1. Introduction

In teaching Bayesian thinking, many authors advocate beginning with a "discrete model" approach, where one lists a plausible list of parameter values and then assigns prior probabilities to these parameter values. This approach has several advantages. First, it is relatively easy to specify a prior probability distribution on a set of values, and second, it is straightforward to compute the posterior and predictive distributions by multiplying and summing prior and likelihood values. This discrete model approach is common in introductory Bayesian texts such as Antleman (1997), Schmitt (1969), Berry (1995), Rossman and Albert (2000), and Bolstad (2007).

One typically illustrates this discrete model approach for standard sampling distributions such as the binomial (unknown proportion), Poisson (unknown mean), and normal (unknown mean and known standard deviation). All of these familiar sampling distributions are part of R, and so this suggests that one can easily implement this discrete model approach in R. There are several functions currently available in R packages that implement discrete priors for specific sampling problems such as `pdisc` for binomial sampling in the `LearnBayes` package and `poisdp` and `normdp` for Poisson and normal sampling in the `Bolstad` package. These particular functions are discussed respectively in Albert (2009) and Bolstad (2007).

There are some features of R that are helpful in writing a general-purpose function for implementing discrete Bayes. A prior can be represented by a single vector, say `prior`, where the elements of `prior` are the probabilities and the names of `prior` contain the parameter values. A R function can accept sampling density functions such as `dnorm`, `dgamma`, and `dbinom` as input arguments. Last, once a discrete Bayes function is written, one can define `print`, `plot`, and `summary` methods for this function to graph, display, and summarize the posterior probability vector in specific ways.

To encourage the teaching of the discrete Bayesian model approach for a wide range of inference problems, this paper illustrates the use of several generic R functions for performing Bayesian calculations for one and two parameter problems. In Section 2, we illustrate the use of a single function `discrete.bayes` that can be used to obtain posterior probabilities and the predictive probability. The main inputs to this function are the sampling density and the prior distribution. In Section 3, a similar function `discrete.bayes.2` is described that implements Bayesian calculations for arbitrary two parameter problems.

2. Discrete Bayes with One Parameter

Suppose we observe a sample $y_1, ..., y_n$ from a sampling density density $f(y|\theta)$ depending

on a single unknown parameter θ. Further suppose that there are k plausible values of θ, $\theta_1, ..., \theta_k$ with respective prior probabilities $P(\theta_1), ..., P(\theta_k)$. The joint probability function of $\theta_j, y = (y_1, ..., y_n)$ is given by the product

$$f(y, \theta_j) = P(\theta_j) \times \prod_{i=1}^{n} f(y_i|\theta_j).$$

We can rewrite this joint probability as

$$f(y, \theta_j) = \left[\frac{P(\theta_j) \prod_{i=1}^{n} f(y_i|\theta_j)}{\sum_{m=1}^{k} P(\theta_m) \prod_{i=1}^{n} f(y_i|\theta_m)}\right] \times \left[\sum_{m=1}^{k} P(\theta_m) \prod_{i=1}^{n} f(y_i|\theta_m)\right].$$

The first term in brackets represents the posterior density of the parameter value θ_j, $P(\theta_j|y)$. The second bracketed term represents the (prior) predictive probability of y, $f(y)$. Both terms are important in a Bayesian analysis. The posterior probabilities $\{P(\theta_j|y)\}$ are useful for performing inference about the parameter, and the predictive probability is useful for assessing the suitability of the Bayesian model and for comparing Bayesian models by means of Bayes factors.

The syntax for our R function has the form

```
discrete.bayes(df,prior,y,...)
```

There are four arguments:

1. `df` is the name of the function defining the sampling density.
2. `prior` is a vector that defines the prior density. The names of the elements of the vector define the parameter values and the entries of the vector give the prior probabilities.
3. `y` is a vector of data values.
4. ... define any further fixed parameter values used in the function.

The output of `discrete.bayes` is a list with two components:

1. `prob` is a vector containing the posterior probabilities $\{P(\theta_j|y)\}$
2. `pred` is a scalar with the prior predictive probability $f(y)$

The output is assigned to the R class `bayes` and `print`, `plot`, and `summary` methods have been assigned to this class. The `print` method will display only the posterior probabilities and the `plot` method will construct a bar graph of the probabilities. The `summary` method will compute the posterior mean and standard deviation and will also display a "highest probability content" interval estimate for the parameter.

