Bayesian Inference for a Proportion (R scripts)

Jingchen (Monika) Hu

MATH 347 Bayesian Statistics

Installing the necessary packages

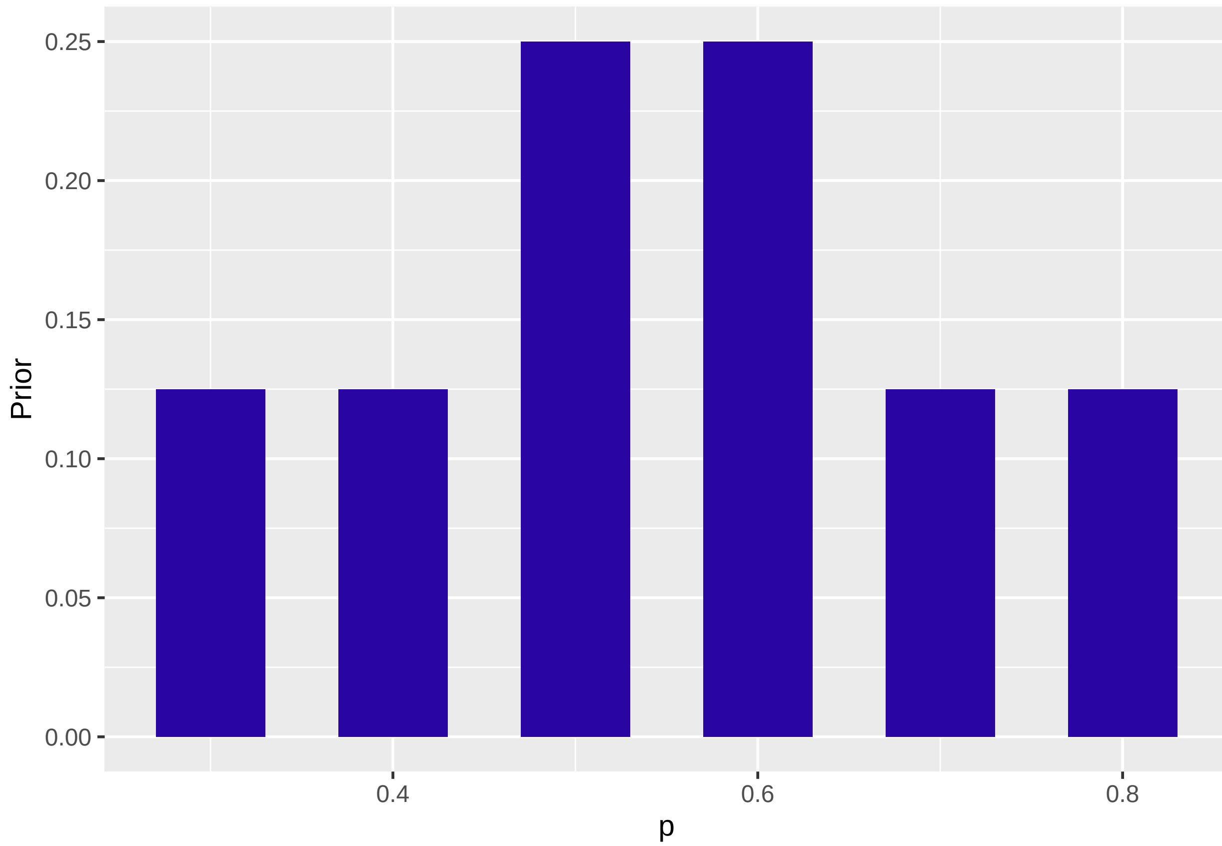
```
install.packages("devtools")
require(devtools)
devtools::install_github("bayesball/ProbBayes")

require(ggplot2)
require(gridExtra)
require(ProbBayes)
require(tidyverse)
crcblue <- "#2905a1"
```

Example: Tokyo Express customers' dining preference

Bayesian inference with discrete priors

Using R/RStudio to express and plot the prior $\pi_{owner}(p)$

```
bayes_table <- data.frame(p = seq(.3, .8, by=.1),
                          Prior = c(0.125, 0.125, 0.250,
                                    0.250, 0.125, 0.125))
ggplot(data=bayes_table, aes(x=p, y=Prior)) +
  geom_bar(stat="identity", fill=crcblue, width = 0.06)
```



Use R/RStudio to compute the likelihood function

```
bayes_table$Likelihood <- dbinom(12, size=20,
                                 prob=bayes_table$p)
bayes_table
```

```
##     p Prior  Likelihood
## 1 0.3 0.125 0.003859282
## 2 0.4 0.125 0.035497440
## 3 0.5 0.250 0.120134354
## 4 0.6 0.250 0.179705788
## 5 0.7 0.125 0.114396740
## 6 0.8 0.125 0.022160877
```

Use R/RStudio to compute and plot the posterior

```
bayesian_crank(bayes_table) -> bayes_table
bayes_table
```

