HW3

Ruiyi Jiang

2023-09-15

1.

(a)

```
cal_euler <- function(n){
  my_euler <- 0
  sum <- 0
  for (i in 1:n){
    sum <- sum + 1/i

  }
  my_euler <- -log(n, base=exp(1)) + sum

  return(my_euler)



}
```

```
true_euler <- 0.577215664901532
```

```
find_n_euler <- function(e){
  for (i in 1:10000000000){
    error <- abs(cal_euler(i) - true_euler)
    if (error < e){

      break
    }


  }



  print(i)



}
```

when $e = 1 * 10^{-3}$, the n is

```
find_n_euler(1*10^(-3))
```

```
## [1] 500
```

when $e = 1 * 10^{-4}$, the n is

```
find_n_euler(1*10^(-4))
```

```
## [1] 5000
```

when $e = 1 * 10^{-5}$, the n is

```
find_n_euler(1*10^(-5))
```

```
## [1] 50000
```

(b)

```
A <- c(0.1,1,10)
```

```
B <- c(0.5,10,30)
```

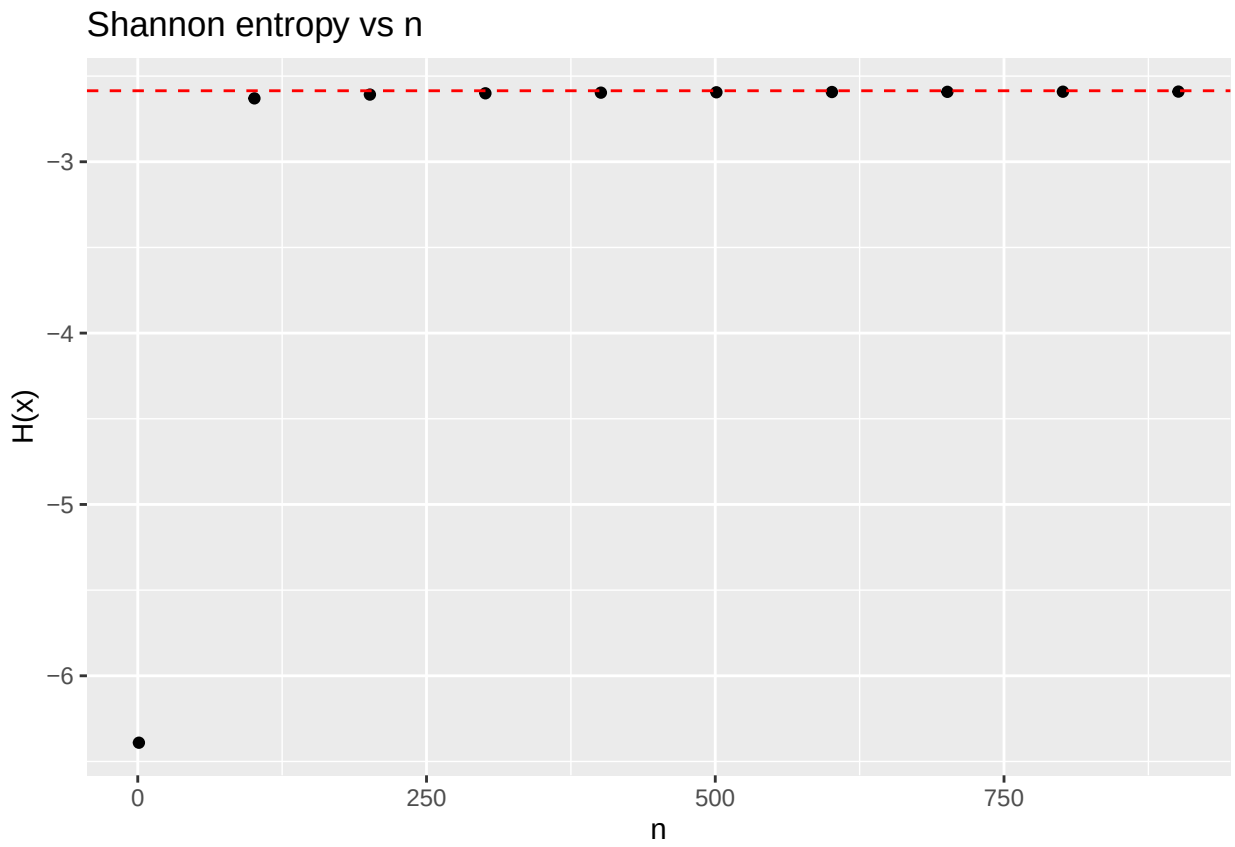
```
my_n <- seq(1,1000,100)

my_res <- matrix(NA,nrow=9,length(my_n) )


l<- 1

for (i in 1:3){
 for (j in 1:3){
  a <- A[i]
  b <- B[j]
   for(k in (1:length(my_n))){
     my_res[l,k] <-  cal_euler(my_n[k])*(1-1/a)+log(b/a)+1



   }
  l<- l+1

  }
   }
```

```
library(ggplot2)
```

```
my_res_df <- as.data.frame(t(my_res))
my_res_df$my_n <- my_n
```

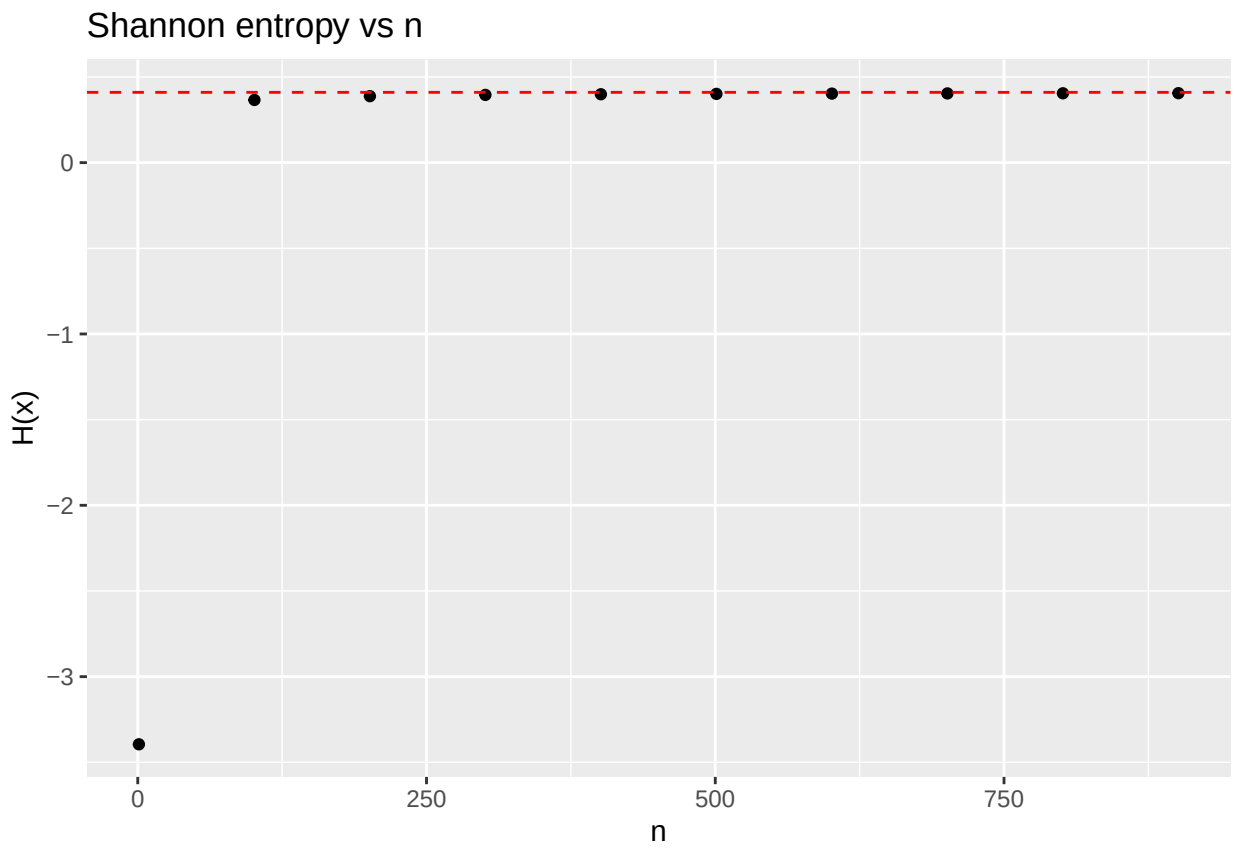
When $\alpha = 0.1, \beta = 0.5$:

```
ggplot(my_res_df,aes(x = my_n, y = V1) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/0.1)+log(5)+1, color =
               "red", linetype = "dashed")
```



