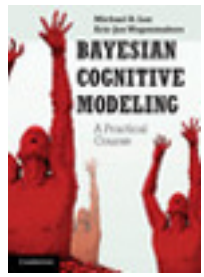


Bayesian Cognitive Modeling

A Practical Course

Michael D. Lee, Eric-Jan Wagenmakers

Book DOI: http://dx.doi.org/10.1017/CBO9781139087759

Online ISBN: 9781139087759

Hardback ISBN: 9781107018457

Paperback ISBN: 9781107603578

Chapter

6 - Latent-mixture models pp. 77-98

Chapter DOI: http://dx.doi.org/10.1017/CBO9781139087759.008

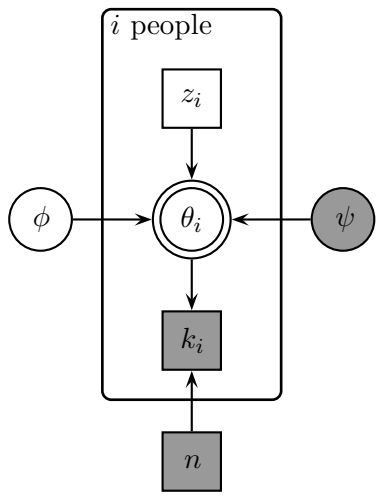
Cambridge University Press

6 Latent-mixture models

6.1 Exam scores

Suppose a group of 15 people sit an exam made up of 40 true-or-false questions, and they get 21, 17, 21, 18, 22, 31, 31, 34, 34, 35, 35, 36, 39, 36, and 35 right. These scores suggest that the first 5 people were just guessing, but the last 10 had some level of knowledge.

One way to make statistical inferences along these lines is to assume there are two different groups of people. These groups have different probabilities of success, with the guessing group having a probability of 0.5, and the knowledge group having a probability greater than 0.5. Whether each person belongs to the first or the second group is a latent or unobserved variable that can take just two values. Using this approach, the goal is to infer to which group each person belongs, and also the rate of success for the knowledge group.



$$
\begin{aligned}
z_i &\sim \text{Bernoulli}(0.5) \\
\phi &\sim \text{Uniform}(0.5, 1) \\
\psi &\leftarrow 0.5 \\
\theta_i &\leftarrow \begin{cases} \phi \text{ if } z_i = 1 \\ \psi \text{ if } z_i = 0 \end{cases} \\
k_i &\sim \text{Binomial}(\theta_i, n)
\end{aligned}
$$

Fig. 6.1 Graphical model for inferring membership of two latent groups, with different rates of success in answering exam questions.

A graphical model for doing this is shown in Figure 6.1. The number of correct answers for the ith person is k_i, and is out of $n = 40$. The probability of success on each question for the ith person is the rate θ_i. This rate is either ψ, if the person is

in the guessing group, or ϕ if the person is in the knowledge group. Which group the ith person belongs to is determined by a binary indicator variable z_i, with $z_i = 0$ if the ith person is in the guessing group, and $z_i = 1$ if the ith person is in the knowledge group.

We assume each of these indicator variables is equally likely to be 0 or 1 a priori, so they have the prior $z_i \sim \text{Bernoulli}\big(1/2\big)$. For the guessing group, we assume that the rate is $\psi = 1/2$. For the knowledge group, we use a prior where all rate possibilities greater than $1/2$ are equally likely, so that $\phi \sim \text{Uniform}\big(0.5, 1\big)$.

This type of model is known as a *latent-mixture* model, because the data are assumed to be generated by two different processes that combine or mix, and important properties of that mixture are unobserved or latent. In this case, the two components that mix are the guessing and knowledge processes, and the group membership of each person is latent.

The script `Exams_1.txt` implements the graphical model in WinBUGS:

```
# Exam Scores
model{
  # Each Person Belongs To One Of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(0.5)
  }
  # First Group Guesses
  psi <- 0.5
  # Second Group Has Some Unknown Greater Rate Of Success
  phi ~ dbeta(1,1)I(0.5,1)
  # Data Follow Binomial With Rate Given By Each Person's Group Assignment
  for (i in 1:p){
    theta[i] <- equals(z[i],0)*psi+equals(z[i],1)*phi
    k[i] ~ dbin(theta[i],n)
  }
}
```

The code `Exams_1.m` or `Exams_1.R` makes inferences about group membership, and the success rate of the knowledge group, using the model.

Exercises

Exercise 6.1.1 Draw some conclusions about the problem from the posterior distribution. Who belongs to what group, and how confident are you?

Exercise 6.1.2 The initial allocations of people to the two groups in this code is random, and so will be different every time you run it. Check that this does not affect the final results from sampling.

Exercise 6.1.3 Include an extra person in the exam, with a score of 28 out of 40. What does their posterior for z tell you? Now add four extra people, all with the score 28 out of 40. Explain the change these extra people make to the inference.

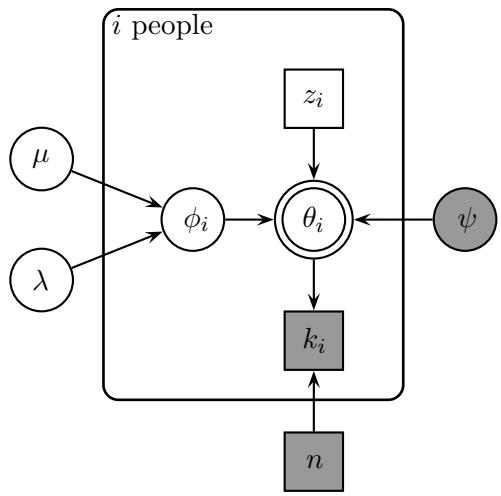
Exercise 6.1.4 What happens if you change the prior on the success rate of the second group to be uniform over the whole range from 0 to 1, and so allow for worse-than-guessing performance?

Exercise 6.1.5 What happens if you change the initial expectation that everybody is equally likely to belong to either group, and have an expectation that people generally are not guessing, with (say), $z_i \sim \text{Bernoulli}(0.9)$?

6.2 Exam scores with individual differences

The previous example shows how sampling can model data as coming from a mixture of sources, and infer properties of these latent groups. But the specific model has at least one big weakness, which is that it assumes all the people in the knowledge group have exactly the same rate of success on the questions.

