Experiments with Hierarchical Multinomial Dirichlet Priors for the Conditional Probability Tables of Discrete Bayesian Networks

Michael L. Thompson

11/23/2020

Contents

Introduction

This is a brief experiment with Hierarchical Multinomial-Dirichlet priors for Conditional Probability Tables (CPTs) of discrete Bayesian Networks (i.e., Bayesian belief networks, BBN).

I demonstrate the following two approaches:

1. `hierMD` – The base case proposed by L. Azzimonti, G. Corani, and M. Zaffalon (*ACZ*) of Imprecise Probability Group @ IDSIA (Azzimonti, Corani, and Zaffalon 2019).

2. `hierMDmix` – A mixture model of parent states that I've proposed.

- (**Note:** Originally, I'd posed the mixture on the Dirichlet parameter vectors. But now, I pose the mixture on conditional probability table columns – see the math below. Haven't done enough testing to determine which is better, but the current choice seems more reasonable.)

My Conclusions (w/out having done any other testing than what's shown below):

- It's unclear whether the more complicated mixture model gives Bayesian inference results that are much different than those of the *ACZ* model.

- In the resulting inferences for sparse data cases, both approaches are an improvement over a flat prior – the traditional `BDeu` (*Bayesian Dirichlet equivalent uniform* or *Laplace smoothing*) approach, in that they yield conditional distributions that are closer to the child node's marginal distribution than are uniform distributions from `BDeu`. These less flat distributions often seem more plausible.

- Given that both approaches are in the Multinomial-Dirichlet family of posterior distributions, it probably is fairly straight forward to approximate the expectation of the posteriors using analytical formulas, esp. for `hierMD`. Even for `hierMDmix`, I think the mixture component weights could be estimated in proportion to the Kullback-Leibler Divergence of each parent state's smoothed estimate of the conditional probability vector from that of the marginal distribution of the child node averaged over the parent states marginal distributions. Or some such Information-Theory-based weighting. . . .

Generative Models

Let:

- Each child node X, with L_X states that are indexed $x_i = 1, \ldots, L_X$, have $P \geq 1$ parent nodes $Y_1, ..., Y_P$.

- Each parent of X be denoted Y_j have L_{Y_j} states that are indexed $y_j = 1, \ldots, L_{Y_j}$.

- All possible combinations of the parent states be indexed by $y = 1, \ldots, (L_{Y_i} \times \cdots \times L_{Y_P})$.

- Additionally, the L_X probability parameters that make up the conditional probability vector over the states of the child node given the y combination of parent states is denoted by $\boldsymbol{\theta}_{X|Y=y}$.

- Similarly, the vector of data counts over the states of the child node given the y combination of parent states is denoted by $\mathbf{n}_{X|Y=y}$.

Then, we have the following models.

Hierarchical Multinomial-Dirichlet, `hierMD`

$$
\begin{aligned}
\boldsymbol{\alpha} &\sim \text{Dirichlet}(\mathbf{1}_{L_X}) \\
s &\sim \text{Student-t}_+(\nu = 4, \mu = 1, \sigma = 1) \\
&\text{For each combination of states, } y, \text{ of all parents,} \\
&\forall y \in 1, \ldots, (L_{Y_1} \times \cdots \times L_{Y_P}) : \\
\boldsymbol{\alpha}_y &= s \times \boldsymbol{\alpha} \\
\boldsymbol{\theta}_{X|Y=y} &\sim \text{Dirichlet}(\boldsymbol{\alpha}_y) \\
\mathbf{n}_{X|Y=y} &\sim \text{Multinomial}(\boldsymbol{\theta}_{X|Y=y}, N_{X|Y=y}) \\
&\forall i \in 1, \ldots, L_X : \\
\boldsymbol{\theta}_{X|Y=y} &= \{\theta_{X=x_i|Y=y}\} \\
\mathbf{n}_{X|Y=y} &= \{n_{X=x_i|Y=y}\} \\
N_{X|Y=y} &= \sum_{i=1}^{L_X} n_{X=x_i|Y=y}
\end{aligned}
$$