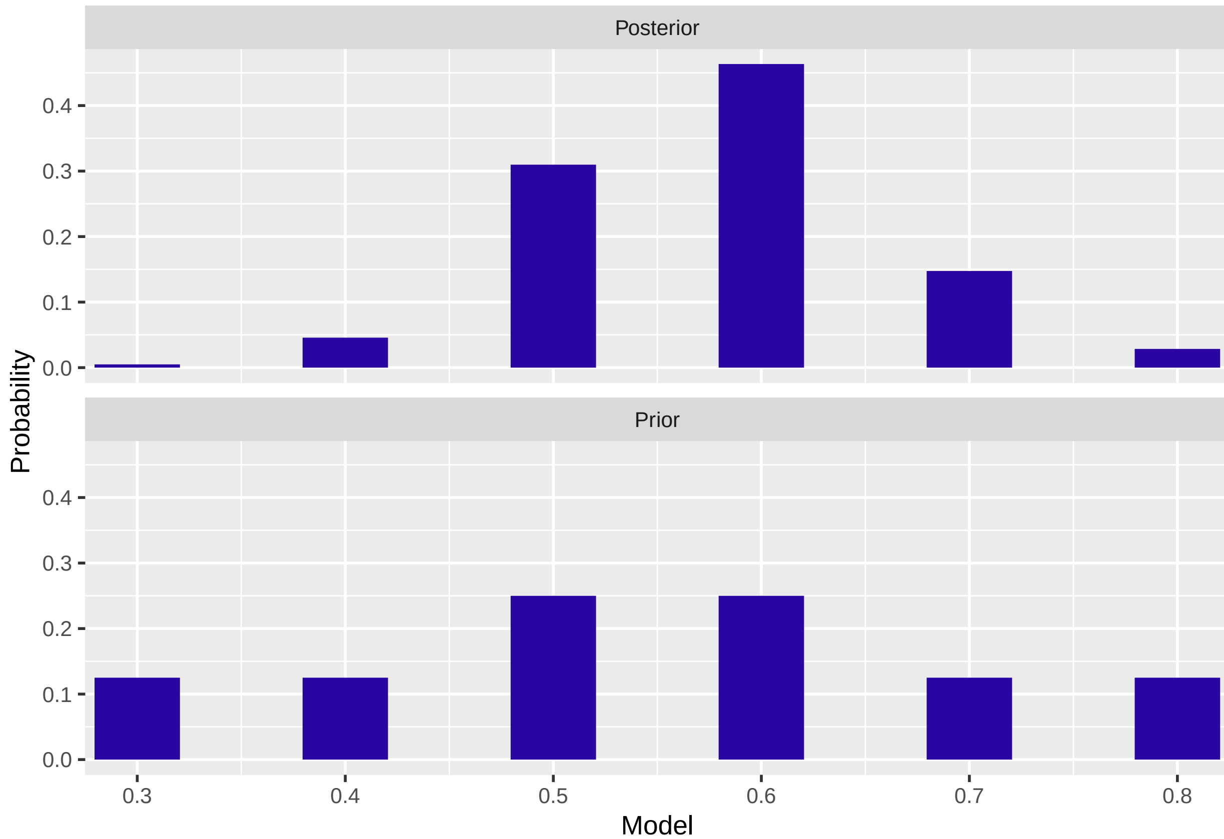
```
##     p Prior  Likelihood      Product   Posterior
## 1 0.3 0.125 0.003859282 0.0004824102 0.004975901
```

```
## 2 0.4 0.125 0.035497440 0.0044371799 0.045768032
## 3 0.5 0.250 0.120134354 0.0300335884 0.309786454
## 4 0.6 0.250 0.179705788 0.0449264469 0.463401326
## 5 0.7 0.125 0.114396740 0.0142995925 0.147495530
## 6 0.8 0.125 0.022160877 0.0027701096 0.028572757
```

```
bayesian_crank(bayes_table) -> bayes_table
sum(bayes_table$Posterior[bayes_table$p > 0.5])
```

```
## [1] 0.6394696
```

```
prior_post_plot(bayes_table, Color = crcblue) +
  theme(text=element_text(size=10))
```