One straightforward way to allow for individual differences in the knowledge group is to extend the model hierarchically. This involves drawing the success rate for each of the people in the knowledge group from an over-arching distribution. One convenient (but not perfect) choice for this "individual differences" distribution is a Gaussian. It is a natural statistical model for individual variation, at least in the absence of any richer theory. But it has the problem of allowing for success rates below zero and above one. An inelegant but practical and effective way to deal with this is simply to restrict the sampled success rates to the valid range.



$$
\begin{aligned}
z_i &\sim \text{Bernoulli}(0.5) \\
\mu &\sim \text{Uniform}(0.5, 1) \\
\lambda &\sim \text{Gamma}(0.001, 0.001) \\
\phi_i &\sim \text{Gaussian}(\mu, \lambda)_{\mathcal{I}(0,1)} \\
\psi &\leftarrow 0.5 \\
\theta_i &\leftarrow \begin{cases} \phi_i \text{ if } z_i = 1 \\ \psi \text{ if } z_i = 0 \end{cases} \\
k_i &\sim \text{Binomial}(\theta_i, n)
\end{aligned}
$$

Fig. 6.2 Graphical model for inferring membership of two latent groups, with different rates of success in answering exam questions, allowing for individual differences in the knowledge group.

A graphical model that implements this idea is shown in Figure 6.2. It extends the original model by having a knowledge group success rate ϕ_i for the ith person. These success rates are drawn from a Gaussian distribution with mean μ and precision

Box 6.1 Assessing and improving convergence

In a perfect world, a single MCMC chain would immediately begin drawing samples from the posterior distribution, and the only computational issue would be how many are needed to form a sufficiently precise approximation. This ideal state of affairs is often not what happens, and latent-mixture models are notorious for needing convergence checks. So, this is a good place to list some checks (see also Gelman, 1996; Gelman & Hill, 2007).

The basic principle is that, when the sampling process has converged, chains with substantially different starting values should be indistinguishable from each other. One implication of this requirement is that chains should vary around a constant mean, so a slow drift up or down signals a problem. And, if the sampling process has converged, each individual chain should look like a "fat hairy caterpillar," because this visual appearance is generated when successive values are relatively independent. As a formal test for convergence, the $\hat{R}$ statistic (Gelman & Rubin, 1992) is widely used. It is basically a measure of between-chain to within-chain variance, and so values close to 1 indicate convergence. As a rule of thumb, values higher than 1.1 are (deeply) suspect. If you were not paying much attention to the `rhat` values WinBUGS is returning to Matlab and R in previous modeling exercises, now is a good time to start checking them.

There are three basic remedies for a lack of convergence, easily implemented in WinBUGS for any model. The first is simply to collect many more samples, or more chains of samples, and wait (and hope) for convergence. The second is to increase the number of *burn-in* samples, which are initial samples in a chain that are discarded. This will be effective if separate chains are sensitive to their starting points, and take some time to converge. A worked example of this is presented in Section 11.2. The third is to *thin* the samples, by retaining only one out of every n. This will be effective if a chain is autocorrelated, with lack of independence between samples. A worked example of this is presented in Section 3.6. There are other, more advanced, methods for improving convergence in WinBUGS, involving changing the model itself. Worked examples of the *parameter expansion* method are presented in Sections 11.3 and 14.2.

λ. The mean μ is given a uniform prior between 0.5 and 1.0, consistent with the original assumption that people in the knowledge group have a greater-than-chance success rate.

Box 6.2 Scripts for graphical models

The scripts that implement graphical models in WinBUGS are declarative, rather than procedural. This means the order of the commands does not matter. All that a script does is define the observed and unobserved variables in a graphical model, saying how they are distributed, and how they relate to each other. This is inherently a structure, rather than a process, and so order is not important. In practice this means, for example, that a separate loop is not needed in a script like `Exam_2.txt` to define `k[i]`, `z[i]`, and `phi[i]`. Exactly the same graphical model would be defined if they were all placed inside one `for (i in 1:p)` loop. Sometimes, however, it is conceptually clearer to use separate loops to implement different parts of a graphical model.

The script `Exams_2.txt` implements the graphical model in WinBUGS:

```
# Exam Scores With Individual Differences
model{
  # Rates Given By Each Person's Group Assignment
  for (i in 1:p){
    theta[i] <- equals(z[i],0)*psi+equals(z[i],1)*phi[i]
    k[i] ~ dbin(theta[i],n)
  }
  # Each Person Belongs To One Of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(0.5)
  }
  # The Second Group Allows Individual Differences
  for (i in 1:p){
    phi[i] ~ dnorm(mu,lambda)I(0,1)
  }
  # First Group Guesses
  psi <- 0.5
  # Second Group Mean, Precision (And Standard Deviation)
  mu ~ dbeta(1,1)I(.5,1) # >0.5 Average Success Rate
  lambda ~ dgamma(.001,.001)
  sigma <- 1/sqrt(lambda)
  # Posterior Predictive For Second Group
  predphi ~ dnorm(mu,lambda)I(0,1)
}
```

Notice that the code includes a variable `predphi` that draws success rates from the inferred Gaussian distribution of the knowledge group.

The code `Exams_2.m` or `Exams_2.R` makes inferences about group membership, the success rate of each person in the knowledge group, and the mean and standard deviation of the over-arching Gaussian for the knowledge group.

Exercises

Exercise 6.2.1 Compare the results of the hierarchical model with the original model that did not allow for individual differences.

Exercise 6.2.2 Interpret the posterior distribution of the variable `predphi`. How does this distribution relate to the posterior distribution for `mu`?

Exercise 6.2.3 In what sense could the latent assignment of people to groups in this case study be considered a form of model selection?

6.3 Twenty questions

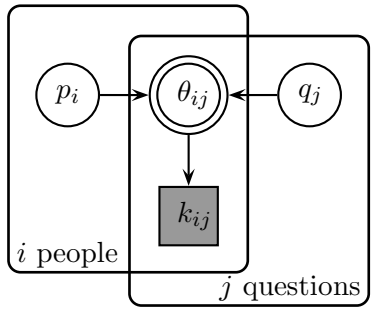
Suppose a group of 10 people attend a lecture, and are asked a set of 20 questions afterwards, with every answer being either correct or incorrect. The pattern of data is shown in Table 6.1. From this pattern of correct and incorrect answers we want to infer two things. The first is how well each person attended to the lecture. The second is how hard each of the questions was.

