Lesson5

## Пункт 1

Для пункту 1 опишимо наші логарифмічні функції щільності для наших розподілів і побудуємо наші вибірки. Також змоделюємо наші значення і подивимось на помилку першого роду і другого роду відповідно.

set.seed(5)  
  
n <- 50  
  
alpha\_H0 <- 2  
beta\_H0 <- 5  
  
alpha\_H1 <- 2  
beta\_H1 <- 4   
  
f0 <- function(x){log(dbeta(x, alpha\_H0, beta\_H0))}  
f1 <- function(x){log(dbeta(x, alpha\_H1, beta\_H1))}  
  
lr <- function(x)sum(sapply(x,f1)-sapply(x,f0))  
  
gen0 <- function(n)rbeta(n, alpha\_H0, beta\_H0)  
gen1 <- function(n)rbeta(n, alpha\_H1, beta\_H1)  
  
alpha <- 0.05  
  
B<-10000  
lr0<-replicate(B,lr(gen0(n)))  
  
Ca<-quantile(lr0,1-alpha)  
Ca

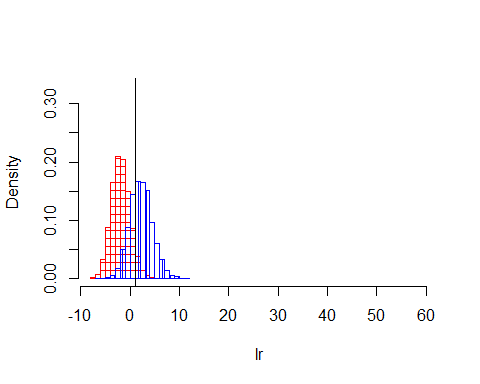
## 95%   
## 1.137396

Бачимо, що для рівня значущості =0.05 ми отримали поріг тесту = 1.137396

lr1<-replicate(B,lr(gen1(n)))  
mean(lr1<Ca)

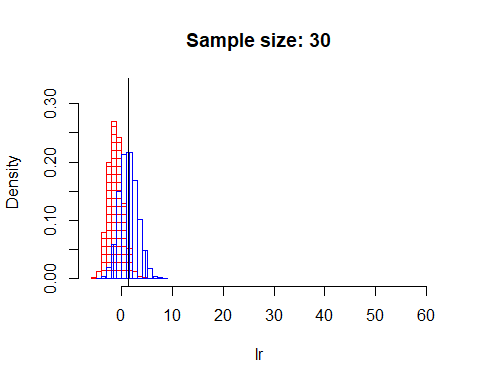
## [1] 0.3248

mi<-min(c(lr0,lr1))  
mx<- 60  
hist(lr0,breaks=15,probability=T,angle=0,density=12,xlim=c(mi,mx),ylim=c(0,0.33),col="red",xlab="lr",main=" ")  
hist(lr1,probability=T,breaks=15,angle=90,density=12, xlim=c(mi,mx),col="blue",add=T)  
abline(v=Ca)

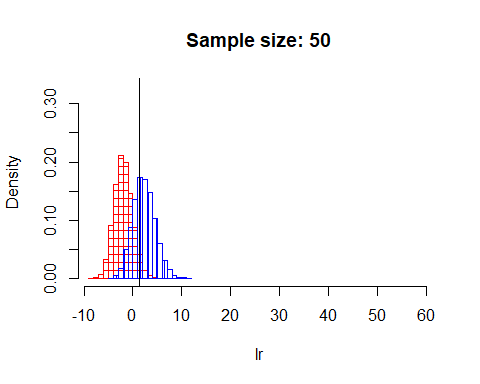


Також ймовірністьпомилки другого роду складатиме 0.3248 (32.48%). Це досить велике значення, і важко на даному етапі оцінити якість нашого тесту. Також враховуючи гістограми, можна сказати, що якість тесту важко зараз оцінювати. Спробуємо подивитись на різні значення n, змінюючи кількість наших значень. Видозмінимо наш код і додамо різні значення n: 30, 50, 100, 200

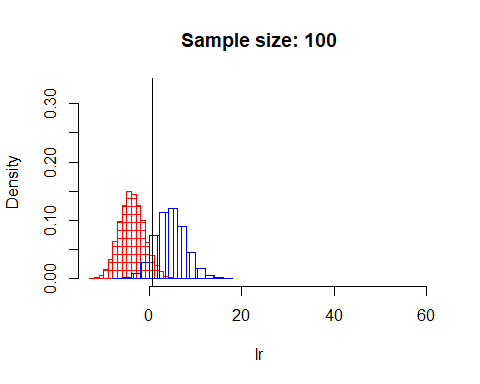
set.seed(5)  
  
alpha\_H0 <- 2  
beta\_H0 <- 5  
  
alpha\_H1 <- 2  
beta\_H1 <- 4   
  
f0 <- function(x) log(dbeta(x, alpha\_H0, beta\_H0))  
f1 <- function(x) log(dbeta(x, alpha\_H1, beta\_H1))  
  
lr <- function(x) sum(sapply(x, f1) - sapply(x, f0))  
  
gen0 <- function(n) rbeta(n, alpha\_H0, beta\_H0)  
gen1 <- function(n) rbeta(n, alpha\_H1, beta\_H1)  
  