Continuous priors - the Beta distribution

Step 1: Prior distribution

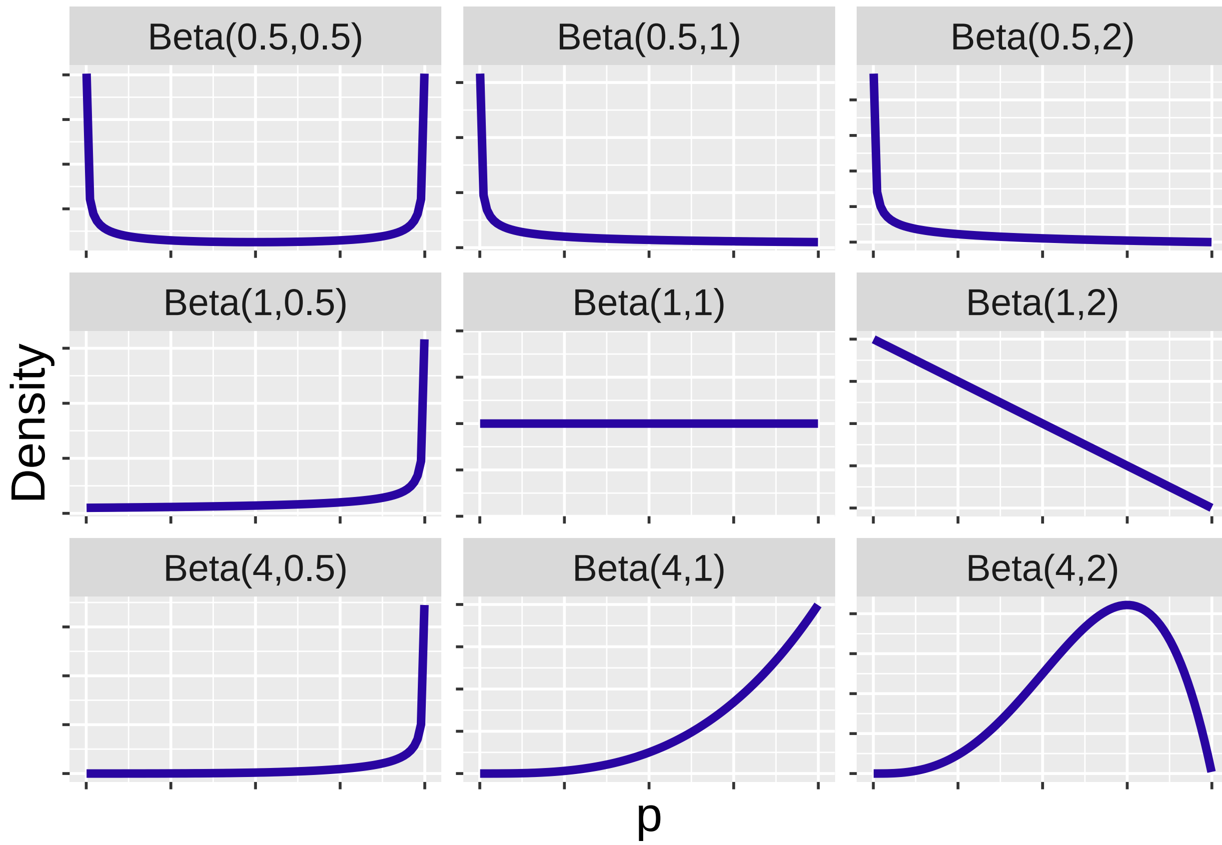
```
bayes_table
```

```
##     p Prior   Likelihood      Product    Posterior
## 1 0.3 0.125 0.003859282 0.0004824102 0.004975901
## 2 0.4 0.125 0.035497440 0.0044371799 0.045768032
```

```
## 3 0.5 0.250 0.120134354 0.0300335884 0.309786454
## 4 0.6 0.250 0.179705788 0.0449264469 0.463401326
## 5 0.7 0.125 0.114396740 0.0142995925 0.147495530
## 6 0.8 0.125 0.022160877 0.0027701096 0.028572757
```

