23847-Lab-2

23847

2024-03-03

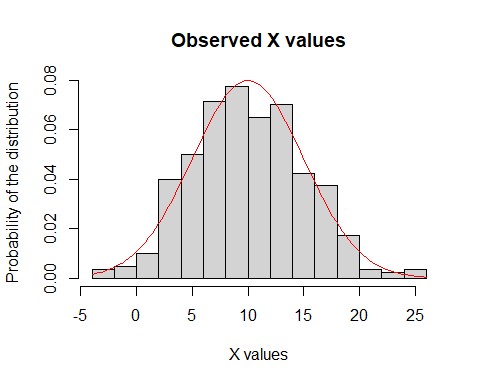
## Problem 1

### a)

We will use the function *rnorm(n, m, s)* to generate 400 realizations with *m := 10* and *s := 5*. Then the *hist()* function will be used to plot a histogram of the vector x. Finally the curve function can be used with the PDF of the normal distribution,

to superimpose N(10, 5^2) onto the histogram.

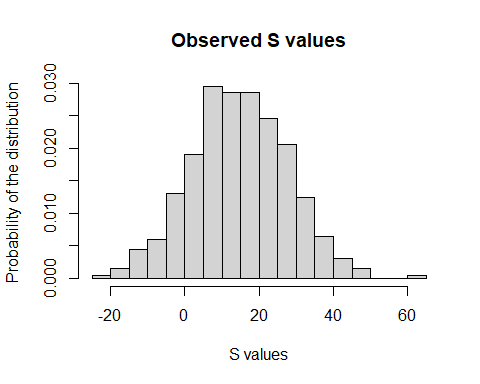
m = 10  
s = 5  
n = 400  
X = rnorm(n, m, s)  
hist(X,breaks = 15, prob = TRUE, main = "Observed X values", ylab = "Probability of the distribution", xlab = "X values")  
curve((2\*pi\*s^2)^(-1/2)\*exp(-(x - m)^2 / (2\*s^2)), add = TRUE, col = "red")



### b)

Much like 1a) we will use the *rnorm(n, m, s)* function, this time with different m and s values. Then we construct the vector S as Y - X as required.

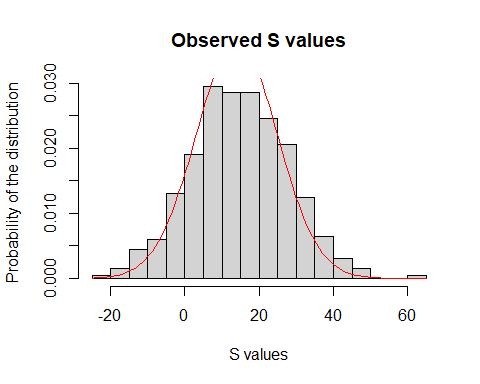
m2 = 24  
s2 = 12  
Y = rnorm(n, m2, s2)  
S = c(Y - X)  
hist(S, breaks = 15, prob = TRUE, main = "Observed S values", xlab = "S values", ylab = "Probability of the distribution")



### c)

S appears to follow a normal distribution. Recall, if X ~ N(a, b2) and Y ~ N(c, d2) then X + Y ~ N(a + c, b2 + d2) which implies S ~ N(24 - 10, 12^2 - 5^2) = N(-14, 119). TO avoid unnecessary computation v will represent the variance as we do not use the standard deviation in the normal distribution.

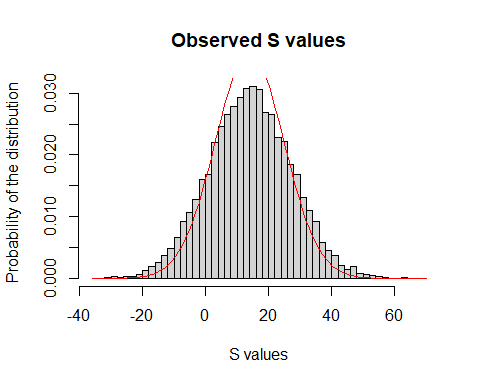
m3 = 14  
v = 119  
hist(S, breaks = 15, prob = TRUE, main = "Observed S values", xlab = "S values", ylab = "Probability of the distribution")  
curve((2\*pi\*v)^(-1/2)\*exp(-(x - m3)^2 / (2\*v)), add = TRUE, col = "red")



### d)

A larger value of n could be used for X and Y. More breaks could be used. The results appear to agree with the estimation of the PDF.