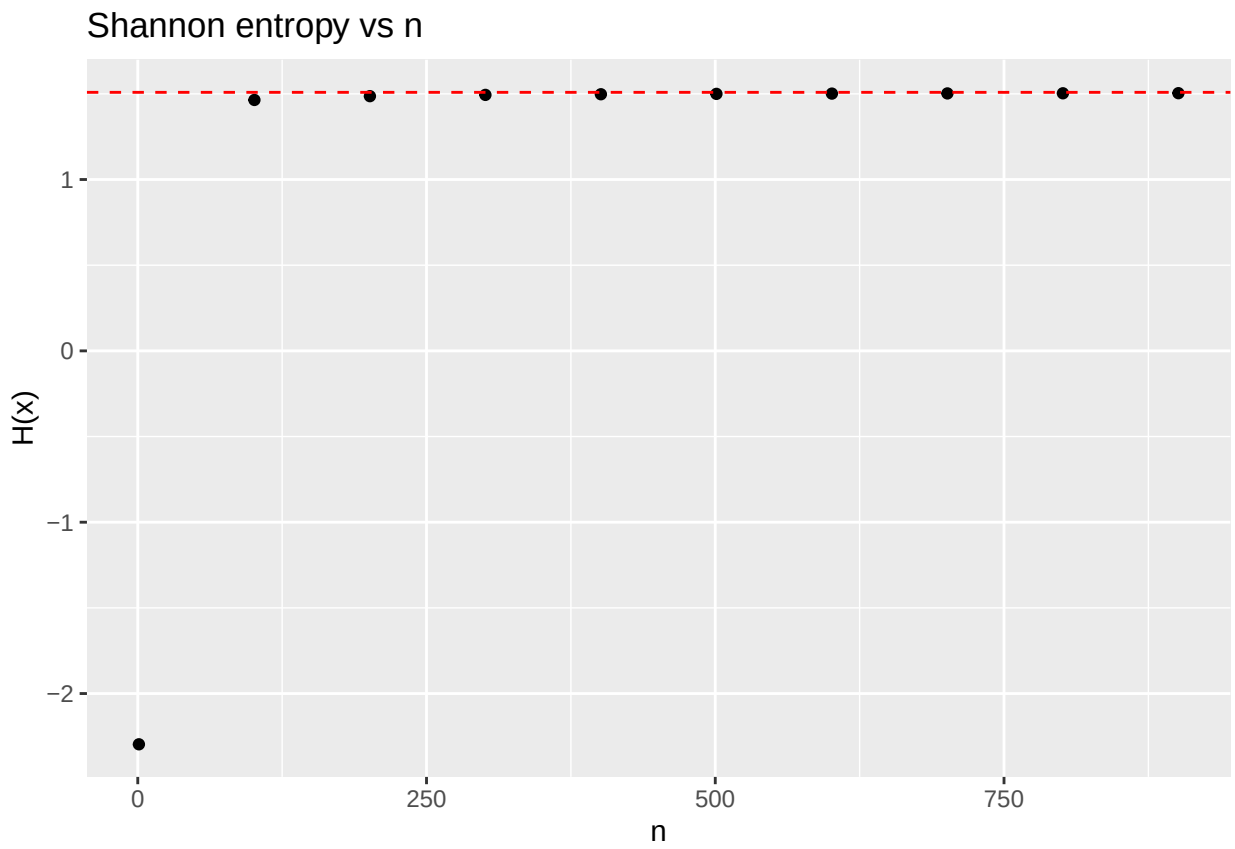
When $\alpha = 0.1, \beta = 10$:

```
ggplot(my_res_df,aes(x = my_n, y = V2) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/0.1)+log(10/0.1)+1, color =
               "red", linetype = "dashed")
```



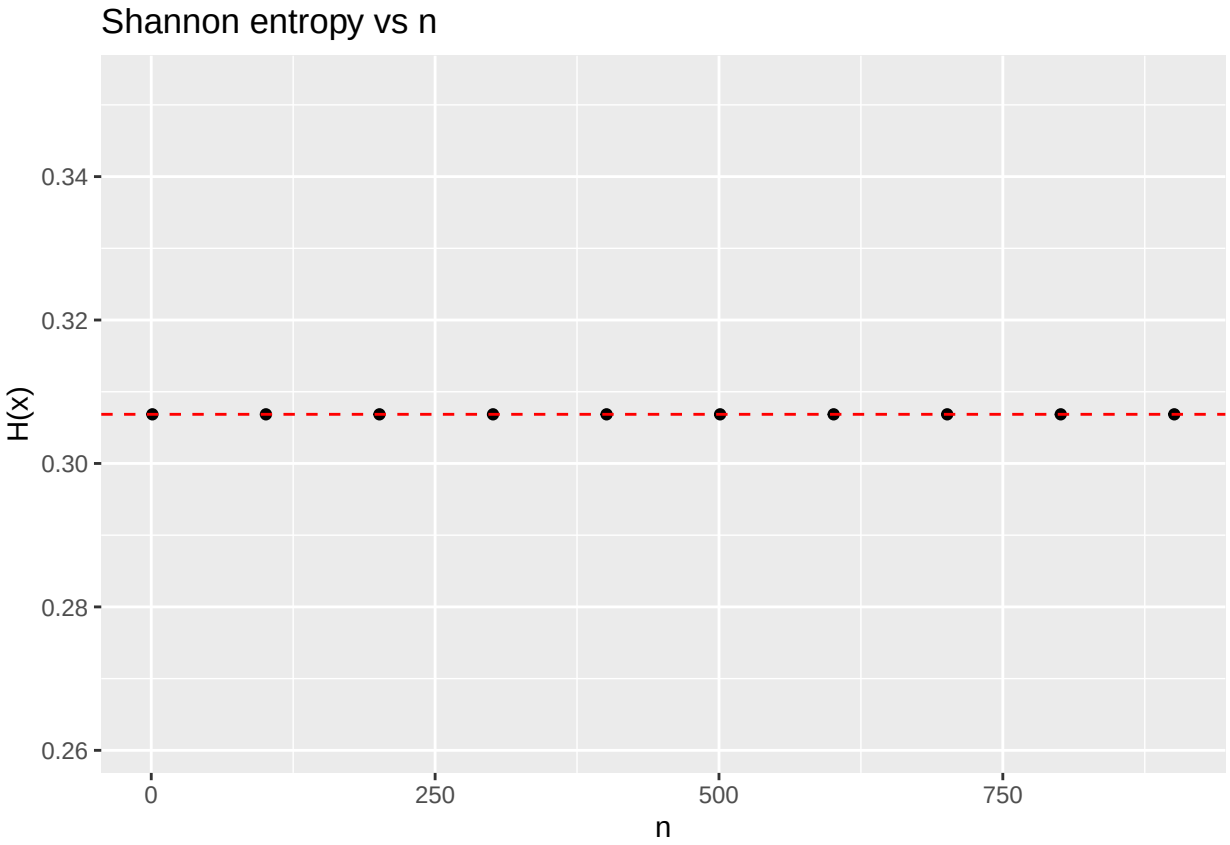
When $\alpha = 0.1, \beta = 30$:

```
ggplot(my_res_df,aes(x = my_n, y = V3) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/0.1)+log(30/0.1)+1, color =
               "red", linetype = "dashed")
```