2.1. Learning about a proportion

As a first example, suppose one wishes to learn about a baseball player's probability of getting a hit p. I believe that a reasonable set of probabilities are 0.20, 0.21, ..., 0.36 and I assign these probabilities the corresponding weights 1, 1, 1, 2, 2, 2, 4, 4, 4, 4, 2, 2, 2, 2, 1, 1, 1. In R, I create a vector `prior` with values 1, 1, 1, 2, 2, 2, ..., and I name the probability entries with the proportion values. The probability vector is normalized by dividing by its sum.

```
> options(digits = 4)
> options(width = 60)
> prior = c(1, 1, 1, 2, 2, 2, 4, 4, 4, 4, 2, 2,
+     2, 2, 1, 1, 1)
> names(prior) = seq(0.2, 0.36, by = 0.01)
> prior = prior/sum(prior)
> prior
```

```
    0.2     0.21     0.22     0.23     0.24     0.25     0.26
0.02778  0.02778  0.02778  0.05556  0.05556  0.05556  0.11111
   0.27     0.28     0.29      0.3     0.31     0.32     0.33
0.11111  0.11111  0.11111  0.05556  0.05556  0.05556  0.05556
   0.34     0.35     0.36
0.02778  0.02778  0.02778
```

The name of the sampling density is the binomial density `dbinom` that corresponds to the sampling density

$$f(y|p) = \binom{n}{y} p^y (1-p)^{n-y}.$$

I observe the player's hitting performance for four periods of 80 at-bats (opportuntities) – for these four periods, he is 20 for 80, 22 for 80, 19 for 80, and 29 for 80. I place the hit counts in the vector `y` and the sample sizes in the vector `n`.

```
> y = c(20, 22, 19, 29)
> n = c(80, 80, 80, 80)
```

We obtain the posterior probabilities and the predictive probability by using `discrete.bayes` with arguments `dbinom`, `prior`, and `y`. We add the additional argument `size=n` which is the vector of fixed sample sizes used in the function `dbinom`.

```
> library(LearnBayes)
> out = discrete.bayes(dbinom, prior, y, size = n)
> print(out)
```

```
       0.2       0.21       0.22       0.23       0.24       0.25
0.0001211  0.0005415  0.0019028  0.0106905  0.0243648  0.0456369
```

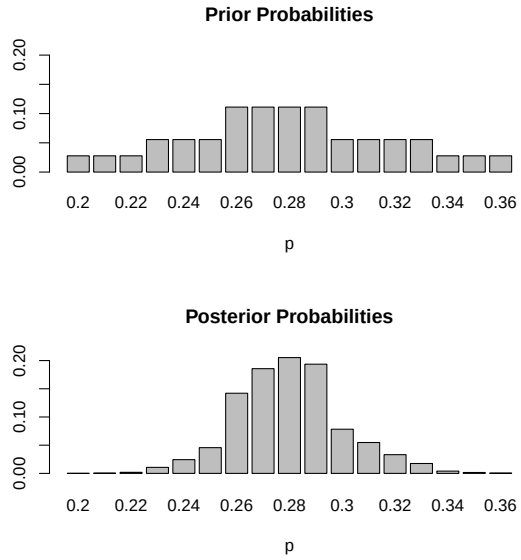

Figure 1: Prior and posterior distributions of a binomial proportion using a discrete prior.

```
     0.26      0.27      0.28      0.29       0.3      0.31
0.1420861 0.1856314 0.2052824 0.1935953 0.0783586 0.0547606
     0.32      0.33      0.34      0.35      0.36
0.0332026 0.0175422 0.0040532 0.0016436 0.0005864
```

The posterior probabilities are stored in the vector `out$prob`. We display the prior and posterior probabilities in Figure 1 using two applications of the `barplot` function.

```
> par(mfrow = c(2, 1))
> barplot(prior, main = "Prior Probabilities", xlab = "p",
+     ylim = c(0, 0.2))
> barplot(out$prob, main = "Posterior Probabilities",
+     xlab = "p")
```

This graph clearly shows how one's prior beliefs are updated on the basis of this hitting data.

