Case Study 2 take 2: True-or-False Exam Scores

1 The data

Suppose a group of 15 people sit an exam made up of 40 true-or-false questions, and each receives a score afterwards. Dataset "TrueFalseScores.csv" on Moodle.

2 Latent class model approach

Suppose the true/false exam has m questions and y_i denotes the score of observation i, $i = 1, \cdots, n$. Assume there are two latent classes and each observation belongs to one of the two latent classes. Let z_i be the class assignment for observation i and π be the probability of being assigned to class 1. Given the latent class assignment z_i for observation i, the score Y_i follows a Binomial distribution with m trials and a class-specific success probability. Since there are only two possible class assignments, all observations assigned to class 1 share the same correct success parameter p_1 and all observations assigned to class 0 share the same success rate parameter p_0. The specification of the data model is expressed as follows:

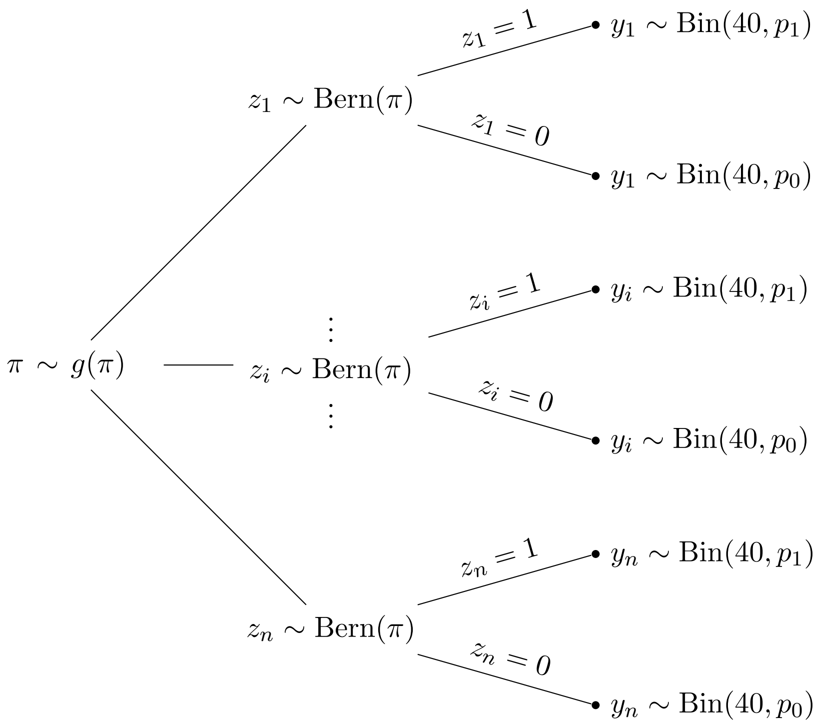
$$\begin{aligned} Y_i = y_i \mid z_i, p_{z_i} &\sim \text{Binomial}(m, p_{z_i}), && (1)\\ z_i \mid \pi &\sim \text{Bernoulli}(\pi). && (2) \end{aligned}$$

In this latent class model there are many unknown parameters. One does not know the class assignment probability π, the class assignments $z_1, ..., z_n$, and the probabilities p_1 and p_0 for the two Binomial distributions. Some possible choices for prior distributions are provided below for inspiration.

(a) The parameters π and $(1 - \pi)$ are the latent class assignment probabilities for the two classes. If additional information is available which indicates, for example, that 1/3 of the observations belonging to class 1, then π is considered as fixed and set to the value of 1/3. If no such information is available, one can consider π as unknown and assign this parameter a prior distribution. A natural choice for prior on a success probability is a Beta prior distribution with shape parameters a and b.

(b) The parameters p_1 and p_0 are the success rates in the Binomial model in the two classes. If one believes that the test takers in class 1 are simply random guessers, then one fixes p_1 to the value of 0.5. Similarly, if one believes that test takers in class 0 have a higher success rate of 0.9, then one sets p_0 to the value 0.9. However, if one is uncertain about the values of p_1 and p_0, one lets either or both success rates be random and assigned prior distributions.

The tree diagram below illustrates the latent class model.



3 Sample JAGS script

Sample JAGS script is provided below. This is the case where $\pi = 1/3$.

One introduces a new variable `theta[i]` that indicates the correct rate value for observation `i`. In the sampling section of the JAGS script, the first block is a loop over all observations, where one first determines the rate `theta[i]` based on the classification value `z[i]`. The `equals` command evaluates equality, for example, `equals(z[i], 0)` returns 1 if `z[i]` equals to `0`, and returns 0 otherwise. This indicates that the rate `theta[i]` will either be equal to `p1` or `p0` depending on the value `z[i]`.

One should note in JAGS, the classification variable `z[i]` takes values of 0 and 1, corresponding to the knowledgeable and guessing classes, respectively. As π is considered fixed and set to 1/3, the variable `z[i]` is assigned a Bernoulli distribution with probability 1/3. To conclude the script, in the prior section the guessing rate parameter `p1` is assigned the value 0.5 and the rate parameter `p0` is assigned a Beta(1, 1) distribution truncated to the interval (0.5, 1) using `T(0.5, 1)`.

```
modelString<-"
model {
## sampling
for (i in 1:N){
   theta[i] <- equals(z[i], 1) * p1 + equals(z[i], 0) * p0
y[i] ~ dbin(theta[i], m)
}
for (i in 1:N){
   z[i] ~ dbern(1/3)
}
## priors
```

```
p1 <- 0.5
p0 ~ dbeta(1,1) T(0.5, 1)
}
"
```

4 Important things to note

- Make sure to consider how many parameters are in the model, and what they are.
- For every parameter, make sure to give a prior distribution, and possibly hyperprior distributions for any hyperparameters you use.
- For any parameter of your interest, make sure to monitor it in the JAGS script so you can track them and summarize them in the posterior.
- Posterior summaries of assignments of z_i can tell us across your MCMC chain, how often is person i being classified in the random guessing group, and how often they are being classified in the knowledgeable group.