BDA Assignment 1

Piero Birello

January 2025

Exercise A: Coin flipping

```
set.seed(1)

p=list()  # list for storing the result/results

chain_probability_estimator = function(k=0.5, chain_length=6, n_flips=50, n_iters=10000)
{
  hit_v = rep(0,n_iters)
  for (i in 1:n_iters){    # loop for generating 1000   50 long binomial sequences
    binom_v = rbinom(n_flips, 1, k)
    countH=0
    countT=0
    hit=0
    for (j in 1:length(binom_v)){   # loop for checking wether a sequence contains a 6 long run
      if (binom_v[j] == 1){
        countH = countH + 1   #if it is head increment the head counter
        countT = 0            #and reset the tail counter
      }else{
        countH = 0            #if it is tail reset the head counter
        countT = countT + 1   #and increment the tail counter
      }
      if (countH == chain_length || countT == chain_length){  #if one of the counter reaches 6
        hit_v[i] = 1          #put 1 to the corresponding element of the hit vector
        break                 #escape from the inner for loop
      }
    }
  }
  estimated_p = sum(hit_v)/n_iters
  return(estimated_p)
}

names = c('ub','b')
ks = c(0.5,0.7)

for (i in 1:2){p[names[i]]=chain_probability_estimator(k=ks[i])}

for (name in names) {
  print(sprintf("p_%s: %.3f", name, p[name]))
}


# for the 5. part of the exercise A

q=list(ub=0.6, b=0.4)   # biased and unbiased coin proportions in the bag

```

```
prob_6seq = p[["ub"]]*q[["ub"]] + p[["b"]]*q[["b"]] # evidence
print( sprintf('prob_6seq: %.2f%%', prob_6seq*100))

prob_biasG6seq = p[["b"]]*q[["b"]]/prob_6seq  # probability of biased coin Given to the six
    long sequence
print( sprintf('prob_biasG6seq: %.2f%%', prob_biasG6seq*100))
```

Exercise B: Beta function and plotting

```
set.seed(1)
library(Hmisc)

# function for extracting samples, plotting and getting summary statistics
# given parameters a and b for the Beta distribution
beta_hist = function(a = 10, b= 7)
{
beta_v = rbeta( 100000 , a , b ) # Beta distribution
hist(beta_v, # Histogram
     breaks = seq(0, 1, by = 0.05),
     freq = FALSE,
     xlim = c(0, 1),
     ylim = c(0, 11),
     col = "blue",
     xlab = "beta hist"
     )
mean_beta = mean(beta_v)  # mean of beta_v
sd_beta = sd(beta_v) #sd of beta_v
abline(v=mean_beta, col="red", lwd=4)
text(x = mean_beta+0.15, y = 7.5, labels = sprintf("mean: %.3f", mean_beta))
return(list(mean=mean_beta,sd=sd_beta))
}

par(mfrow = c(2, 2)) # 2x2 plotting grid

# define iterables
a = c(10,50,100)
b = c(7,35,70)
mean_beta = c()
sd_beta = c()

# iterate
for (i in 1:3) {
  result = beta_hist(a[i],b[i])
  mean_beta[i] = result$mean
  sd_beta[i] = result$sd
}

print(mean_beta)

# Errorbar plot
x_vals = a + b
errbar(x_vals, mean_beta,
       yplus = mean_beta + sd_beta,
       yminus = mean_beta - sd_beta,
       col = "red",
       errbar.col = "blue",
       ylim = c(0.45,0.75),
       xlab = "a + b",
       ylab = "mean_beta")
```

