STAT 447 Assignment 7

Caden Hewlett

2024-03-07

Question 1: Installing and Running Stan

Setting up the beta-binomial environment:

Here, we have observed data n and k.

We have a parameter p, where $p \sim \text{beta}(\alpha, \beta)$ and $k \sim \text{bin}(n, p)$.

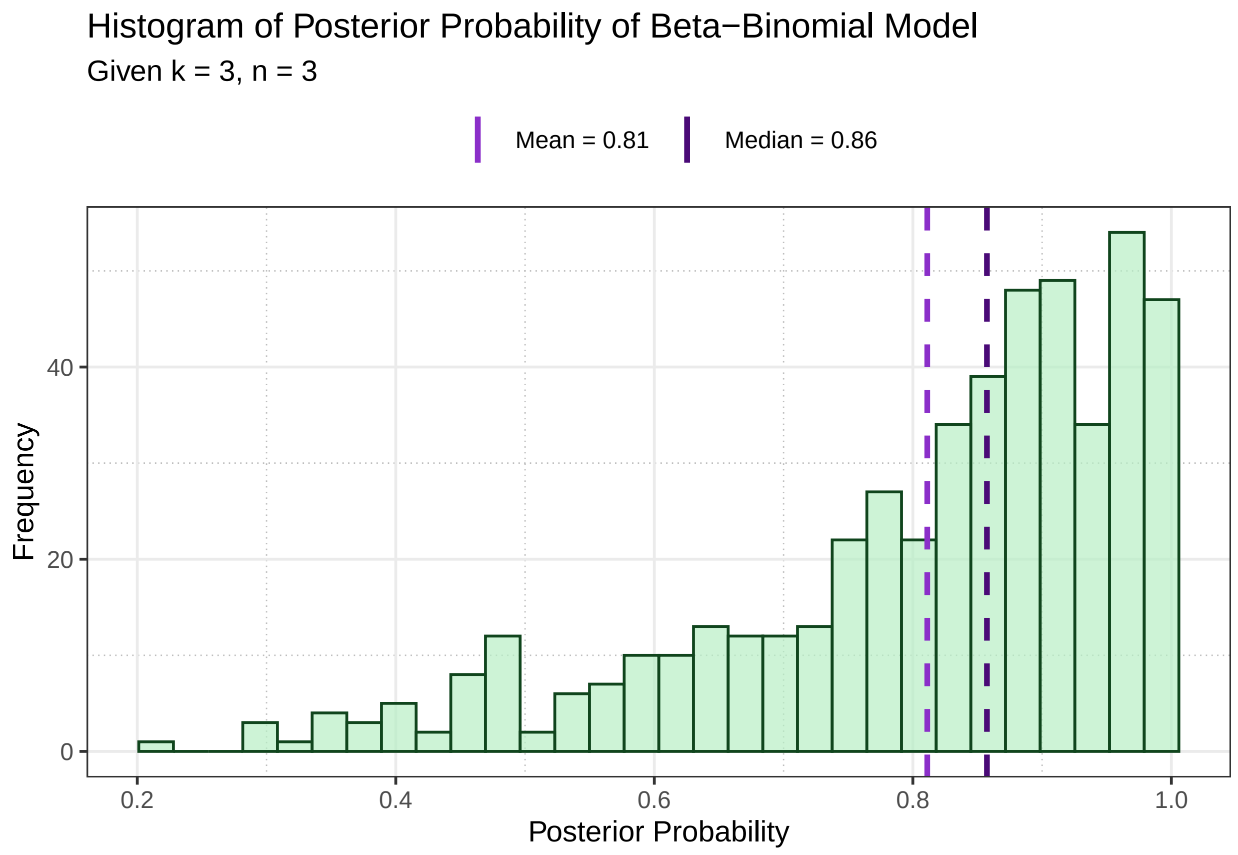
```
data {
  int<lower=0> n;          // number of trials
  int<lower=0,upper=n> k; // number of successes
}

parameters {
  real<lower=0,upper=1> p; // p in [0, 1]
}

model {
  // prior
  p ~ beta(1,1);

  // likelihood
  k ~ binomial(n, p);
}
```