2.2. Learning about the number of successes in a finite population

There are a variety of sampling densities included in the base package of R that can be used in `discrete.bayes`. It is easy to make slight modifications to these functions for specific problems that don't quite fit within these function definitions. To illustrate this situation, suppose a small community has 100 voters and one is interested in estimating the number in favor of a school levy. One takes a random sample of 20 voters without replacement from the population and 12 support the levy. One is interested in learning about the unknown

number M in the population who are supportive. Here the likelihood of M is given by the hypergeometric probability

$$L(M) = \text{Prob}(12/20 \text{ in support}|M) = \frac{\binom{M}{12}\binom{100-M}{8}}{\binom{100}{20}}.$$

Since the R function `dhyper` uses a different parameterization, we define the new function `dhyper2` that gives the probability of x successes in a sample of n with a population size of N and population number of successes M.

```
> dhyper2=function(x,M,sample.size=n,pop.size=N)
+  dhyper(x,M,N-M,n)
```

In this finite population example, there are 101 possible values for the population number of levy supporters M from 0 to 100 and we assign a uniform prior on these values. Using the `rep` function, we place the probabilities 1/101, ..., 1/101 in the vector `M` and assign the names 0, ..., 100 to these probabilities.

```
> prior=rep(1/101,101)
> names(prior)=0:100
```

We define the given values of the sample size n, population size N, and observed number of sample successes x. Then we use the function `discrete.bayes` with inputs the hypergeometric sampling function `dhyper2`, the prior density `prior` and the observed data `x` to update the posterior probabilities.

```
> n=20; x=12; N=100
> s=discrete.bayes(dhyper2,prior,x,
+  sample.size=n,pop.size=N)
```

The `plot` method is used to graph the posterior probabilities of M displayed in Figure 2.

```
> par(mfrow=c(1,1))
> plot(s,xlab="M",ylab="Probability")
```

The `summary` method computes the posterior mean and standard deviation of M and outputs a 90% interval estimate for the parameter. In this example, this interval includes values small than 50, so there is insufficient evidence to conclude that a majority of the voters support the levy.

```
> summary(s)
```

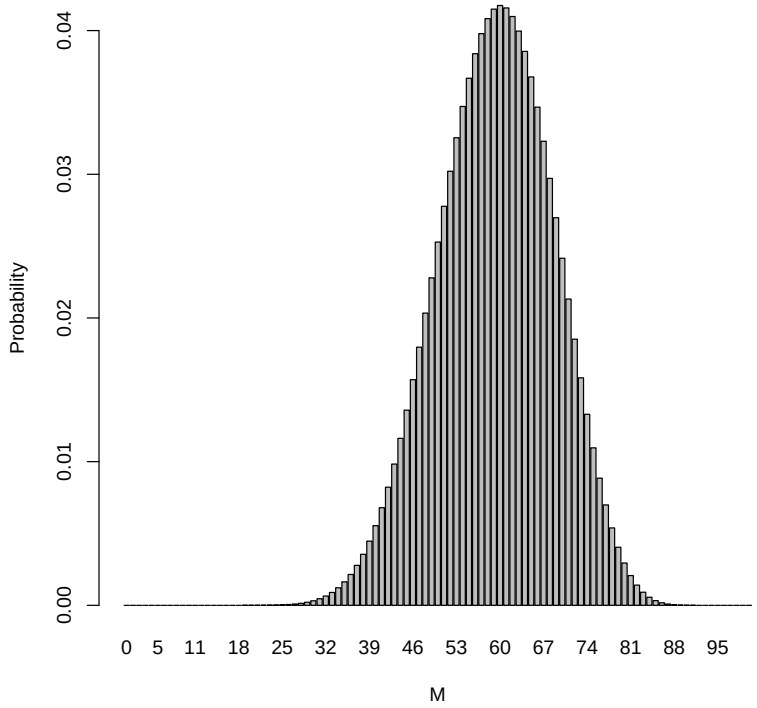

Figure 2: Posterior probabilities of the number M in support of the levy in the population.

```
$mean
[1] 59.27

$sd
[1] 9.26

$coverage
[1] 0.9061

$set
 [1] 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
[19] 62 63 64 65 66 67 68 69 70 71 72 73 74
```

2.3. Learning about a Poisson rate

As a third example, suppose you observe the number of home runs y in a specific number of opportunities n. Since home runs are rare events, it is common to assume that y has a