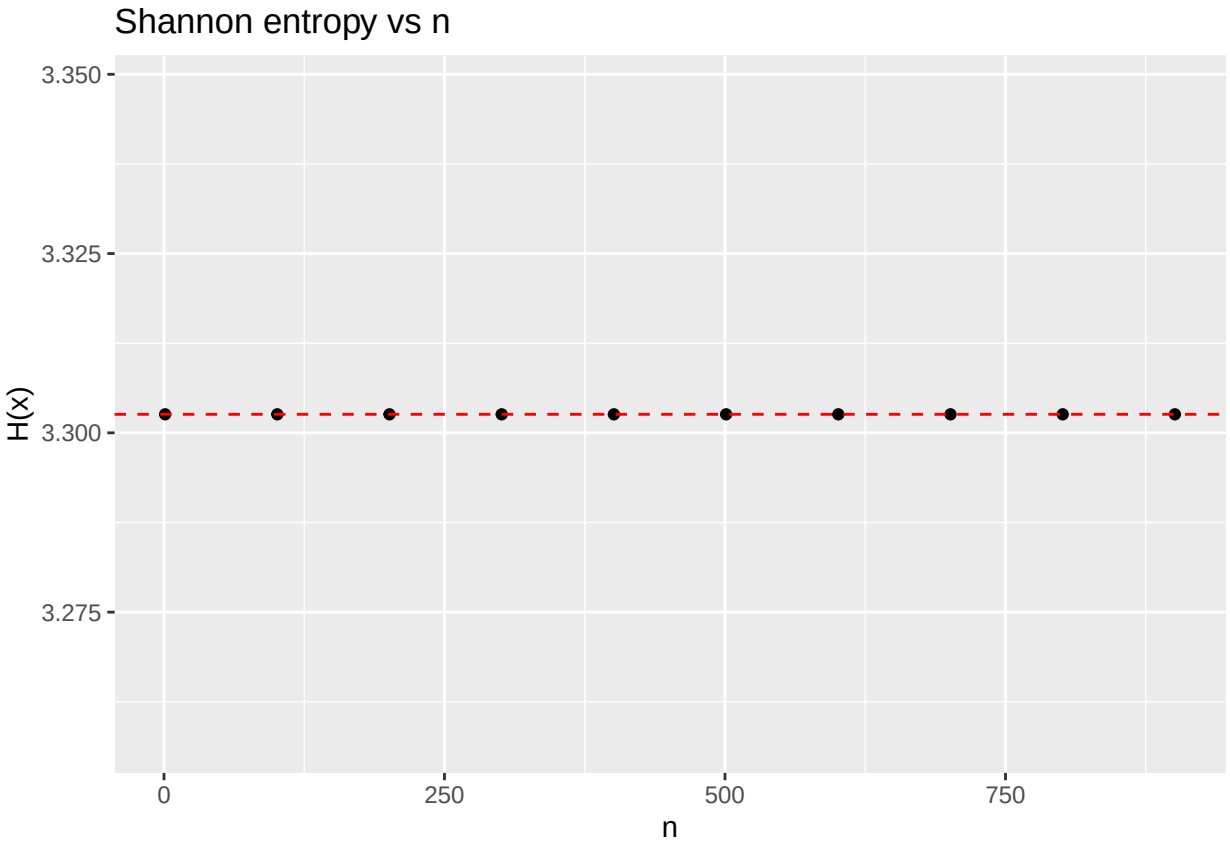
When $\alpha = 1, \beta = 0.5$:

```
ggplot(my_res_df,aes(x = my_n, y = V4) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/1)+log(0.5/1)+1, color =
               "red", linetype = "dashed")
```



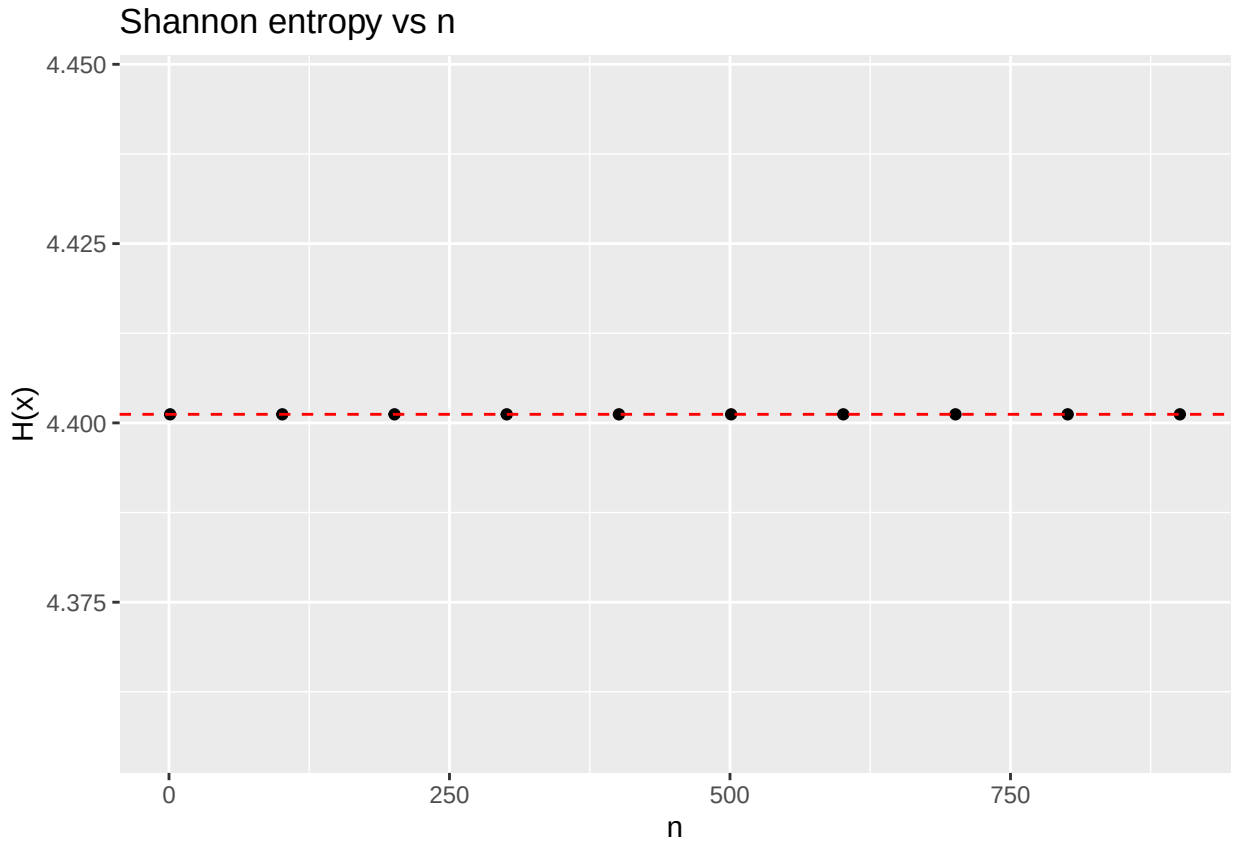
When $\alpha = 1, \beta = 10$:

```
ggplot(my_res_df,aes(x = my_n, y = V5) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/1)+log(10/1)+1, color =
               "red", linetype = "dashed")
```



