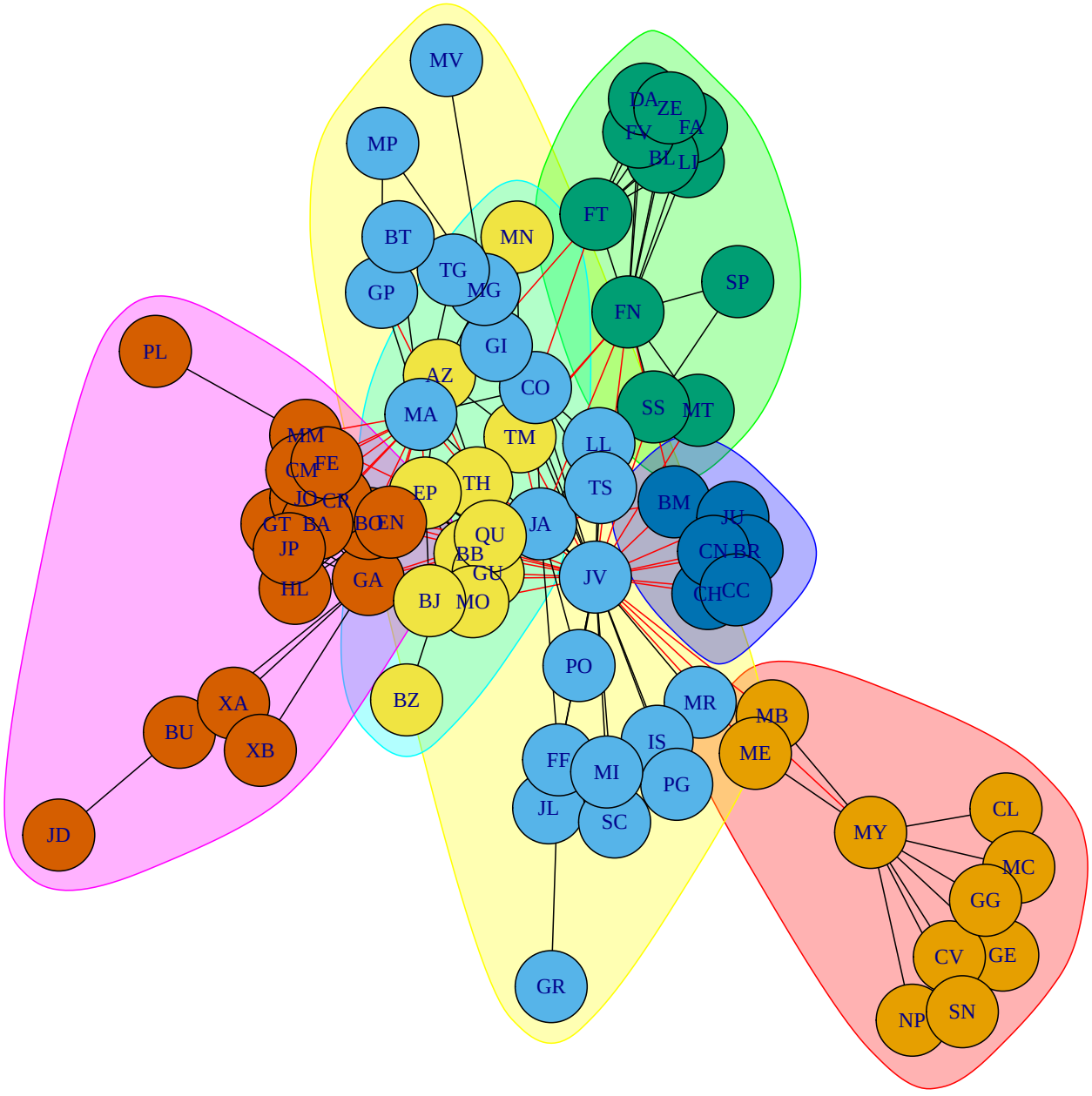

Figure 3: Clustered Les Miserables Communities

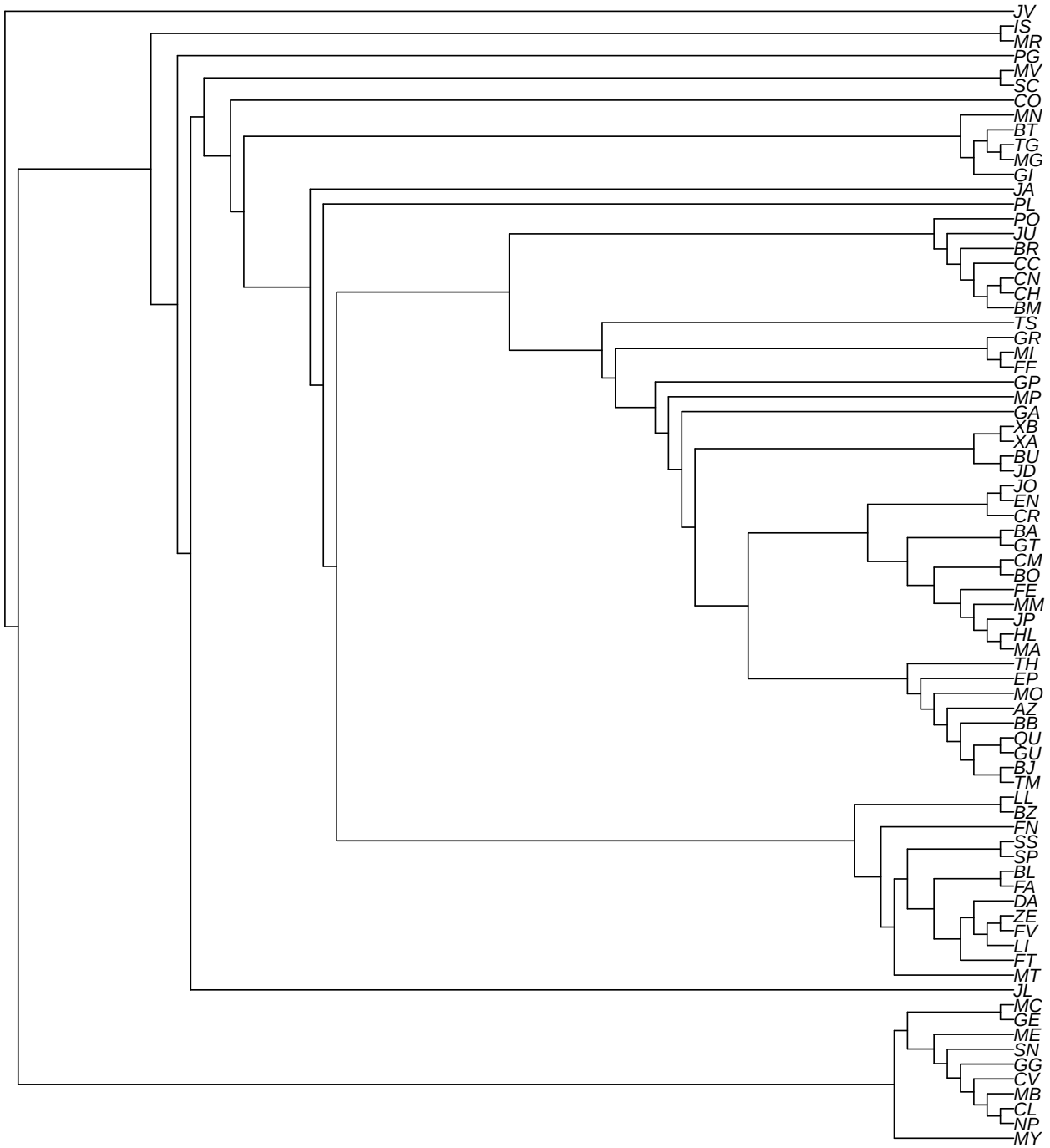

Figure 4: Dendrogram of MCMC-based Sampling of Les Miserables Network

1.2 Link-prediction of noisy versions of the Dolphin network dataset

For section 1.2 we were asked to focus on the `dolphins` network only. The helper function `link.prediction` was written such that a user-specified percentage of random sampling of edges was conducted and the sampled edges were subsequently deleted from the complete datasets, with the purpose of creating a 'noisy' dataset. We wanted to see how well we were able to predict the edges that were deleted from the complete dataset.

```
set.seed(1)
link.prediction(dolphins, 5)
```

```
##      v1       v2        predicted probability
## [1,] "DN63"   "PL"      "0.0148"
## [2,] "Double" "SN4"     "0.0768"
## [3,] "Bumper" "Thumper" "0.0469"
## [4,] "DN21"   "Web"     "0.299"
## [5,] "DN63"   "Number1" "0.0511"
## [6,] "DN16"   "Wave"    "0.0565"
## [7,] "Jet"    "Web"     "0.3853"
## [8,] "SN4"    "Topless" "0.0824"
```

`link.prediction` was set to a 5 percent sampling deletion (8 edges of the 159) (See attached `R` code hw3.R for `link.prediction`). The eight edges were able to be predicted by `igraph::hrg.predict`. `igraph::hrg.predict` uses HRG MCMC-sampling to predict missing edges from a network. This done using the optimum model – proportional to their likelihood. With the 5% deletion, the edges were predicted, but with low probabilities for the most part, only two have >10% probability of being predicted.