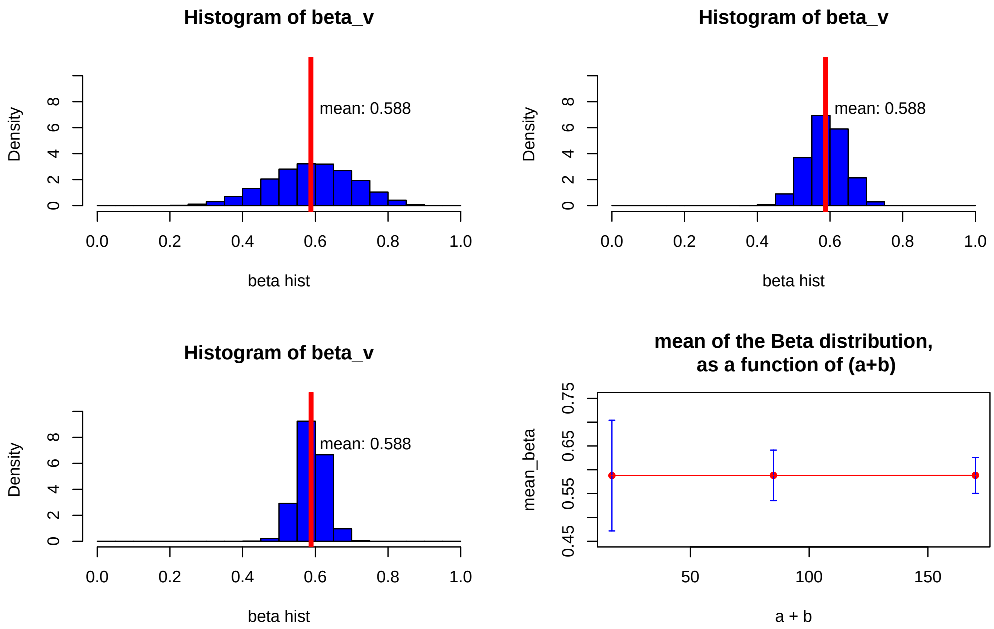

Figure 1: Result of exercise B. In the last plot, I preferred to maintain means in red and display the standard deviation in blue, for consistency.

```
lines(x_vals, mean_beta, col = "red")
title("mean of the Beta distribution,\n as a function of (a+b)")

dev.print(file="ExB.pdf", pdf) #, height = 800, width=800)
dev.off()
ExB.pdf", pdf) #, height = 800, width=800)
dev.off()
```

Exercise C: Coin flipping again

Denote the two admissible models M_α, $\alpha \in \{1, 2\}$. Our data consists of two sequences $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$. Then the Bayes factor κ can be computed as:

$$\kappa = \frac{p(\mathbf{y}^{(1)}, \mathbf{y}^{(2)} | M_1)}{p(\mathbf{y}^{(1)}, \mathbf{y}^{(2)} | M_2)} \tag{1}$$

$$= \frac{p(\mathbf{y}^{(1)} | M_1)\, p(\mathbf{y}^{(2)} | M_1)}{p(\mathbf{y}^{(1)} | M_2)\, p(\mathbf{y}^{(2)} | M_2)} \tag{2}$$

with:

$$p(\mathbf{y}|M_\alpha) = \sum_k \prod_i p(y_i | k, M_\alpha) p(k | M_\alpha) \tag{3}$$

and:

$$p(y_i | k, M_\alpha) = p(y_i | k) = k^{y_i} \, (1 - k)^{1 - y_i} \tag{4}$$

We obtain $\kappa = 1.76841693183448$.

```
# Define variables for the two bags
p_k_given_M1 = c(0.5, 0.5)  # probability of k given model 1
k_M1 = c(0.7, 0.5)  # k values in model 1
p_k_given_M2 = rep(1/9, 9)  # probability of k given model 2
k_M2 = seq(0.1, 0.9, by=0.1)  # k values in model 2

# Single lists
p_k = list(p_k_given_M1, p_k_given_M2)
k = list(k_M1, k_M2)

# Tosses data
y = rbind(
  c(0, 1, 0, 0, 1, 0, 1, 0),
  c(0, 1, 1, 1, 0, 1, 1, 1)
)

# Define bernoulli distribution
bernoulli_seq = function(y, k) {
  prod(k^y * (1 - k)^(1 - y))
}

# Define likelihood of model alpha
likelihood = function(y, alpha) {
  sum(sapply(1:length(k[[alpha]]), function(i) {
    bernoulli_seq(y, k[[alpha]][i]) * p_k[[alpha]][i]
  }))
}

# Calculate likelihoods
likelihood_M1 = prod(apply(y, 1, function(y_) likelihood(y_, alpha=1)))
likelihood_M2 = prod(apply(y, 1, function(y_) likelihood(y_, alpha=2)))
K = likelihood_M1 / likelihood_M2

# Print results
print(paste("Likelihood M1:", likelihood_M1))
print(paste("Likelihood M2:", likelihood_M2))
print(paste("K:", K))
```