alpha <- 0.05  
B <- 10000  
  
sample\_sizes <- c(30, 50, 100, 200)  
  
for (n in sample\_sizes) {  
 lr0 <- replicate(B, lr(gen0(n)))  
 Ca <- quantile(lr0, 1 - alpha)  
 lr1 <- replicate(B, lr(gen1(n)))  
 p\_value <- mean(lr1 < Ca)  
  
 mi <- min(c(lr0, lr1))  
 mx <- 60  
 hist(lr0, breaks = 15, probability = TRUE, angle = 0, density = 12, xlim = c(mi, mx), ylim = c(0, 0.33), col = "red", xlab = "lr", main = paste("Sample size:", n))  
 hist(lr1, probability = TRUE, breaks = 15, angle = 90, density = 12, xlim = c(mi, mx), col = "blue", add = TRUE)  
 abline(v = Ca)  
  
 cat("Sample size:", n, "Critical value:", Ca, "P-value:", p\_value, "\n")  
}



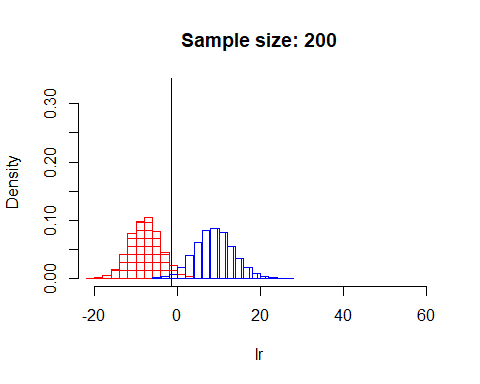
## Sample size: 30 Critical value: 1.222057 P-value: 0.4927



## Sample size: 50 Critical value: 1.26122 P-value: 0.3381



## Sample size: 100 Critical value: 0.6657194 P-value: 0.1146



## Sample size: 200 Critical value: -1.493837 P-value: 0.009

Бачимо, що при збільшені кількості вибірки, а також при розумних рівнях значущості бачимо, що наші дані свідчать на користь альтернативи.

Перевіримо різні значення, щоб подивитись на найменший розмір n, при якому тест матиме ймовірність помилок першого і другого роду, що не перевищують 0.05

set.seed(5)  
  
  
alpha\_H0 <- 2  
beta\_H0 <- 5  
alpha\_H1 <- 2  
beta\_H1 <- 4  
alpha\_level <- 0.05  
B <- 10000  
  
generate\_data <- function(a, b, n) {  
 return(rbeta(n, a, b))  
}  
  
log\_likelihood\_ratio <- function(x, a\_H0, b\_H0, a\_H1, b\_H1) {  
 return(sum(log(dbeta(x, a\_H1, b\_H1) / dbeta(x, a\_H0, b\_H0))))  
}  
  
estimate\_errors <- function(n) {  
 data\_H0 <- replicate(B, generate\_data(alpha\_H0, beta\_H0, n))  
 lr\_H0 <- apply(data\_H0, 2, function(x) log\_likelihood\_ratio(x, alpha\_H0, beta\_H0, alpha\_H1, beta\_H1))  
 Ca <- quantile(lr\_H0, 1 - alpha\_level)  
 alpha\_error <- mean(lr\_H0 > Ca)  
 data\_H1 <- replicate(B, generate\_data(alpha\_H1, beta\_H1, n))  
 lr\_H1 <- apply(data\_H1, 2, function(x) log\_likelihood\_ratio(x, alpha\_H0, beta\_H0, alpha\_H1, beta\_H1))  
 beta\_error <- mean(lr\_H1 < Ca)  
 return(c(alpha\_error, beta\_error))  
}  
  
optimal\_n <- NULL  
for (n in seq(10, 150, by=10)) {  
 errors <- estimate\_errors(n)  
 cat("n:", n, "Alpha error:", errors[1], "Beta error:", errors[2], "\n")  
 if (errors[1] <= alpha\_level && errors[2] <= alpha\_level) {  
 optimal\_n <- n  
 break  
 }  
}

## n: 10 Alpha error: 0.05 Beta error: 0.7341   
## n: 20 Alpha error: 0.05 Beta error: 0.5974   
## n: 30 Alpha error: 0.05 Beta error: 0.5038   
## n: 40 Alpha error: 0.05 Beta error: 0.4181   
## n: 50 Alpha error: 0.05 Beta error: 0.3421   
## n: 60 Alpha error: 0.05 Beta error: 0.2725   
## n: 70 Alpha error: 0.05 Beta error: 0.2122   
## n: 80 Alpha error: 0.05 Beta error: 0.1768   
## n: 90 Alpha error: 0.05 Beta error: 0.1332   
## n: 100 Alpha error: 0.05 Beta error: 0.1098   
## n: 110 Alpha error: 0.05 Beta error: 0.081   
## n: 120 Alpha error: 0.05 Beta error: 0.0608   
## n: 130 Alpha error: 0.05 Beta error: 0.043

optimal\_n

## [1] 130

Бачимо, що оптимальним значенням, при якому ймовірність помилок першого і другого роду не буде перевищувати 0.05 буде при n = 130