Hierarchical Multinomial-Dirichlet Mixture, `hierMDmix`

$$
\begin{aligned}
\boldsymbol{\alpha} &\sim \text{Dirichlet}(\mathbf{1}_{L_X}) \\
s &\sim \text{Student-t}_+(\nu = 4, \mu = 1, \sigma = 1) \\
\mathbf{w}_X &\sim \text{Dirichlet}(\mathbf{1}_P) \\
\mathbf{w}_X &= \{w_{X,j}\} \forall j \in 1, \ldots, P \\
&\text{For each state, } y_j, \text{ of each parent, } Y_j, \\
&\forall j \in 1, \ldots, P; \text{ and } \forall y_j \in 1, \ldots, L_{Y_j} : \\
\boldsymbol{\alpha}_{Y_j=y_j} &= s \times \boldsymbol{\alpha} \\
\boldsymbol{\theta}_{Y_j=y_j} &\sim \text{Dirichlet}(\boldsymbol{\alpha}_{Y_j=y_j}) \\
&\text{For each combination of states, } y, \text{ of all parents,} \\
&\forall y \in 1, \ldots, (L_{Y_1} \times \cdots \times L_{Y_P}) : \\
\boldsymbol{\theta}_{X|Y=y} &= \sum_{j=1}^{P} w_{X,j} \times \boldsymbol{\theta}_{Y_j=y_j^*}; \\
&\qquad \text{where } y_j^* \text{ is state of } Y_j \text{ corresponding to parent combo indexed by } y \\
\mathbf{n}_{X|Y=y} &\sim \text{Multinomial}(\boldsymbol{\theta}_{X|Y=y}, N_{X|Y=y}) \\
&\forall i \in 1, \ldots, L_X : \\
\boldsymbol{\theta}_{X|Y=y} &= \{\theta_{X=x_i|Y=y}\} \\
\mathbf{n}_{X|Y=y} &= \{n_{X=x_i|Y=y}\} \\
N_{X|Y=y} &= \sum_{i=1}^{L_X} n_{X=x_i|Y=y}
\end{aligned}
$$

Comments

For both models, it is instructive to look at what the posterior distribution for the conditional probability vector $\boldsymbol{\theta}_{X|Y=y}$ over child node X's L_X states looks like when there are either very little or no data for the combination of parent states $Y = y$, i.e., when $\mathbf{n}_{X|Y=y}$ nears or equals the zero vector. In both models, the posterior will look like the prior of $\boldsymbol{\theta}_{X|Y=y}$, which is Dirichlet($\boldsymbol{\alpha}_y$); but the models calculate this prior's $\boldsymbol{\alpha}_y$ parameter vector differently.

The `hierMD` model uses a population-level (or global) prior $\boldsymbol{\alpha}_y$ for child node X, which depending upon the strength of s (the effective sample size of the prior), will tend towards the marginal distribution of counts for each state of X. That is, it will look like the proportions computed from each element of vector $\mathbf{n}_X \equiv \sum_{y=1} \mathbf{n}_{X|Y=y}$, with a bit of smoothing.

On the other hand, the `hierMDmix` model uses a mixture of parent-state-level (or local) parameter vectors for child node X, which depending upon the strength of s, will tend to a $\mathbf{w}_X$-weighted mixture of the conditional distribution of counts for each state of X given each state y_j of each parent node Y_j.

Importantly, for child nodes X with a single parent ($P = 1$), `hierMD` and `hierMDmix` give the same result for $\boldsymbol{\theta}_{X|Y=y}$. (For example, node `age` in the example problem below.)

Below we look at a BBN for predicting the rating of a movie given the viewer features, movie features, and past ratings, then the child node `gender` has two parent nodes, `movie_rating` (below denoted as `Str.ctr_79`) and `occupation`.

In a sparsely measured combo of parent states like lowest level of `movie_rating` and `occupation` of `"doctor"`, the `hierMD` model will estimate a posterior distribution over the `gender` states of `"F"` and `"M"` that looks like the overall population distribution – in this BBN's dataset, that is about 29% and 71%, respectively – but slightly smoothed by a uniform hyperprior.

On the other hand, the `hierMDmix` model will estimate a posterior over `gender` that is a $\mathbf{w}_X$-weighted mixture of the empirical conditional probability distributions (i.e., the data proportions) of `gender` given `movie_rating` at its lowest level and of `gender` given `occupation` equal `"doctor"`, each slightly smoothed with the population-level prior.

Prepare the R Environment

Here, we load the packages we need.

```
library(magrittr) # I always use piping!
library(tidyverse) # Thank heaven & Hadley Wickham for the `tidyverse`!
library(ggridges)

# Bayesian network packages
library(bnlearn)
library(gRain)
# Implementation of **Stan** probabilistic programming language
library(rstan)
rstan_options(auto_write = TRUE)

select <- dplyr::select
```

This section includes a hidden **R** code chunk that defines functions we need (rather than `source`-ing a script). View this R Markdown document's source to see the functions.

Load the Bayesian Belief Network (BBN)