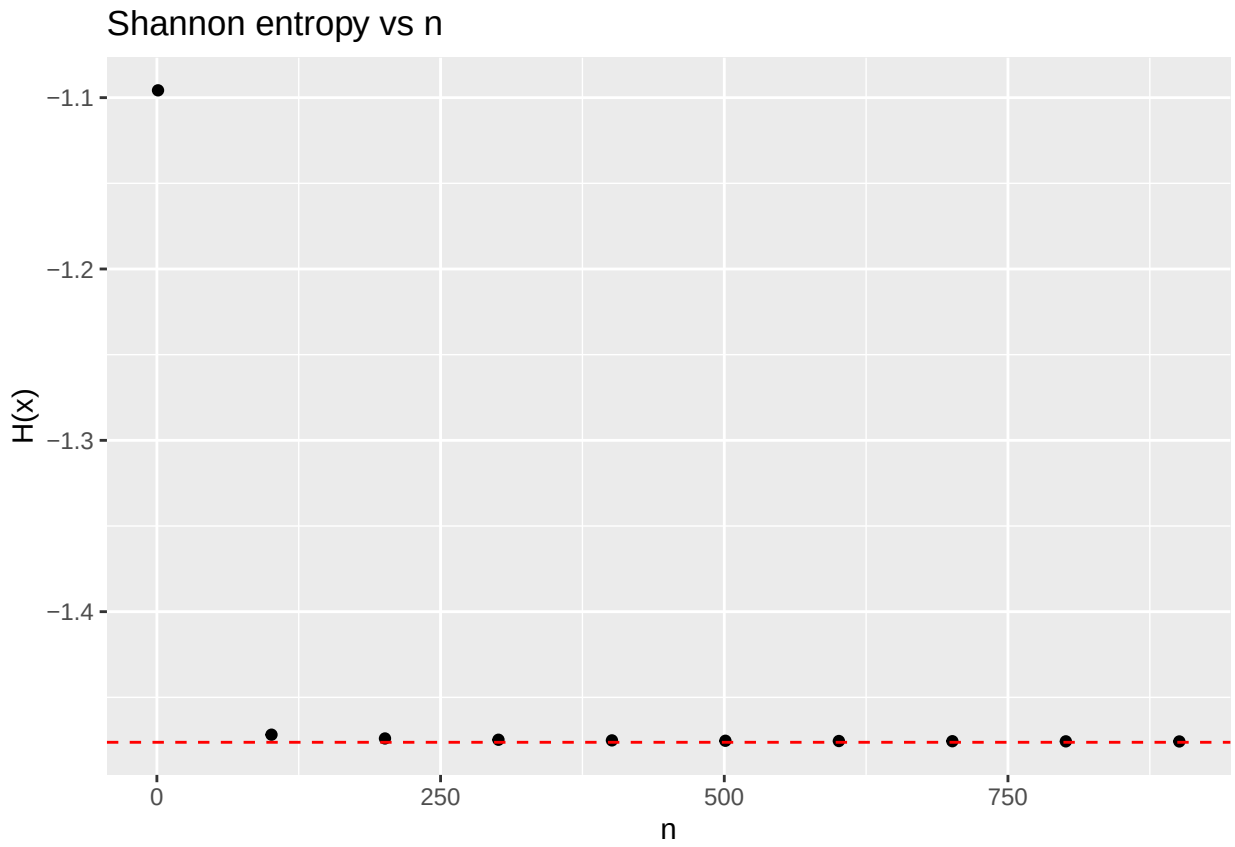
When $\alpha = 1, \beta = 30$:

```
ggplot(my_res_df,aes(x = my_n, y = V6) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/1)+log(30/1)+1, color =
               "red", linetype = "dashed")
```



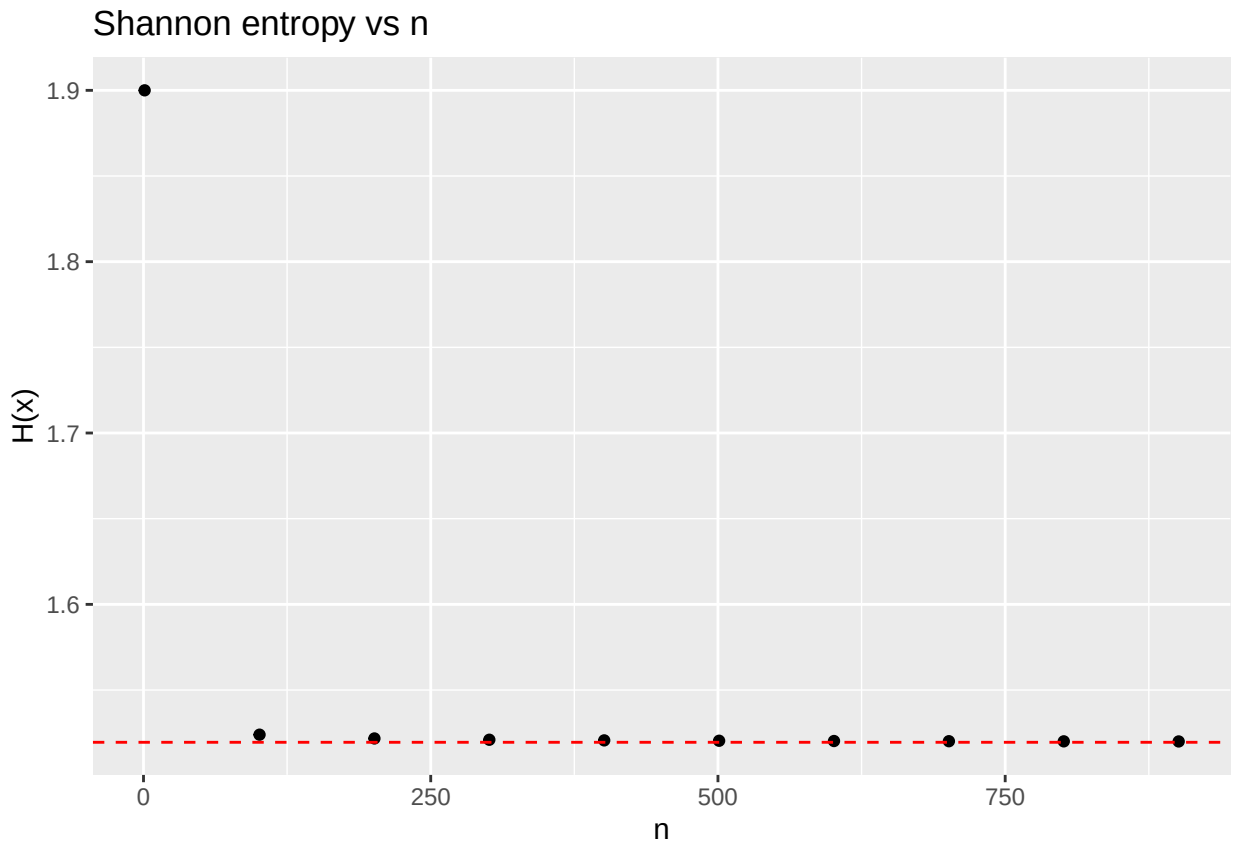
When $\alpha = 10, \beta = 0.5$:

```
ggplot(my_res_df,aes(x = my_n, y = V7) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/10)+log(0.5/10)+1, color =
               "red", linetype = "dashed")
```



