DATA MINING 2 ASSIGNMENT- STA 6704

BAYSIAN NETWORKS

NAME: SUSHMITHA MANI

UCF ID: 5016977

ASSIGNMENT: BAYSIAN NETWORKS

DATA:

The dataset used in this assignment has been changed. The dataset chosen for this assignment is coronary dataset from the data package available in R. This dataset tends to work much better and efficient with bnlearn compared to the other available datasets. The dataset is initially in the matrix format which is then converted into the dataframe format. The dataset consists of 6 columns which includes: Smoking, M. Work, P. Work, Pressure, Proteins, Family. This has been used further for model building and scoring and further analysis in this assignment. The necessary packages for this assignment are installed.

PART 1: BUILDING THE MODELS

To build the models, the fastest models are used for each category as follows. The models and their respective graphs are attached:

Score based Algorithm:

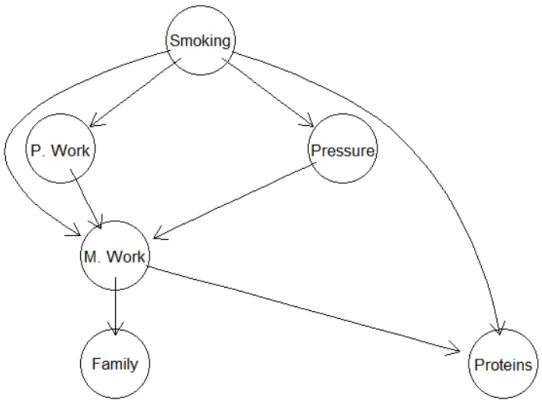
The fastest score based algorithm used is hc(). The following are the results produced for the hc() algorithm:

```
> model_hc

  Bayesian network learned via Score-based methods

  model:
   [Smoking][P. Work|Smoking][Pressure|Smoking][M. Work|Smoking:P. Work:Pressure]
   [Proteins|Smoking:M. Work][Family|M. Work]
  nodes:                                 6
  arcs:                                  8
    undirected arcs:                     0
    directed arcs:                       8
  average markov blanket size:           3.00
  average neighbourhood size:            2.67
  average branching factor:              1.33

  learning algorithm:                    Hill-Climbing
  score:                                 BIC (disc.)
  penalization coefficient:              3.759032
  tests used in the learning procedure:  65
  optimized:                             TRUE
```



HC Visualization

Hybrid Algorithm:

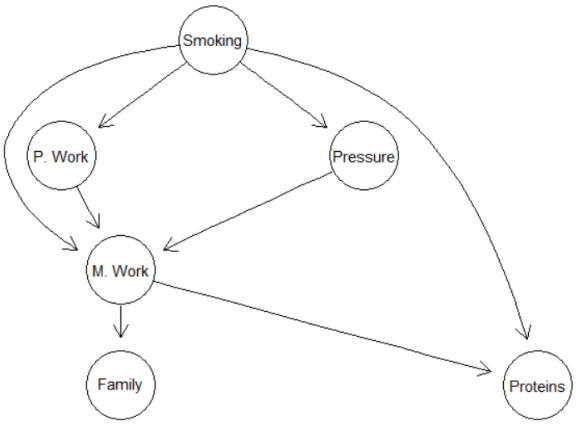
The fastest hybrid algorithm used is h2pc(). The results are as follows:

```
> model_h2pc

  Bayesian network learned via Hybrid methods

  model:
   [Smoking][P. Work|Smoking][Pressure|Smoking][M. Work|Smoking:P. Work:Pressure]
   [Proteins|Smoking:M. Work][Family|M. Work]
  nodes:                                6
  arcs:                                 8
    undirected arcs:                    0
    directed arcs:                      8
  average markov blanket size:          3.00
  average neighbourhood size:           2.67
  average branching factor:             1.33

  learning algorithm:                   Hybrid^2 Parent Children
  constraint-based method:              Hybrid Parents and Children
  conditional independence test:        Mutual Information (disc.)
  score-based method:                   Hill-Climbing
  score:                                BIC (disc.)
  alpha threshold:                      0.05
  penalization coefficient:             3.759032
  tests used in the learning procedure: 411
  optimized:                            TRUE
```



h2pc() visualization

Constraint based Algorithm:

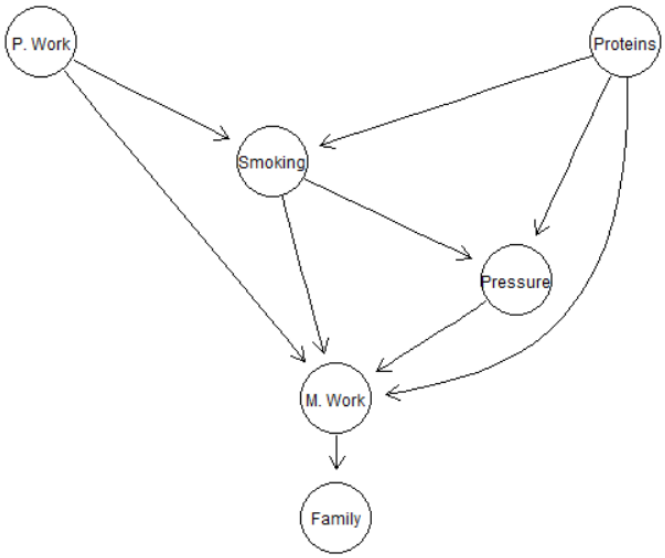
The fastest constraint based algorithm used is iamb(). The results are as follows:

```
> model_iamb

  Bayesian network learned via Constraint-based methods

  model:
   [P. Work][Proteins][Smoking|P. Work:Proteins][Pressure|Smoking:Proteins]
   [M. Work|Smoking:P. Work:Pressure:Proteins][Family|M. Work]
  nodes:                                6
  arcs:                                 9
    undirected arcs:                    0
    directed arcs:                      9
  average markov blanket size:          3.67
  average neighbourhood size:           3.00
  average branching factor:             1.50

  learning algorithm:                   IAMB-FDR
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure: 340
```



Iamb() visualization

Local Discovery Algorithm:

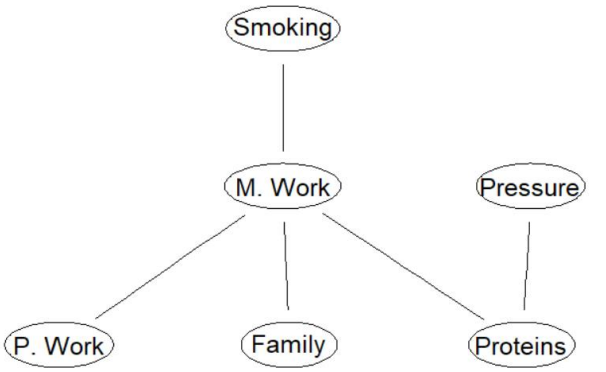
The fastest local discovery algorithm used is arcane(). The graph produced in this algorithm is undirected and it was necessary to make the graphs directed in order to produce the scores. The results are as follows:

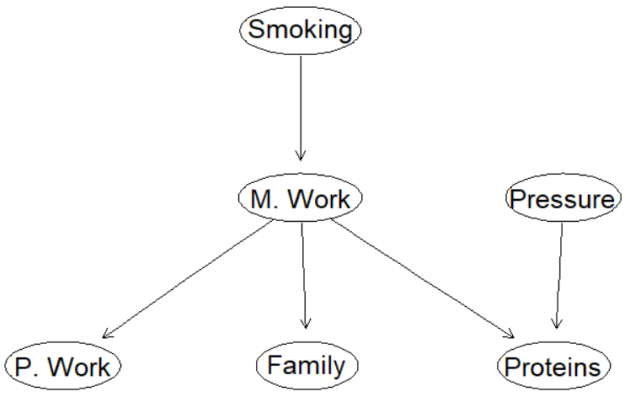
```
> model_aracne

  Bayesian network learned via Pairwise Mutual Information methods

  model:
    [undirected graph]
  nodes:                                6
  arcs:                                 5
    undirected arcs:                    5
    directed arcs:                      0
  average markov blanket size:          1.67
  average neighbourhood size:           1.67
  average branching factor:             0.00

  learning algorithm:                   ARACNE
  mutual information estimator:         Maximum Likelihood (disc.)
  tests used in the learning procedure: 15
```





PART 2 : SCORING OF THE MODELS

The scoring for the different algorithms are calculated and generated as follows. The scoring is by default “bic”, unless specified explicitly. The scoring of the different algorithm are tabulated in the end of this section.

AIC() scoring:

```
> bnlearn::score(model_hc, data_coronary, type="aic")
[1] -6668.589
> bnlearn::score(model_h2pc, data_coronary, type="aic")
[1] -6668.589
> bnlearn::score(model_iamb, data_coronary, type="aic")
[1] -6645.867
> bnlearn::score(model_aracne, data_coronary, type="aic")
[1] -6725.402
```

BIC() scoring:

```
> bnlearn::score(model_hc, data_coronary, type="bic")
[1] -6721.011
> bnlearn::score(model_h2pc, data_coronary, type="bic")
[1] -6721.011
> bnlearn::score(model_iamb, data_coronary, type="bic")
[1] -6723.12
> bnlearn::score(model_aracne, data_coronary, type="bic")
[1] -6758.51
```

Loglik() scoring:

```
> bnlearn::score(model_hc, data_coronary, type="loglik")
[1] -6649.589
> bnlearn::score(model_h2pc, data_coronary, type="loglik")
[1] -6649.589
> bnlearn::score(model_iamb, data_coronary, type="loglik")
[1] -6617.867
> bnlearn::score(model_aracne, data_coronary, type="loglik")
[1] -6713.402
```

