Bayesian Network's

Name:- Aravindan K S (1910110080)
Sanjana Thakur(2010110562)

Problem Statement:

We want to investigate the usage patterns of different means of transport, with a focus on cars and trains.

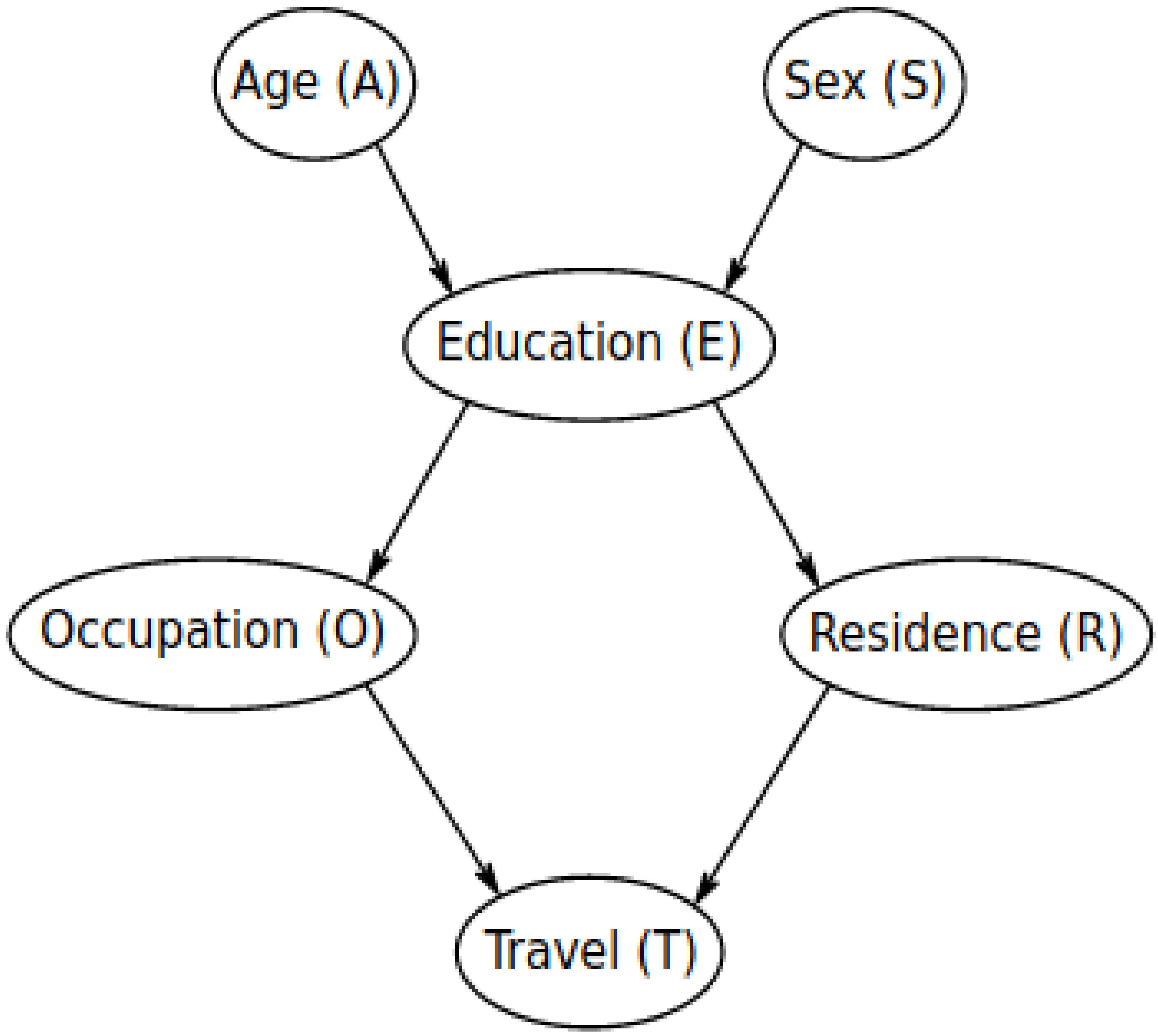
Here we will use **approximate inference method** to find the most likely explanation for Sex and Transport just by looking for the combination of their states that has the highest probability.