Table 6.1 Correct and incorrect answers for 10 people on 20 questions.

	Question																			
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Person 1	1	1	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	0	0
Person 2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 3	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
Person 4	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
Person 5	1	0	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	0	0
Person 6	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0
Person 7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Person 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 9	0	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
Person 10	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0

One way to make these inferences is to specify a model of how a person's attentiveness and a question's difficulty combine to give an overall probability that the question will be answered correctly. A very simple model involves assuming that each person listens to some proportion of the lecture, and that each question has some probability of being answered correctly if the person was listening at the right point in the lecture.

A graphical model that implements this idea is shown in Figure 6.3. Under the model, if the ith person's probability of listening is p_i, and the jth question's probability of being answered correctly if the relevant information is heard is q_j,



$$p_i, q_j \sim \text{Beta}(1,1)$$
$$\theta_{ij} \leftarrow p_i q_j$$
$$k_{ij} \sim \text{Bernoulli}(\theta_{ij})$$

Fig. 6.3 Graphical model for inferring the rate people listened to a lecture, and the difficulty of the questions.

then the probability the ith person will answer the jth question correctly is just $\theta_{ij} = p_i q_j$. The observed pattern of correct and incorrect answers, where $k_{ij} = 1$ if the ith person answered the jth question correctly, and $k_{ij} = 0$ if they did not, then is a draw from a Bernoulli distribution with probability θ_{ij}.

The script `TwentyQuestions.txt` implements the graphical model in WinBUGS:

```
# Twenty Questions
model{
  # Correctness Of Each Answer Is Bernoulli Trial
  for (i in 1:np){
    for (j in 1:nq){
      k[i,j] ~ dbern(theta[i,j])
    }
  }
  # Probability Correct Is Product Of Question By Person Rates
  for (i in 1:np){
    for (j in 1:nq){
      theta[i,j] <- p[i]*q[j]
    }
  }
  # Priors For People and Questions
  for (i in 1:np){
    p[i] ~ dbeta(1,1)
  }
  for (j in 1:nq){
    q[j] ~ dbeta(1,1)
  }
}
```

The code `TwentyQuestions.m` or `TwentyQuestions.R` makes inferences about the data in Table 6.1 using the model.

Exercises

Exercise 6.3.1 Draw some conclusions about how well the various people listened, and about the difficulties of the various questions. Do the marginal posterior distributions you are basing your inference on seem intuitively reasonable?

Exercise 6.3.2 Now suppose that three of the answers were not recorded, for whatever reason. Our new data set, with missing data, now takes the form shown in Table 6.2. Bayesian inference will automatically make predictions about these missing values (i.e., "fill in the blanks") by using the same probabilistic model that generated the observed data. Missing data are entered as `nan` ("not a number") in Matlab, and `NA` ("not available") in R or WinBUGS. Including the variable `k` as one to monitor when sampling will then provide posterior values for the missing values. That is, it provides information about the relative likelihood of the missing values being each of the possible alternatives, using the statistical model and the available data. Look through the Matlab or R code to see how all of this is implemented in the second data set. Run the code, and interpret the posterior distributions for the three missing values. Are they reasonable inferences?

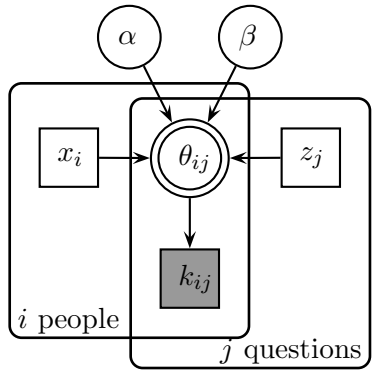
Table 6.2 Correct, incorrect, and missing answers for 10 people on 20 questions.

	Question																			
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Person 1	1	1	1	1	0	0	1	1	0	1	0	0	**?**	0	0	1	0	1	0	0
Person 2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 3	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
Person 4	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
Person 5	1	0	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	0	0
Person 6	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0
Person 7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Person 8	0	0	0	0	**?**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 9	0	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
Person 10	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	**?**	0	0

Exercise 6.3.3 The definition of the accuracy for a person on a question in terms of the product $\theta_{ij} = p_i q_j$ is very simple to understand, but other models of the interaction between person ability and question difficulty are used in psychometric models. For example, the Rasch model (e.g., Andrich, 1988) uses $\theta_{ij} = \exp(p_i - q_j) / (1 + \exp(p_i - q_j))$. Change the graphical model to implement the Rasch model.

6.4 The two-country quiz

Suppose a group of people take a historical quiz, and each answer for each person is scored as correct or incorrect. Some of the people are Thai, and some are Moldovan.



$$\alpha \sim \text{Uniform}(0,1) \quad \text{if home-country, P(correct)}$$
$$\beta \sim \text{Uniform}(0,\alpha) \quad \text{if away-country, P(correct)}$$
$$x_i \sim \text{Bernoulli}(0.5) \quad \text{P(home country)}$$
$$z_j \sim \text{Bernoulli}(0.5) \quad \text{P(home question)}$$
$$\theta_{ij} \leftarrow \begin{cases} \alpha & \text{if } x_i = z_j \\ \beta & \text{if } x_i \neq z_j \end{cases}$$
$$k_{ij} \sim \text{Bernoulli}(\theta_{ij})$$

Fig. 6.4 Graphical model for inferring the country of origin for people and questions.

Some of the questions are about Thai history, and it is more likely the answer would be known by a Thai person than a Moldovan. The rest of the questions are about Moldovan history, and it is more likely the answer would be known by a Moldovan than a Thai.

We do not know who is Thai or Moldovan, and we do not know the content of the questions. All we have are the data shown in Table 6.3. Spend some time just looking at the data, and try to infer which people are from the same country, and which questions relate to their country.

Table 6.3 Correct and incorrect answers for 8 people on 8 questions.

	Question							
	A	B	C	D	E	F	G	H
Person 1	1	0	0	1	1	0	0	1
Person 2	1	0	0	1	1	0	0	1
Person 3	0	1	1	0	0	1	0	0
Person 4	0	1	1	0	0	1	1	0
Person 5	1	0	0	1	1	0	0	1
Person 6	0	0	0	1	1	0	0	1
Person 7	0	1	0	0	0	1	1	0
Person 8	0	1	1	1	0	1	1	0

A good way to make these inferences formally is to assume there are two types of answers. For those where the nationality of the person matches the origin of the question, the answer will be correct with high probability. For those where a person is being asked about the other country, the answer will have a very low probability of being correct.

