# Bayesian Networks in R

# Install R package “bnlearn”

- “bnlearn” package in R
  - Learn the graphical structure of Bayesian Networks
  - Estimate their parameters
  - Perform some useful inference

- Install “bnlearn”
  - > install.packages(“bnlearn”)

# Creating Bayesian Networks

- Learn the network structure
- Use “learning.test” data set included in “bnlearn”
  - > library(bnlearn)
  - > head(learning.test)
- Incremental Association learning algorithm (iamb)
  - > data(learning.test)
  - > pdag = iamb(learning.test)
  - > pdag

# Creating Bayesian Networks

- Other structural learning algorithms in “bnlearn”
  - gs (Grow-Shrink)
  - fast.iamb (Fast Incremental Association)
  - mmpc (Max-Min Parents and Children)
  - si.hiton.pc (Semi-Interleaved HITON-PC constraint-based algorithms)

  - e.g.
    - > pdag = mmpc(learning.test)

# Creating Bayesian Networks

- Use the known topological ordering of the nodes or causal relationships from the experimental setting
  - > colnames(learning.test)
  - > dag = pdag2dag(pdag, ordering = c("A", "B", "C", "D", "E", "F")
  - > dag

- Problem
  - 같은 데이터 "learning.test" 에 대하여 "mmpc"를 사용한 결과가 "iamb"를 사용한 결과와 어떻게 다른가?

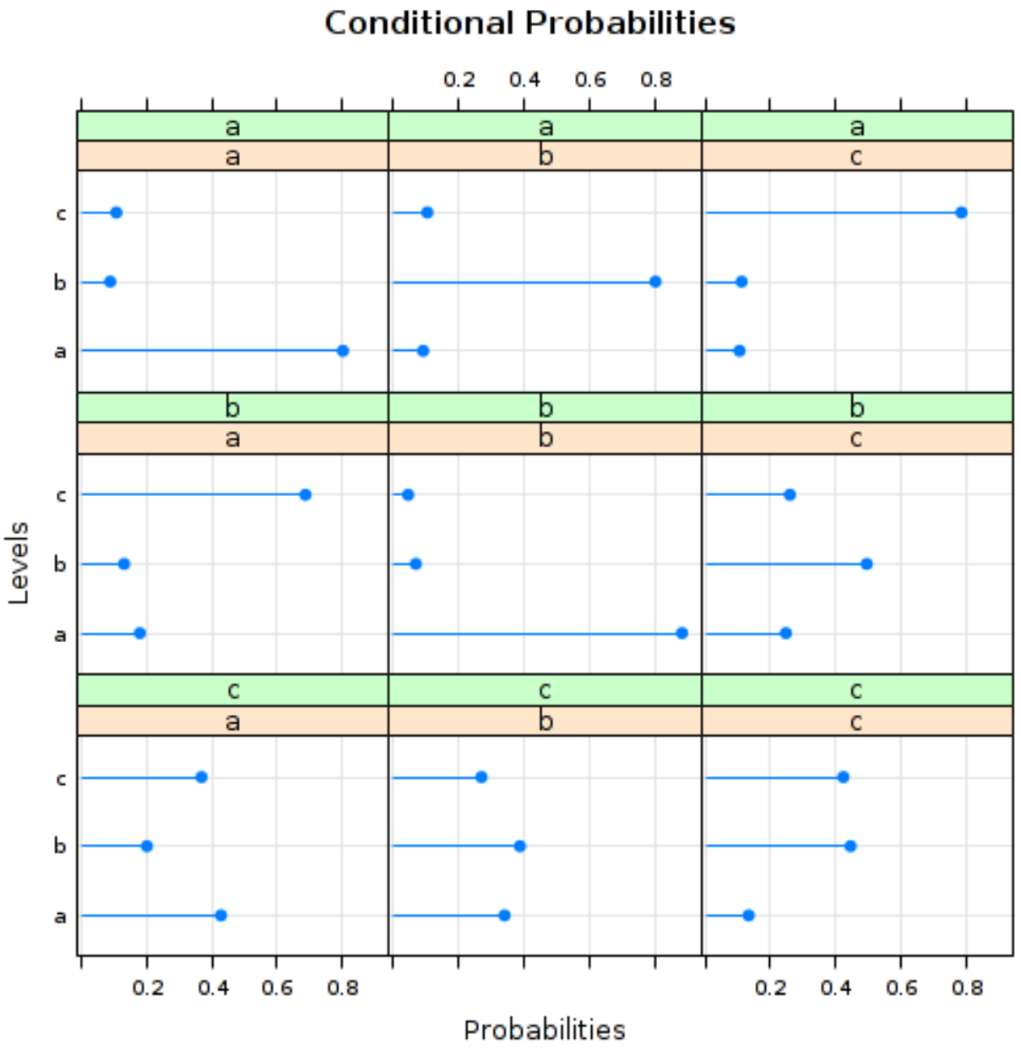
# Fitting the parameters

- Estimate the parameters in the form of conditional probability tables
- > fit = bn.fit(dag, learning.test)
- > fit
- > fit$D

```
  Parameters of node D (multinomial distribution)

Conditional probability table:

, , C = a

   A
D        a      b      c
  a 0.8008 0.0925 0.1053
  b 0.0902 0.8021 0.1117
  c 0.1089 0.1054 0.7830

, , C = b

   A
D        a      b      c
  a 0.1808 0.8830 0.2470
  b 0.1328 0.0702 0.4939
  c 0.6864 0.0468 0.2591
```

# Fitting the parameters

- > bn.fit.barchart(fit$D)
- > bn.fit.dotplot(fit$D)



# Model averaging

- Bootstrap-based inference
  - Bootstrap – test using random sampling with replacement
  - Measure arc strength (*i.e*. the degree of confidence for an arc)
- > boot = boot.strength(learning.test, R = 500, algorithm = “hc”)
  - ‘R = 500’ means that bootstrap is replicated 500 times
  - hc (hill climbing) is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a sing element of the solution

# Model averaging

- > avg.boot = averaged.network(boot, threshold = 0.85)
- > avg.boot
  - How many arcs are undirected?

- > undirected.arcs(avg.boot)

- > dag = set.arc(avg.boot, “A”, “B”)
- > dag