Poisson distribution with mean $n\lambda$ where λ corresponds to the true home run rate:

$$f(y|\lambda) = \frac{(n\lambda)^y \exp(-n\lambda)}{y!},\ y = 0, 1, ...$$

The R Poisson density function `dpois` only depends on the mean λ. But we can write a new function `dpois2` with an extra input n corresponding to the interval size.

```
> dpois2 = function(y, lambda, n = 1) dpois(y, n *
+     lambda)
```

Now we can use the function `discrete.bayes` using the sampling density defined in `dpois2`. We initially believe the home run rate λ can be one of the size values 0.02, 0.04, 0.06, 0.08, 0.10, 0.12 with probabilities 0.1, 0.1, 0.2, 0.3, 0.2, 0.1.

```
> prior = c(0.1, 0.1, 0.2, 0.3, 0.2, 0.1)
> names(prior) = seq(0.02, 0.12, by = 0.02)
```

We observe 20 home runs in 400 at-bats and update our probabilities using `discrete.bayes`. We display and plot the posterior probabilities of the rates in Figure 3.

```
> y = 20
> n = 400
> out = discrete.bayes(dpois2, prior, y, n = 400)
> print(out)
```

```
     0.02      0.04      0.06      0.08       0.1      0.12
7.908e-04 2.782e-01 6.206e-01 9.848e-02 1.910e-03 1.228e-05
```

```
> plot(out, main = "Posterior Probabilities",
+     xlab = "lambda")
```

We are pretty confident the player's true home run rate λ is between 0.04 and 0.06.

2.4. Binomial or Poisson sampling?

One can use discrete models to illustrate model comparison by the use of values from the prior predictive distribution. In the previous example, we illustrated a Poisson sampling model for home run data. Suppose instead that we let y be binomial with parameters n and p. We assign the same prior on p as we used for λ in the Poisson example.

```
> prior = c(0.1, 0.1, 0.2, 0.3, 0.2, 0.1)
> names(prior) = seq(0.02, 0.12, by = 0.02)
```

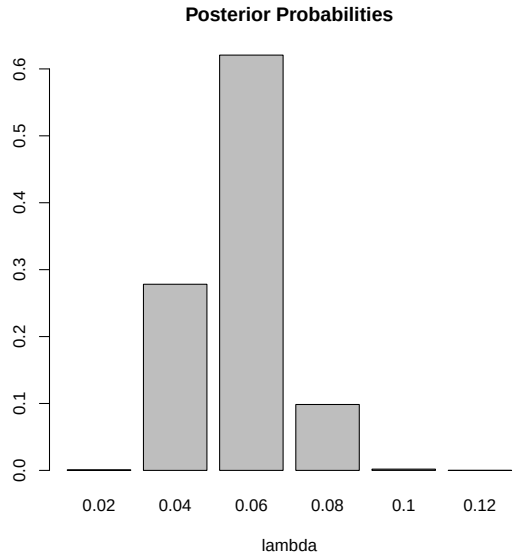

Figure 3: Posterior distribution of a Poisson rate for home run example.

Then we update the probabilities using `discrete.bayes` with sampling density `dbinom`:

```
> y = 20
> n = 400
> out.binom = discrete.bayes(dbinom, prior, y, size = n)
> print(out.binom)
```

```
     0.02      0.04      0.06      0.08       0.1      0.12
6.822e-04 2.829e-01 6.310e-01 8.427e-02 1.150e-03 4.309e-06
```

We display the values of the two predictive probabilities from the Poisson and binomial models and compute the Bayes factor in support of the binomial model.

```
> c(out$pred, out.binom$pred)
```

```
[1] 0.02010 0.01986
```

```
> BF = out.binom$pred/out$pred
> BF
```

```
[1] 0.988
```

The Bayes factor is close to one, indicating that the two sampling models using the same prior beliefs are very similar.

The prior beliefs of this particular person are concentrated about $p = .08$, indicating that he or she believes that this player is a strong home run hitter. Suppose a second person believes the player is more ordinary and assigns the following prior.

```
> prior.new = c(0.2, 0.3, 0.2, 0.1, 0.1, 0.1)
> names(prior.new) = seq(0.02, 0.12, by = 0.02)
```

We compute the predictive probability from this new prior model using `discrete.bayes`:

```
> y = 20
> n = 400
> out.binom2 = discrete.bayes(dbinom, prior.new,
+     y, size = n)
> BF = out.binom2$pred/out.binom$pred
> BF

[1] 1.510
```

Here the Bayes factor is 1.510, indicating that the new prior is more consistent with the observed home run rate $y/n = 0.05$ than the initial prior.