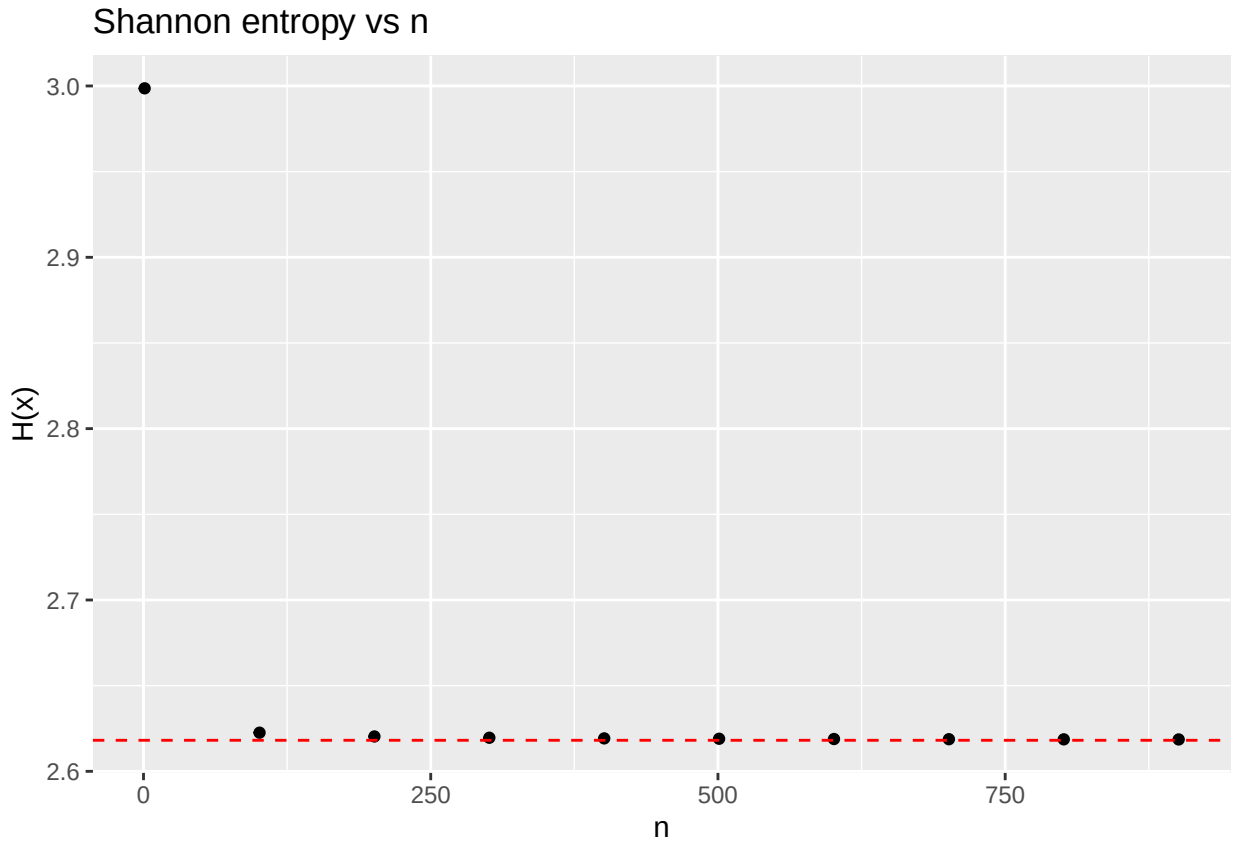
When $\alpha = 10, \beta = 10$:

```
ggplot(my_res_df,aes(x = my_n, y = V8) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/10)+log(10/10)+1, color =
             "red", linetype = "dashed")
```



When $\alpha = 10, \beta = 30$:

```
ggplot(my_res_df,aes(x = my_n, y = V9) ) + geom_point() +
  labs(x = "n", y = "H(x)", title = "Shannon entropy vs n") +
  geom_hline(yintercept = true_euler*(1-1/10)+log(30/10)+1, color =
               "red", linetype = "dashed")
```



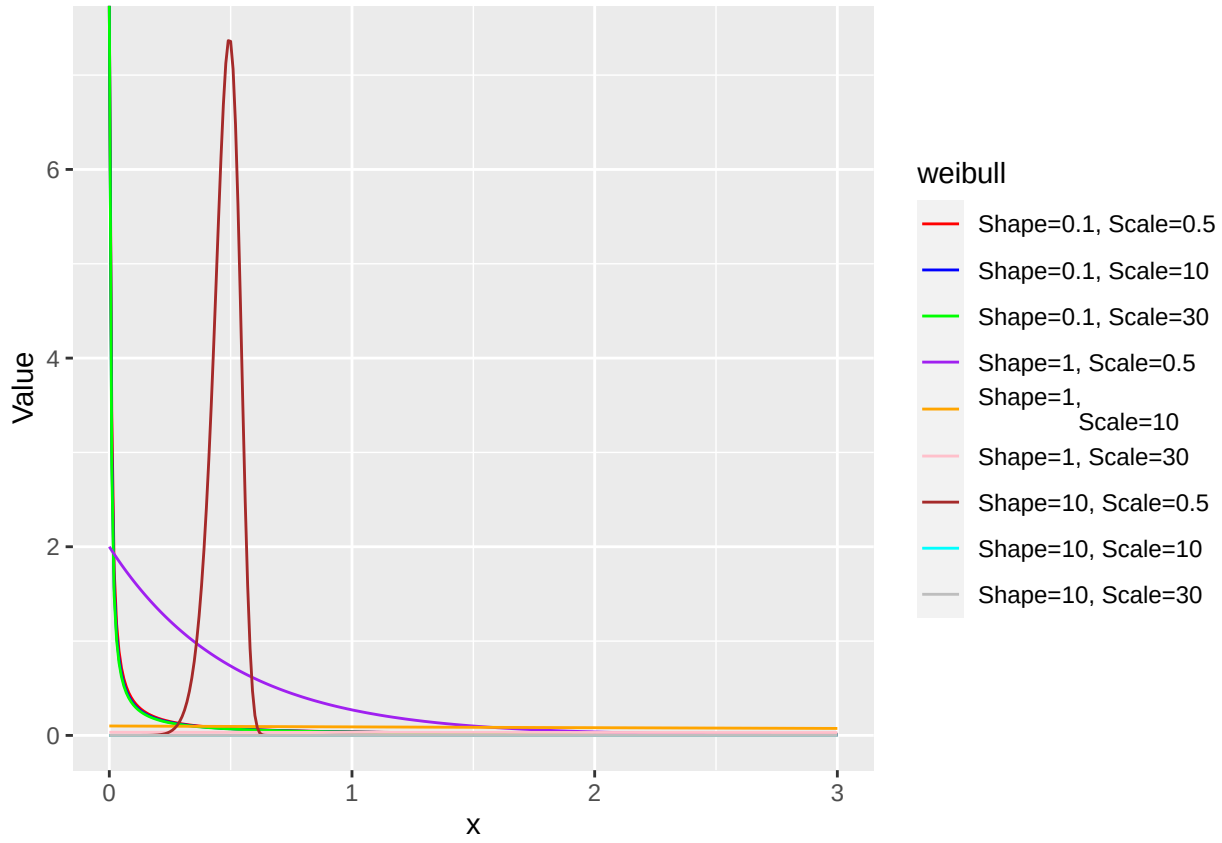
The performance of approximation is depend on the parameter of γ. When the coefficient of γ is exist(no matter large or small), need around 400 iterations to approximate. Although when the coefficient of γ is small, the converge speed will be faster. When the coefficient of γ is 0, it will not influence the value of the H(x).

(c) Here I think the combination of $\alpha = 1, \beta = 30$ gives the largest Shannon entropy and when $\alpha = 0.1, \beta = 0.5$ gives the smallest Shannon entropy.

```
library(ggplot2)
x <- seq(0, 3, by = 0.01)

dweibull_1 <- dweibull(x,shape = 0.1,0.5)
dweibull_2 <- dweibull(x,shape = 0.1,10)
dweibull_3 <- dweibull(x,shape = 0.1,30)
dweibull_4 <- dweibull(x,shape = 1,0.5)
dweibull_5 <- dweibull(x,shape = 1,10)
dweibull_6 <- dweibull(x,shape = 1,30)
dweibull_7 <- dweibull(x,shape = 10,0.5)
dweibull_8 <- dweibull(x,shape = 10,10)
dweibull_9 <- dweibull(x,shape = 10,30)

dweibull_df <- data.frame(x)

dweibull_df$dw1 <- dweibull_1
dweibull_df$dw2 <- dweibull_2
dweibull_df$dw3 <- dweibull_3
```

```
dweibull_df$dw4 <- dweibull_4
dweibull_df$dw5 <- dweibull_5
dweibull_df$dw6 <- dweibull_6
dweibull_df$dw7 <- dweibull_7
dweibull_df$dw8 <- dweibull_8
dweibull_df$dw9 <- dweibull_9
```

```
library(tidyr)

long_data <- gather(dweibull_df, key = "weibull", value = "Value", -x )



ggplot(long_data, aes(x = x, y = Value, color = weibull)) +
  geom_line()+scale_color_manual(values = c("red", "blue", "green",
                                            "purple", "orange",
                                            "pink", "brown", "cyan",
                                            "gray"),
    labels = c("Shape=0.1, Scale=0.5", "Shape=0.1, Scale=10",
               "Shape=0.1, Scale=30","Shape=1, Scale=0.5", "Shape=1,
               Scale=10", "Shape=1, Scale=30","Shape=10, Scale=0.5",
               "Shape=10, Scale=10", "Shape=10, Scale=30"))
```



2.