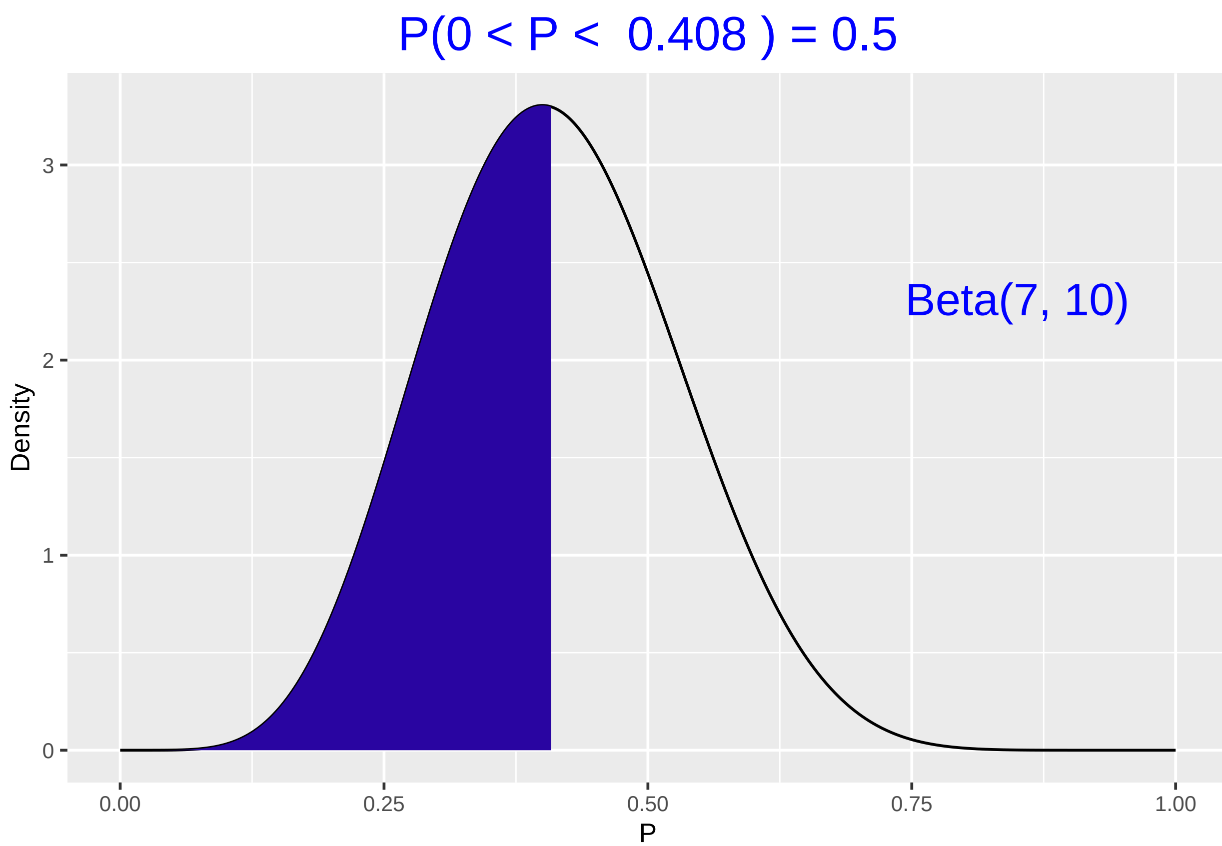
Examples of Beta curves

```
betapars <- matrix(c(0.5, 0.5,
                     0.5, 1,
                     0.5, 2,
                     1, 0.5,
                     1, 1,
                     1, 2,
                     4, 0.5,
                     4, 1,
                     4, 2),
                     9, 2, byrow = TRUE)
p <- seq(.001, .999, length.out = 100)
BETA <- NULL
for (j in 1:9){
  df <- data.frame(p = p, Density = dbeta(p,
                    betapars[j, 1], betapars[j, 2]))
  df$Type <-  paste("Beta(", betapars[j, 1],
                    ",", betapars[j, 2], ")",
                    sep = "")
  BETA <- rbind(BETA, df)
}
ggplot(BETA, aes(p, Density)) +
  geom_line(color = crcblue, size = 1.5) +
  facet_wrap(~ Type, scale = "free") +
  increasefont() +
  theme(axis.text=element_blank())
```



Choose a Beta curve to represent prior opinion

```
beta_quantile(0.5, c(7, 10), Color = crcblue) +
  theme(text=element_text(size=10))
```



Use `beta.select` to choose a Beta curve

```r
beta.select(list(x = 0.55, p = 0.5),
            list(x = 0.80, p = 0.9))
```

```
## [1] 3.06 2.56
```