3. Discrete Bayes with Two Parameters

3.1. The R function

The discrete model approach can be extended easily for problems, such as normal sampling or the comparison of Poisson rates, where there are two unknown parameters. A prior distribution is now represented by a matrix, where the rows and columns are labeled with the values of the two parameters and the entries of the matrix correspond to the prior probabilities.

The syntax for the R function has the form

```
discrete.bayes.2(df,prior,y,...)
```

There are four arguments:

1. `df` is the name of the function defining the sampling density with two parameters.
2. `prior` is a matrix that defines the prior density. The row names and column names of the elements of the vector define respectively the names of parameter 1 and parameter 2 and the entries of the matrix give the prior probabilities.

3. `y` is a matrix of data values, where each row corresponds to a single observation.
4. ... define any further fixed parameter values used in the function.

The output of `discrete.bayes.2` is a list with two components:

1. `prob` is a matrix containing the posterior probabilities
2. `pred` is a scalar with the prior predictive probability

The output is assigned to the R class `bayes2`. A `plot` method assigned to this class uses the `image` function to display the probability matrix as gray rectangles where higher probabilities correspond to darker rectangles.

3.2. Learning about two proportions

Berry (1996) illustrates using discrete prior models to learn about the equality of two proportions. In Example 8.5, he is interested in seeing if a particular basketball player Ford has a "hot hand". Suppose p_1 denotes the probability that Ford is successful in shooting a free throw following a miss and p_2 denotes the probability the Ford is successful following a successful shot. Berry is interested in estimating the proportions given the prior information that there is no hot hand and the proportions p_1 and p_2 are equal.

We observe y_1, y_2 independent where y_1 is binomial(n_1, p_1), y_2 is binomial(n_2, p_2). We're interested in testing the hypothesis that $p_1 = p_2$. The sampling density is programmed using the following R function `twoproplike`.

```
> twoproplike = function(y, p1, p2, size1, size2) dbinom(y[1],
+     size = size1, prob = p1) * dbinom(y[2], size = size2,
+     prob = p2)
```

Berry considers the following "testing" prior for this situation. Each proportion can be one of the nine values 0.1, ..., 0.9. Based on some survey data, Berry assumes the prior probability that $p_1 = p_2$ is equal to 45/108, so the probability that the proportions are different is $1 - 45/108$. Given this constraint, each proportion pair in the set $\{p_1 = p_2\}$ and the set $\{p_1 \neq p_2\}$ is assigned the same probability.

We set up a prior probability matrix `prior` the reflects the above information. The row names for `prior` are the values of p_1, and the column names are values of p_2.

```
> p1 = seq(0.1, 0.9, length = 9)
> p2 = p1
> prior = matrix(0, 9, 9) + 7/864
> diag(prior) = 5/108
> dimnames(prior)[[1]] = p1
> dimnames(prior)[[2]] = p2
```

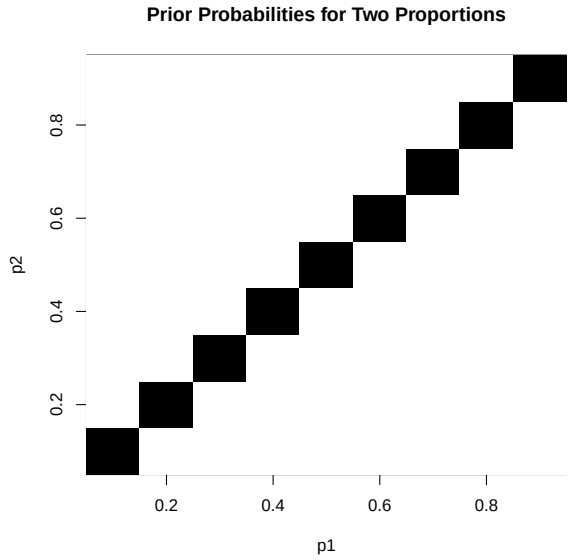

Figure 4: Image display of the prior distribution for two proportions in testing problem.

