HW 10

Jared Brotamonte

12/6/2023

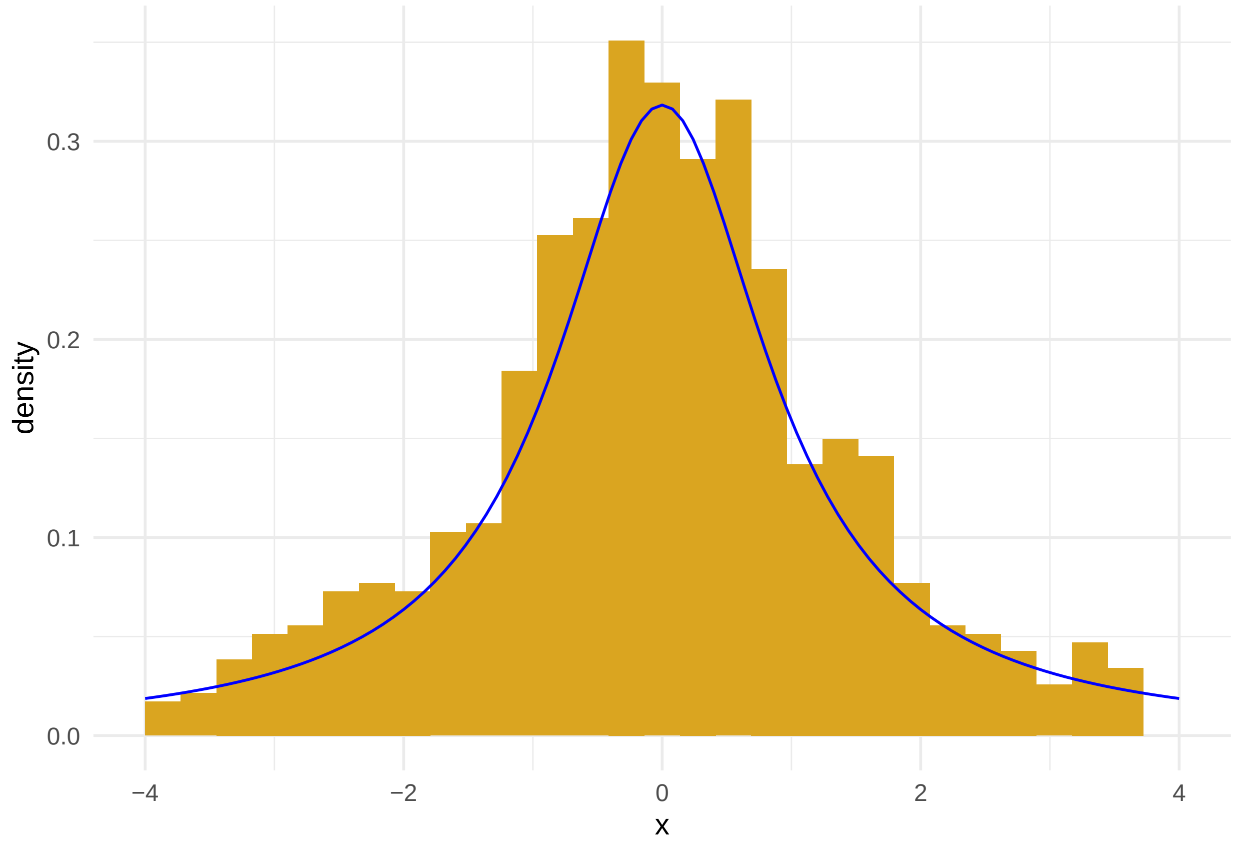
Exercise 1

(a)

```r
# Function to draw random samples from Cauchy distribution using Inverse CDF
draw_cauchy_samples <- function(n) {
  # get n random values from U(0,1)
  probs <- runif(n)

  # Inverse CDF for Cauchy distribution
  x <- qcauchy(probs, location = 0, scale = 1)

  return(x)
}

# Number of samples
n_samples <- 1000

# Draw 1000 random samples from Cauchy distribution
cauchy_samples <- data.frame(x = draw_cauchy_samples(n_samples))

# Plot the histogram and the true Cauchy density
cauchy_samples %>%
  ggplot() +
  geom_histogram(aes(x = x, y = ..density..), fill = "goldenrod", bins = 30) +
  stat_function(fun = dcauchy, color = "blue", args = list(location = 0, scale = 1)) +
  xlim(c(-4, 4)) +  # Set x-axis limits to [-4, 4]
  theme_minimal()
```

```
## Warning: Removed 153 rows containing non-finite values (stat_bin).
```