(a) Here $x_n \sim exp(n)$,

$$\lim_{n->\infty} p(|x_n - 0| \geq \epsilon) = \lim_{n->\infty} p(x_n \geq \epsilon) = \lim_{n->\infty} e^{-n\epsilon} = 0$$

So x_n converges in probability to 0.

(b)

```
my_x <- seq(from = 10, to = 1e5, length.out = 100)
```

```
my_r <- my_x
my_rexp <- list()


set.seed(1)
for (i in 1:100){
  a <- c(rexp(my_r[i], rate = my_r[i]))
  my_rexp[[i]] <-  a

}
```

```
my_e <-  c(1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7)
```

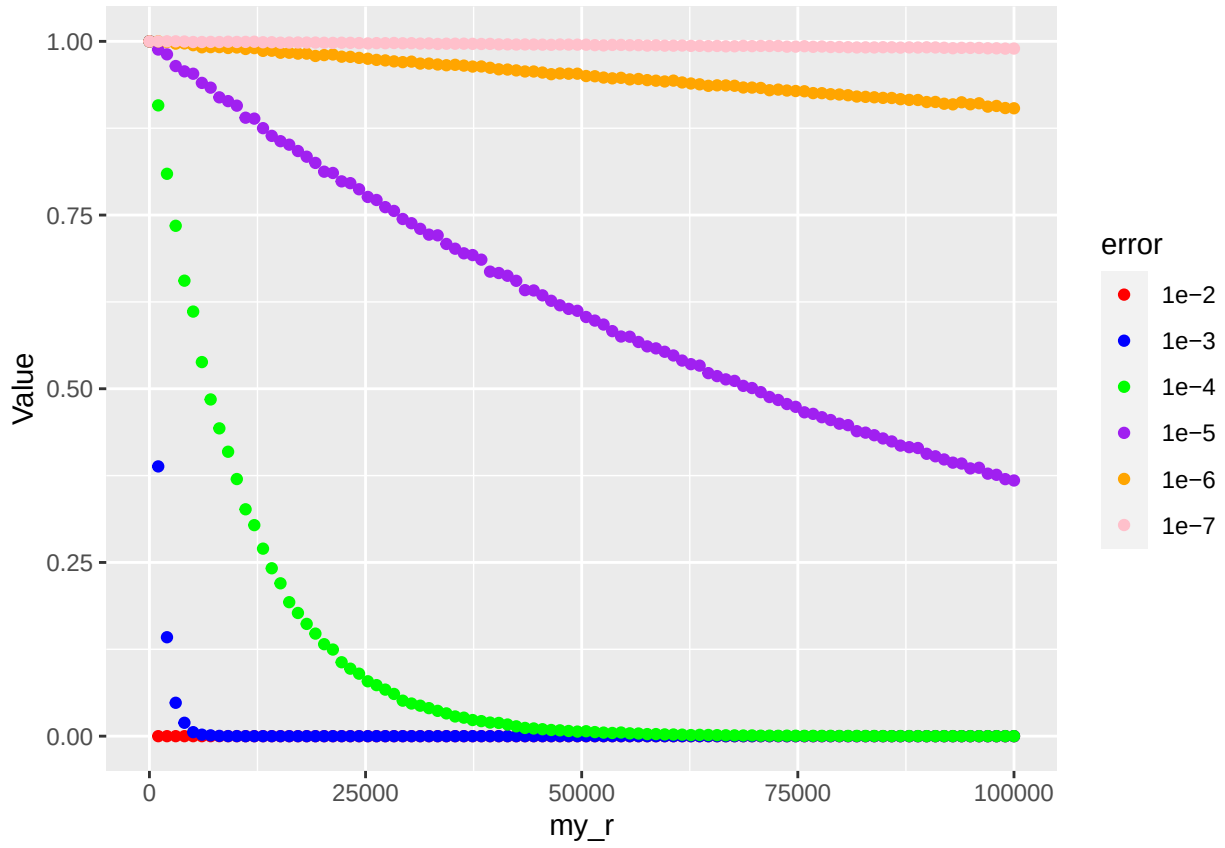
```
my_prop <- function(my_data = my_rexp,my_sample_size = my_r,e){
 my_propinf <- c()


 for (i in 1:100){
  my_propinf[i] <- sum(my_rexp[[i]]>e)

 }
 return(my_propinf/my_sample_size)

}
```

```
my_1e2 <- my_prop(e=1e-2)
my_1e3 <- my_prop(e=1e-3)
my_1e4 <- my_prop(e=1e-4)
my_1e5 <- my_prop(e=1e-5)
my_1e6 <- my_prop(e=1e-6)
my_1e7 <- my_prop(e=1e-7)
```

```
my_data <- data.frame(my_r, my_1e2, my_1e3,my_1e4,
                      my_1e5, my_1e6,my_1e7)
```

```
my_long_data <- gather(my_data, key = "error", value = "Value",-my_r)
```

```
ggplot(my_long_data, aes(x = my_r, y = Value, color = error)) +
  geom_point()+scale_color_manual(values = c("red", "blue", "green",
                                             "purple", "orange",
                                             "pink", "brown", "cyan",
                                             "gray"),
    labels = c("1e-2", "1e-3", "1e-4","1e-5", "1e-6", "1e-7"))
```



3.

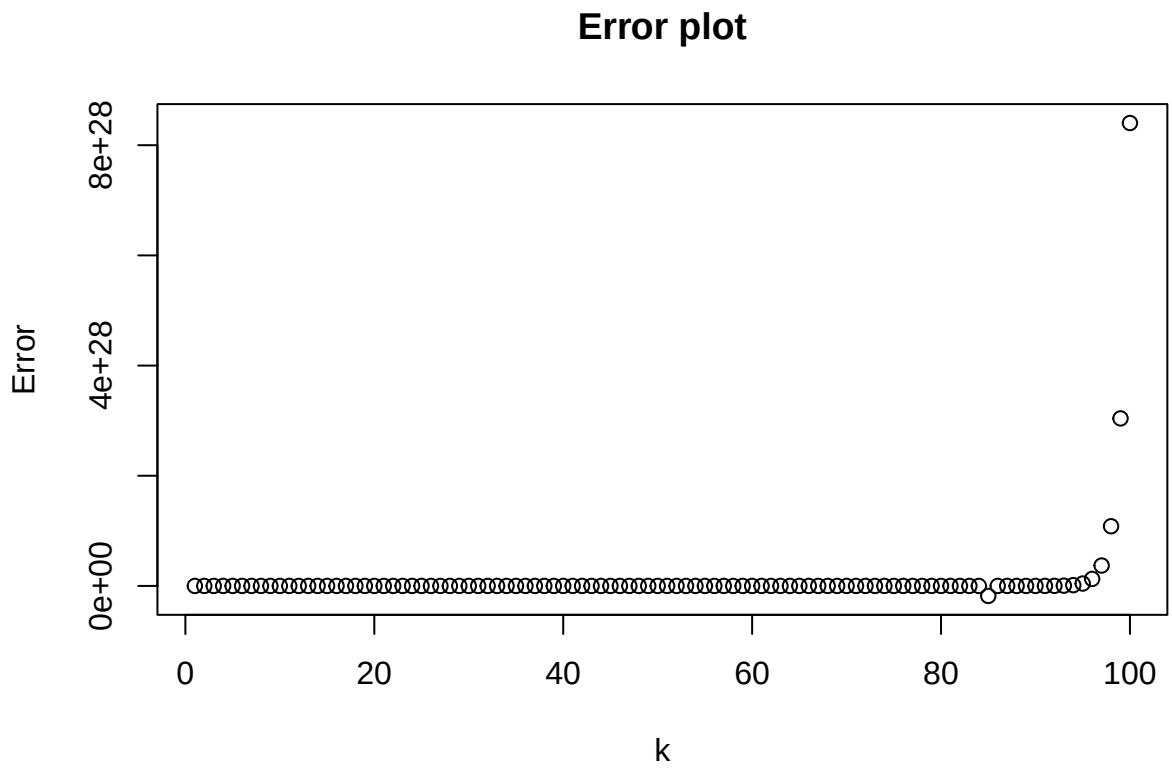
(a)

```
my_inte_3 <- c()
my_integrand3 <- function(k,t) {return(exp(k*cos(t)))}
result <- NA
for (i in 1:100){
  result <- integrate(function(t) my_integrand3(k=i,t) , lower = -pi, upper = pi)
  my_inte_3[i] <- result$value
}

 my_inte_3 <- my_inte_3/(2*pi)
```

```
my_bes <- NA
for (i in 1:100){

 my_bes[i] <- besselI(x = i, nu=0, expon.scaled = FALSE)

}
```

```
my_plot_3 <-  my_inte_3 - my_bes

plot(c(1:100),my_plot_3,main = "Error plot",
```

```
     xlab = "k",
     ylab = "Error")
```