We graph the prior below first using `discrete.bayes.2` with no data to output the probabilities in a special R class. A `plot` method for this class is written for this class that uses the R base function `image` with gray colors where darker colors correspond to higher probabilities. Using the `plot` method for this class, one obtains an image plot of this prior probability matrix in Figure 4. It is clear from this figure that the prior is concentrated along the diagonal where the proportions are equal.

```
> out=discrete.bayes.2(twoproplike,prior)
> plot(out,xlab="P1",ylab="P2",
+  main = "Prior Probabilities")
```

The data is $(y_1, n_1) = (17, 22)$ and $(y_1, n_2) = (36, 51)$, indicating that Ford was 17 for 22 following a miss and 36 for 51 following a successful shot. The values y_1 and y_2 are stored in the 2 by 1 matrix `y`. We use the function `discrete.bayes.2` using the sampling density `twoproplike`, the prior probability matrix `prior` and the data matrix `y`. The fixed binomial sample sizes are indicated through the arguments `size1 = 22, size2 = 51`.

```
> y = matrix(c(17, 36), 1, 2, byrow = TRUE)
> out = discrete.bayes.2(twoproplike, prior, y, size1 = 22,
+     size2 = 51)
```

The posterior probabilities, stored in `S$prob`, are plotted using the `plot` method in Figure 5.

```
> plot(out, xlab="P1", ylab="P2",
+  main = "Posterior Probabilities")
```

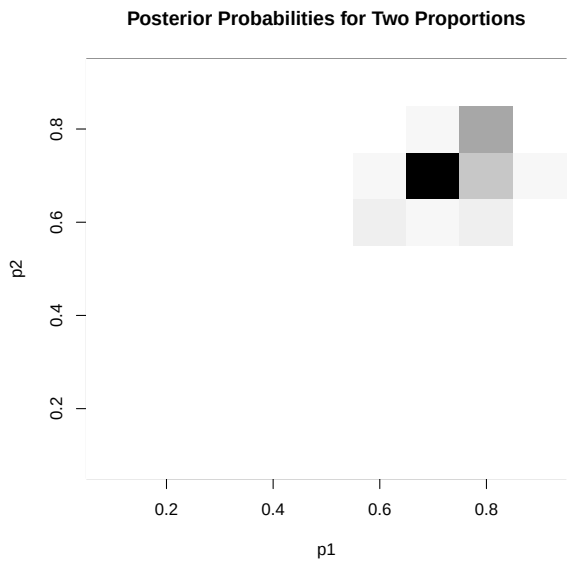

Figure 5: Image plot of the posterior probabilities for two proportions.

One can compute the posterior probability that the proportions are equal by summing the probabilities on the diagonal.

```
> sum(diag(out$prob))
```

```
[1] 0.7223
```

The prior probability that $p_1 = p_2$ is 0.4167 and the posterior probability of equality is 0.7223, which indicates that there is support from the data for the hypothesis of equal proportions. In other words, one has a stronger belief that there is not a hot hand effect from the data.

3.3. Learning about the mean and standard deviation of a normal distribution

Suppose I am interested in the sleeping habits of my statistics students. I assume that the hours of sleep for the 16 students, $y_1, ..., y_{16}$, are a random sample from a normal population with mean μ and standard deviation σ. The likelihood function of (μ, σ) is given by

$$L(\mu, \sigma) = \prod_{i=1}^{16} \phi(y_i, \sigma, \mu),$$

where $\phi()$ is the normal(μ, σ) density available as the R function `dnorm`.

We assign a grid of values of plausible values to the mean μ and σ:

```
> mu = seq(5, 9, by = 0.2)
> sigma = seq(0.5, 3, by = 0.2)
```

The following function `prior.two.parameters` (contained in the `LearnBayes package` will accept as input two vectors and output a matrix of uniform prior probabilities over the mesh grid defined by the two vectors.

```
> prior.two.parameters = function(parameter1, parameter2) {
+     prior = matrix(1, length(parameter1), length(parameter2))
+     prior = prior/sum(prior)
+     dimnames(prior)[[1]] = parameter1
+     dimnames(prior)[[2]] = parameter2
+     prior
+ }
```

Here we use this function to construct a uniform prior over the grid of values of μ and σ.

```
> prior = prior.two.parameters(mu, sigma)
```

We input the sample of sleeping times and put it in a vector.

```
> y = c(4.25, 9.25, 7, 9.16, 6.25, 6.75,
+     7.5, 9.5, 8, 7.25, 7.91, 7.5, 8, 7, 8.5, 6.5)
```