n = 10000  
X = rnorm(n, m, s)  
Y = rnorm(n, m2, s2)  
S = c(Y - X)  
hist(S, breaks = 50, prob = TRUE, main = "Observed S values", xlab = "S values", ylab = "Probability of the distribution")  
curve((2\*pi\*v)^(-1/2)\*exp(-(x - m3)^2 / (2\*v)), add = TRUE, col = "red")

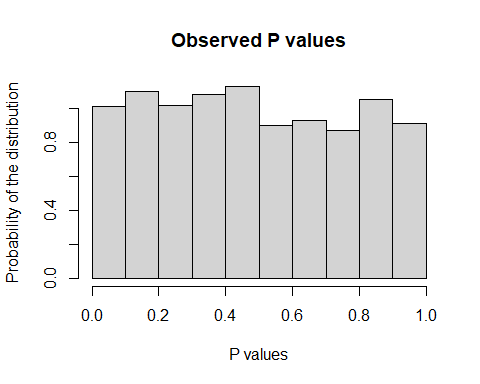


## Problem 2

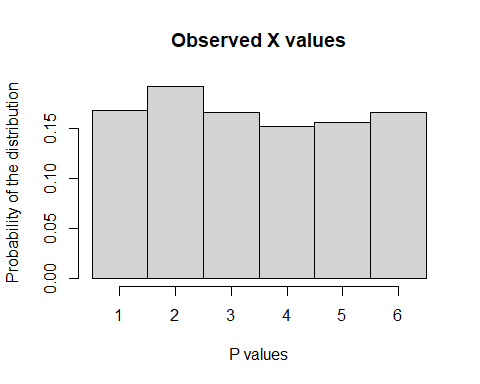
### a)

We will use *runif(n, a, b)* to generate a random value from the Unif(0, 1) distribution for P

n = 1000  
Psample = runif(n, 0, 1)  
Xsample = numeric(n)  
for(i in 1:n){  
 Xsample[i] = rbinom(1, 5, Psample[i]) + 1  
}  
hist(Psample, prob = TRUE, main = "Observed P values", xlab = "P values", ylab = "Probability of the distribution")



hist(Xsample, breaks = c(0.5:6.5), prob = TRUE, main = "Observed X values", xlab = "P values", ylab = "Probability of the distribution")



### b)

We will use the *table()* function to count the occurrences of each value of i and then, assuming all events are equally likely, we may divide it by the sample space, which is n in this case. Each value will be assigned to p\_i for i in {1, …, 6} respectively. Instead of having 6 of the same lines each we may store these probabilities in a vector *probX* where the index corresponds to i.

probX = numeric(6)  
for(i in 1:6){  
 probX[i] = table(Xsample)[[as.character(i)]] / n  
}

### c)

by calculating *sum(probX) / 6* we can see the average value of *probX* is 0.1666667 ~ 1/6 so it agrees with his claim. To further verify we may increase n and calculate again, to which we see it still agrees.

sum(probX) / 6

## [1] 0.1666667

# Verifying  
n = 100000  
Psample = runif(n, 0, 1)  
Xsample = numeric(n)  
for(i in 1:n){  
 Xsample[i] = rbinom(1, 5, Psample[i]) + 1  
}  
probX = numeric(6)  
for(i in 1:6){  
 probX[i] = table(Xsample)[[as.character(i)]] / n  
}  
sum(probX) / 6

## [1] 0.1666667

### d)

#### i)

Firstly we shall define a function, *experiment* which takes a parameter *n > 0* so as to not have repeated code. trying to run experiment(4) we get an error that the subscript is out of bounds. Initially min(6, n) was used as it was assumed the for loop was attempting to call for 6 values instead of 4 however this did not fix it. It seems the error is that the table function does not like it when values that do not occur (which will be guaranteed for n < 6) are present in the Xsample. To fix this we will sum the number of values equal to *i* and then divide by *i* as an improvised counter. Now we see for n < 5 the values are roughly 1/3n. For n=1 or n = 2 we get a probabiliy of zero.

#### ii)

As seen in **2c)** a large value of n agrees with Dr Watson’s claim

In conclusion Dr Watson’s claim is only accurate for large values of n.

experiment = function(n){  
 if(n <= 0) {  
 0  
 }else{  
 Psample = runif(n, 0, 1)  
 Xsample = numeric(n)  
 for(i in 1:n){  
 Xsample[i] = rbinom(1, 5, Psample[i]) + 1  
 }  
 probX = numeric(6)  
 for(i in 1:min(6, n)){  
 probX[i] = sum(Xsample == i) / n  
 }  
 sum(probX) / 6  
 }  
 }  
  
#i)   
print(experiment(5))

## [1] 0.1666667

#ii)  
print(experiment(100000))

## [1] 0.1666667