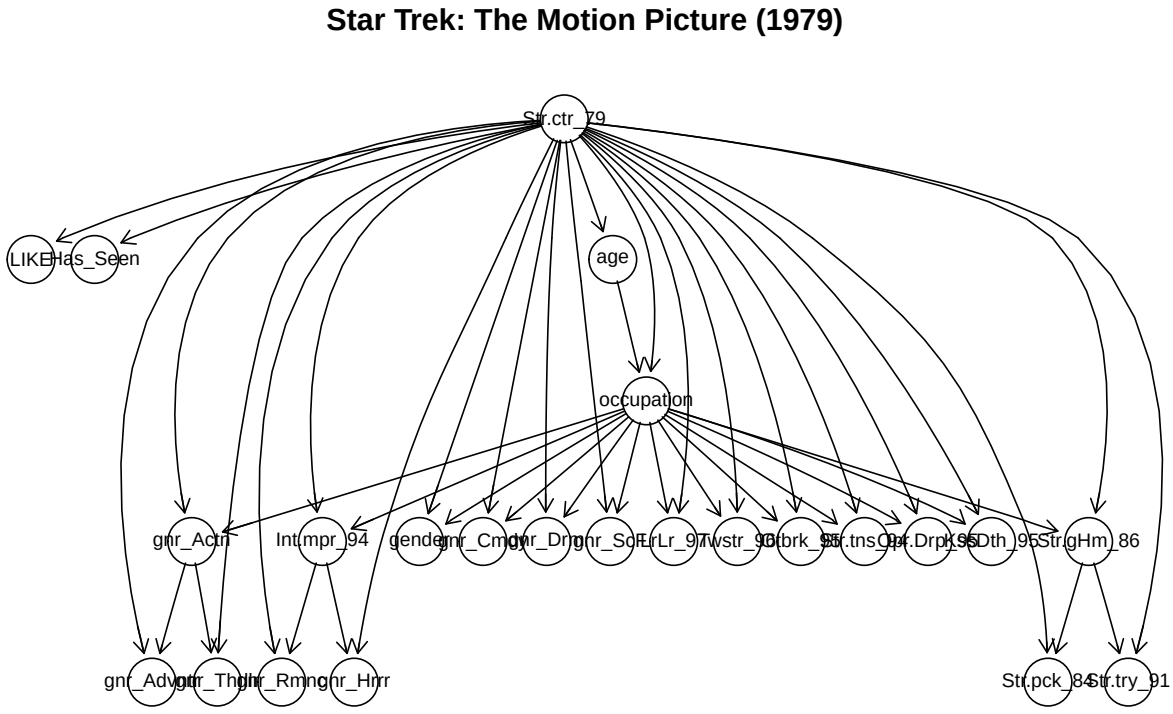
We'll use a BBN that I've generated before. The network is one of many in an ensemble of BBNs used to build a Movie Recommender System – see my presentation at the 8th Annual BayesiaLab Conference (2020) and the code and PDF of the slides at this github repository, "Bayesian Analysis".

The particular BBN shown here predicts the viewer's rating for the movie "Star Trek: The Motion Picture" (1979) given any combination of viewer features – age, gender, or occupation – movie genre, and ratings of other movies. (It was based on 694 observations in the smallest version of the MovieLens dataset.)

```
# Get a BBN to experiment with.
# Use it to simulate data, too.
bbn <- loadHuginNet(
  file = "bbn_StarTrek_79.net",
  description = "Predicts ratings of 'Star Trek: The Motion Picture (1979)'"
)
```

Visualize the BBN

```
# plot it
g1 <- bbn$dag  %>%
  as.bn() %>%
  graphviz.plot( render = FALSE )
g1 %>%
  plot(
    attrs = list(node=list(fontsize="32")),
    main = 'Star Trek: The Motion Picture (1979)'
  )
```



Perform Test Inferences

Just to demonstrate some of the nature of the BBN we're experimenting with, we perform a few inferences on the base BBN.

We plan on concentrating on inferences between the viewer features (`age`, `gender`, `occupation`) and the viewing (`Has_Seen`), liking (`LIKE`), and rating (`Str.ctr_79`) nodes of the BBN.

```r
# The movie nodes have states that are in 1-point deviations from a viewer's
# median rating over all movies the viewer has seen. Assume dev. of zero or
# greater means viewer liked the movie.
movie_node <- "Str.ctr_79"
querygrain(bbn,"Has_Seen") %>% map(round,3)
```

```
## $Has_Seen
## Has_Seen
##   yes    no
## 0.125 0.875
```

```r
querygrain(bbn,"LIKE") %>% map(round,3)
```

```
## $LIKE
## LIKE
##    yes     no unseen
##  0.065  0.060  0.875
```

```r
querygrain(bbn,"LIKE", evidence = c(Has_Seen="yes") ) %>% map(round,3)
```

```
## $LIKE
## LIKE
##    yes     no unseen
##  0.521  0.479  0.000
```

```r
querygrain(bbn,c("LIKE","Has_Seen"),type="conditional") %>% round(3)
```

```
##         Has_Seen
## LIKE       yes no
##   yes    0.521  0
##   no     0.479  0
##   unseen 0.000  1
```

```r
querygrain(bbn,nodes = c("Has_Seen","age"),type="conditional") %>% round(3)
```

```
##         age
## Has_Seen yrs_0_15 yrs_15_25 yrs_25_33 yrs_33_44 yrs_44_110
##      yes     0.01     0.103     0.157     0.162      0.084
##      no      0.99     0.897     0.843     0.838      0.916
```

```r
querygrain(
  bbn,
  nodes = c(movie_node,"age"),
  evidence = c(Has_Seen="yes"),
  type="conditional"
) %>% round(3)
```

```
##           age
## Str.ctr_79 yrs_0_15 yrs_15_25 yrs_25_33 yrs_33_44 yrs_44_110
```

```
##      r2=-3     0.167     0.116     0.026     0.001     0.002
##      r3=-2     0.167     0.116     0.175     0.118     0.293
##      r4=-1     0.167     0.345     0.275     0.264     0.235
##      r5=0      0.167     0.345     0.349     0.411     0.351
##      r6=1      0.167     0.078     0.150     0.176     0.060
##      r7=2      0.167     0.001     0.026     0.030     0.060
##      unseen    0.000     0.000     0.000     0.000     0.000
```