Six discrete variables to be used with there labels are:

1) Age (**A**): (Young, Adult, Old)
2) Sex (**S**): (Male, Female)
3) Education (**E**): (high, uni)
4) Occupation (**O**) : (emp, self)
5) Residence (**R**) : (small, big)
6) Travel (**T**) : (car, train ,others)

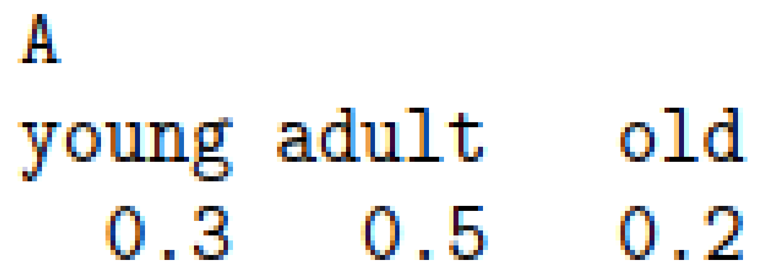
Graphical Representation

Directed Acyclic Graph (DAG): shows the dependence relationship between the variables

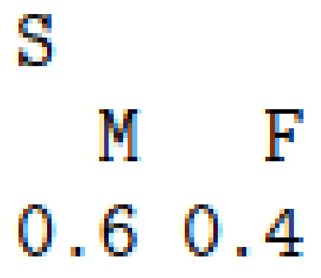


Probabilistic Representation:

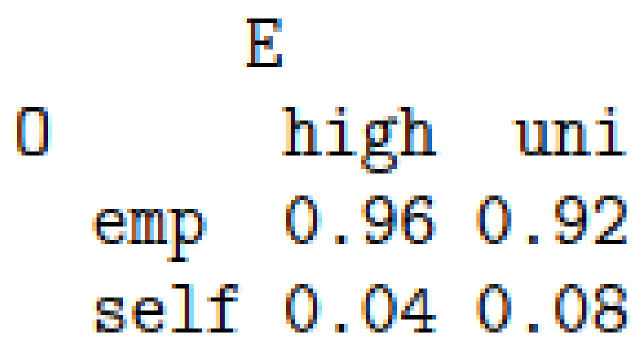
To complete the BN modelling survey, we will now specify a joint probability distribution over these variables



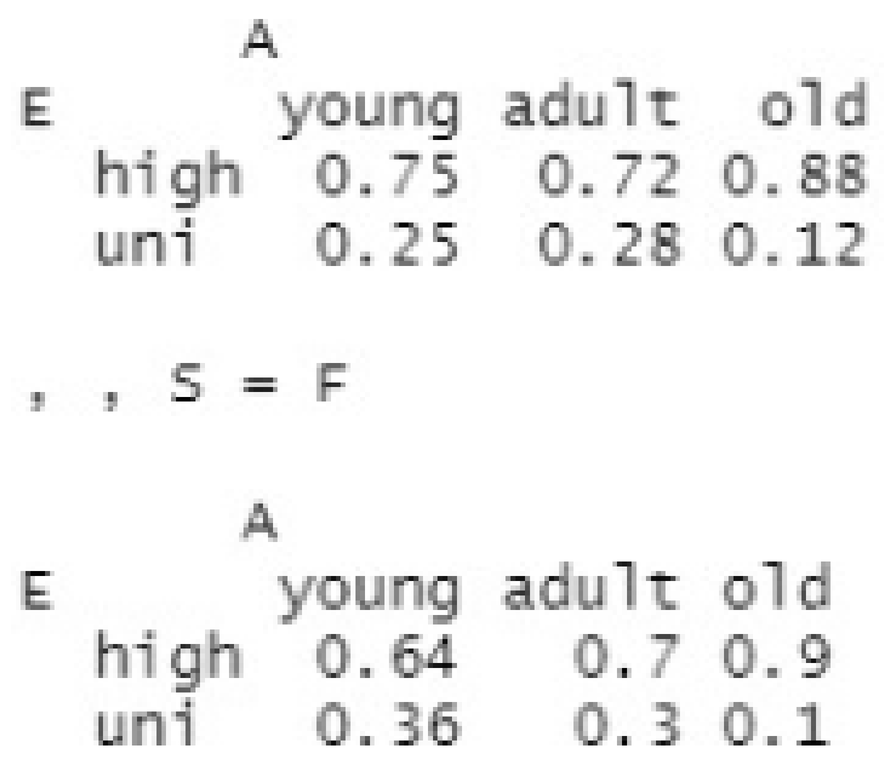
```
A
young adult   old
  0.3   0.5   0.2
```



