```
install.packages('MASS')
library('MASS')

#Part 1
#a
#P(x;p,n)=(n x)(p)^x(1−p)(n−x)
#P(X=0)=(4!/(0!*4!))*(0.5)^0*(0.5)^4=0.0625
#P(X=1)=(4!/(1!*3!))*(0.5)^1*(0.5)^3=0.25
#P(X=2)=(4!/(2!*2!))*(0.5)^2*(0.5)^2=0.375
#P(X=3)=(4!/(3!*1!))*(0.5)^3*(0.5)^1=0.25
#P(X=4)=(4!/(4!*0!))*(0.5)^4*(0.5)^0=0.0625
dbinom(0:4,4,.5)

#b
#P(X<0)
#P(X<0)=P(X=0)=0.0625
pbinom(0,4,.5)
#P(0<X<1)
#P(0<X<1)=P(X=1)+P(X=0)-P(X=0)=0.25+0.0625-0.0625=0.25
pbinom(1,4,.5) - pbinom(0,4,.5)
#P(1<X<2)
#P(1<X<2)=P(X=2)+P(X=1)+P(X=0)-P(X=1)-P(X=0)=0.375
pbinom(2,4,.5) - pbinom(1,4,.5)
#P(2<X<3)
#P(2<X<3)=P(X=3)+P(X=2)+P(X=1)+P(X=0)-P(X=2)-P(X=1)-P(X=0)=0.25
pbinom(3,4,.5) - pbinom(2,4,.5)
#P(3<X<4)
#P(3<X<4)=P(X=4)+P(X=3)+P(X=2)+P(X=1)+P(X=0)-P(X=3)-P(X=2)-P(X=1)-P(X=0)=0.0625
pbinom(4,4,.5) - pbinom(3,4,.5)
#P(X>4)
#P(X>4)=1-P(X<4); P(X<4)=P(X=4)+P(X=3)+P(X=2)+P(X=1)+P(X=0)=1; P(X>4)=0
1 - pbinom(4,4,.5)

#c
plot(0:4,dbinom(0:4,4,.5),type='h',ylab='f(x)',xlab='x',main='PMF of Bin(4,0.5)',ylim=c(0.0,0.4))
plot(-2:6,pbinom(-2:6,4,.5),type='s',ylab='F(x)',xlab='x',main='CDF of
Bin(4,0.5)',ylim=c(0.0,1.0),xlim=c(-1,5))

#Part 2
#a
#(5.3-4.7)/(8-2)=0.6/6=0.1=10%

#b
z <- runif(100000,2,8)
length(z[z>4.7&z<5.3])/length(z)

#c
#The P(4.7<x̄<5.3) should be around 76%

#d
ptm <- proc.time()
w <- c()
for (n in 1:100000){
  w <- c(w,mean(runif(48,2,8)))
}
proc.time() - ptm
#time to run code was 15.552
length(w[w>4.7&w<5.3])/length(w)
#The P(4.7<x̄<5.3) is about 76%

#e
ptm <- proc.time()
colMeans(matrix(runif(4800000,2,8),nrow=48,ncol=100000))
proc.time() - ptm
#time to run code was 0.283
y3 <- colMeans(matrix(runif(4800000,2,8),nrow=48,ncol=100000))
length(y3[y3>4.7&y3<5.3])/length(y3)
#The P(4.7<x̄<5.3) is about 76%

#f
ptm <- proc.time()
matrix1 <- matrix(data=1,nrow=48,ncol=100000)
matrix1
for (j in 1:100000){
  for (i in 1:48){
    matrix1[i,j]=runif(1,2,8)
  }
}
colMeans(matrix1)
proc.time() - ptm
#time to run code was 9.214
length(colMeans(matrix1)[colMeans(matrix1)>4.7&colMeans(matrix1)<5.3])/
length(colMeans(matrix1))
#The P(4.7<x̄<5.3) is about 76%

#g
#The most efficient in terms of execution time is clearly approach 2 where a
#matrix of values is created and the columns are averaged.