A graphical model that implements this idea is shown in Figure 6.4. The rate α is the (expected to be high) probability of a person from a country correctly

answering a question about their country's history. The rate β is the (expected to be low) probability of a person correctly answering a question about the other country's history. To capture the knowledge about the rates, the priors constrain $\alpha \geq \beta$, by defining `alpha` $\sim$ `dunif(0,1)` and `beta` $\sim$ `dunif(0,alpha)`. At first glance, this might seem inappropriate, since it specifies a prior for one parameter in terms of another (unknown, and being inferred) parameter. Conceptually, it is clearer to think of this syntax as a (perhaps clumsy) way to specify a *joint* prior over α and β in which the $\alpha \geq \beta$. Graphically, the parameter space over (α, β) is a unit square, and the prior being specified is the half of the square on one side of the diagonal line $\alpha = \beta$.

In the remainder of the graphical model, the binary indicator variable x_i assigns the ith person to one or other country, and z_j similarly assigns the jth question to one or other country. The probability the ith person will answer the jth question correctly is θ_{ij}, which is simply α if the country assignments match, and β if they do not. Finally, the actual data k_{ij} indicating whether or not the answer was correct follow a Bernoulli distribution with rate θ_{ij}.

The script `TwoCountryQuiz.txt` implements the graphical model in WinBUGS:

```
# The Two Country Quiz
model{
  # Probability of Answering Correctly
  alpha ~ dunif(0,1)    # Match
  beta ~ dunif(0,alpha) # Mismatch
  # Group Membership For People and Questions
  for (i in 1:nx){
    x[i] ~ dbern(0.5)
    x1[i] <- x[i]+1   seemingly, just to make x = {1,2}
  }
  for (j in 1:nz){
    z[j] ~ dbern(0.5)
    z1[j] <- z[j]+1
  }
  # Probability Correct For Each Person-Question Combination By Groups
  for (i in 1:nx){
    for (j in 1:nz){
      theta[i,j,1,1] <- alpha
      theta[i,j,1,2] <- beta      make a 2x2 matrix
      theta[i,j,2,1] <- beta      Alpha on diagonal
      theta[i,j,2,2] <- alpha     Beta on off-diagonal
    }
  }
  # Data Are Bernoulli By Rate
  for (i in 1:nx){
    for (j in 1:nz){
      k[i,j] ~ dbern(theta[i,j,x1[i],z1[j]])
    }
  }
}
```

The code `TwoCountryQuiz.m` or `TwoCountryQuiz.R` makes inferences about the data in Table 6.3 using the model.

Exercises

Exercise 6.4.1 Interpret the posterior distributions for `x[i]`, `z[j]`, `alpha`, and `beta`. Do the formal inferences agree with your original intuitions?

Exercise 6.4.2 The priors on the probabilities of answering correctly capture knowledge about what it means to match and mismatch, by imposing an order constraint $\alpha \geq \beta$. Change the code so that this information is not included, by using priors `alpha~dbeta(1,1)` and `beta~dbeta(1,1)`. Run a few chains against the same data, until you get an inappropriate, and perhaps counter-intuitive, result. The problem that is being encountered is known as model indeterminacy or label-switching. Describe the problem, and discuss why it comes about.

Exercise 6.4.3 Now suppose that three extra people enter the room late, and begin to take the quiz. One of them (Late Person 1) has answered the first four questions, the next (Late Person 2) has only answered the first question, and the final new person (Late Person 3) is still sharpening their pencil, and has not started the quiz. This situation can be represented as an updated data set, now with missing data, as in Table 6.4. Interpret the inferences the model makes about the nationality of the late people, and whether or not they will get the unfinished questions correct.

Table 6.4 Correct, incorrect, and missing answers for 8 people and 3 late people on 8 questions.

	Question							
	A	B	C	D	E	F	G	H
Person 1	1	0	0	1	1	0	0	1
Person 2	1	0	0	1	1	0	0	1
Person 3	0	1	1	0	0	1	0	0
Person 4	0	1	1	0	0	1	1	0
Person 5	1	0	0	1	1	0	0	1
Person 6	0	0	0	1	1	0	0	1
Person 7	0	1	0	0	0	1	1	0
Person 8	0	1	1	1	0	1	1	0
Late Person 1	1	0	0	1	?	?	?	?
Late Person 2	0	?	?	?	?	?	?	?
Late Person 3	?	?	?	?	?	?	?	?

Exercise 6.4.4 Finally, suppose that you are now given the correctness scores for a set of 10 new people, whose data were not previously available, but who form part of the same group of people we are studying. The updated data set is shown in Table 6.5. Interpret the inferences the model makes about the nationality of the new people. Revisit the inferences about the late people, and whether or not they will get the unfinished questions correct. Does the

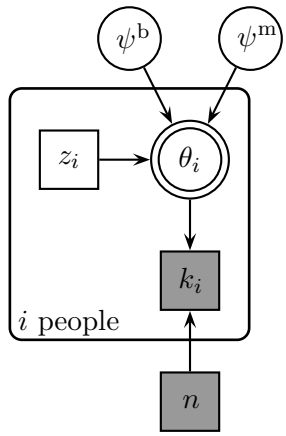
inference drawn by the model for the third late person match your intuition? There is a problem here. How could it be fixed?

Table 6.5 Correct, incorrect, and missing answers for 8 people, 3 late people, and 10 new people on 8 questions.