I notice that when the k is not that large, the calculation is accurate. When the k is getting large, the approcation is not that good.

When k $= 1 * 10^4$

```
k = 1*10^4
#integrate(function(t) my_integrand3(k=1*10^4,t) , lower = -pi, upper = pi)
```

It will gives the error: Error in integrate(function(t) my_integrand3(k = 1 * 10^4, t), lower = -pi, : non-finite function value

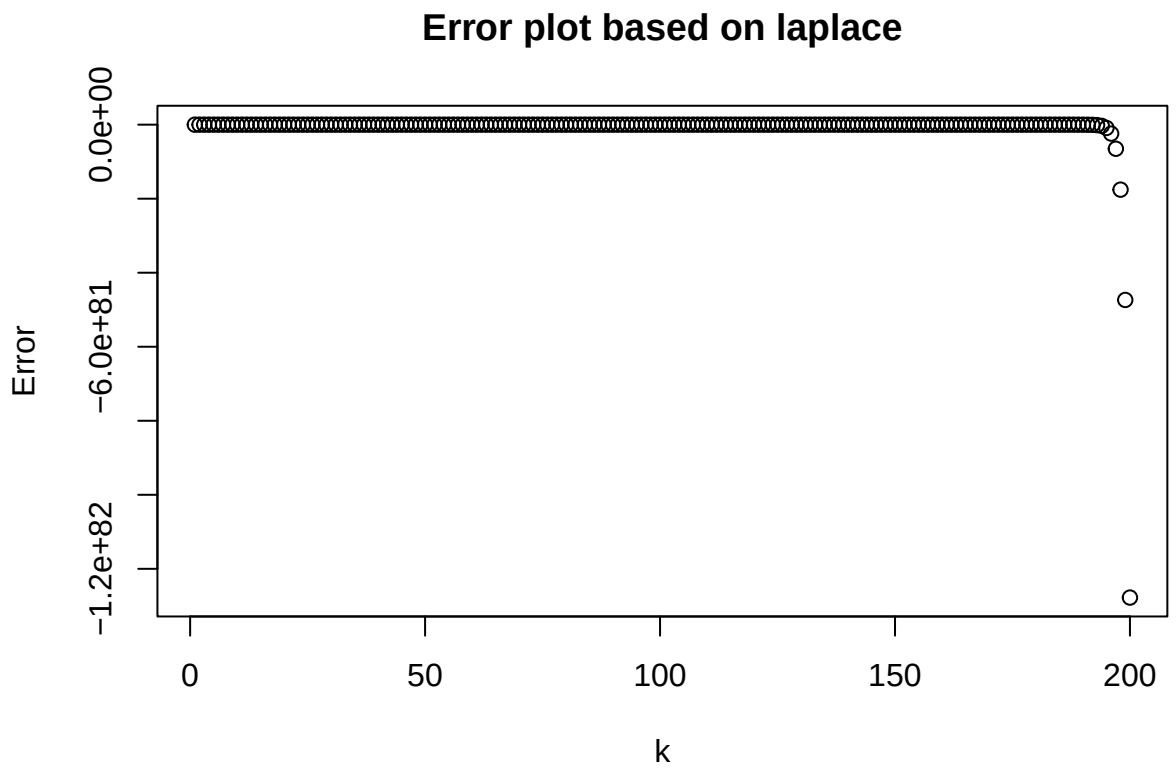
(b)

```
my_lap <- NA

for (i in 1:200){
  my_lap[i] <- exp(i)/sqrt(2*pi*i)
}
```

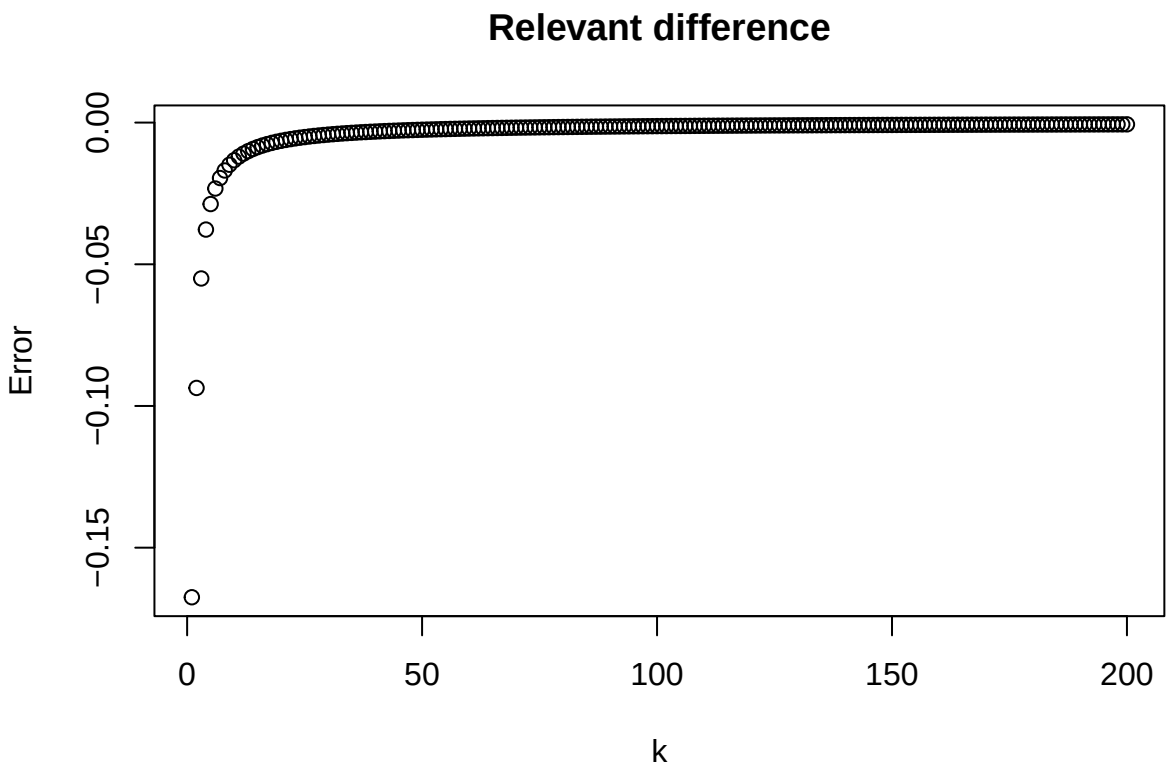
```
my_bes_b <- NA
for (i in 1:200){
```

```
 my_bes_b[i] <- besselI(x = i, nu=0, expon.scaled = FALSE)

}
```

```
my_plot_b <-  my_lap - my_bes_b

plot(c(1:200),my_plot_b,main = "Error plot based on laplace",
     xlab = "k",
     ylab = "Error")
```




```
my_plot_b_r <-  ( my_lap - my_bes_b)/my_lap

plot(c(1:200),my_plot_b_r,main = "Relevant difference",
     xlab = "k",
     ylab = "Error")
```



Here I noticed that when talking about the absolute difference, the k is less than 190, the performance is pretty good. But when the k is close to 200, it shows some diverge.