Generate Data

We'll also use the BBN to generate data that will serve as the bases for traditionally Laplace-smoothed parameter estimates and to see how sparsely the data lie in this high-dimensional discrete-variate space.

```
rng_seed <- 31
N   <- 1000L
df  <- bbn %>%
  simulate( nsim = N, seed = rng_seed ) %>%
  as_tibble()
```

Re-estimate the parameters of the BBN so they match this new dataset.

```
# Update the CPT parameters of the bbn
# Note that the nodes `LIKE` and `Has_Seen` are deterministic (i.e., logical).
cptl <- bbn$cptlist[setdiff(names(bbn$cptlist),"Str.ctr_79")] %>%
  map(
  ~ {
    names(dimnames(.x)) %>%
      { #print(.[[1]])
        tb <- tabMarg( table(select(df,all_of(.))), .)
        if(.[[1]] %in% c("LIKE","Has_Seen")){
          smth <- 0
        } else {
          smth <- 1/length(tb)
        }
        as.parray(tb + smth)
      }
  }
)
cptl2 <- c(
  list(
    Str.ctr_79= c(prop.table(table(df$Str.ctr_79)))[] %>%
      array( dim=length(.),dimnames=list(Str.ctr_79=names(.)))
  ),
  cptl
)
bbn_new <- grain(compileCPT(cptl2))

bbn <- bbn_new
```

Code in the Stan Probabilistic Programming Language

We use package **rstan**, based upon the **Stan** probabilistic programming language. Here, we show the programs, one each for the hierarchical Multinomial Dirichlet approach of *ACZ*, and the mixture variant

that I propose. The code below shows how straight-forward it is to implement these approaches in **Stan**. The `hierMD` code is less than 25 lines!

(Although it is possible to have `knitr` execute the code compilation directly from the **Stan** code chunks, we'll forego that and compile the programs from their respective files on disk in a later **R** code chunk.)

`hierMD` Stan Program

This first **Stan** program is adapted from the original by *ACZ* (under the GPLv3 license) – see "Hierarchical BN parameter estimation". Here, I've copied the code from my file `hierMD_MLT.stan` without its banner comments acknowledging Azzimonti *et al.* and stating the license terms – see my **Stan** files `hierMD_MLT.stan` and `hierMDmix_MLT.stan` for the full banners including the GPLv3 license statement.

This version of *ACZ*'s approach adds a proper likelihood statement and puts a prior on the magnitude of the Dirichlet parameter, `N_prior`, which in the original code was a fixed input value `s`. I've also dropped the estimation of the posterior of the marginal probability distribution parameters for the parent states (`thetaY` in the original *ACZ* code).

```
data {
  int<lower=2> n_st_ch; // number of child states
  int<lower=2> n_st_pr; // number of total combos of parent states
  int<lower=0> N_ch_pr[n_st_ch,n_st_pr]; // number of cases at all combos of parents & child
  vector<lower=0>[n_st_ch] alpha_0; // hyperparameter for Dirichlet priors
}
parameters {
  simplex[n_st_ch] theta[n_st_pr]; // conditional probability table parameters
  simplex[n_st_ch] alpha_norm; // population-level parameter for Dirichlet priors, normalized
  real<lower=0> N_prior; // number of cases represented by prior
}
transformed parameters {
  vector<lower=0>[n_st_ch] alpha; // population-level parameter for Dirichlet priors
  alpha = N_prior * alpha_norm;
}
model {
  alpha_norm ~ dirichlet(alpha_0); // prior
  N_prior    ~ student_t(4,1,1);   // prior
  for (i_st_pr in 1:n_st_pr){
    theta[i_st_pr]    ~ dirichlet( alpha ); // prior
    N_ch_pr[,i_st_pr] ~ multinomial( theta[i_st_pr] ); // likelihood
  }
}
```