Then, we run the MCMC to find $\mathbb{P}(p \mid \{k, n\} = \{3, 3\})$. As in, the posterior success probability given three subsequent successes.

```r
require(rstan)

fit = sampling(
  test,
  seed = 123,
  data = list(n = 3, k = 3),
  chains = 1,
  iter = 1000
)

q1model = extract(fit)
```

We can also use `ggplot2` to make a nice histogram of the output.

```r
ggplot(data.frame(q1model$p), aes(x = q1model.p)) +
  geom_histogram(
    bins = 30,
    fill = "#B7EFC5",
    color = "#10451D",
    alpha = 0.7
  ) +
  geom_vline(aes(xintercept = mean(q1model$p), color = "Mean"),
             linetype = "dashed",
             linewidth = 1) +
  geom_vline(
    aes(xintercept = median(q1model$p), color = "Median"),
    linetype = "dashed",
    linewidth = 1
  ) +
  scale_color_manual(
    name = "",
    values = c("Mean" = "#8B2FC9",
               "Median" = "#4A0A77"),
    labels = c(paste("Mean =", round(mean(q1model$p), 2)),
               paste("Median =", round(median(q1model$p), 2)))
  ) +
  labs(
    title = "Histogram of Posterior Probability of Beta-Binomial Model",
    subtitle = "Given k = 3, n = 3",
    x = "Posterior Probability",
    y = "Frequency"
  ) +
  theme_bw() +
  theme(
    legend.position = "top",
    panel.grid.minor = element_line(colour = "gray", linetype = "dotted")
  ) +
  guides(color = guide_legend(override.aes =
                                list(linetype = c("solid", "solid"))))
```



So, both from the histogram we can see the posterior median is approximately 0.86.

Precisely, it is the value below:

```
median(q1model$p)
```

```
## [1] 0.8573115
```

Question 2: Regression in Stan

In this question, we will analyze the Hubble data encountered earlier in the course but using different priors, and with Stan instead of simPPLe. First, we access and preview the data:

```
df = read.csv("hubble.csv")[1:24, c(3:4)]
colnames(df) = c("distance", "velocity")
velocity = df$velocity/1000
distance = df$distance
kable(t(head(df)))
```

	1	2	3	4	5	6
distance	0.032	0.03	0.214	0.263	0.275	0.275
velocity	170.000	290.00	-130.000	-70.000	-185.000	-220.000

Then, we write a Stan model following the same structure as the model from last time we analyzed this data except that the following priors should be used:

For the slope parameter, `student_t(3, 0, 100)` and for the standard deviation parameter and exponential with rate 0.001.

For completeness, we detail the full model below. Let d_i be the i-th observed distance.

In Stan, the t distriubtion takes parameters $\{\nu, \mu, \sigma\}$, which are degrees of freedom, mean and variance. We let $\{\nu, \mu, \sigma\} = \{3, 0, 100\}$. Further, we let the rate parameter λ of the exponential distribution be $1/1000 = 0.001$.

$$
\begin{aligned}
\beta &\sim t(3, 0, 100) \\
\sigma &\sim \exp(0.001) \\
v_i \mid \{\beta, \sigma\} &\sim N\big((\beta \times d_i), \sigma\big)
\end{aligned}
$$

In Stan, we would configure this as follows:

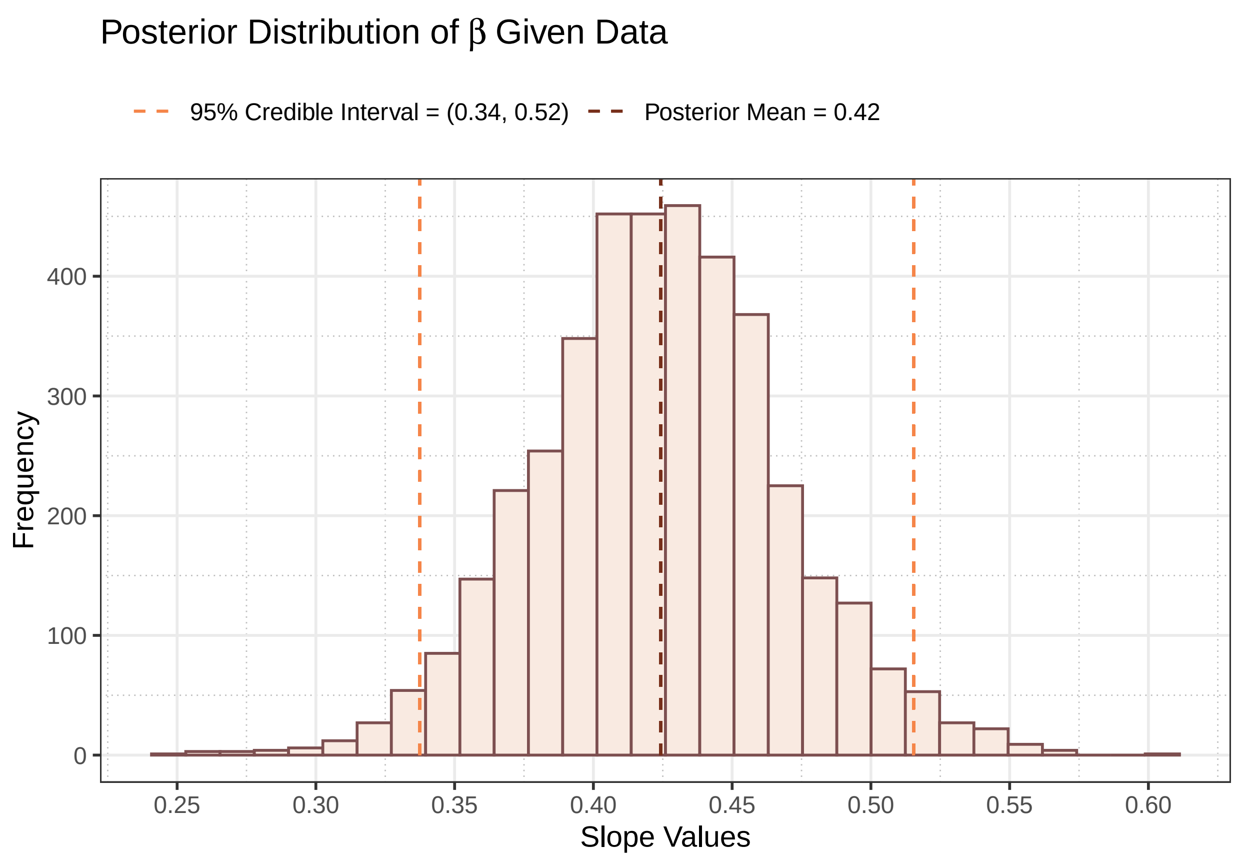
```
// The input data are the distances 'd' of length 'N'.
data {
  int<lower=0> N;
  vector[N] d;
  vector[N] v;
}

// The parameters accepted by the model. Our model
// accepts two parameters 'beta' which is the slope
// and 'sigma', which is the variance (heteroskedastic)
parameters {
  real beta;
  real<lower=0> sigma;
}

// The model to be estimated.
// We model the velocity 'v' to be normally distributed
// with mean 'beta * d_i'and standard deviation 'sigma'.
model {
  beta ~ student_t(3, 0, 100);
  sigma ~ exponential(0.001);
  for (i in 1:N){
    v[i] ~ normal(beta * d[i], sigma);
  }
}
```

Now, we run Stan for $M = 2000$ iterations and report a histogram on the quantity of interest, i.e., the slope parameter.

```
reg_fit = sampling(
  regression_model,
  seed = 1990,
  data = list(N = length(distance),
              d = distance,
              v = velocity),
  #chains = 4, # default
  iter = 2000
)
# extract data from our fit
reg_vals = extract(reg_fit)
# and some summary stats
qL = quantile(reg_vals$beta, 0.025)
qU = quantile(reg_vals$beta, 0.975)
xbar = mean(reg_vals$beta)
```

Then, the histogram is below (NOTE: The code required to produce this plot object became very long, so I set `include = FALSE` on the cell defining the histogram. The plot object `p2` is below:)

```
p2
```



Question 3

We initialize the helper/background functions provided in a hidden code cell.