Updating the Beta prior

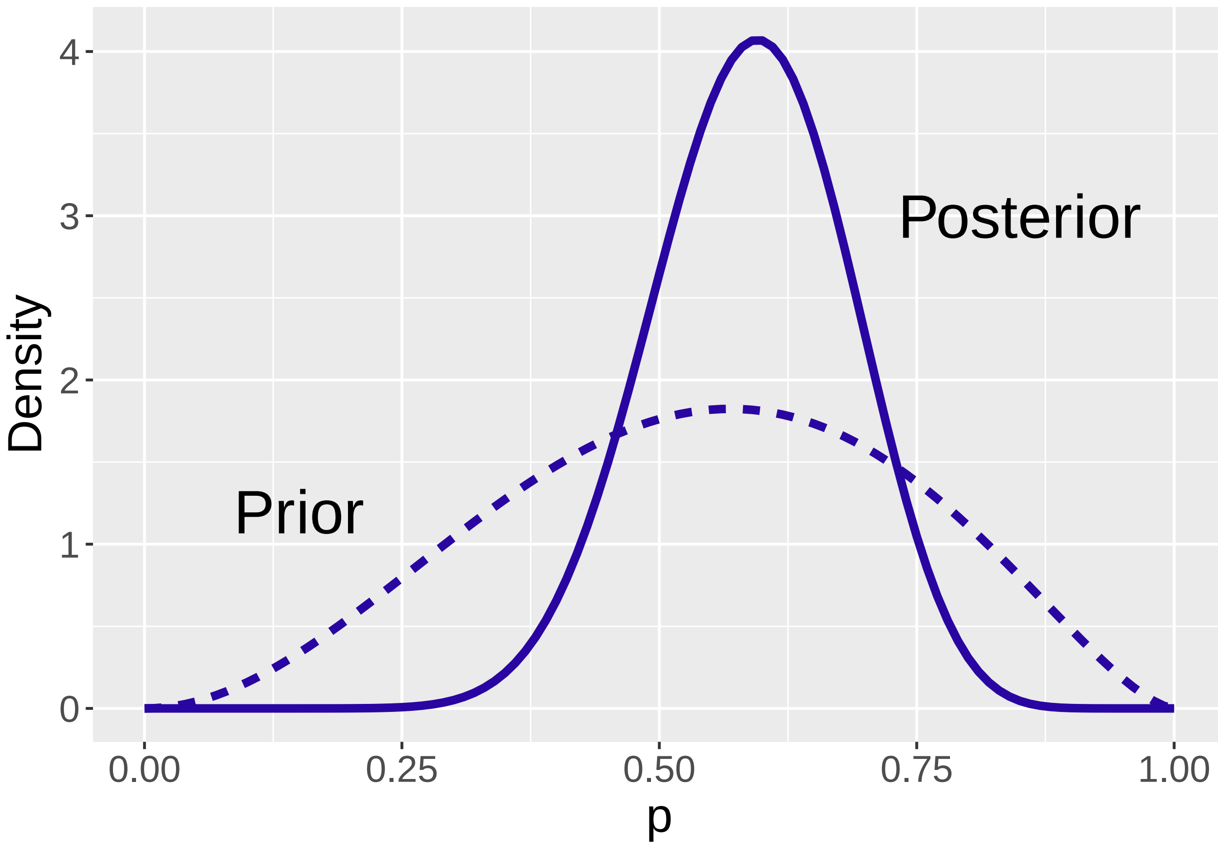
Use R/RStudio to compute and plot the posterior

```r
ab <- c(3.06, 2.56)
yny <- c(12, 8)
(ab_new <- ab + yny)
```

```
## [1] 15.06 10.56
```

```r
ggplot(data.frame(x=c(0, 1)), aes(x)) +
  stat_function(fun=dbeta, geom="line",
                aes(linetype="solid"),
                size=1.5, color = crcblue,
```

```
                 args=list(shape1=ab[1],
                           shape2=ab[2])) +
  stat_function(fun=dbeta, geom="line",
                aes(linetype="dashed"),
                size=1.5, color = crcblue,
                args=list(shape1=ab_new[1],
                          shape2=ab_new[2])) +
  xlab("p") + ylab("Density") +
  theme(legend.position = "none") +
  annotate(geom = "text", x = .15, y = 1.2,
           size = 8, label = "Prior") +
  annotate(geom = "text", x = .85, y = 3,
           size = 8, label = "Posterior") +
  increasefont()
```