`hierMDmix` Stan Program

And, here's the **Stan** implementation of my mixture variant, `hierMDmix`. (See file `hierMDmix_MLT2.stan` for the full banner of acknowledgments to *ACZ* and the license terms.)

```
data {
  int<lower=2> n_st_ch; // number of child states
  int<lower=2> n_st_pr; // number of total combos of parent states
  int<lower=0> N_ch_pr[n_st_ch,n_st_pr]; // number of cases at all combos of parents & child
  vector<lower=0>[n_st_ch] alpha_0; // hyperparameter for Dirichlet priors
  int <lower=1> n_parent; // number of parents
```

```
  int <lower=2> n_st_pr_i[n_parent]; // number of states for each parent
  int i_st_pr[n_st_pr,n_parent]; // state of each parent for each combo of parents
}
transformed data {
  int n_st_pr_sum; // sum of number of states for parents
  vector[n_parent] alpha_pr_mix; // Dirichlet parameters for mixture
  n_st_pr_sum  = sum(n_st_pr_i);
  alpha_pr_mix = rep_vector(1,n_parent);
}
parameters {
  simplex[n_st_ch] theta_i[n_st_pr_sum];  // conditional probability table parameters
  simplex[n_st_ch] alpha_norm; // population-level parameter for Dirichlet priors, normalized
  real<lower=0> N_prior;  // number of cases represented by prior
  simplex[n_parent] w_mix; // mixture probabilities on the parents alpha_i
}
transformed parameters {
  vector<lower=0>[n_st_ch] theta[n_st_pr]; // mixture alpha
  vector<lower=0>[n_st_ch] alpha; // population level
  alpha = N_prior * alpha_norm;  // population-level
  for( i_st in 1:n_st_pr ){
    theta[i_st] = rep_vector(0,n_st_ch); // initialize as zeros
    for( i in 1:n_parent ){
      theta[i_st] += w_mix[i] * theta_i[ sum(head(n_st_pr_i,i-1)) + i_st_pr[i_st,i] ];
    }
  }
}
model {
  alpha_norm ~ dirichlet( alpha_0 );      // prior
  N_prior    ~ student_t( 4, 1, 1 );      // prior
  w_mix      ~ dirichlet( alpha_pr_mix ); // prior
  {
    int i_st = 0;
    for( i in 1:n_parent ){
      for( j in 1:n_st_pr_i[i]){
        i_st += 1;
        theta_i[i_st] ~ dirichlet( alpha ); // prior
      }
    }
  }
  for (i_st in 1:n_st_pr){
    N_ch_pr[,i_st] ~ multinomial( theta[i_st] );  // likelihood
  }
}
```

Now, we compile the **Stan** programs from file.

```
# STAN COMPILATION
# Compile the Stan programs of the two model variants.
sm_hierMD    <- stan_model(file="hierMD_MLT.stan",    model_name="hierMD")
sm_hierMDmix <- stan_model(file="hierMDmix_MLT2.stan", model_name="hierMDmix")
```

Estimation of the Conditional Probability Tables (CPT)

We show application of the `gen_CPT_hierMD()` function, which performs Variational Bayesian Infernce on both models to generate CPT parameters first using the `hierMD` method then the `hierMDmix` method for a single named child node. It also computes the parameters using Laplace smoothing (`BDeu`).

We could do it for every child node in the BBN, but instead, we're just going to do it on the three viewer feature nodes: `gender`, `age`, and `occupation`.

```
# CPT ESTIMATION
rslt_gender <- gen_CPT_hierMD( bbn, df, ch_name = "gender" )
rslt_age    <- gen_CPT_hierMD( bbn, df, ch_name = "age" )
rslt_occ    <- gen_CPT_hierMD( bbn, df, ch_name = "occupation" )

rslt <- list(gender=rslt_gender, age = rslt_age, occupation = rslt_occ)

# # Do All Child Nodes, excluding deterministic nodes:
# rslt <- setdiff( bbn$universe$nodes, c(movie_node,"LIKE","Has_Seen") ) %>%
#   set_names(.,.) %>%
#   map( ~ {print(.x);gen_CPT_hierMD(bbn, df, ch_name = .x)} )
```

Display the CPT parameters of the `gender` node for each method.

```
# Test the gender-node revision
theta_BDeu_eps  <- rslt_gender$theta_BDeu_eps
theta_hierMD    <- rslt_gender$theta_hierMD
theta_hierMDmix <- rslt_gender$theta_hierMDmix

df %>%
  filter(str_detect(occupation,"programmer")) %$%
  table(gender,Str.ctr_79) %>%
  {list(`OCCURRENCES: occupation == "programmer"`= .)}
```

```
## $`OCCURRENCES: occupation == "programmer"`
##       Str.ctr_79
## gender r2=-3 r3=-2 r4=-1 r5=0 r6=1 r7=2 unseen
##      F     0     0     0    0    0    0      4
##      M     0     1     0    9    5    0     38
```

```
cpt <- bbn$cptlist$gender
n_st_pr_i <- dim(cpt)[-1]

# The 15th occupation is "programmer".
# cat("bbn cpt, original:\n")
# (cpt[,,15,drop=FALSE]) %>% round(3)
cat("\ntheta_BDeu_eps:\n")
```

```
##
## theta_BDeu_eps:
```

```
theta_BDeu_eps[(15-1)*n_st_pr_i[[1]] + (1:n_st_pr_i[[1]]),] %>%
  t() %>%
  round(3) %>%
  {dimnames(.)<- dimnames(cpt)[-3]; .}
```

```
##       Str.ctr_79
## gender r2=-3 r3=-2 r4=-1 r5=0  r6=1 r7=2 unseen
##      F   0.5 0.003   0.5    0 0.001  0.5  0.095
##      M   0.5 0.997   0.5    1 0.999  0.5  0.905
```

```
cat("\ntheta_hierMD:\n")
```

```
##
## theta_hierMD:
```

```
theta_hierMD[(15-1)*n_st_pr_i[[1]] + (1:n_st_pr_i[[1]]),] %>%
  t() %>%
  round(3) %>%
  {dimnames(.)<- dimnames(cpt)[-3]; .}
```

```
##       Str.ctr_79
## gender r2=-3 r3=-2 r4=-1  r5=0  r6=1 r7=2 unseen
##      F 0.268 0.192 0.284 0.042 0.091 0.27  0.102
##      M 0.732 0.808 0.716 0.958 0.909 0.73  0.898
```

```
cat("\ntheta_hierMDmix:\n")
```

```
##
## theta_hierMDmix:
```

```
theta_hierMDmix[(15-1)*n_st_pr_i[[1]] + (1:n_st_pr_i[[1]]),] %>%
  t() %>%
  round(3) %>%
  {dimnames(.)<- dimnames(cpt)[-3]; .}
```

```
##       Str.ctr_79
## gender r2=-3 r3=-2 r4=-1  r5=0  r6=1  r7=2 unseen
##      F 0.069 0.079 0.044 0.045 0.056 0.059  0.114
##      M 0.931 0.921 0.956 0.955 0.944 0.941  0.886
```