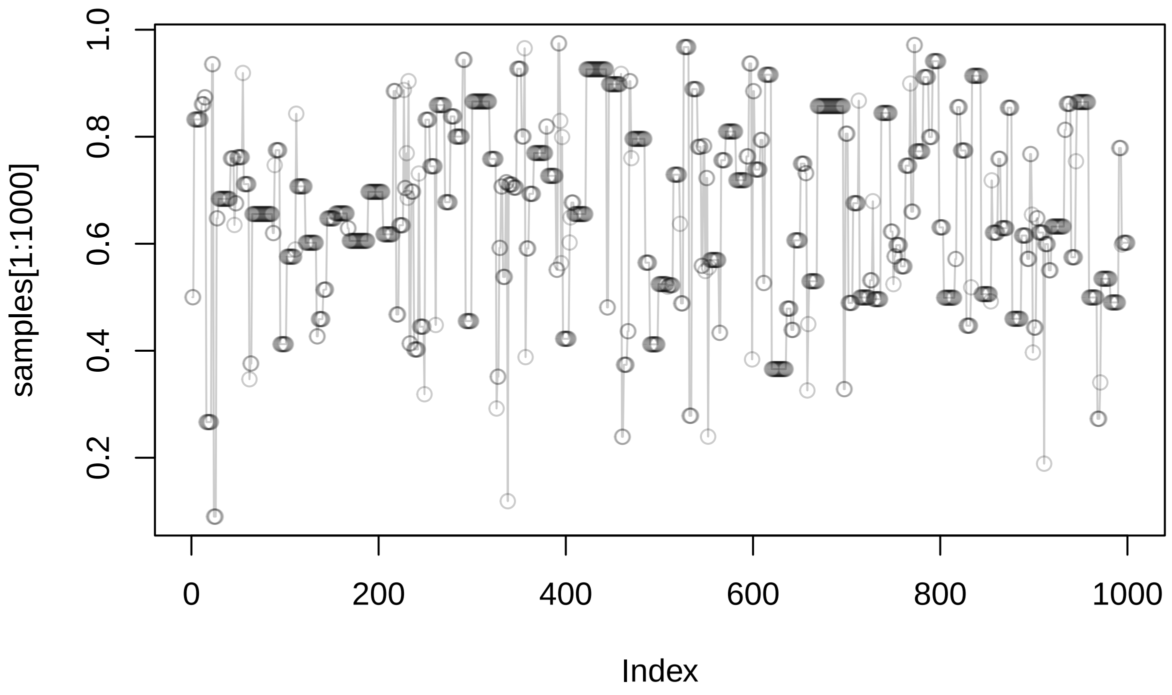
Then, we implement our Metropolis-Hastings Algorithm below.

```
# define functions to be used within M.H.
Q <- function(x){  rnorm(1, mean = x) }
R <- function(x_tilde, x_m, gamma_fn){ gamma_fn(x_tilde) / gamma_fn(x_m) }
A <- function(R){ rbinom(1, size = 1, p = min(1, R)) }
```

```r
# simple Metropolis-Hastings algorithm (normal proposal)
simple_mh = function(gamma, initial_point, n_iters) {
  # declare numeric X
  X = numeric(n_iters)
  dim = length(initial_point)
  # set X_0 = initial
  X[1] = initial_point
  # then we will return a vector of length m+1
  for (m in 2:(n_iters+1)){
    # declare proposed x value
    x_m <- X[m-1]
    # compute q(X^{m-1})
    x_tilde <- Q(x_m)
    # compute ratio \gamma(\tilde{X}^{m}) / \gamma(X^{m-1})
    R_m <- R(x_tilde, x_m, gamma_fn = gamma)
    # compute bernoulli trial at p = min\{1, R^{(m)} \}
    A_m <- A(R_m)
    # accept or reject proposal
    X[m] <- ifelse(A_m == 1,  x_tilde, x_m)
  }
  return(X)
}
```

Then, we will create the trace plot:

```r
set.seed(447)
# call algorithm
samples = simple_mh(gamma_beta_binomial, 0.5, 1000)
# create trace plot
plot(samples[1:1000],
     type = "o",
     col = rgb(
       red = 0,
       green = 0,
       blue = 0,
       alpha = 0.2
     ))
```


samples[1:1000]
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000
Index