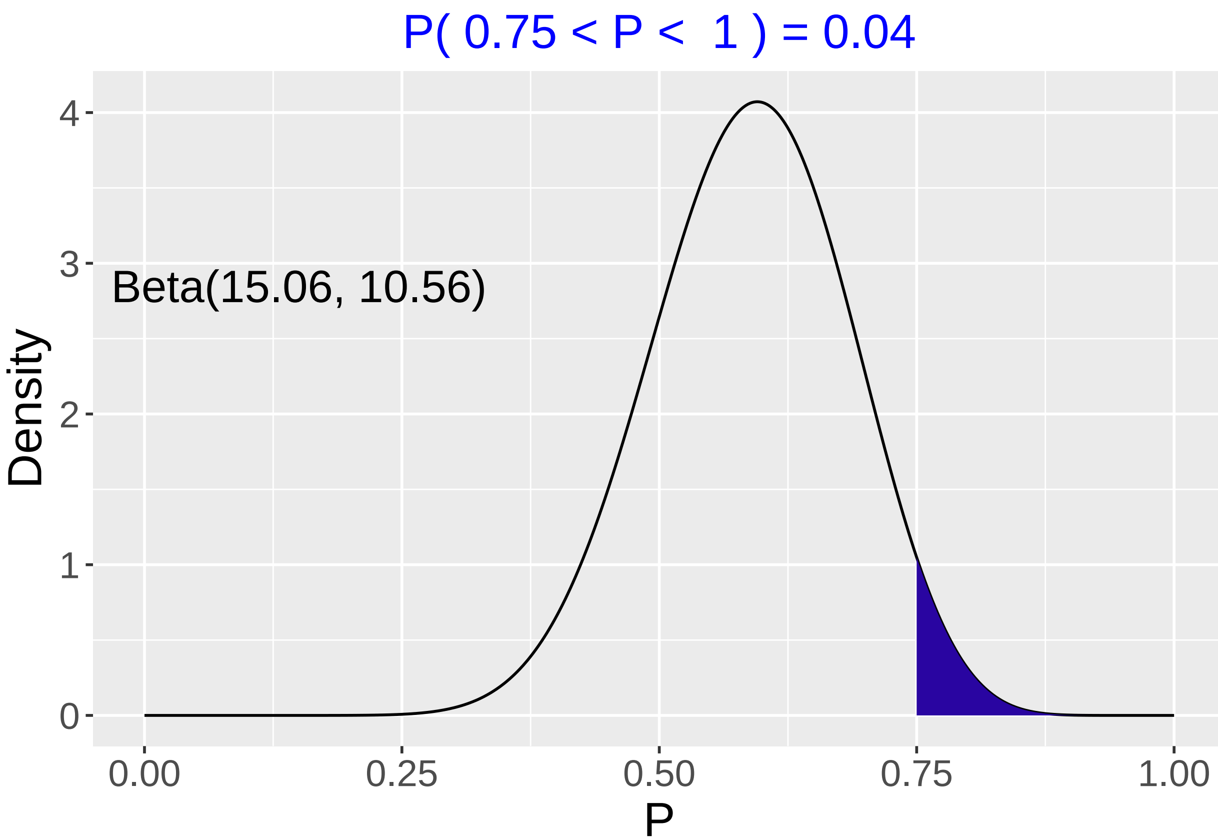


Bayesian inference with continuous priors

Bayesian hypothesis testing

```
beta_area(lo = 0.75, hi = 1.0, shape_par = c(15.06, 10.56),
          Color = crcblue) +
```

```
  theme(text=element_text(size=18))
```




```
beta_area(lo = 0.75, hi = 1.0, shape_par = c(15.06, 10.56))
```

```
pbeta(1, 15.06, 10.56) - pbeta(0.75, 15.06, 10.56)
```

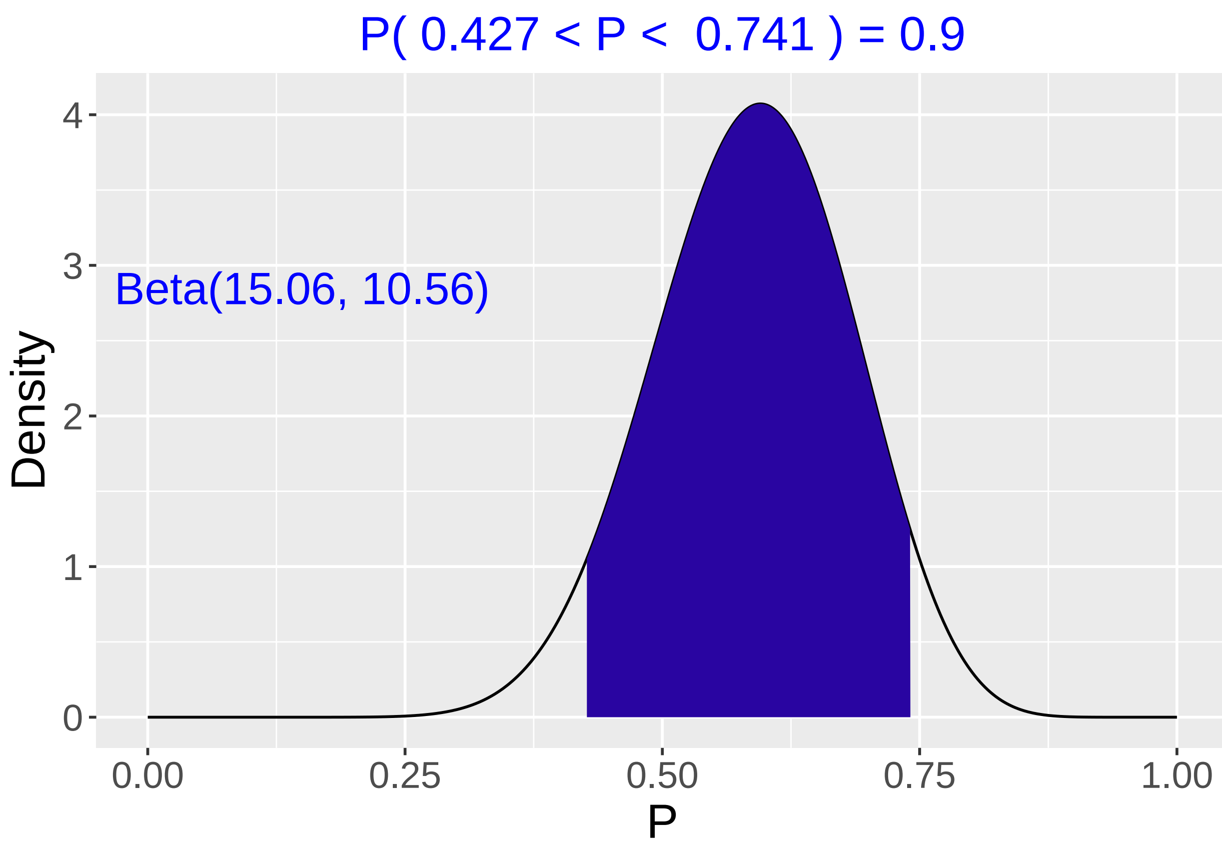
```
## [1] 0.03973022
```

```
S <- 1000
BetaSamples <- rbeta(S, 15.06, 10.56)
sum(BetaSamples >= 0.75)/S
```

```
## [1] 0.05
```