1.3 Repeating with 2.2 with more deletions (15% and 40%)

```
link.prediction(dolphins, 15)
```

```
##       v1           v2         predicted probability
##  [1,] "DN63"       "PL"       "0.015"
##  [2,] "Double"     "SN4"      "0.0594"
##  [3,] "Bumper"     "Thumper"  "0.0429"
##  [4,] "DN21"       "Web"      "0.1553"
##  [5,] "DN63"       "Number1"  "0.0492"
##  [6,] "DN16"       "Wave"     "0.0571"
##  [7,] "Jet"        "Web"      "0.3896"
##  [8,] "SN4"        "Topless"  "0.0786"
##  [9,] "Haecksel"   "Topless"  "0.1972"
## [10,] "CCL"        "Grin"     "0.0465"
## [11,] "Beescratch" "Number1"  "0.1182"
## [12,] "Jonah"      "MN83"     "0.2065"
## [13,] "Double"     "Zap"      "0.0816"
## [14,] "Kringel"    "SN100"    "0.1076"
## [15,] "Hook"       "TR99"     "0.089"
## [16,] "DN63"       "SN9"      "0.003"
## [17,] "Beak"       "TR77"     "0.1385"
## [18,] "DN16"       "Web"      "0.1223"
## [19,] "Scabs"      "Shmuddel" "0.2525"
## [20,] "Shmuddel"   "TR88"     "0.0207"
```

```
## [21,] "Stripes"    "TSN83"    "0.0248"
## [22,] "Mus"        "Notch"    "0.1216"
## [23,] "Bumper"     "Zipfel"   "0.0241"
## [24,] "Beescratch" "Knit"     "0.08"
```

```
link.prediction(dolphins, 40)
```

```
##       v1            v2            predicted probability
##  [1,] "DN63"        "PL"          "0.0256"
##  [2,] "Double"      "SN4"         "0.0411"
##  [3,] "Bumper"      "Thumper"     "0.0208"
##  [4,] "DN21"        "Web"         "0.2015"
##  [5,] "DN63"        "Number1"     "0.0256"
##  [6,] "DN16"        "Wave"        "0.0055"
##  [7,] "Jet"         "Web"         "0.2516"
##  [8,] "SN4"         "Topless"     "0.0765"
##  [9,] "Haecksel"    "Topless"     "0.0655"
## [10,] "CCL"         "Grin"        "0.1153"
## [11,] "Beescratch"  "Number1"     "0.033"
## [12,] "Jonah"       "MN83"        "0.1388"
## [13,] "Double"      "Zap"         "0.0813"
## [14,] "Kringel"     "SN100"       "0.0595"
## [15,] "Hook"        "TR99"        "0.0162"
## [16,] "DN63"        "SN9"         "0.0064"
## [17,] "Beak"        "TR77"        "0.0597"
## [18,] "DN16"        "Web"         "0.1075"
## [19,] "Scabs"       "Shmuddel"    "0.0261"
## [20,] "Shmuddel"    "TR88"        "0.0125"
## [21,] "Stripes"     "TSN83"       "0.0167"
## [22,] "Mus"         "Notch"       "0.0794"
## [23,] "Bumper"      "Zipfel"      "0.0189"
## [24,] "Beescratch"  "Knit"        "0.0873"
## [25,] "Double"      "Oscar"       "0.0284"
## [26,] "MN105"       "Scabs"       "0.0243"
## [27,] "DN16"        "Feather"     "0.0691"
## [28,] "Hook"        "Scabs"       "0.0456"
## [29,] "Scabs"       "TR99"        "0.028"
## [30,] "Oscar"       "PL"          "0.1007"
## [31,] "Scabs"       "SN4"         "0.0581"
## [32,] "SN4"         "SN9"         "0.064"
## [33,] "SN9"         "TSN103"      "0.0474"
## [34,] "Jet"         "MN23"        "0.0211"
## [35,] "Fish"        "TR77"        "0.0384"
## [36,] "Fish"        "SN96"        "0.0621"
## [37,] "MN60"        "Topless"     "0.1183"
## [38,] "DN21"        "Jet"         "0.1658"
## [39,] "SN4"         "Stripes"     "0.0556"
## [40,] "Grin"        "Scabs"       "0.2628"
## [41,] "Patchback"   "Trigger"     "0.2718"
## [42,] "SN63"        "TSN103"      "0.0444"
## [43,] "Shmuddel"    "Thumper"     "0.0163"
## [44,] "SN100"       "SN4"         "0.0415"
## [45,] "Hook"        "SN4"         "0.0517"
## [46,] "Haecksel"    "Vau"         "0.0478"
```

```
## [47,] "DN21"        "Feather"     "0.133"
## [48,] "Ripplefluke" "Zig"         "0.0056"
## [49,] "Bumper"      "SN96"        "0.0447"
## [50,] "Five"        "Trigger"     "0.0068"
## [51,] "Grin"        "Shmuddel"    "0.136"
## [52,] "Double"      "Topless"     "0.0151"
## [53,] "Feather"     "Ripplefluke" "0.0733"
## [54,] "Haecksel"    "MN83"        "0.0616"
## [55,] "Feather"     "Gallatin"    "0.3177"
## [56,] "Beak"        "Haecksel"    "0.0646"
## [57,] "Jet"         "Number1"     "0.0731"
## [58,] "DN21"        "Wave"        "0.006"
## [59,] "Hook"        "SN63"        "0.0455"
## [60,] "MN105"       "Patchback"   "0.2599"
## [61,] "Kringel"     "Thumper"     "0.1147"
## [62,] "Beescratch"  "Notch"       "0.0234"
## [63,] "SN100"       "Zap"         "0.0368"
## [64,] "Topless"     "Zap"         "0.0207"
```

`link.prediction` was used to randomly delete 15% and 40% of the `dolphin` graph, respectively. It seems as the edges can always be predicted regardless of the percent deletion, however, the probability to be predicted remains relatively low for the most part, but as sample deletion increases, there are more predicted values of >10% than in smaller deletions.

Problem 2

Exercise 3.11 from Koller et al. was adopted for this problem. The following Bayesian Network (`B`) by Judea Pearl was considered:

The original Bayesian Network `B` (Figure 5) was marginalized over the `alarm` node. Subsequently, a Bayesian Network `B'` was constructed such that it was the minimal I-map for the marginal distribution P_B(B, E, T, N, J, M). The minimal I-map for the marginalized network Bayesian Network `B'` was constructed such that the independencies from `B` were preserved (Figure 6) . We utilized the `ggm::inducedDAG` function in `R` to construct this I-map.