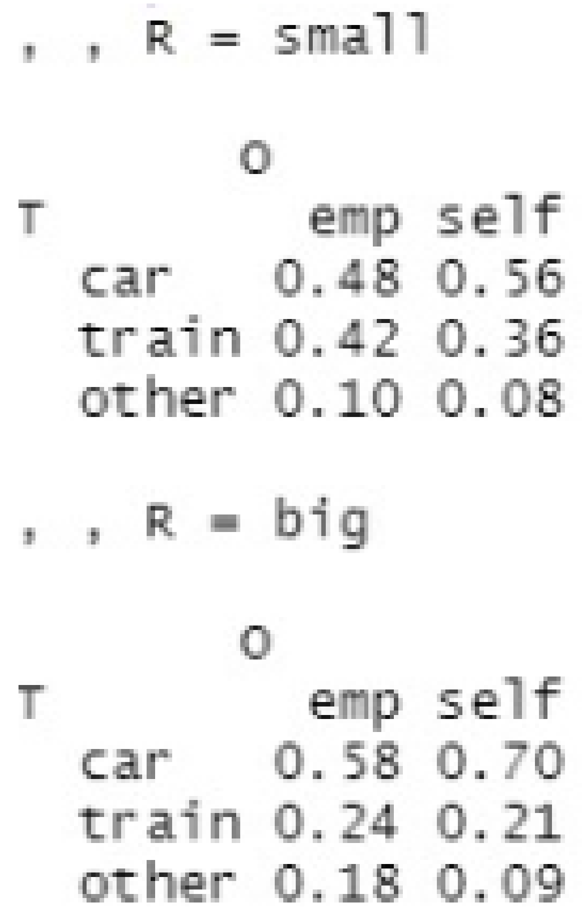
```
S
  M   F
0.6 0.4
```



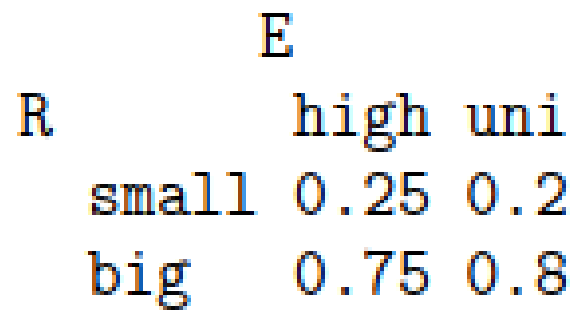
```
        E
O        high  uni
  emp    0.96 0.92
  self   0.04 0.08
```



```
        A
E        young adult  old
  high   0.75  0.72 0.88
  uni    0.25  0.28 0.12

, , S = F

        A
E        young adult old
  high   0.64   0.7 0.9
  uni    0.36   0.3 0.1
```