	Question							
	A	B	C	D	E	F	G	H
New Person 1	1	0	0	1	1	0	0	1
New Person 2	1	0	0	1	1	0	0	1
New Person 3	1	0	0	1	1	0	0	1
New Person 4	1	0	0	1	1	0	0	1
New Person 5	1	0	0	1	1	0	0	1
New Person 6	1	0	0	1	1	0	0	1
New Person 7	1	0	0	1	1	0	0	1
New Person 8	1	0	0	1	1	0	0	1
New Person 9	1	0	0	1	1	0	0	1
New Person 10	1	0	0	1	1	0	0	1
Person 1	1	0	0	1	1	0	0	1
Person 2	1	0	0	1	1	0	0	1
Person 3	0	1	1	0	0	1	0	0
Person 4	0	1	1	0	0	1	1	0
Person 5	1	0	0	1	1	0	0	1
Person 6	0	0	0	1	1	0	0	1
Person 7	0	1	0	0	0	1	1	0
Person 8	0	1	1	1	0	1	1	0
Late Person 1	1	0	0	1	**?**	**?**	**?**	**?**
Late Person 2	0	**?**	**?**	**?**	**?**	**?**	**?**	**?**
Late Person 3	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**

6.5 Assessment of malingering

Armed with the knowledge from the previous sections, we now consider the practical challenge of detecting if people cheat on a test. For example, people who have been in a car accident may seek financial compensation from insurance companies by feigning cognitive impairment such as pronounced memory loss. When these people are confronted with a memory test that is intended to measure the extent of their impairment, they may deliberately under-perform. This behavior is called malingering, and it may be accompanied by performance much worse than that displayed by real amnesiacs. Sometimes, for example, malingerers may perform substantially below chance.



$$\psi^{\mathrm{b}} \sim \mathrm{Uniform}(0.5, 1)$$ some skill level

$$\psi^{\mathrm{m}} \sim \mathrm{Uniform}(0, \psi^{\mathrm{b}})$$ malingerers perform below skill level

$$z_i \sim \mathrm{Bernoulli}(0.5)$$

$$\theta_i \leftarrow \begin{cases} \psi^{\mathrm{b}} & \text{if } z_i = 0 \\ \psi^{\mathrm{m}} & \text{if } z_i = 1 \end{cases}$$

$$k_i \sim \mathrm{Binomial}(\theta_i, n)$$

Fig. 6.5 Graphical model for the detection of malingering.

Malingering is not, however, always easy to detect, but is naturally addressed by latent-mixture modeling. Using this approach, it is possible to infer which of two categories—those who malinger, and those who are truthful or *bona fide*—each person belongs to, and quantify the confidence in each of these classifications.

We consider an experimental study on malingering, in which each of $p = 22$ participants completed a memory test (Ortega, Wagenmakers, Lee, Markowitsch, & Piefke, 2012). One group of participants was told to do their best. These are the *bona fide* participants. The other group of participants was told to under-perform by deliberately simulating amnesia. These are the malingerers. Out of a total of $n = 45$ test items, the participants get 45, 45, 44, 45, 44, 45, 45, 45, 45, 45, 30, 20, 6, 44, 44, 27, 25, 17, 14, 27, 35, and 30 correct. Because this was an experimental study, we know that the first 10 participants were *bona fide* and the next 12 were instructed to malinger.

The first analysis is straightforward, and uses the graphical model shown in Figure 6.5. We assume that all *bona fide* participants have the same ability, and so have the same rate ψ_{b} of answering each question correctly. For the malingerers, the rate of answering questions correctly is given by ψ_{m}, and $\psi_{\mathrm{b}} > \psi_{\mathrm{m}}$.

The script `Malingering_1.txt` implements the graphical model in WinBUGS:

```
# Malingering
model{
  # Each Person Belongs to One of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(0.5)
    z1[i] <- z[i]+1
  }
  # Bona Fide Group has Unknown Success Rate Above Chance
  psi[1] ~ dunif(0.5,1)
  # Malingering Group has Unknown Success Rate Below Bona Fide
  psi[2] ~ dunif(0,psi[1])
  # Data are Binomial with Group Rate for Each Person
  for (i in 1:p){
```

```
    theta[i] <- psi[z1[i]]
    k[i] ~ dbin(theta[i],n)
  }
}
```

Notice the restriction in the `dunif` definition of `psi[2]`, which prevents the indeterminacy or label-switching problem by ensuring that $\psi_b > \psi_m$.

The code `Malingering_1.m` or `Malingering_1.R` applies the model to the data.

Exercise

Exercise 6.5.1 What are your conclusions about group membership? Did all of the participants follow the instructions?

6.6 Individual differences in malingering

As before, it may seem restrictive to assume that all members of a group have the same chance of answering correctly. So, now we assume that the ith participant in each group has a unique rate of answering questions correctly, θ_i, which is constrained by group-level distributions. In Section 6.2, we used group-level Gaussians. The problem with that approach is that values can lie outside the range 0 to 1. These values were just censored in Section 6.2, but this is not quite technically correct, and is certainly not elegant.[1]

One of several alternatives is to assume that instead of being Gaussian, the group-level distribution is Beta(α, β). Because the Beta distribution is defined on the interval from 0 to 1 it respects the natural boundaries of rates. So we now have a model in which each individual binomial rate parameter is constrained by a group-level beta distribution. This complete model is known as the beta-binomial (e.g., Merkle, Smithson, & Verkuilen, 2011; J. B. Smith & Batchelder, 2010).

It is useful to transform the α and β parameters from the beta distribution to a group mean $\mu = \alpha/(\alpha + \beta)$ and a measure $\lambda = \alpha + \beta$ that can be conceived of as a precision, in the sense that as it increases the variability of the distribution decreases. It is then straightforward to assign uniform priors to both μ_b, the group-level mean for the *bona fide* participants, and μ_m, the group-level mean for the malingerers. This assignment does not, however, reflect our knowledge that $\mu_b > \mu_m$. To capture this knowledge, we could define `dunif(0,mubon)`, as done in the previous model.

However, for this model we apply a different approach. We first define μ_m as the additive combination of μ_b and a difference parameter, so that $\text{logit}(\mu_m) = \text{logit}(\mu_b) - \mu_d$. Note that this is an additive combination on the logit scale,

[1] WinBUGS conceptually conflates censoring and truncation in the `I(,)` notation, which is the cause of the technical problem. JAGS has the advantage of dealing with these two related concepts coherently.

does Dan Y. know what this means? maybe yes.