TABULATED SCORES:

Algorithm	**AIC()**	**BIC()**	**Loglik()**
score based algorithm – hc()	-6668.589	-6721.011	-6649.589
Hybrid Algorithm – h2pc()	-6668.589	-6721.011	-6649.589
Constraint based algorithm – iamb()	-6645.867	-6723.12	-6617.867
Local Discovery algorithm – Aracne()	-6725.402	-6758.51	-6713.402

From the above tabulated column, it is found that AIC() and BIC() are almost similar to each other. But when these two are compared, BIC() is a better performing algorithm for scoring than AIC(). BIC() tends to perform better in situations of high dimensionality data in real time when compared to that of AIC(). Hence the scoring under BIC() is sorted and considered. Below the table is formulated to obtain the best algorithm.

Algorithm:	**BIC()**
score based algorithm – hc()	-6721.011
Hybrid Algorithm – h2pc()	-6721.011
Constraint based algorithm – iamb()	-6723.12
Local Discovery algorithm – Aracne()	-6758.51

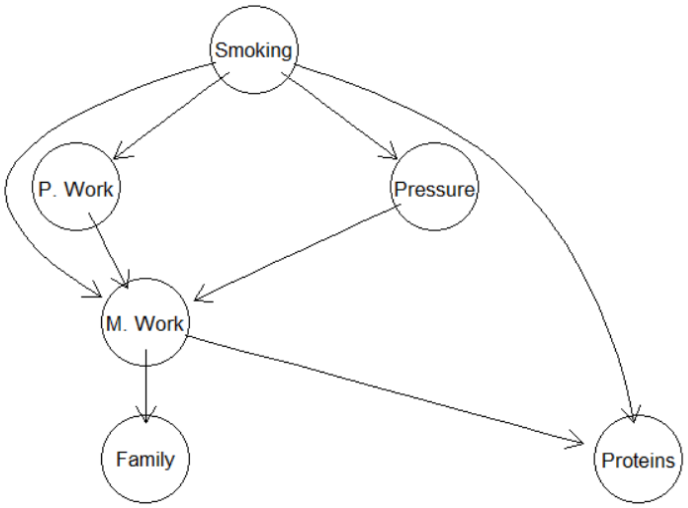
From the scoring comparison under BIC(), it is found that the score based algorithm (hc) and hybrid algorithm (h2pc) are similar to each. But when compared, it is found that hc is a better algorithm for realistic approach. Hence score based algorithm is the best algorithm.

PART 3

CONSIDERING HC ALGORITHM AS THE FINAL MODEL SELECTED:

The hc() algorithm which is part of the score model algorithm is the fastest performing algorithm. This algorithm tends to perform better and faster compared to the algorithms. It is easy to implement in real time situations. This algorithm is dependent on the values obtained by the next values in the dataset and runs until the point where there is no more higher values obtained. The hc algorithm is efficient for baysian networks from the model's quality and computational perspective. The visualization of the hc algorithm is as follows:

FINAL MODEL SELECTED AND VISUALIZATION



PART 4 – PREDICTION OF THE TARGET VARIABLE

The target variable used for the prediction of the values is the smoking column, which tends to indicate if the person smokes or not. This target variable is predicted with the other variables present in the dataset. The following are the predictions observed for the target variable.

```
> pred = predict(fit, "Smoking",data_coronary)
> pred
   [1] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [10] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [19] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [28] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [37] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [46] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [55] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [64] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [73] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
  [82] 1.699387 1.699387 1.699387 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
  [91] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [100] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [109] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [118] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [127] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [136] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [145] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [154] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [163] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [172] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [181] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380
 [190] 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.063380 1.699387 1.699387
 [199] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
 [208] 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387 1.699387
```

The accuracy of this model is obtained as follows:

```
> accuracy(object = pred, x =data_coronary$Smoking)
                     ME      RMSE       MAE       MPE     MAPE
Test set 1.138109e-17 0.4724202 0.4463617 -11.15904 33.47713
```

The various accuracy models are obtained. From the obtained accuracy results, we observe that the value produced by RMSE(Root mean square error) is the value of error rate by the square root of MSE. Lower the value of RMSE is better indication of the model fitting accurately, and the obtained value is 0.4724 which indicates that the model fits well.

As conclusion, Hill Climbing algorithms are very suitable to work on projects which have very high dimensional data. The hill climbing algorithm is very beneficial in the case of optimization problems in several high level projects. This model works efficiently and reduces the time consumption which is an important factor at projects in organizations. It is very effective in work spaces such as job scheduling, designing of circuits, automatic programming, management of portfolios and so on. The errors produced in this algorithm is less which can be tested using the accuracy. The accuracy reflects the efficiency of the model, for which the HC algorithm stands out when compared to that of the others.