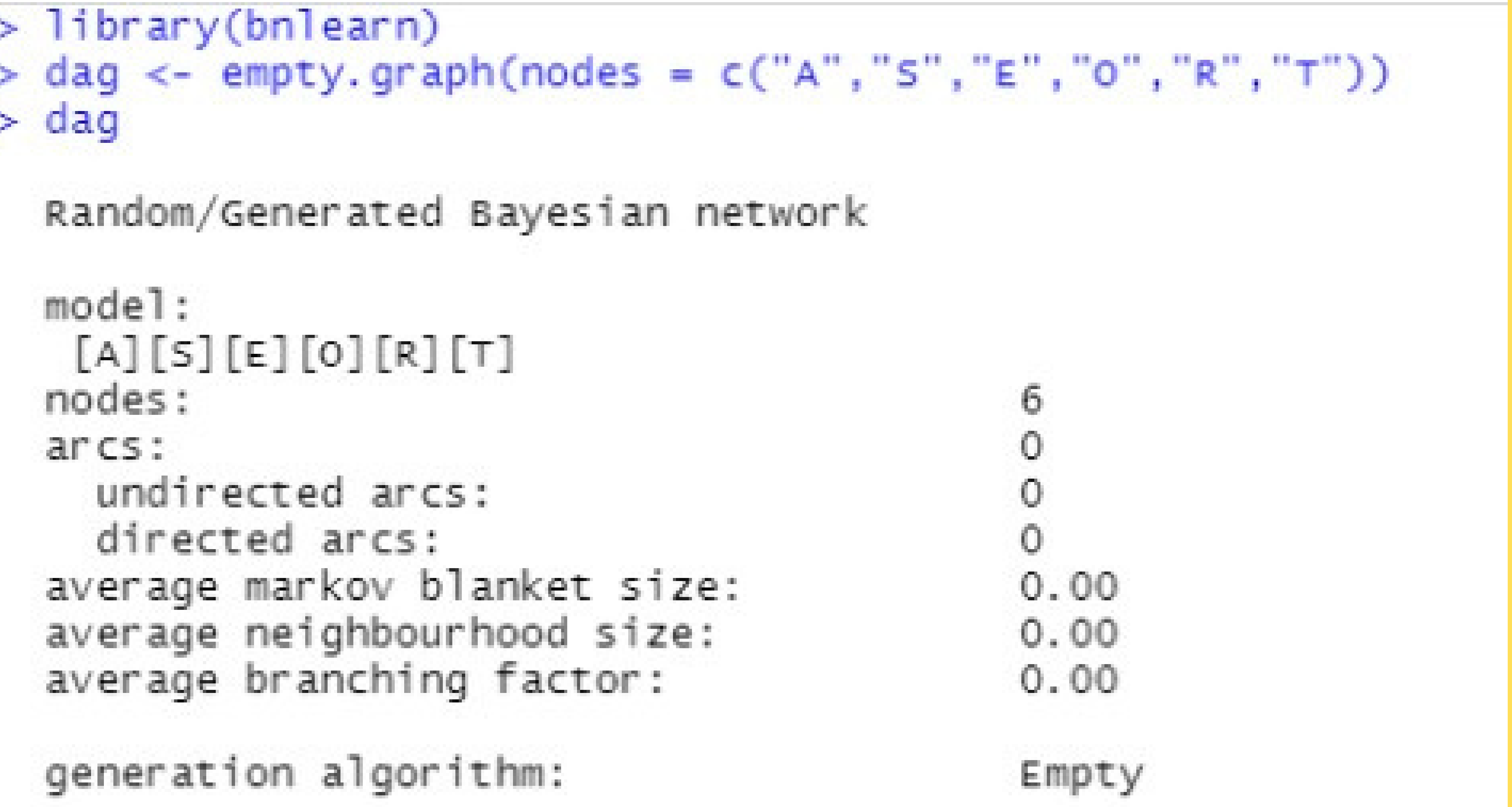


```
, , R = small

       O
T        emp self
  car   0.48 0.56
  train 0.42 0.36
  other 0.10 0.08

, , R = big

       O
T        emp self
  car   0.58 0.70
  train 0.24 0.21
  other 0.18 0.09
```



```
        E
R        high uni
  small  0.25 0.2
  big    0.75 0.8
```