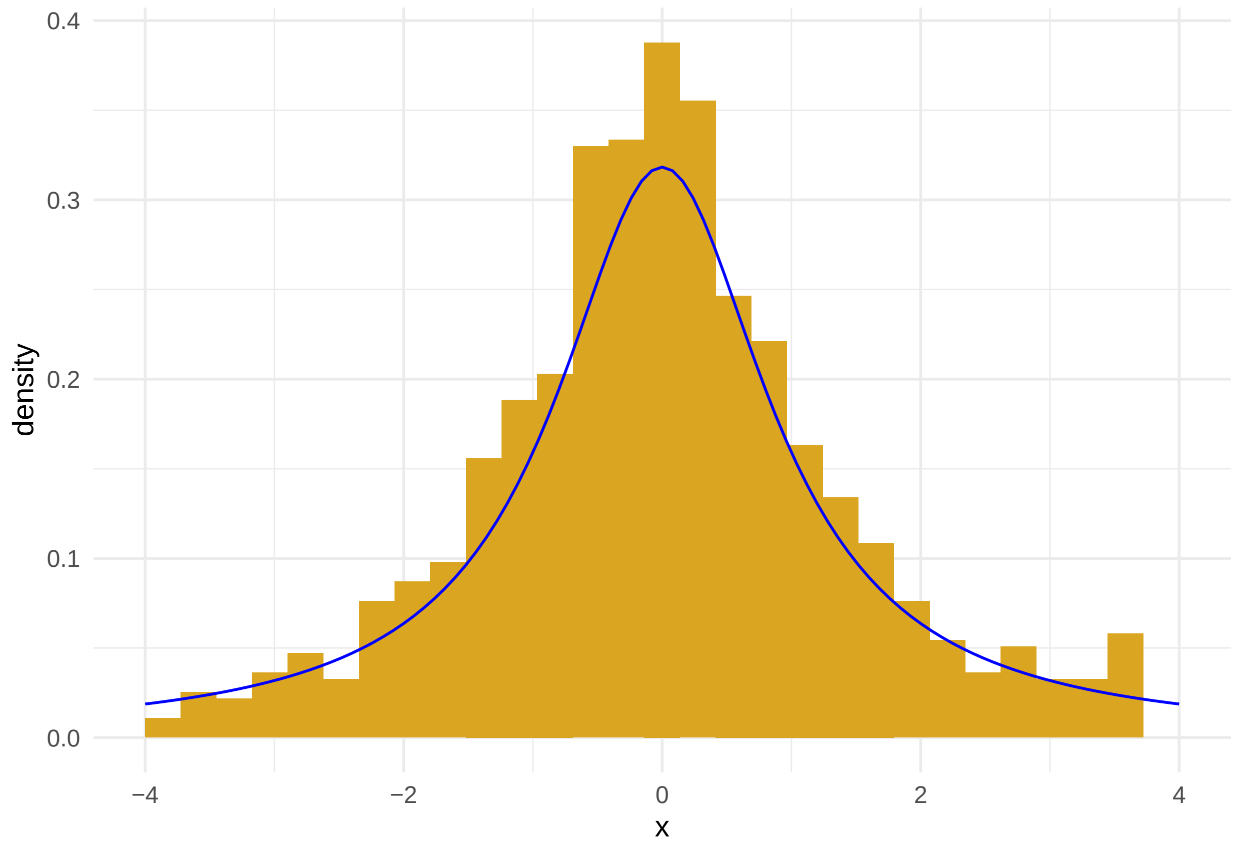
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