Revision of the BBN

Now, let's load the CPTs into separate BBN and evaluate how they differ from the original BBN in terms of inferences.

```
# REVISION OF BBN
bbn_list <- list( hierMD=bbn, hierMDmix=bbn )
for(ch_name in c("age","gender","occupation")){

  bbn_list <- revise_bbn(
    bbn_list,
    ch_name         = ch_name,
    theta_hierMD    = rslt[[ch_name]]$theta_hierMD,
    theta_hierMDmix = rslt[[ch_name]]$theta_hierMDmix
  )
```

```
}

bbn_hierMD    <- bbn_list$hierMD
bbn_hierMDmix <- bbn_list$hierMDmix
```

Impact on Bayesian Inference

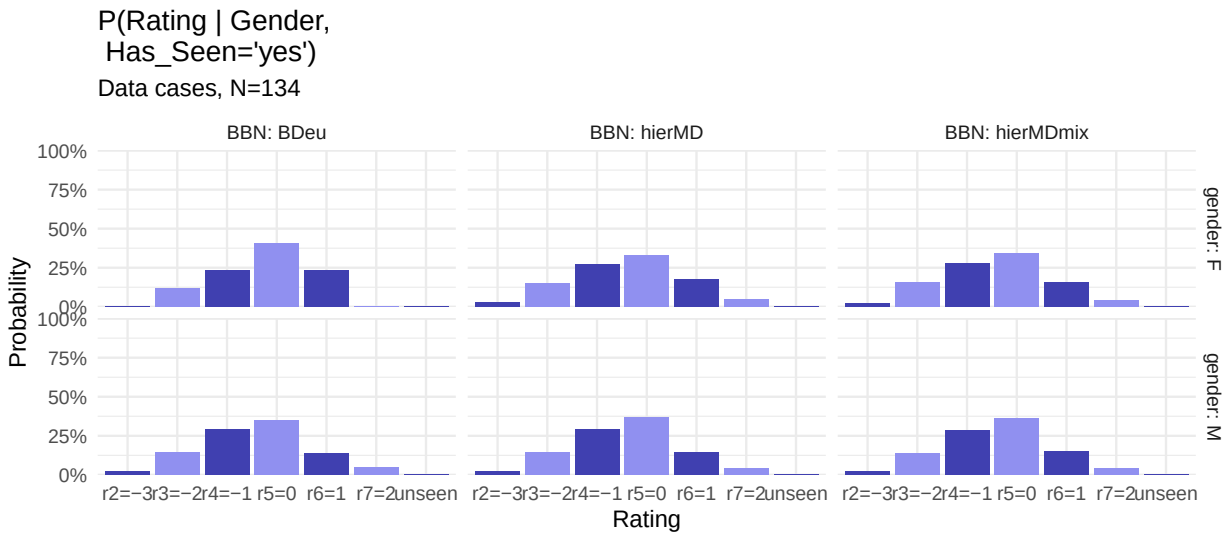
The space is so high-dimensional, and we only have `N=1000` cases. So, most of the combos are not measured. Yet, the models will make inferences for any combination of node values.

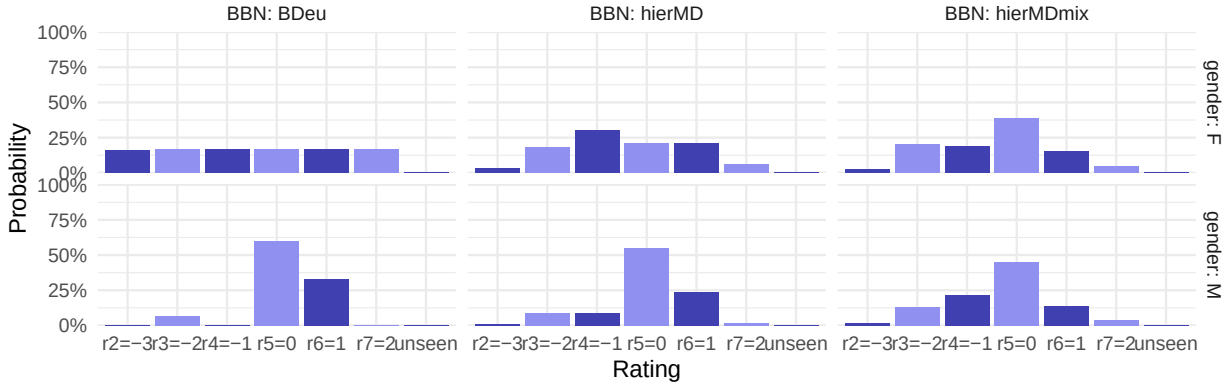
(What's actually needed is feedback to the practitioner that the network is highly uncertain about any inferences in such instances. That's where having the full posterior distributions of the CPT parameters is helpful. But, we don't explore that here.)