Problem 3

A toy directed acyclic graph (DAG) was assembled in order to determine d-separation between nodes (Figure 7).

Determine if the following statements are "`TRUE`" or "`FALSE`" based on the DAG:

The helper function `dCon` was written for questions pertaining to 'd-connected' nodes

```
dCon <- function(dSep){
        if(dSep==FALSE){
                as.logical(dSep+1)
        }
        else{
                as.logical(dSep*0)
        }
}
```

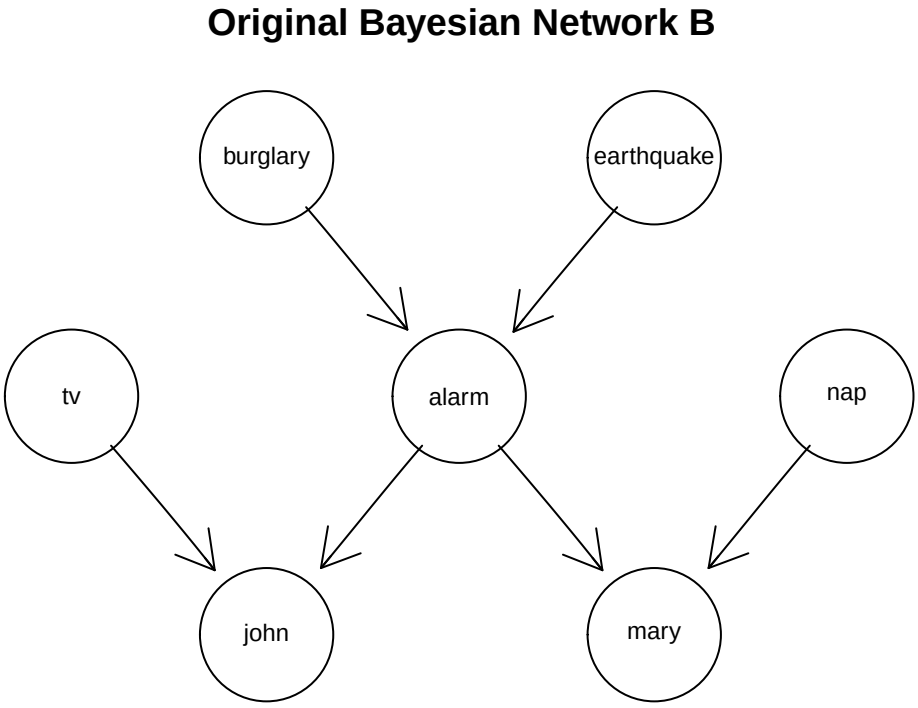

Figure 5: Original Bayesian Network B

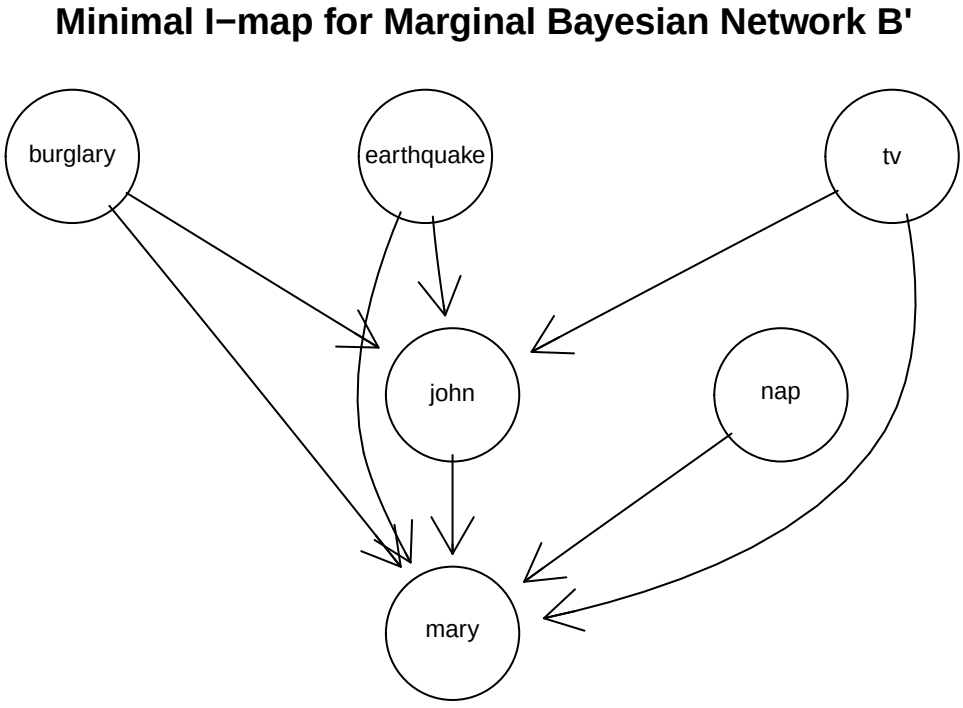

Figure 6: Bayesian Network B'

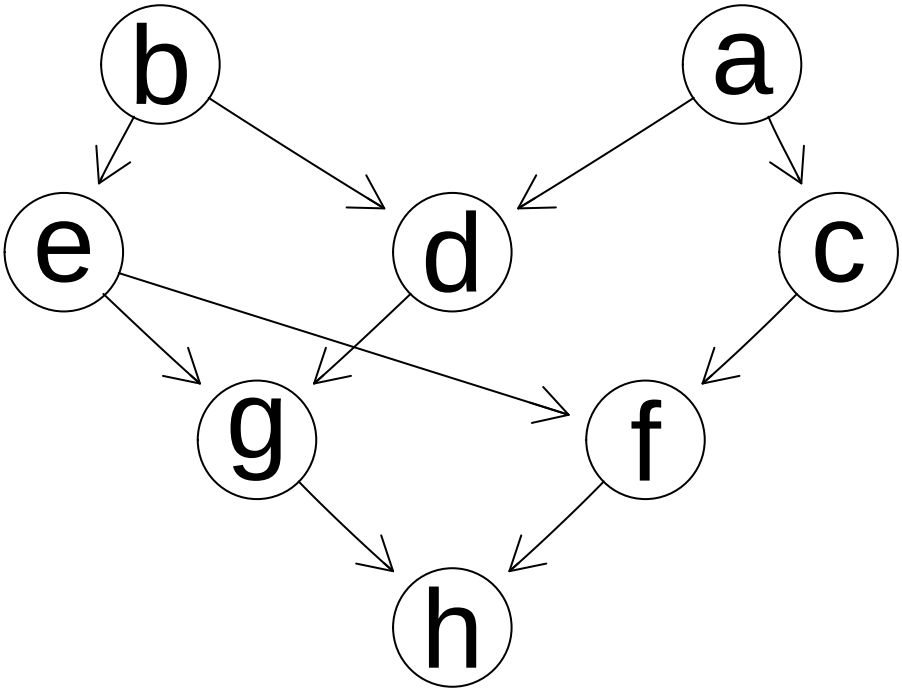

Figure 7: Toy DAG Model

A) C and G are d-separated

```
dSep(as(dag.list, "matrix"), "c", "g", NULL)
```

```
## [1] FALSE
```

B) C and E are d-separated

```
dSep(as(dag.list, "matrix"), "c", "e", NULL)
```

```
## [1] TRUE
```

C) C and E are d-connected given evience about G

```
dCon(dSep(as(dag.list, "matrix"), "c", "e", "g"))
```

```
## [1] TRUE
```

D) A and G are d-connected given evidence about D and E

```
dCon(dSep(as(dag.list, "matrix"), "a", "g", c("d","e")))
```

```
## [1] FALSE
```

E) A and G are d-connected given evidence on D

```
dCon(dSep(as(dag.list, "matrix"), "a", "g", "d"))
```

```
## [1] TRUE
```