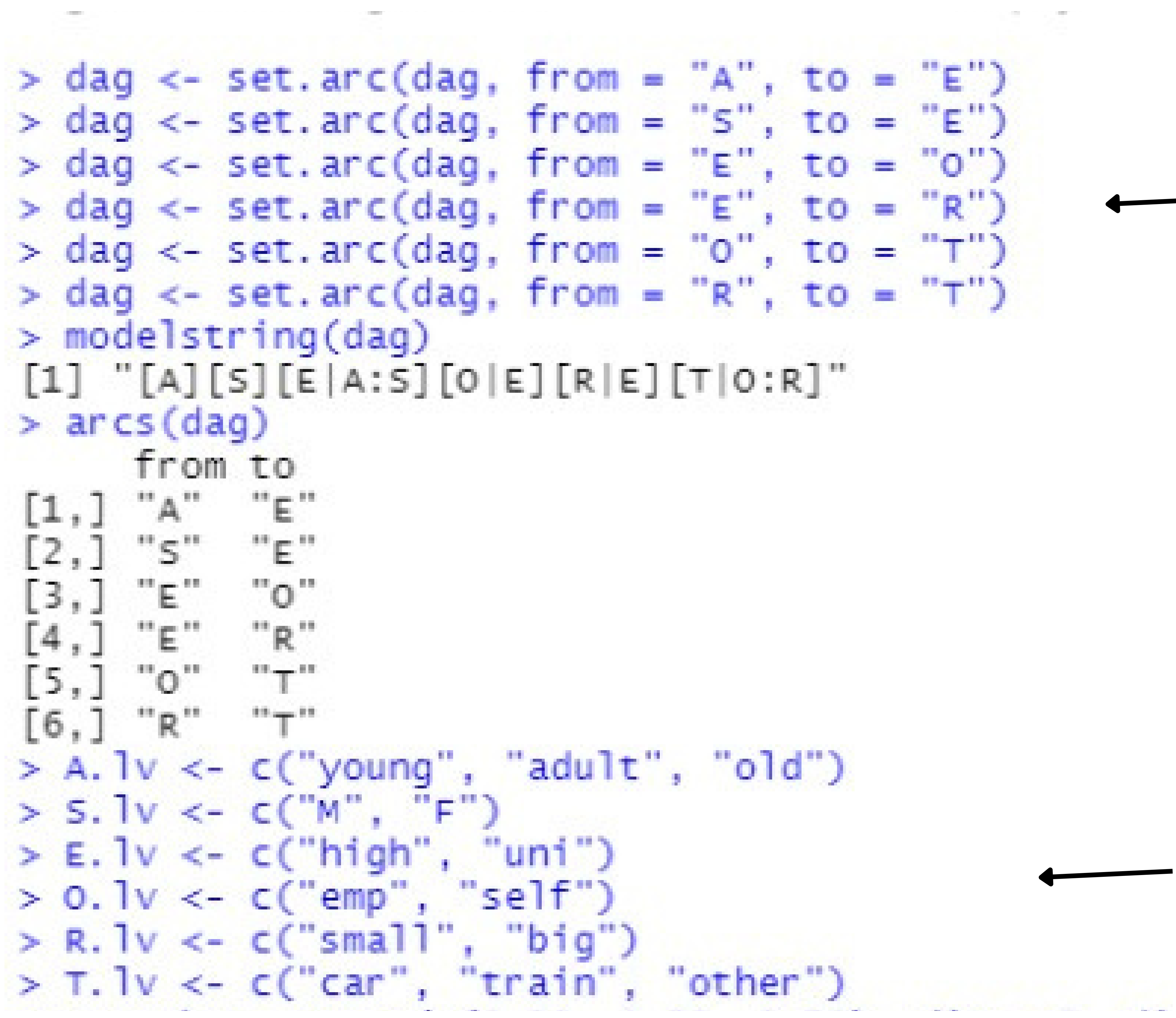
Initial Code:



```
> library(bnlearn)
> dag <- empty.graph(nodes = c("A","S","E","O","R","T"))
> dag

  Random/Generated Bayesian network

  model:
   [A][S][E][O][R][T]
  nodes:                                 6
  arcs:                                  0
    undirected arcs:                     0
    directed arcs:                       0
  average markov blanket size:           0.00
  average neighbourhood size:            0.00
  average branching factor:              0.00

  generation algorithm:                  Empty
```

bnlearn:

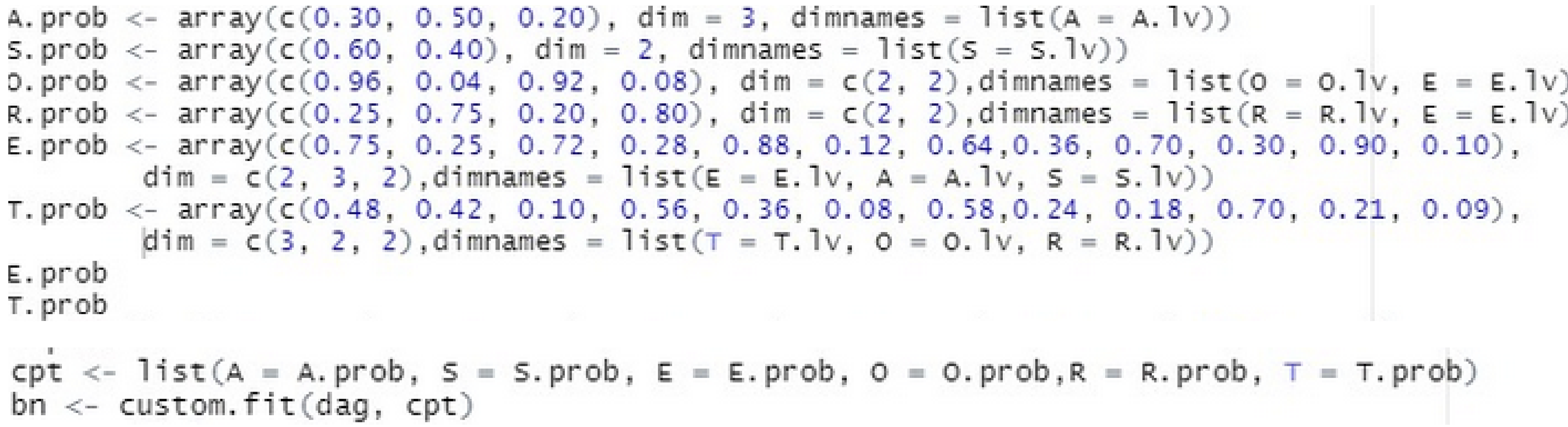
To create and manipulate DAGs in the context of BNs, estimate their parameters and perform some useful inference.



```
> dag <- set.arc(dag, from = "A", to = "E")
> dag <- set.arc(dag, from = "S", to = "E")
> dag <- set.arc(dag, from = "E", to = "O")
> dag <- set.arc(dag, from = "E", to = "R")
> dag <- set.arc(dag, from = "O", to = "T")
> dag <- set.arc(dag, from = "R", to = "T")
> modelstring(dag)
[1] "[A][S][E|A:S][O|E][R|E][T|O:R]"
> arcs(dag)
     from to
[1,] "A"  "E"
[2,] "S"  "E"
[3,] "E"  "O"
[4,] "E"  "R"
[5,] "O"  "T"
[6,] "R"  "T"
> A.lv <- c("young", "adult", "old")
> S.lv <- c("M", "F")
> E.lv <- c("high", "uni")
> O.lv <- c("emp", "self")
> R.lv <- c("small", "big")
> T.lv <- c("car", "train", "other")
```

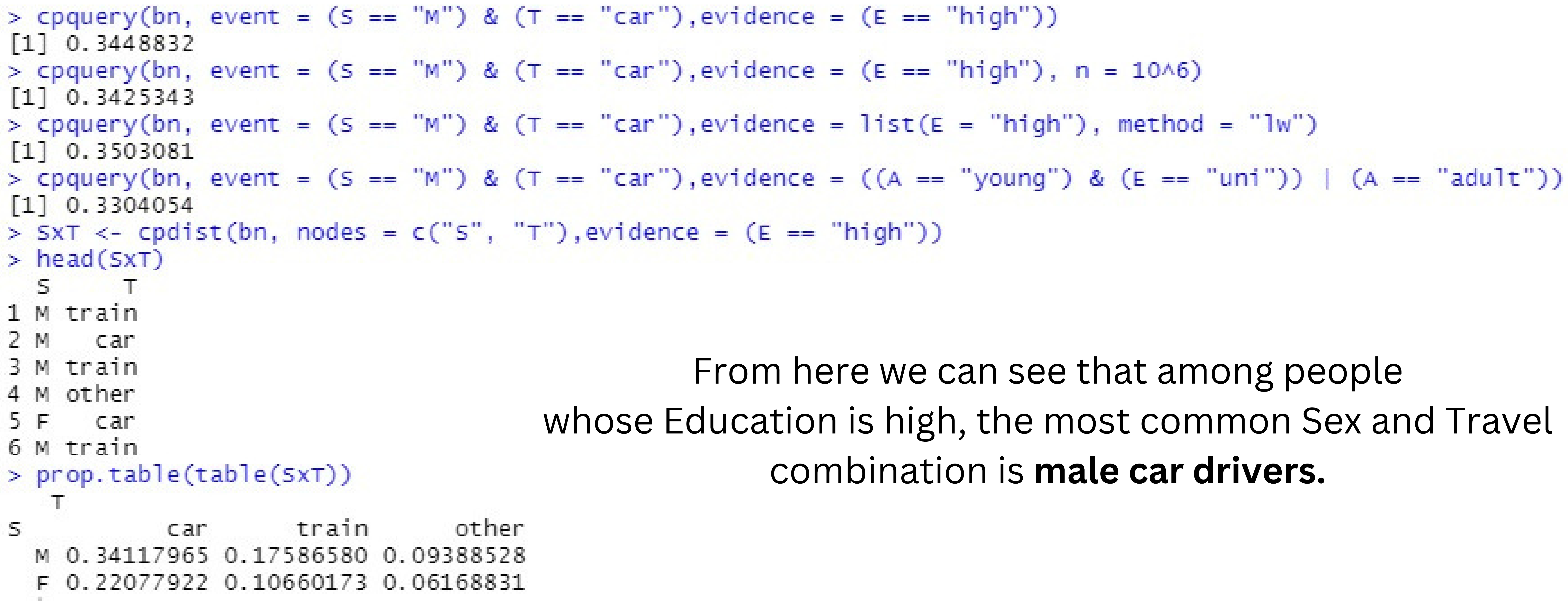
Here we added the arcs that encode the direct dependencies between the variables in the survey