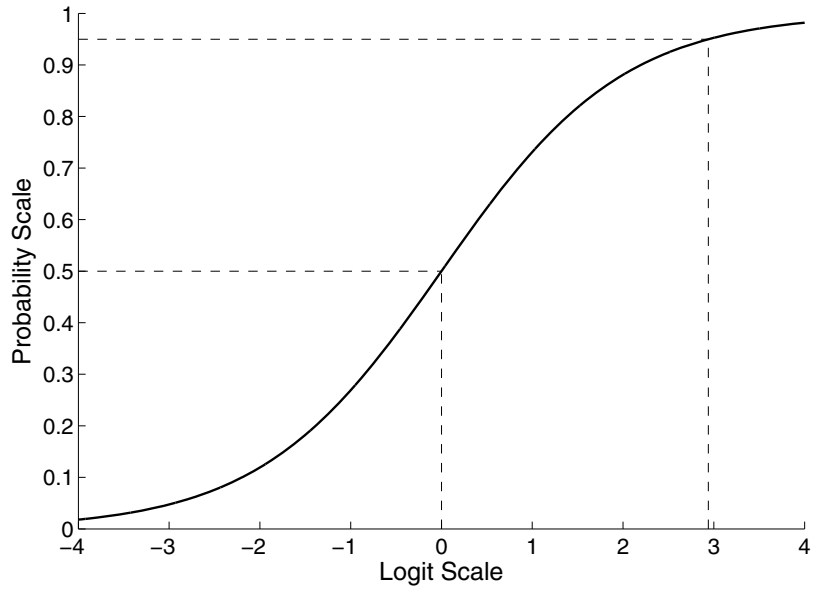


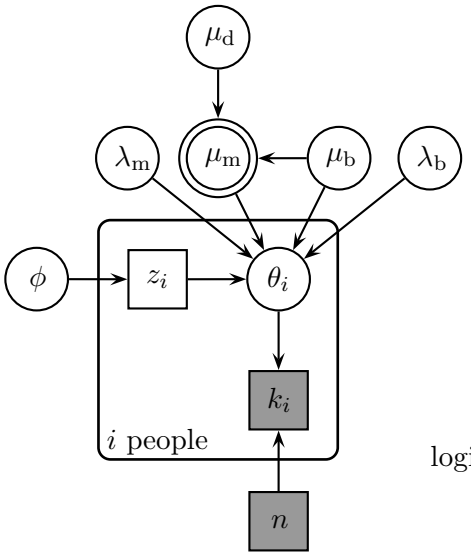
Fig. 6.6 The logit transformation. Probabilities range from 0 to 1 and are mapped to the entire set of real numbers using the logit transform.

as is customary in beta-binomial models. The logit transformation is defined as $\text{logit}(\theta) \equiv \ln(\theta/(1-\theta))$ and it transforms values on the rate scale, ranging from 0 to 1, to values on the logit scale, ranging from $-\infty$ to ∞. The logit transformation is shown in Figure 6.6, including two specific examples with the logit value 0 corresponding to probability 0.5, and the logit probability 2.94 corresponding to probability 0.95.

The prior for $\mu_{\text{d}} \sim \text{Gaussian}\big(0, 0.5\big)_{\mathcal{I}(0,\infty)}$ is a positive-only Gaussian distribution. This ensures that the group mean of the *bona fide* participants is always larger than that of the malingerers. Finally, note that the base rate of malingering ϕ, which was previously fixed to 0.5, is now assigned a relatively wide beta prior distribution that is centered around 0.5. This means the model uses the data to infer group membership and at the same time learn about the base rate.

A graphical model that implements the above ideas is shown in Figure 6.7. The script `Malingering_2.txt` implements the graphical model in WinBUGS:

```
# Malingering, with Individual Differences
model{
  # Each Person Belongs to One of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(phi) # phi is the Base Rate
    z1[i] <- z[i]+1
  }
  # Relatively Uninformative Prior on Base Rate
  phi ~ dbeta(5,5)
  # Data are Binomial with Rate Given by
  # Each Person's Group Assignment
```



$$\mu_b \sim \text{Beta}(1,1)$$ bona fide group mean, unif

$$\mu_d \sim \text{Gaussian}(0,0.5)_{\mathcal{I}(0,\infty)}$$ B - M, positive, gaussian

$$\lambda_b \sim \text{Uniform}(40,800)$$ higher precision for bona fide?

$$\lambda_m \sim \text{Uniform}(4,100)$$ lower precision for malingerer?

$$z_i \sim \text{Bernoulli}(\phi)$$ group membership, bernouilli(base_rate)

$$\theta_i \sim \begin{cases} \text{Beta}(\mu_b\lambda_b, (1-\mu_b)\,\lambda_b) & \text{if } z_i = 0 \\ \text{Beta}(\mu_m\lambda_m, (1-\mu_m)\,\lambda_m) & \text{if } z_i = 1 \end{cases}$$ bona fide coin / malingerer coin

$$k_i \sim \text{Binomial}(\theta_i, n)$$

$$\text{logit}\mu_m \leftarrow \text{logit}\mu_b - \mu_d$$ logit (malingerer group mean)

$$\phi \sim \text{Beta}(5,5)$$ prior on group membership base_rate; peaked at 0.5

Fig. 6.7 Graphical model for inferring membership of two latent groups, consisting of malingerers and *bona fide* participants.

```
  for (i in 1:p){
    k[i] ~ dbin(theta[i,z1[i]],n)
    theta[i,1] ~ dbeta(alpha[1],beta[1])
    theta[i,2] ~ dbeta(alpha[2],beta[2])
  }
  # Transformation to Group Mean and Precision
  alpha[1] <- mubon * lambdabon
  beta[1] <- lambdabon * (1-mubon)
  # Additivity on Logit Scale
  logit(mumal) <- logit(mubon) - mudiff
  alpha[2] <- mumal * lambdamal
  beta[2]  <- lambdamal * (1-mumal)
  # Priors
  mubon ~ dbeta(1,1)
  mudiff ~ dnorm(0,0.5)I(0,) # Constrained to be Positive
  lambdabon ~ dunif(40,800)
  lambdamal ~ dunif(4,100)
}
```

The code `Malingering_2.m` or `Malingering_2.R` allows you to draw conclusions about group membership and the success rate of the two groups.

Exercises

Exercise 6.6.1 Is the inferred rate of malingering consistent with what is known about the instructions given to participants?

Exercise 6.6.2 Assume you know that the base rate of malingering is 10%. Change the WinBUGS script to reflect this knowledge. Do you expect any differences?

Exercise 6.6.3 Assume you know for certain that participants 1, 2, and 3 are *bona fide*. Change the code to reflect this knowledge.

Exercise 6.6.4 Suppose you add a new participant. What number of questions answered correctly by this participant would lead to the greatest uncertainty about their group membership?

Exercise 6.6.5 Try to solve the label-switching problem by using the `dunif(0,mubon)` approach instead of the logit transform.

Exercise 6.6.6 Why are the priors for λ_b and λ_m different?