#Part 3
a <- 1:10
b <- 50:55
riffle <- function(x,y){
  i=1
  v <- c()
  if (length(x) >= length(y)) length1=y else length1=x
  if (length(x) >= length(y)) length2=x else length2=b
  while (i-1<length(length1)){
    v <- c(v,x[i])
    v <- c(v,y[i])
    i <- i+1
  }
  v <- c(v,length2[i:length(length2)])
  v
}
riffle(a,b)
```

```
#Part 4
insert <- function(x, where, what){
  if (where <= length(x)+1) append(x,what,where-1) else{
    print('Warning Message: index exceeds the dimensions of the vector')
    c(x,rep(NA,(where-length(a))-1),what)
  }
}
insert(a,15,100)
insert(a,6,100)

#Part 5
#a
#H0:μ1=μ2; Ha:μ1 ≠ μ2; Ha:μ1 < μ2; Ha:μ1 > μ2

#b
#The pooled variance estimate is represented by Sp=sqrt(((n1-1)*s1+(n2-1)*s2))/(n1+n2-2)
#where n1 and n2 are the sample sizes and s1 and s2 are the sample variances

#c
#The test statistic is equal to T=X̄1-X̄2/(Sp*sqrt(1/n1+1/n2))
#The null distribution is |T| < t1-α/2,v

#d
#Ha:μ1 ≠ μ2; t(1-α/2,v) P-value needs to be greater than T to accept null hypothesis
#Ha:μ1 < μ2; t(α,v) P-value needs to be greater than T to accept null hypothesis
#Ha:μ1 > μ2; t(1-α,v)  P-value needs to be less than T to accept null hypothesis

#f
test <- function(y1,y2,alt='two-sided',lev=0.95){
  if (alt=='two-sided'){
    add <- 0
    for (a in y1){
      add <- add + (a-mean(y1))^2
    }

    add2 <- 0
    for (b in y2){
      add2 <- add2 + (b-mean(y2))^2
    }
    test_statistic <- (mean(y1)-mean(y2))/(sqrt((add2/(length(y2)-1))/length(y2)+(add/
(length(y1)-1))/length(y1)))

    critical_value <- qt(1-((1-lev)/2),length(y1)+length(y2)-2)

    sd1 <- sqrt(add/length(y1))
    sd2 <- sqrt(add2/length(y2))
    p_value <- (mean(y1)-mean(y2))/sqrt((sd1/length(y1))+sd2/length(y2))

    SE <- sqrt((((add/(length(y1)-1))/length(y1)) + (add2/(length(y2)-1))/length(y2)))
    CI <- c(mean(y1)-mean(y2)-critical_value*SE,mean(y1)-mean(y2)+critical_value*SE)
    print(CI)
    print(test_statistic)

  }
  if (alt=='less'){
    add <- 0
    for (a in y1){
      add <- add + (a-mean(y1))^2
    }

    add2 <- 0
    for (b in y2){
      add2 <- add2 + (b-mean(y2))^2
    }
    test_statistic <- (mean(y1)-mean(y2))/(sqrt((add2/(length(y2)-1))/length(y2)+(add/
(length(y1)-1))/length(y1)))

    critical_value <- qt(1-((1-lev)),length(y1)+length(y2)-2)

    sd1 <- sqrt(add/length(y1))
    sd2 <- sqrt(add2/length(y2))
    p_value <- (mean(y1)-mean(y2))/sqrt((sd1/length(y1))+sd2/length(y2))

    SE <- sqrt((((add/(length(y1)-1))/length(y1)) + (add2/(length(y2)-1))/length(y2)))
    CI <- c(mean(y1)-mean(y2)-critical_value*SE,mean(y1)-mean(y2)+critical_value*SE)
    print(CI)
    print(test_statistic)
  }
  if (alt=='greater'){
    add <- 0
    for (a in y1){
      add <- add + (a-mean(y1))^2
    }

    add2 <- 0
    for (b in y2){
      add2 <- add2 + (b-mean(y2))^2
    }
    test_statistic <- (mean(y1)-mean(y2))/(sqrt((add2/(length(y2)-1))/length(y2)+(add/
(length(y1)-1))/length(y1)))

    critical_value <- qt(((1-lev)),length(y1)+length(y2)-2)

    sd1 <- sqrt(add/length(y1))
    sd2 <- sqrt(add2/length(y2))
    p_value <- (mean(y1)-mean(y2))/sqrt((sd1/length(y1))+sd2/length(y2))

    SE <- sqrt((((add/(length(y1)-1))/length(y1)) + (add2/(length(y2)-1))/length(y2)))
    CI <- c(mean(y1)-mean(y2)-critical_value*SE,mean(y1)-mean(y2)+critical_value*SE)
    print(CI)
    print(test_statistic)
    print(p_value)
  }

}

usa <- subset(Cars93,Origin %in% c('USA'))[['MPG.city']]
non_usa <- subset(Cars93,Origin %in% c('non-USA'))[['MPG.city']]
```

```
test(c(non_usa),c(usa),alt='two-sided',lev=0.95)
```