(b)

```r
# Load required libraries
library(ggplot2)

# Function to draw random samples from Cauchy distribution
draw_cauchy_samples_accept_reject <- function(n) {
  # define Cauchy density
  dcauchy_custom <- function(x) {
    return(dcauchy(x, location = 0, scale = 1))
  }

  # define parameters
  M <- 2.6
  n_samples <- n

  # create a vector to store samples
  samples <- c()
  count <- 0

  while(length(samples) < n_samples) {
    count <- count + 1

    # sample at random from x in the interval [-4, 4]
    x <- runif(1, -4, 4)

    # calculate f(x)
    fx <- dcauchy_custom(x)

    # calculate M * g(x)
    Mgx <- M * dunif(x, -4, 4)  # Using a uniform distribution for g(x) in the interval [-4, 4]

    # calculate acceptance probability
    prob <- fx / Mgx

    # draw random sample from U(0, 1)
    q <- runif(1)

    # accept or reject sample
    if (q < prob) {
      samples <- append(samples, x)
    }
  }

  return(samples)
}

# Number of samples
n_samples <- 1000

# Draw 1000 random samples from Cauchy distribution using Accept-Reject method
cauchy_samples_accept_reject <- draw_cauchy_samples_accept_reject(n_samples)

# Plot the histogram and the true Cauchy density
data.frame(x = cauchy_samples_accept_reject) %>%
  ggplot() +
  geom_histogram(aes(x = x, y = ..density..), fill = "goldenrod", bins = 30) +
  stat_function(fun = dcauchy, color = "blue", args = list(location = 0, scale = 1)) +
  xlim(c(-4, 4)) +  # Set x-axis limits to [-4, 4]
  theme_minimal()
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



(c)

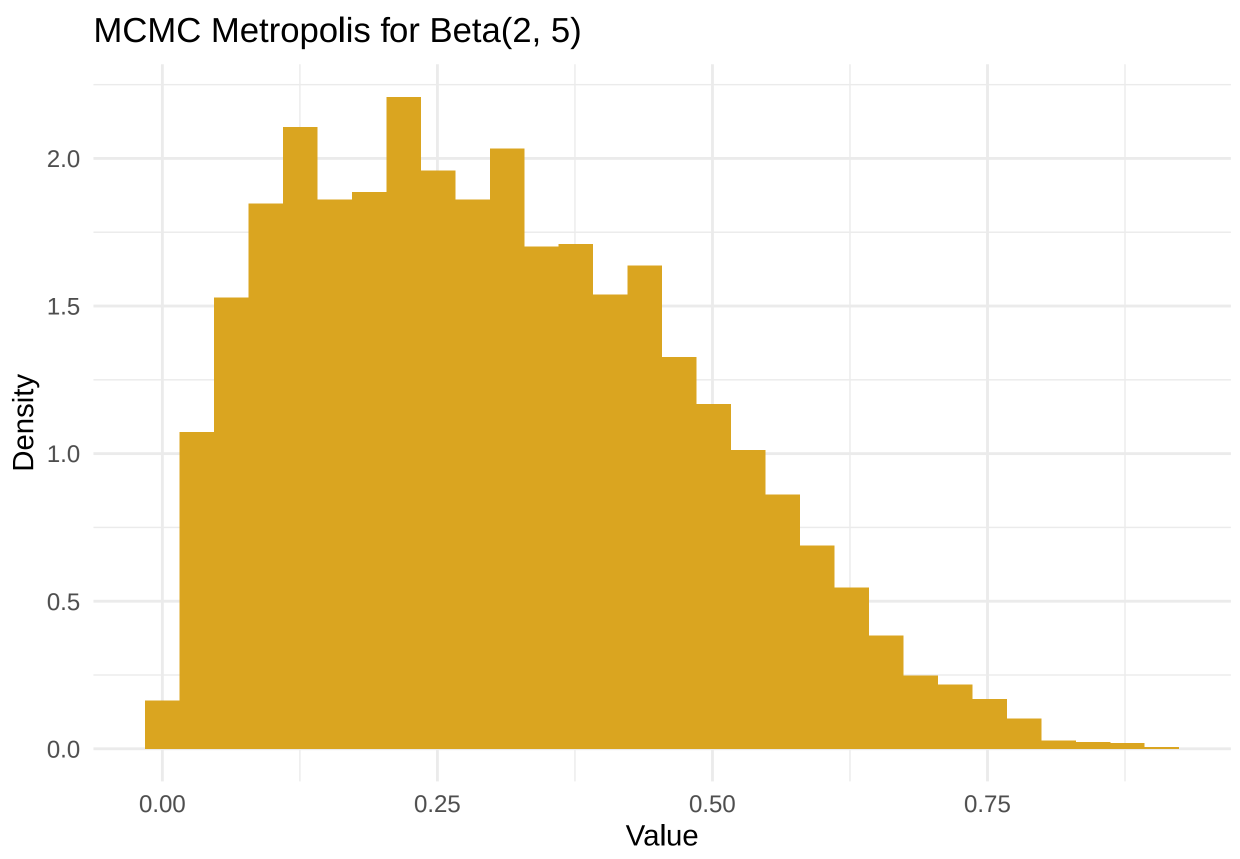
The histograms of part a and b look largely similar.

Exercise 2

(a)

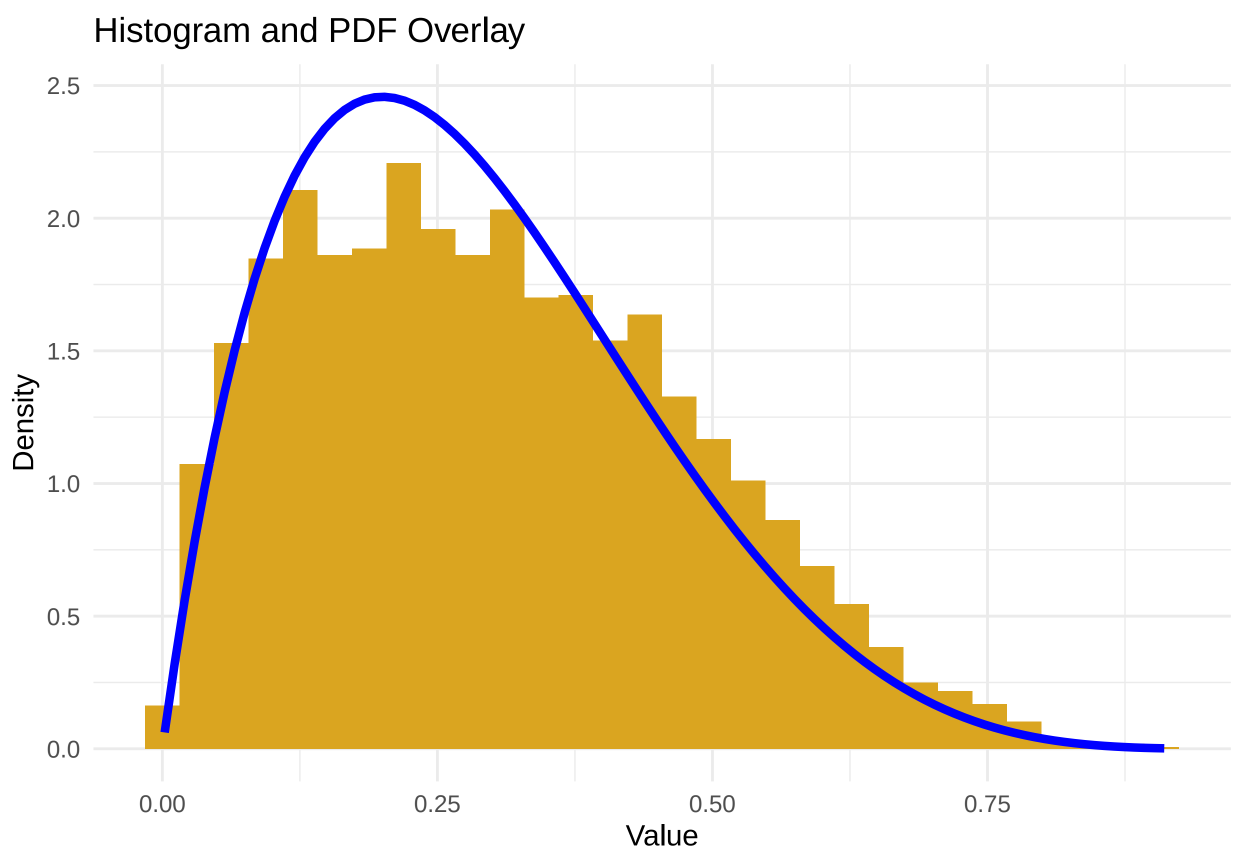
```
# Define the Beta distribution function
dbeta_custom <- function(x) {
  return(dbeta(x, shape1 = 2, shape2 = 5))
}

