Assignment Covering Chapters 2 & 3 of *Bayesian Computation with R*

Chris Hayduk

March 15, 2019

1. **Chapter 2: Introduction to Bayesian Thinking** Write your own code for the following:
   (a) Simulating a sample from a posterior distribution when you have a histogram prior.
   (b) Prior predictive density function, `pdiscp()`.
   (c) Prior predictive density function, `pbetap()`.
   (d) Discrete credible interval, `discint`.

Solution:

(a) Let's start by defining a function to generate the values for the histogram prior:

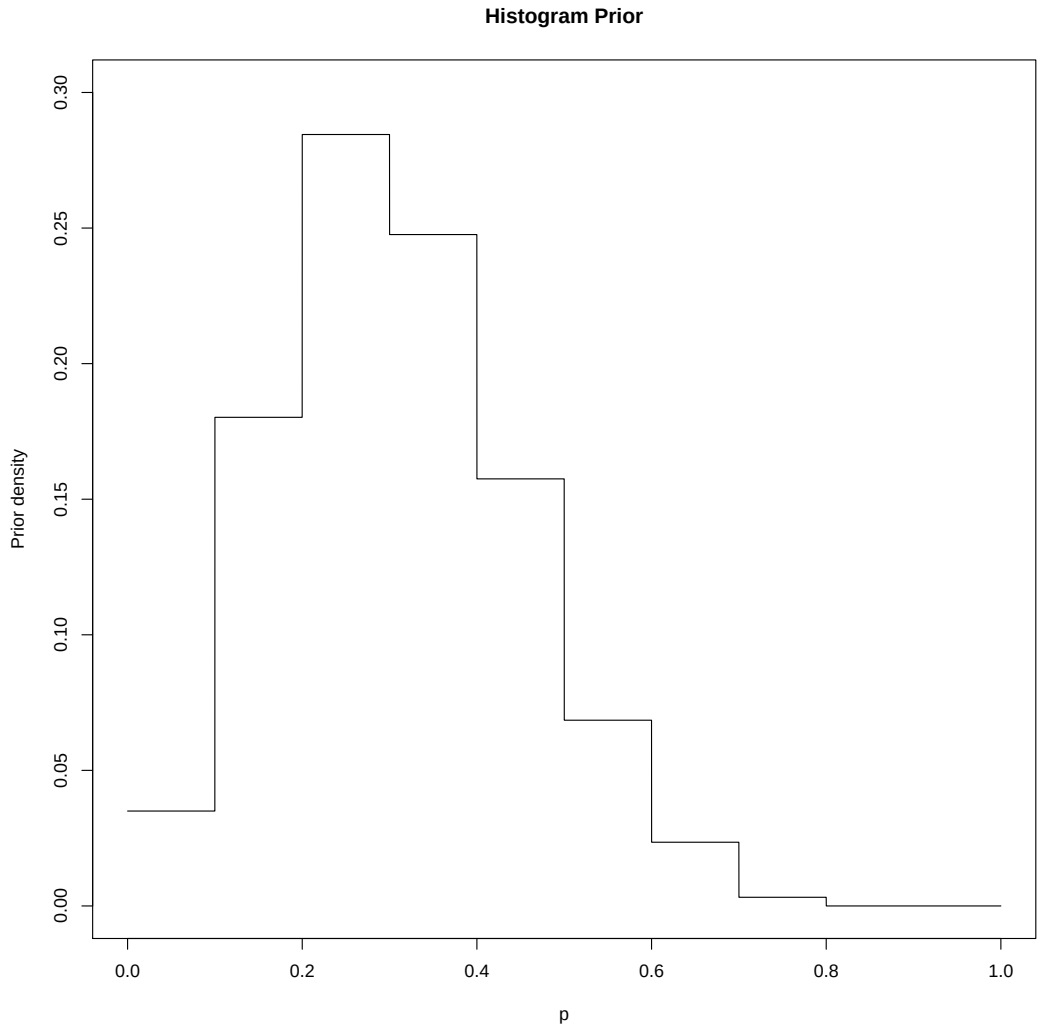
```
#Function to generate prior values for each value in x
get_histprior_value <- function(x, histprior){
  vec <- rep(NA, length=length(x))
  for(i in 1:length(vec)){
    new_vec <- histprior[histprior[,1]>=x[i],]

    #Check if new_vec is matrix or vector and index accordingly
    if(!is.null(ncol(new_vec))){
      vec[i] <- new_vec[1,2]
    } else{
      vec[i] <- new_vec[2]
    }
  }

  return(vec)
}
```

