#Exercise-1

#Computing Coverage Probability for different values of p and sample size.

#defined a function conf.int for computing the 95% confidence interval with each random variable coming from a binomial trial.

conf.int <- function(n, p, m=10000, alpha=0.05){

x <- rbinom(m, n, p)

#p.hat is the estimate probability of p computed as sampel mean.

p.hat <-x/n

#ci <- p.hat + c(-1,1)\*qnorm(1 - (alpha/2))\*sqrt(p.hat\*(1-p.hat)/n)

#lci is the lower limit of confidence interval computed for a normal distribution with 95 % confidence interval.

lci <- p.hat - qnorm(1 - (alpha/2))\*sqrt(p.hat\*(1-p.hat)/n)

#uci is the upper limit of confidence interval computed for a normal distribution with 95 % confidence interval.

uci <- p.hat + qnorm(1 - (alpha/2))\*sqrt(p.hat\*(1-p.hat)/n)

return(sum(lci < p & uci> p)/m)

}

#Vectorize the returned value from function conf.int

conf.vec <- Vectorize(conf.int)

#n.vec is a vector of all the sample sizes to be considered

n.vec <- c(5, 10, 30, 50, 100)

#p.vec is a vector of all the values of p to be considered

p.vec <- c(0.05, 0.1, 0.25, 0.5, 0.9, 0.95)

#outer function returns a product of n.vec and p.vec with conf.vec function returning values with its parameters taking values from n.vec and p.vec.

cov.mat <-outer(n.vec, p.vec, conf.vec)

#rownames takes row values from n.vec

rownames(cov.mat) <- n.vec

#colnames takes column values from p.vec

colnames(cov.mat) <- p.vec

#printing the value of cov.mat in the form of a matrix

print(cov.mat)

#plot() function generates a plot with its y-label and x-label

plot(NA, xlim = c(0,1), ylim = c(0,1), ylab = "Coverage Probability", xlab = "p")

#lines() function plots a line on the plot with values from cov.mat

for(i in seq\_along(n.vec)){

lines(p.vec, cov.mat[i,], type = "b", col = i)

}

#abline() function draws a straight line on the plot for value 0.95 i.e. the value of confidence interval

abline(h = 0.95)

#legend() function specifies the line for each value of n.

legend("bottom", col = seq(5), lwd = 1, legend = paste0("n = ", n.vec))