Bayesian credible intervals

```
beta_interval(0.9, c(15.06, 10.56), Color = crcblue) +
  theme(text=element_text(size=18))
```




```r
c(qbeta(0.05, 15.06, 10.56), qbeta(0.95, 15.06, 10.56))
```

```
## [1] 0.4266788 0.7410141
```

```r
S <- 1000; BetaSamples <- rbeta(S, 15.06, 10.56)
quantile(BetaSamples, c(0.05, 0.95))
```

```
##        5%       95%
## 0.4299751 0.7395138
```

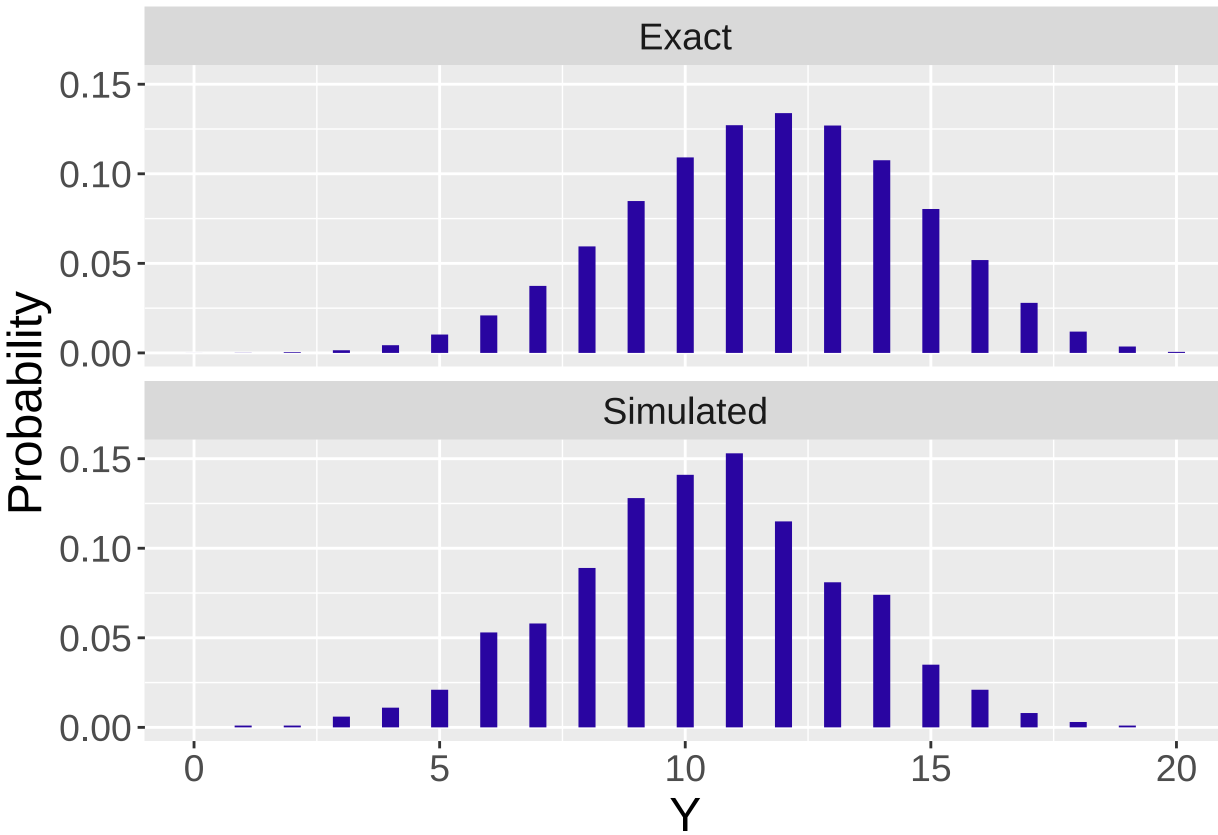
Use R/RStudio to make Bayesian predictions

```r
S <- 1000
a <- 3.06; b <- 2.56
n <- 20; y <- 12
m <- 20
pred_p_sim <- rbeta(S, a + y, b + n - y)
pred_y_sim <- rbinom(S, m, pred_p_sim)
sum(pred_y_sim >=5 & pred_y_sim <= 15)/S
```

```
## [1] 0.897
```

```r
a <- 3.06; b <- 2.56
n <- 20; y <- 12
prob <- pbetap(c(a + y, b + n - y), 20, 0:20)
```