The posterior probabilities are computed using the function `discrete.bayes.2`.

```
> source("discrete.bayes.2.R")
> out = discrete.bayes.2(dnorm, prior, y)
```

The output of the function `discrete.bayes.2` is assigned to a R class. We use the `plot` method to graph the posterior probabilities and the graph is displayed in Figure 6.

```
> plot(out, xlab="MU", ylab="SIGMA")
```

Using the optional argument `marginal`, this `plot` method also computes and displays the marginal probabilities for each parameter. We use this `plot` method with `marginal = 1` to display the marginal posterior density of μ and the method with `marginal = 2` to obtain the marginal posterior density of σ. (See Figures 7 and 8.)

```
> plot(out, marginal=1, xlab="MU")
```

```
> plot(out, marginal=2, ylab="SIGMA")
```

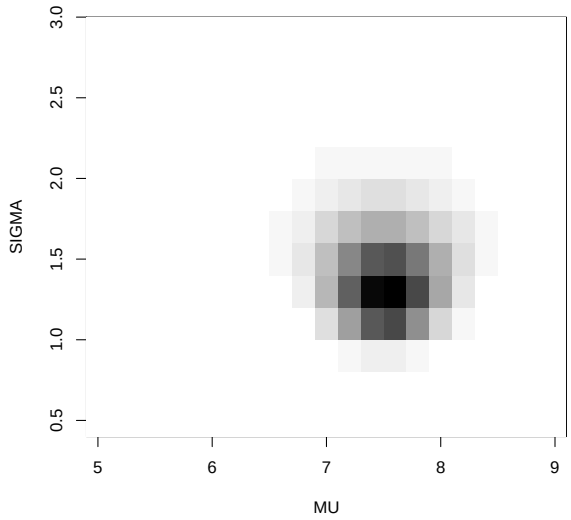

Figure 6: Image display of the posterior distribution for the mean and standard deviation for the sleeping data example.

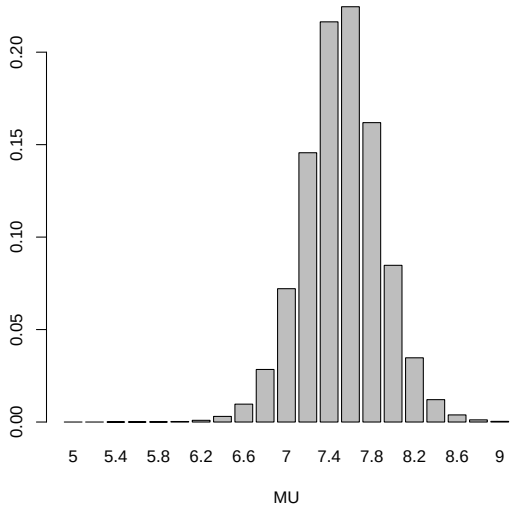

Figure 7: Marginal posterior distribution of the mean for the sleeping data example.

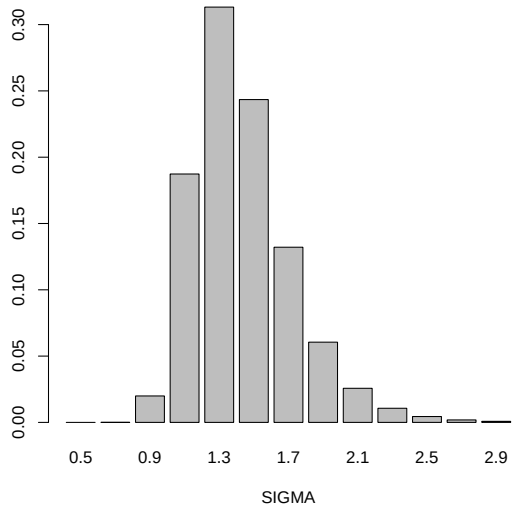

Figure 8: Marginal posterior distribution of the standard deviation for the sleeping data example.