Now let's prepare the data for input into our new `get_histprior_value` function:

```
#Generate points for interval
interval <- seq(0.1, 1, by = 0.1)

#Prior probability
prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0)
prior <- prior/sum(prior)

#Create the histogram prior
histprior <- sample(interval, 10000, replace=TRUE, prob = prior)
histprior <- table(histprior)/sum(table(histprior))
histprior <- as.matrix(histprior)
names <- rownames(histprior)
rownames(histprior) <- NULL
histprior <- cbind(as.numeric(names),as.numeric(histprior))
histprior <- rbind(histprior, c(0.9, 0))
histprior <- rbind(histprior, c(1.0, 0))
```

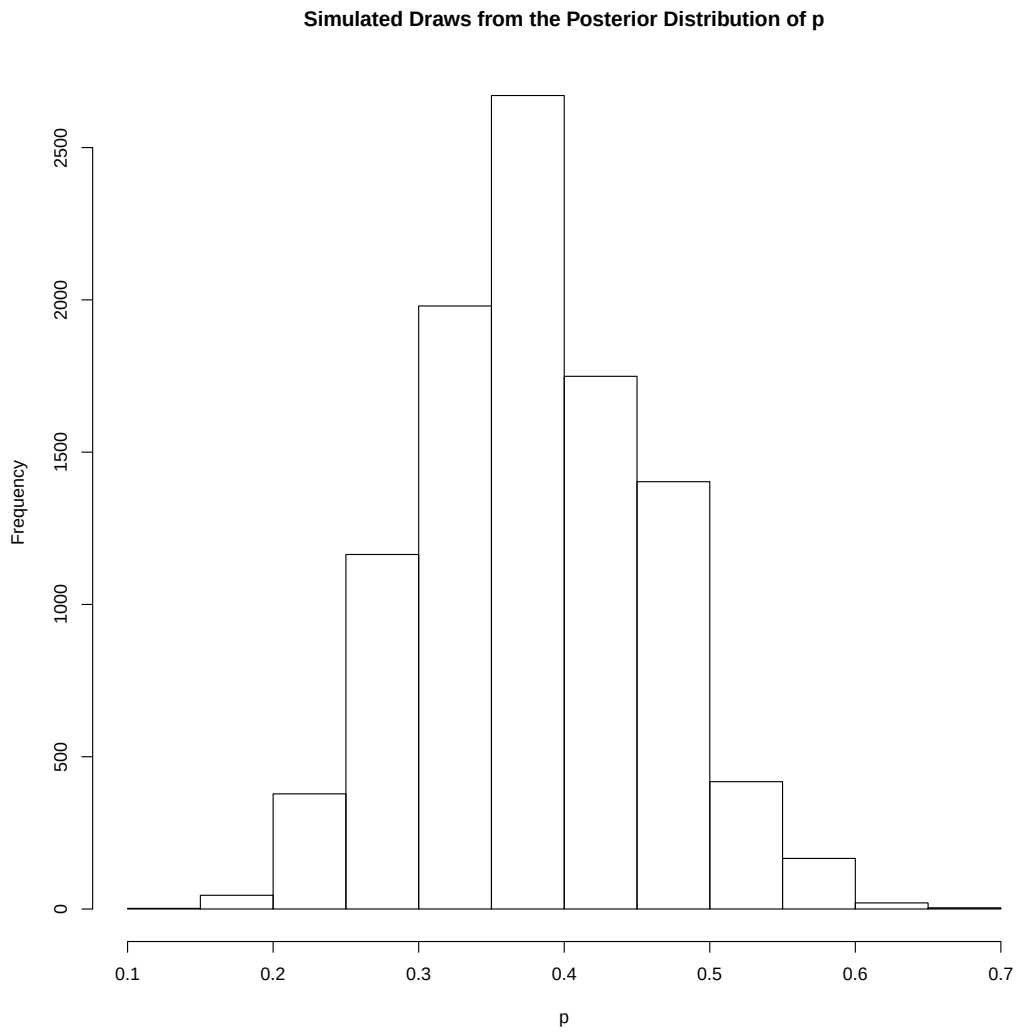
Now let's plot the prior:

```
#Output histogram of prior
curve(get_histprior_value(x, histprior), from=0, to = 1,
      ylim = c(0, 0.3), n = 10000,
      xlab="p", ylab = "Prior density", main = "Histogram Prior")
```