But when talking Relevant difference, it not show too much difference around 200. However, it shows some difference when k is small.

4.

(a)

```
library(SDaA)
```

```
## Warning: package 'SDaA' was built under R version 4.3.1
```

```
##
## Attaching package: 'SDaA'
```

```
## The following object is masked from 'package:ggplot2':
##
##     seals
```

```
library(ggplot2)
n <- nrow(agstrat)
t.X <- sum(agpop[,4]) # Population total - 1987
N <- nrow(agpop)
```

```r
m.X <- t.X/N
t.x.hat <- sum(agstrat[, 4]) # Sample total - 1987
t.y.hat <- sum(agstrat[, 3]) # Sample total - 1992

m.x <- t.x.hat/n
m.y <- t.y.hat/n

s.x <- sd(agstrat[, 4])
s.y <- sd(agstrat[, 3])

D <- cov(agstrat[, 4],agstrat[, 3])


R.hat <- m.y/m.x
s.xy <- cov(agstrat[,3],agstrat[,4])

dicr_Var.R.hat <- 1/(n*m.X^2) * (s.y^2 + R.hat^2*s.x^2 - 2*R.hat*s.xy )
dicr_Var.R.hat
```

```
## [1] 3.965322e-05
```

```r
my_Var.y.R <- m.X^2*dicr_Var.R.hat
my_Var.y.R
```

```
## [1] 3885193
```

The difference is calculated by:

```r
r <- cor(agstrat[,3],agstrat[,4])
# Since the true X is known, we use m.X in our variance calculation
Var.R.hat <- (1/n)*(1-(n-1)/(N-1))*(R.hat^2*s.x^2+s.y^2-2*R.hat*r*s.x*s.y)/m.X^2
Var.y.R <- m.X^2*Var.R.hat
my_Var.y.R - Var.y.R
```

```
## [1] 377534.2
```

(b)

```r
R.hat +c(-1,1)*qnorm(.975)*sqrt(Var.R.hat)
```

```
## [1] 0.9782885 1.0017427
```

```r
R.hat +c(-1,1)*qnorm(.975)*sqrt(dicr_Var.R.hat)
```

```
## [1] 0.9776735 1.0023576
```

There will be 95% probability the 1992 census will be the confidence interval times the 1987 census.

(c) When I face with doing the analysis to a client. I will told them these are different way to approach the true value. When you tolerance the large variance, the estimator will contain some bias, if you hope get a non-bias eastimator, variance will be larger. So the different result based on what's you need about the estimator.

5.

(a)

```
url <- "https://raw.githubusercontent.com/dsy109/Supplemental/main/Courses/705/wine.data"
my_wine <- read.csv(url, header = FALSE)
```

```
my_wine_1 <- my_wine[my_wine$V1==1, ,]
my_wine_1 <- my_wine_1[ ,-1 ]
```

```
my_wine_2 <- my_wine[my_wine$V1==2, ,]
my_wine_2 <- my_wine_2[ ,-1 ]
```

```
my_wine_3 <- my_wine[my_wine$V1==3, ,]
my_wine_3 <- my_wine_3[ ,-1 ]
```

```
S1 <- cov(my_wine_1)
S2 <- cov(my_wine_2)
S3 <- cov(my_wine_3)
```

```
mynorm_matrix <- matrix(NA,nrow = 5, ncol = 3)
mynorm_matrix[1,1] <- norm(S1 - S2, type = "1")
mynorm_matrix[2,1] <- norm(S1 - S2, type = "2")
mynorm_matrix[3,1] <- norm(S1 - S2, type = "I")
mynorm_matrix[4,1] <- norm(S1 - S2, type = "M")
mynorm_matrix[5,1] <- norm(S1 - S2, type = "F")


mynorm_matrix[1,2] <- norm(S1 - S3, type = "1")
mynorm_matrix[2,2] <- norm(S1 - S3, type = "2")
mynorm_matrix[3,2] <- norm(S1 - S3, type = "I")
mynorm_matrix[4,2] <- norm(S1 - S3, type = "M")
mynorm_matrix[5,2] <- norm(S1 - S3, type = "F")

mynorm_matrix[1,3] <- norm(S2 - S3, type = "1")
mynorm_matrix[2,3] <- norm(S2 - S3, type = "2")
mynorm_matrix[3,3] <- norm(S2 - S3, type = "I")
mynorm_matrix[4,3] <- norm(S2 - S3, type = "M")
mynorm_matrix[5,3] <- norm(S2 - S3, type = "F")
```

```
mynorm_df <- as.data.frame(mynorm_matrix)
colnames(mynorm_df) <- c("S1_S2","S1_S3","S2_S3")
```

```
mynorm_df
```

```
##      S1_S2    S1_S3    S2_S3
## 1 26377.98 36792.14 12663.91
## 2 24469.12 35834.33 11570.07
## 3 26377.98 36792.14 12663.91
## 4 24356.08 35824.12 11468.04
## 5 24470.76 35834.33 11570.24
```

I think the cultivar 2 and cultivar 3 seems most similar , cultivar 1 and cultivar 3 are least similar.

(b)

```
library(ICSNP)
```

```
## Warning: package 'ICSNP' was built under R version 4.3.1
```

```
## Loading required package: mvtnorm
```

```
## Warning: package 'mvtnorm' was built under R version 4.3.1
```

```
## Loading required package: ICS
```

```
## Warning: package 'ICS' was built under R version 4.3.1
```

```
T1 <- tyler.shape(my_wine_1)
T2 <- tyler.shape(my_wine_2)
T3 <- tyler.shape(my_wine_3)
```

```
myt_matrix <- matrix(NA,nrow = 5, ncol = 3)
myt_matrix[1,1] <- norm(T1 - T2, type = "1")
myt_matrix[2,1] <- norm(T1 - T2, type = "2")
myt_matrix[3,1] <- norm(T1 - T2, type = "I")
myt_matrix[4,1] <- norm(T1 - T2, type = "M")
myt_matrix[5,1] <- norm(T1 - T2, type = "F")


myt_matrix[1,2] <- norm(T1 - T3, type = "1")
myt_matrix[2,2] <- norm(T1 - T3, type = "2")
myt_matrix[3,2] <- norm(T1 - T3, type = "I")
myt_matrix[4,2] <- norm(T1 - T3, type = "M")
myt_matrix[5,2] <- norm(T1 - T3, type = "F")

myt_matrix[1,3] <- norm(T2 - T3, type = "1")
myt_matrix[2,3] <- norm(T2 - T3, type = "2")
myt_matrix[3,3] <- norm(T2 - T3, type = "I")
myt_matrix[4,3] <- norm(T2 - T3, type = "M")
myt_matrix[5,3] <- norm(T2 - T3, type = "F")
```

```
myt_df <- as.data.frame(myt_matrix)
colnames(myt_df) <- c("T1_T2","T1_T3","T2_T3")
```

```
myt_df
```

```
##      T1_T2    T1_T3    T2_T3
## 1 103086.3 102989.8 673.5626
## 2 100446.5 100618.7 377.2677
## 3 103086.3 102989.8 673.5626
## 4 100416.8 100597.6 271.8539
## 5 100446.6 100618.7 444.0864
```

The cultivar 2 and cultivar 3 are the most similar. The is not much difference between cultivar 1 and cultivar 2 compare to cultivar 1 and cultivar 3.