4. Concluding Remarks

In teaching Bayesian thinking, there needs to be a good balance of concepts and computation. In using R, one would like the student to focus on the interpretation of the results rather than the generation of the R code. Towards this goal, one is generally interested in using functions that have the potential of wide applicability. In a typical Bayesian analysis, there are two important issues related to the choice of sampling model and prior. First, one wonders if the posterior inference is sensitive to reasonable changes in the sampling distribution and/or prior. A related issue is goodness of fit – are there particular choices of prior and sampling model that are more consistent with the observed data? The sensitivity issue is addressed by looking at changes in, say a particular posterior mean, when one changes the prior or sampling model. The goodness of fit issue is addressed by comparing the predictive density of the observed for different models.

The functions `discrete.bayes` and `discrete.bayes.2` (contained in release 2.1 of the `LearnBayes` package) are general-use functions that can be used to address both the sensitivity and goodness of fit issues. They require little programming by the student and can be used to perform posterior updating for a wide variety of priors and sampling distributions. By the use of the `pred` component, the student can easily compare different models (different priors or different sampling distributions) by the use of Bayes factors. The functions will hopefully encourage the instructor to go beyond the common sampling distributions such as the binomial, normal, and Poisson that are typically used in Bayesian teaching.

References

Albert, J. (2009), *Bayesian Computation with R*, Springer.

Antleman, G. (1997), *Elementary Bayesian Statistics*, Cheltenham: Edward Elgar Publishing.

Berry, D. A. (1995), *Basic Statistics: A Bayesian Perspective*, Belmont, CA: Wadsworth.

Bolstad, W. M. (2007), *Introduction to Bayesian Statistics*, Wiley.

Rossman, A. and Albert, J. (2000), *Workshop Statistics: Discovery with Data, A Bayesian Approach*, Key College.

Schmitt, S. A. (1969), *Measuring Uncertainty: An Elementary Introduction to Bayesian Statistics*, Reading, MA: Addison-Wesley.

Appendix: Code for R functions

R function `discrete.bayes` and `print`, `plot`, and `summary` methods.

```
discrete.bayes=function (df, prior, y, ...)
{
    param = as.numeric(names(prior))
    lk = function(j) prod(df(y, param[j], ...))
    likelihood = sapply(1:length(param), lk)
    pred = sum(prior * likelihood)
    prob = prior * likelihood/pred
    obj = list(prob = prob, pred = pred)
    class(obj) <- "bayes"
    obj
}

print.bayes=function(x)
  x$prob

plot.bayes=function(x,...)
  barplot(x$prob,...)

summary.bayes=function(s,coverage=.9)
{
    x = as.numeric(names(s$prob))
    p = s$prob

    post.mean=sum(x*p)
```

```
    post.sd=sqrt(sum((x-post.mean)^2*p))

    names(p)=NULL
    n = length(x)
    sp = sort(p, index.return = TRUE)
    ps = sp$x
    i = sp$ix[seq(n, 1, -1)]
    ps = p[i]
    xs = x[i]
    cp = cumsum(ps)
    ii = 1:n
    j = ii[cp >= coverage]
    j = j[1]
    eprob = cp[j]
    set = sort(xs[1:j])
    v = list(mean=post.mean,sd=post.sd,coverage = eprob, set = set)
    return(v)
}
```

R function `discrete.bayes.2` **and** `plot` **method.**

```
discrete.bayes.2=function(df,prior,y=NULL,...)
{
like=function(i,...)
  if(is.matrix(y)==TRUE)
  df(y[i,],param1,param2,...) else
  df(y[i],param1,param2,...)

n.rows=dim(prior)[1]
n.cols=dim(prior)[2]
param1=as.numeric(dimnames(prior)[[1]])
param2=as.numeric(dimnames(prior)[[2]])
param1=outer(param1,rep(1,n.cols))
param2=outer(rep(1,n.rows),param2)

likelihood=1
if(length(y)>0)
{
n=ifelse(is.matrix(y)==FALSE,length(y),dim(y)[1])
for(j in 1:n)
 likelihood=likelihood*like(j,...)
}
product=prior*likelihood
pred=sum(prior*likelihood)
prob=prior*likelihood/pred
```

```
obj=list(prob=prob,pred=pred)
class(obj)<-"bayes2"
obj
}

plot.bayes2=function(S,marginal=0,...)
if(marginal==0)image(as.numeric(dimnames(S$prob)[[1]]),
       as.numeric(dimnames(S$prob)[[2]]),S$prob,
       col=gray(1-(0:32)/32),...) else
   if(marginal==1) barplot(apply(S$prob,1,sum),...) else
   barplot(apply(S$prob,2,sum),...)
```