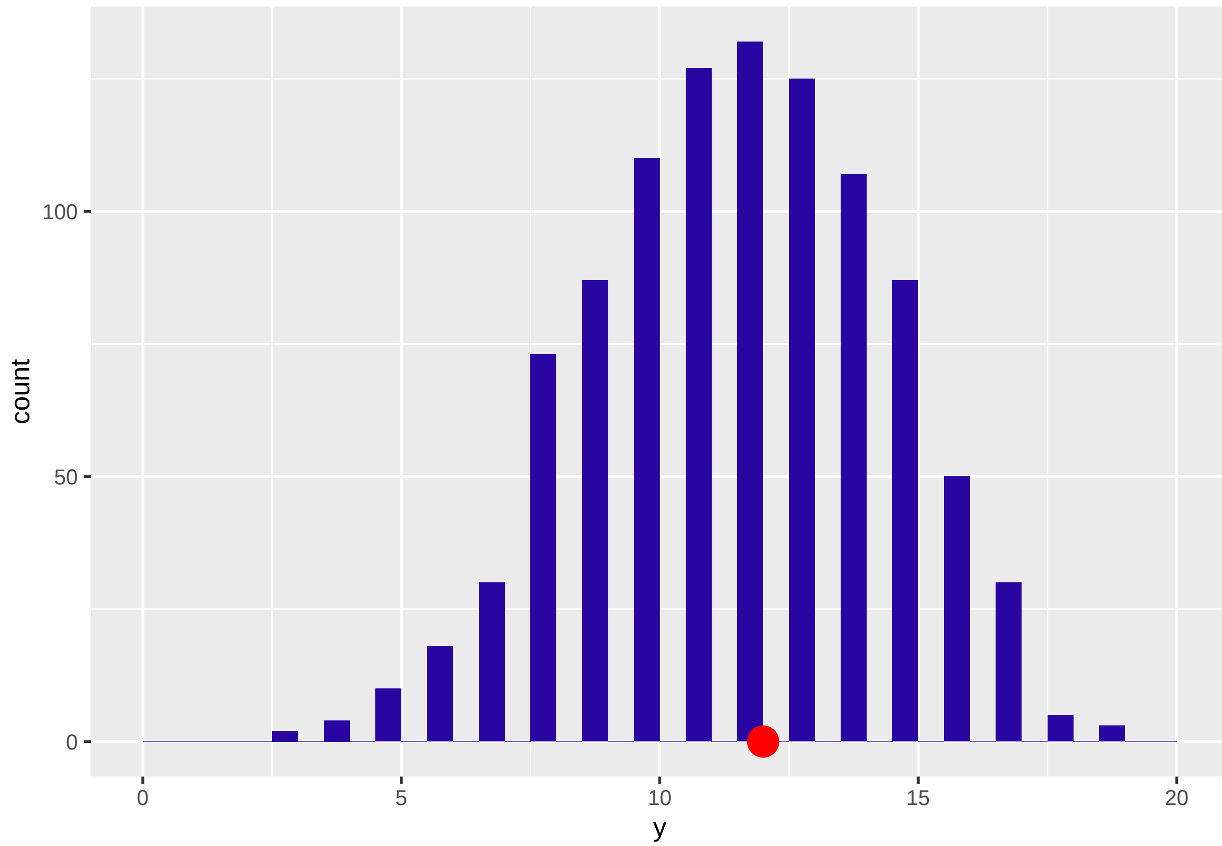
```r
T1 <- data.frame(Y = 0:10,
                 Probability = round(prob[1:11], 3))
T2 <- data.frame(Y = 11:20,
                 Probability = round(prob[12:21], 3))
T2 <- rbind(T2, data.frame(Y = 21, Probability = 999))

set.seed(123)
S <- 1000
pred_p_sim <- rbeta(S, a + y, a + b + n - y)
pred_y_sim <- rbinom(S, n, pred_p_sim)

data.frame(Y = pred_y_sim) %>%
  group_by(Y) %>% summarize(N = n()) %>%
  mutate(Probability = N / sum(N),
         Type = "Simulated")  %>%
  select(Type, Y, Probability) -> S1

S2 <- data.frame(Type = "Exact",
                 Y = 0:20,
                 Probability = prob)

S <- rbind(S1, S2)
ggplot(S, aes(Y, Probability)) +
  geom_segment(aes(xend = Y, yend = 0),
               size = 3,
               lineend = "butt",
               color = crcblue) +
  facet_wrap(~ Type, ncol=1) +
  theme(text=element_text(size=18))
```



Use R/Rstudio to perform posterior predictive checking

```
S <- 1000
a <- 3.06; b <- 2.56
n <- 20; y <- 12
newy = as.data.frame(rep(NA, S))
names(newy) = c("y")

set.seed(123)
for (s in 1:S){
  pred_p_sim <- rbeta(1, a + y, b + n - y)
  pred_y_sim <- rbinom(1, n, pred_p_sim)
  newy[s,] = pred_y_sim
}
```

```
ggplot(data=newy, aes(newy$y)) +
  geom_histogram(breaks=seq(0, 20, by=0.5), fill = crcblue) +
  annotate("point", x = 12, y = 0, colour = "red", size = 5) +
  xlab("y") + theme(text=element_text(size=10))
```




```r
sum(newy > y)/S
```

```
## [1] 0.407
```

```r
1 - sum(newy > y)/S
```

```
## [1] 0.593
```

Recap