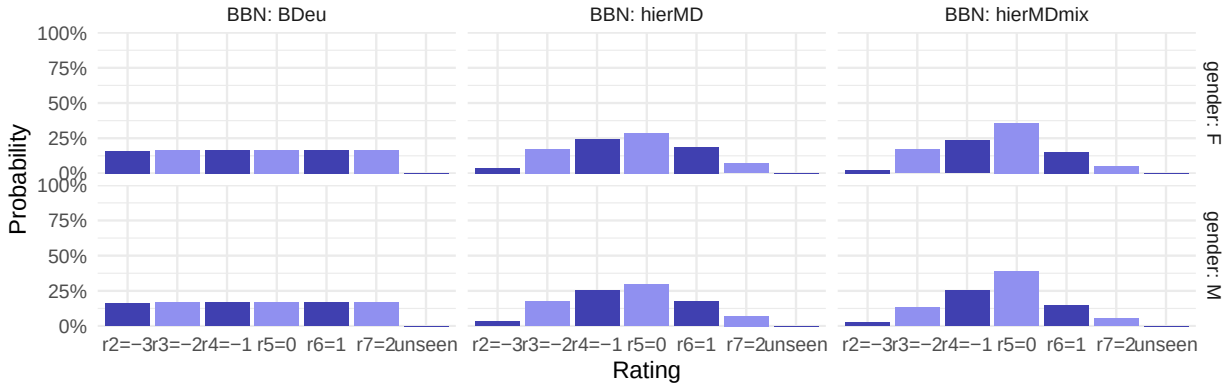
Visualization

Let's contrast the methods by plots. Each set has a different number of data cases, N, from more to less to zero. (The code chunks for the latter two plots are hidden.)

```
qdf <- list(BDeu = bbn) %>% c(bbn_list) %>%
  imap_dfr(
    ~  {
      querygrain(
        .x, nodes = c("Str.ctr_79","gender"),
        evidence=list(Has_Seen="yes"), type="conditional"
      ) %>%
        as.data.frame() %>%
        rownames_to_column("Rating") %>%
        as_tibble() %>%
        pivot_longer(cols=-Rating,names_to="gender", values_to="pr") %>%
        mutate(BBN = .y)
    }
  )
qdf %>% {
  ggplot(.,aes(x=Rating,y=pr,fill=Rating)) +
    geom_col() +
    facet_grid( gender ~ BBN, labeller = label_both) +
    scale_fill_cyclical(values = c("#4040B0", "#9090F0")) +
    scale_y_continuous(
      expand = c(0, 0), limits = c(0,1),
      labels=scales::percent
    ) +
    labs(
      x="Rating", y= "Probability",
      title="P(Rating | Gender,\n\tHas_Seen='yes')",
      subtitle = "Data cases, N=134"
    ) + theme_minimal()
} %>% print()
```


P(Rating | Gender,
Has_Seen='yes')
Data cases, N=134
BBN: BDeu
BBN: hierMD
BBN: hierMDmix
Probability
100%
75%
50%
25%
0%
gender: F
gender: M
r2=−3r3=−2r4=−1
r5=0
r6=1
r7=2unseen
Rating


P(Rating | Gender,
Occupation='programmer',
Has_Seen='yes')
Data cases, N=13
BBN: BDeu
BBN: hierMD
BBN: hierMDmix
Probability
100%
75%
50%
25%
0%
gender: F
gender: M
r2=−3r3=−2r4=−1
r5=0
r6=1
r7=2unseen
Rating


P(Rating | Gender,
Occupation='doctor',
Has_Seen='yes')
Data cases, N=0
BBN: BDeu
BBN: hierMD
BBN: hierMDmix
Probability
100%
75%
50%
25%
0%
gender: F
gender: M
r2=−3r3=−2r4=−1
r5=0
r6=1
r7=2unseen
Rating

Impact on Generalized Bayes Factor, GBF(H:E)

The Generalized Bayes Factor, GBF(H:E), has been shown to be a good metric for hypothesis (H) confirmation given evidence (E) and for use in generating relevant explanations (a ranked list of H's) of observed evidence (E). However, it is very sensitive to noisy estimates of prior and posterior probabilities for sparsely measured cases.