6.7 Alzheimer's recall test cheating

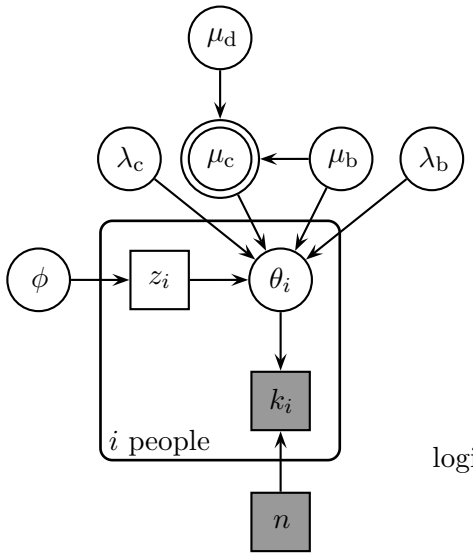
In this section, we apply the same latent-mixture model shown in Figure 6.7 to different memory test data. Simple recognition and recall tasks are an important part of screening for Alzheimer's Disease and Related Disorders (ADRD), and are sometimes administered over the telephone. This practice raises the possibility of people cheating by, for example, writing down the words they are being asked to remember.

The data we use come from an informal experiment, in which 118 people were either asked to complete the test normally, or instructed to cheat. The particular test used was a complicated sequence of immediate and delayed free recall tasks, which we simplify to give a simple score correct out of 40 for each person. By design, there are 61 *bona fide* people who are known to have done the task as intended, and 57 people who are known to have cheated.

This graphical model is shown in Figure 6.8, and is essentially the same as for the previous example on malingering in Figure 6.7. It changes the names of variables from malingering to cheating as appropriate, uses different priors on the precisions of the group distributions, and makes the mean of accuracy rate for the cheaters *higher* than that of the *bona fide* people, since the impact of cheating is to recall more words than would otherwise be the case.

The script `Cheating.txt` implements the analysis in WinBUGS:

```
# Cheating Latent-Mixture Model
model{
  # Each Person Belongs to One of Two Latent Groups
  for (i in 1:p){
     z[i] ~ dbern(phi) # phi is the Base Rate
     z1[i] <- z[i]+1
  }
  # Relatively Uninformative Prior on Base Rate
  phi ~ dbeta(5,5)
  # Data are Binomial with Rate Given by
  # Each Person's Group Assignment
  for (i in 1:p){
```



$$
\begin{aligned}
\mu_{\mathrm{b}} &\sim \mathrm{Beta}(1,1) \\
\mu_{\mathrm{d}} &\sim \mathrm{Gaussian}(0,0.5)_{\mathcal{I}(0,\infty)} \\
\lambda_{\mathrm{b}} &\sim \mathrm{Uniform}(5,50) \\
\lambda_{\mathrm{c}} &\sim \mathrm{Uniform}(5,50) \\
z_i &\sim \mathrm{Bernoulli}(\phi) \\
\theta_i &\sim \begin{cases} \mathrm{Beta}(\mu_{\mathrm{b}}\lambda_{\mathrm{b}}, (1-\mu_{\mathrm{b}})\,\lambda_{\mathrm{b}}) & \text{if } z_i = 0 \\ \mathrm{Beta}(\mu_{\mathrm{c}}\lambda_{\mathrm{c}}, (1-\mu_{\mathrm{c}})\,\lambda_{\mathrm{c}}) & \text{if } z_i = 1 \end{cases} \\
k_i &\sim \mathrm{Binomial}(\theta_i, n) \\
\mathrm{logit}\mu_{\mathrm{c}} &\leftarrow \mathrm{logit}\mu_{\mathrm{b}} + \mu_{\mathrm{d}} \\
\phi &\sim \mathrm{Beta}(5,5)
\end{aligned}
$$

Fig. 6.8 Graphical model for inferring membership of two latent groups, consisting of cheaters and *bona fide* people in a memory test.

```
    k[i] ~ dbin(theta[i,z1[i]],n)
    thetatmp[i,1] ~ dbeta(alpha[1],beta[1])
    theta[i,1] <- max(.01,min(.99,thetatmp[i,1]))
    thetatmp[i,2] ~ dbeta(alpha[2],beta[2])
    theta[i,2] <- max(.01,min(.99,thetatmp[i,2]))
  }
  # Transformation to Group Mean and Precision
  alpha[1] <- mubon * lambdabon
  beta[1] <- lambdabon * (1-mubon)
  # Additivity on Logit Scale
  logit(muche) <- logit(mubon) + mudiff # Note the "+"
  alpha[2] <- muche * lambdache
  beta[2] <- lambdache * (1-muche)
  # Priors
  mubon ~ dbeta(1,1)
  mudiff ~ dnorm(0,0.5)I(0,) # Constrained to be Positive
  lambdabon ~ dunif(5,40)
  lambdache ~ dunif(5,40)
  # Correct Count
  for (i in 1:p){
    pct[i] <- equals(z[i],truth[i])
  }
  pc <- sum(pct[1:p])
}
```

Note that the script includes a variable `pc` that keeps track of the accuracy of each classification made in sampling by comparing each person's latent assignment to the known truth from the experimental design.

The code `Cheating.m` or `Cheating.R` applies the graphical model to the data. We focus our analysis of the results firstly on the classification accuracy of the

Box 6.3 Undefined real result

In WinBUGS, error messages are called traps, and some traps are more serious than others. One of the more serious regularly occurring traps is "undefined real result." This trap indicates numerical overflow or underflow caused by a sample with very low likelihood. This can happen when you have not specified your model well enough. In particular, the prior distribution for a standard deviation may be too wide, thus allowing extreme values that are highly unlikely in light of the data (and, most often, also unlikely in light of prior knowledge). Initial values may also be to blame, and this is why it can be better to specify those yourself instead of having WinBUGS pick them automatically. Although the "undefined real result" trap can be a nuisance, in the end it may actually help you improve your model.

model. The top panel of Figure 6.9 summarizes the data, showing the distribution of correctly recalled words in both the *bona fide* and cheater groups. It is clear that cheaters generally recall more words, but that there is overlap between the groups.

One way to provide a benchmark classification accuracy is to consider the best possible cut-off. This is a total correct score below which a person is classified as *bona fide*, and at or above which they are classified as a cheater. The line in the bottom panel in Figure 6.9 shows the classification accuracy for all possible cut-offs, which peaks at 86.4% accuracy using the cut-off of 35. The gray distribution at the left of the panel is the posterior distribution of the `pc` variable, showing the range of accuracy achieved by the latent-mixture model.