So to give joint probabilist distribution to our variable first we will define levels



```
A.prob <- array(c(0.30, 0.50, 0.20), dim = 3, dimnames = list(A = A.lv))
S.prob <- array(c(0.60, 0.40), dim = 2, dimnames = list(S = S.lv))
O.prob <- array(c(0.96, 0.04, 0.92, 0.08), dim = c(2, 2),dimnames = list(O = O.lv, E = E.lv)
R.prob <- array(c(0.25, 0.75, 0.20, 0.80), dim = c(2, 2),dimnames = list(R = R.lv, E = E.lv)
E.prob <- array(c(0.75, 0.25, 0.72, 0.28, 0.88, 0.12, 0.64,0.36, 0.70, 0.30, 0.90, 0.10),
           dim = c(2, 3, 2),dimnames = list(E = E.lv, A = A.lv, S = S.lv))
T.prob <- array(c(0.48, 0.42, 0.10, 0.56, 0.36, 0.08, 0.58,0.24, 0.18, 0.70, 0.21, 0.09),
           dim = c(3, 2, 2),dimnames = list(T = T.lv, O = O.lv, R = R.lv))
E.prob
T.prob

cpt <- list(A = A.prob, S = S.prob, E = E.prob, O = O.prob,R = R.prob, T = T.prob)
bn <- custom.fit(dag, cpt)
```

Main Code: Approximate Inference



```
> cpquery(bn, event = (S == "M") & (T == "car"),evidence = (E == "high"))
[1] 0.3448832
> cpquery(bn, event = (S == "M") & (T == "car"),evidence = (E == "high"), n = 10^6)
[1] 0.3425343
> cpquery(bn, event = (S == "M") & (T == "car"),evidence = list(E = "high"), method = "lw")
[1] 0.3503081
> cpquery(bn, event = (S == "M") & (T == "car"),evidence = ((A == "young") & (E == "uni")) | (A == "adult"))
[1] 0.3304054
> SxT <- cpdist(bn, nodes = c("S", "T"),evidence = (E == "high"))
> head(SxT)
  S     T
1 M train
2 M   car
3 M train
4 M other
5 F   car
6 M train
> prop.table(table(SxT))
   T
S          car      train      other
  M 0.34117965 0.17586580 0.09388528
  F 0.22077922 0.10660173 0.06168831
```

From here we can see that among people whose Education is high, the most common Sex and Travel combination is **male car drivers.**

Rejection Sampling

- Monte Carlo simulations are used to randomly generate observations from the BN. We use these observations to compute approximate estimates of the conditional probabilities we are interested in.

- For discrete BNs, we implement approximate inference using rejection sampling.

- In rejection sampling, we generate random independent observations from the BN. Then we count how many match the evidence we are conditioning on and how many of those observations also match the event whose probability we are computing; the estimated conditional probability is the ratio between the latter and the former.

- This approach is implemented in bnlearn in the cpquery and cpdist functions.

- cpquery returns the probability of a specific event given some evidence.

Likelihood Weighting

- Likelihood weighting generates random observations in such a way that all of them match the evidence, and re-weights them appropriately when computing the conditional probability for the query.

- It can be accessed from cpquery by setting method = "lw".

- With likelihood weighting, cpquery returned a conditional probability that is very close to the exact value without generating 10^6 random observations in the process.

- The implementation of likelihood weighting in cpquery is not flexible enough to compute a query with composite evidence.

- cpdist returns a data frame containing the random observations for the variables in nodes that match evidence.

Thank You