Given that the `hierMD` and `hierMDmix` approaches both spread probability mass throughout the sparse CPTs that is more consistent with the population-level distributions, we would expect the GBF(H:E) for cases of either sparsely measured hypotheses H or evidence E to be less noisy, though somewhat biased due to the shrinkage towards the population marginal distributions that these priors induce.

```
# IMPACT ON GENERALIZED BAYES FACTOR
# Compare models impact on GBF(H:E) under different case profiles as
# the "hypothesis" H, given the "evidence" E = LIKE = "yes".
list( BDeu = bbn, hierMD = bbn_hierMD, hierMDmix = bbn_hierMDmix ) %>%
  imap_dfr(
    ~ gbf(bbn = .x, gndr = "F", occ = "doctor", age = "yrs_33_44") %$%
      map(at_cond,signif,3) %>%
      as_tibble() %>%
      mutate(model=.y) %>%
      select(model,everything())
  )
```

```
## # A tibble: 3 x 7
##   model       gbf    wte     joint joint_post prior_odds post_odds
##   <chr>     <dbl>  <dbl>     <dbl>      <dbl>      <dbl>     <dbl>
## 1 BDeu      0.913 -0.397 0.0000287  0.0000262  0.0000287 0.0000262
## 2 hierMD    1.1    0.414 0.0027     0.00297    0.00271   0.00298
## 3 hierMDmix 1.26   1.01  0.000236   0.000298   0.000236  0.000298
```

```
list( BDeu = bbn, hierMD = bbn_hierMD, hierMDmix = bbn_hierMDmix ) %>%
  imap_dfr(
    ~ gbf(bbn = .x, gndr = "M", occ = "programmer", age = "yrs_33_44") %$%
      map(at_cond,signif,3) %>%
      as_tibble() %>%
      mutate(model=.y) %>%
      select(model,everything())
  )
```

```
## # A tibble: 3 x 7
##   model       gbf   wte  joint joint_post prior_odds post_odds
##   <chr>     <dbl> <dbl>  <dbl>      <dbl>      <dbl>     <dbl>
## 1 BDeu       1.89  2.76 0.0426     0.0775     0.0445    0.084
## 2 hierMD     1.64  2.14 0.0222     0.0358     0.0227    0.0371
## 3 hierMDmix  1.28  1.06 0.0198     0.0251     0.0202    0.0257
```

```
list( BDeu = bbn, hierMD = bbn_hierMD, hierMDmix = bbn_hierMDmix ) %>%
  imap_dfr(
    ~ gbf(bbn = .x, gndr = "F", occ = "programmer", age = "yrs_33_44") %$%
      map(at_cond,signif,3) %>%
      as_tibble() %>%
      mutate(model=.y) %>%
```

```
      select(model,everything())
  )
```

```
## # A tibble: 3 x 7
##   model       gbf   wte     joint joint_post prior_odds post_odds
##   <chr>     <dbl> <dbl>     <dbl>      <dbl>      <dbl>     <dbl>
## 1 BDeu       1.37 1.38  0.0000389  0.0000535  0.0000389 0.0000535
## 2 hierMD     1.21 0.821 0.00239    0.00289    0.00240   0.00290
## 3 hierMDmix  1.21 0.811 0.00106    0.00128    0.00106   0.00128
```

```
list( BDeu = bbn, hierMD = bbn_hierMD, hierMDmix = bbn_hierMDmix ) %>%
  imap_dfr(
    ~ gbf(bbn= .x, gndr = "F", occ = "administrator", age = "yrs_33_44") %$%
      map(at_cond,signif,3) %>%
      as_tibble() %>%
      mutate(model=.y) %>%
      select(model,everything())
  )
```

```
## # A tibble: 3 x 7
##   model       gbf    wte     joint joint_post prior_odds post_odds
##   <chr>     <dbl>  <dbl>     <dbl>      <dbl>      <dbl>     <dbl>
## 1 BDeu      0.608 -2.16  0.0000431  0.0000262  0.0000431 0.0000262
## 2 hierMD    1.29   1.1   0.00363    0.00466    0.00364   0.00469
## 3 hierMDmix 1.11   0.466 0.0111     0.0124     0.0113    0.0125
```

Thoughts

Of course, much more work would be needed to show if and when the `hierMDmix` method adds value over that of the `hierMD` method. But, *ACZ* have already shown the value of `hierMD` over traditional smoothing calculation of Bayesian network CPTs.

It is nice to see that either of these approaches is so easily implemented using **Stan**. Moreover, *ACZ* provide **R** & **Stan** source code (under the GPLv3 license) – "Hierarchical BN parameter estimation" – to perform the variational Bayesian inference for `hierMD` both with and without using **Stan**.

Finally, having the full posterior (approximately) of the CPT parameters is a nice feature of these two methods. In the future, one should exploit this by reporting or visualizing the uncertainty quantification in risk assessment and decision analysis.

About

-Michael L. Thompson,

LinkedIn profile

LICENSE

References

Azzimonti, Laura, Giorgio Corani, and Marco Zaffalon. 2019. "Hierarchical Estimation of Parameters in Bayesian Networks." *Computational Statistics & Data Analysis* 137: 67–91. https://doi.org/10.1016/j.csda.2019.02.004.