Using a generative model to solve classification problems is unlikely to work as well as the best discriminative methods from machine learning and statistics. This is not because of failings of the Bayesian approach, but because the models we develop are imperfect accounts of how data are generated. If the focus is purely on prediction, other statistical approaches, including especially ones that combine the best aspects of generative and discriminative modeling, may be superior (e.g., Lasserre, Bishop, & Minka, 2006).

The advantage of the generative model is in providing details about the underlying processes assumed to produce the data, particularly by quantifying uncertainty. A good example of this important feature is shown in Figure 6.10, which shows the relationship between the total correct raw data score, and the posterior uncertainty about classification as a cheater, for each person. The broken lines connecting 35 people and a classification probability of 0.5 shows that the model infers people with scores above 35 as more likely than not to be cheaters. But it also shows how certain the model is about each classification, which provides more information (and more probabilistically coherent information) than many machine-learning methods.

This information about uncertainty is useful, for example, if there are costs or utilities associated with different classification decisions. Suppose that raising a

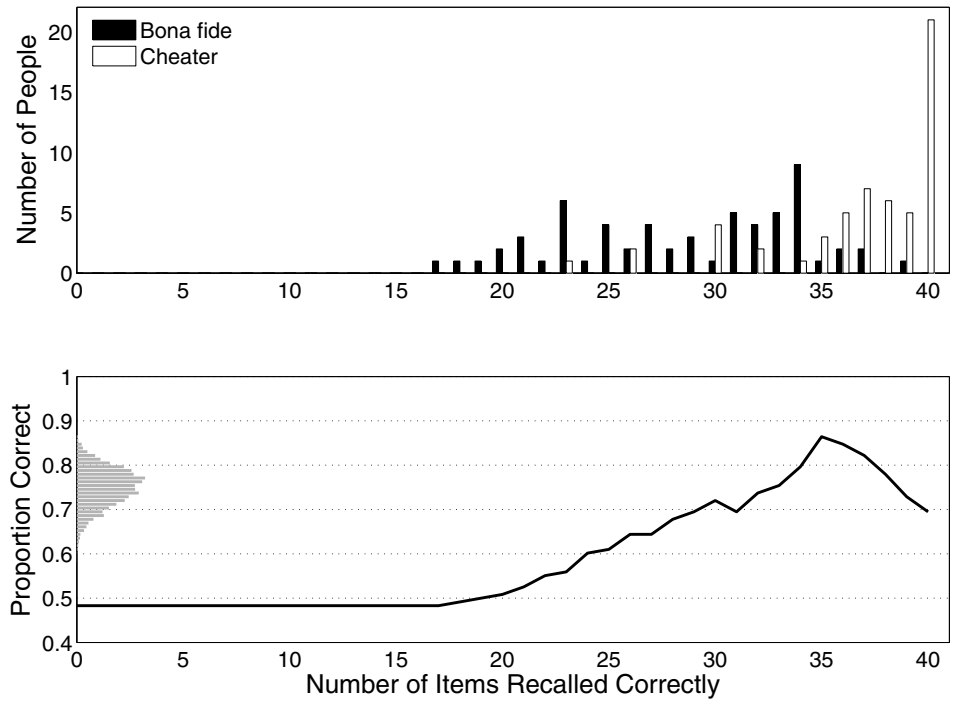


Fig. 6.9 The distribution of total correct recall scores for the Alzheimer's data, and classification performance. The top panel shows the distribution of scores for the *bona fide* and cheater groups. The bottom panel shows, with the line, the accuracy achieved using various cut-offs to separate the groups, and, with the distribution, the accuracy achieved by the latent-mixture model.

false-alarm and suspecting someone of cheating on the screening test costs \$25, perhaps through a wasted follow-up-in-person test, but that missing someone who cheated on the screening test costs \$100, perhaps through providing insurance that should have been withheld. With these utilities, the decision should be to classify people as *bona fide* only if it is four times more likely than them being a cheater. In other words, we need 80% certainty they are *bona fide*. The posterior distribution of the latent assignment variable $\mathbf{z}$ provides exactly this information. Under this set of utilities, as shown by the broken lines connecting 30 people and a classification probability of 0.2 in Figure 6.10, only people with a total correct score below 30 (not 35) should be treated as *bona fide*.

Exercises

Exercise 6.7.1 Suppose the utilities are very different, so that a false alarm costs \$100, because of the risk of litigation in a false accusation, but misses are relatively harmless, costing \$10 in wasted administrative costs. What decisions should be made about *bona fide* and cheating people now?

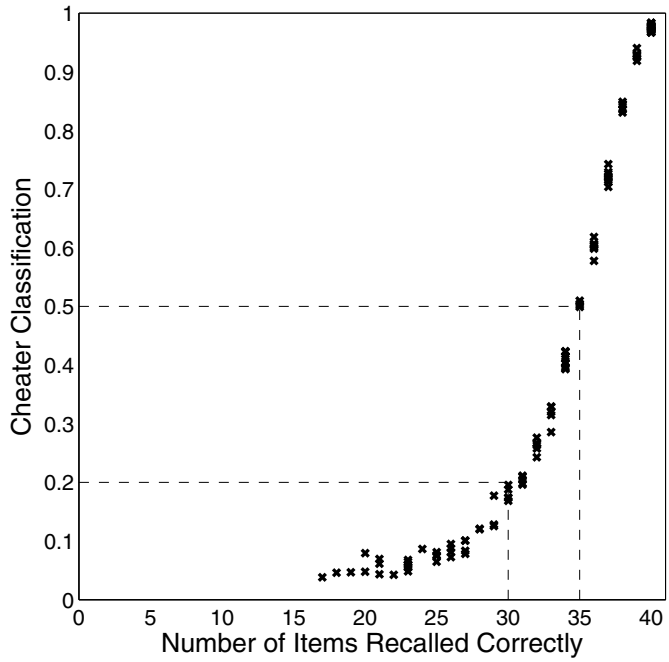


Fig. 6.10 The relationship between the number of items recalled correctly, and the posterior classification as belonging to the cheater group. Each cross corresponds to a person.

Exercise 6.7.2 What other potential information, besides the uncertainty about classification, does the model provide? Give at least one concrete example.