Finally, let's generate and plot the posterior:

```
s <- 11
f <- 16

p <- seq(0, 1, length = 10000)

post <- get_histprior_value(p, histprior) * dbeta(p, s+1, f+1)

post <- post/sum(post)

ps <- sample(p, replace = TRUE, prob = post)

hist(ps, xlab="p", main="Simulated Draws from the Posterior Distribution of p")
```



(b) Let's start with `pdiscp()`:

```
pdiscp <- function(p, prior, m, ys){
  pred <- rep(NA, length = length(ys))

  #Loop through possible y values
  for(i in 1:length(ys)){
    val <- 0
    #Loop through possible proportions in discrete prior
    for(j in 1:length(p)){
      f <- choose(m, ys[i]) * (p[j])^(ys[i]) * (1-p[j])^(m-ys[i])
      g <- prior[j]
      val <- val + (f*g)
    }
    pred[i] <- val
  }

  #Return vector of probabilities for each y value
  return(pred)
}
```

```r
#Test run
p <- seq(0.05, 0.95, by=0.1)
prior <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0)
prior <- prior/sum(prior)
m <- 20
ys <- 0:20
pred <- pdiscp(p, prior, m, ys)

round(cbind(0:20, pred), 3)
```

```
##           pred
##  [1,]  0 0.020
##  [2,]  1 0.044
##  [3,]  2 0.069
##  [4,]  3 0.092
##  [5,]  4 0.106
##  [6,]  5 0.112
##  [7,]  6 0.110
##  [8,]  7 0.102
##  [9,]  8 0.089
## [10,]  9 0.074
## [11,] 10 0.059
## [12,] 11 0.044
## [13,] 12 0.031
## [14,] 13 0.021
## [15,] 14 0.013
## [16,] 15 0.007
## [17,] 16 0.004
## [18,] 17 0.002
## [19,] 18 0.001
## [20,] 19 0.000
## [21,] 20 0.000
```