# MCMC Metropolis algorithm for Beta distribution
n_samples <- 10000
count <- 0
samples <- c()
burn_in <- 100

# Initial point
x <- runif(1)
```

```r
while (length(samples) < (n_samples + burn_in)) {
  count <- count + 1

  # Calculate f(x)
  fx <- dbeta_custom(x)

  # Generate candidate point at random from a uniform distribution
  x.star <- runif(1)

  # Calculate f(x.star)
  fx.star <- dbeta_custom(x.star)

  # Accept or reject sample
  if (fx.star > fx) {
    samples <- append(samples, x.star)
    x <- x.star
  } else {
    # Draw random sample from U(0,1)
    q <- runif(1)
    if (q < (fx.star / fx)) {
      samples <- append(samples, x.star)
      x <- x.star
    }
  }
}

# Remove burn-in
samples <- samples[(burn_in + 1):length(samples)]

# Plot the sample
data.frame(x = samples) %>%
  ggplot() +
  geom_histogram(aes(x = x, y = ..density..), fill = "goldenrod", bins = 30) +
  labs(title = "MCMC Metropolis for Beta(2, 5)", x = "Value", y = "Density") +
  theme_minimal()
```



(b)

```
# Define the Beta distribution function
dbeta_custom <- function(x) {
  return(dbeta(x, shape1 = 2, shape2 = 5))
}

# Plot histogram and overlay PDF
data.frame(x = samples) %>%
  ggplot() +
  geom_histogram(aes(x = x, y = ..density..), fill = "goldenrod", bins = 30) +
  stat_function(fun = dbeta_custom, color = "blue", size = 1.5) +
  labs(title = "Histogram and PDF Overlay", x = "Value", y = "Density") +
  theme_minimal()
```


Histogram and PDF Overlay
Density
Value
0.0
0.5
1.0
1.5
2.0
2.5
0.00
0.25
0.50
0.75