(c) Now let's try `pbetap()`:

```
pbetap <- function(ab, m, ys){
  pred <- rep(NA, length = length(ys))

  for(i in 1:length(ys)){
    choose <- choose(m, ys[i])

    beta_num <- beta(ab[1] + ys[i], ab[2] + m - ys[i])

    beta_denom <- beta(ab[1], ab[2])

    pred[i] <- choose*(beta_num/beta_denom)
  }

  #Return vector of probabilities for each y value
  return(pred)
}
```

```r
#Test run
ab <- c(3.26, 7.19)
m <- 20
ys <- 0:20
pred <- pbetap(ab, m, ys)
print(pred)
```

```
##  [1] 1.812205e-02 4.511485e-02 7.248106e-02 9.456396e-02 1.084896e-01
##  [6] 1.135841e-01 1.106895e-01 1.015339e-01 8.821389e-02 7.280839e-02
## [11] 5.712007e-02 4.252974e-02 2.994441e-02 1.981684e-02 1.221463e-02
## [16] 6.917948e-03 3.527752e-03 1.568885e-03 5.764529e-04 1.575142e-04
## [21] 2.438293e-05
```

```r
round(cbind(0:20, pred), 3)
```

```
##           pred
##  [1,]  0 0.018
##  [2,]  1 0.045
##  [3,]  2 0.072
##  [4,]  3 0.095
##  [5,]  4 0.108
##  [6,]  5 0.114
##  [7,]  6 0.111
##  [8,]  7 0.102
##  [9,]  8 0.088
## [10,]  9 0.073
## [11,] 10 0.057
## [12,] 11 0.043
## [13,] 12 0.030
## [14,] 13 0.020
## [15,] 14 0.012
## [16,] 15 0.007
## [17,] 16 0.004
## [18,] 17 0.002
## [19,] 18 0.001
## [20,] 19 0.000
## [21,] 20 0.000
```

(d) Finally, here's `discint`:

```
discint <- function(dist, covprob){
  total_prob <- 0
  y_list <- c()

  #Find max probability value
  #Centers credible interval around y value with max probability
  i <- which.is.max(dist[,2])
  total_prob <- total_prob + dist[i,2]
  y_list <- c(y_list, dist[i,1])

  i_prev <- i-1
  i_next <- i+1

  while(total_prob < covprob){
    #Check if previous row is within range of matrix
    if(i_prev >= 1 & i_prev <= nrow(dist)){
      prev_val <- dist[i_prev,2]
    } else{
      prev_val <- -1
    }

    #Check if next row is within range of matrix
    if(i_next >= 1 & i_next <= nrow(dist)){
      next_val <- dist[i_next,2]
    } else{
      next_val <- -1
    }

    #Compare values for i_next and i_prev
    if(dist[i_next,2] >= dist[i_prev,2]){
      total_prob <- total_prob + dist[i_next, 2]
      y_list <- c(y_list, dist[i_next, 1])
      i_next <- i_next + 1
    } else{
      total_prob <- total_prob + dist[i_prev, 2]
      y_list <- c(y_list, dist[i_prev, 1])
      i_prev <- i_prev - 1
    }
  }

  output <- list(prob = total_prob, set = sort(y_list))

  return(output)
}
```

```r
#Test run
p <- rbeta(1000, 3.26, 7.19)
y <- rbinom(1000, 20, p)
freq <- table(y)
ys <- as.integer(names(freq))
predprob <- freq/sum(freq)

dist <- cbind(ys, predprob)
covprob <- 0.9

discint(dist, covprob)
```

```
## $prob
## [1] 0.913
##
## $set
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```

2. **Chapter 2: Introduction to Bayesian Thinking** Show how the author derived the predictive density formula below with a $beta(a, b)$ prior (pg. 31 of book):

$$\begin{aligned} f(\tilde{y}) &= \int f_B(\tilde{y}|m,p)g(p)dp \\ &= \binom{m}{\tilde{y}} \frac{B(a+\tilde{y}, b+m-\tilde{y})}{B(a,b)} \end{aligned}$$

   where $\tilde{y} = 0, \ldots, m$.

3. **Chapter 2: Introduction to Bayesian Thinking** Exercise 5.

4. **Chapter 2: Introduction to Bayesian Thinking** Exercise 6.

5. **Chapter 3: Single-Parameter Models** Write your own versions of the functions `beta.binomial.mix()` and `pbetat()`.

6. **Chapter 3: Single-Parameter Models** On pg. 55, Albert writes, "One can show that the posterior probability that the coin is fair is given by:"

$$\lambda(y) = \frac{0.5P_0(Y \leq 5)}{0.5P_0(Y \leq 5) + 0.5P_1(Y \leq 5)}.$$

   Show this.

7. **Chapter 3: Single-Parameter Models** Do all Exercises at the end of the chapter.