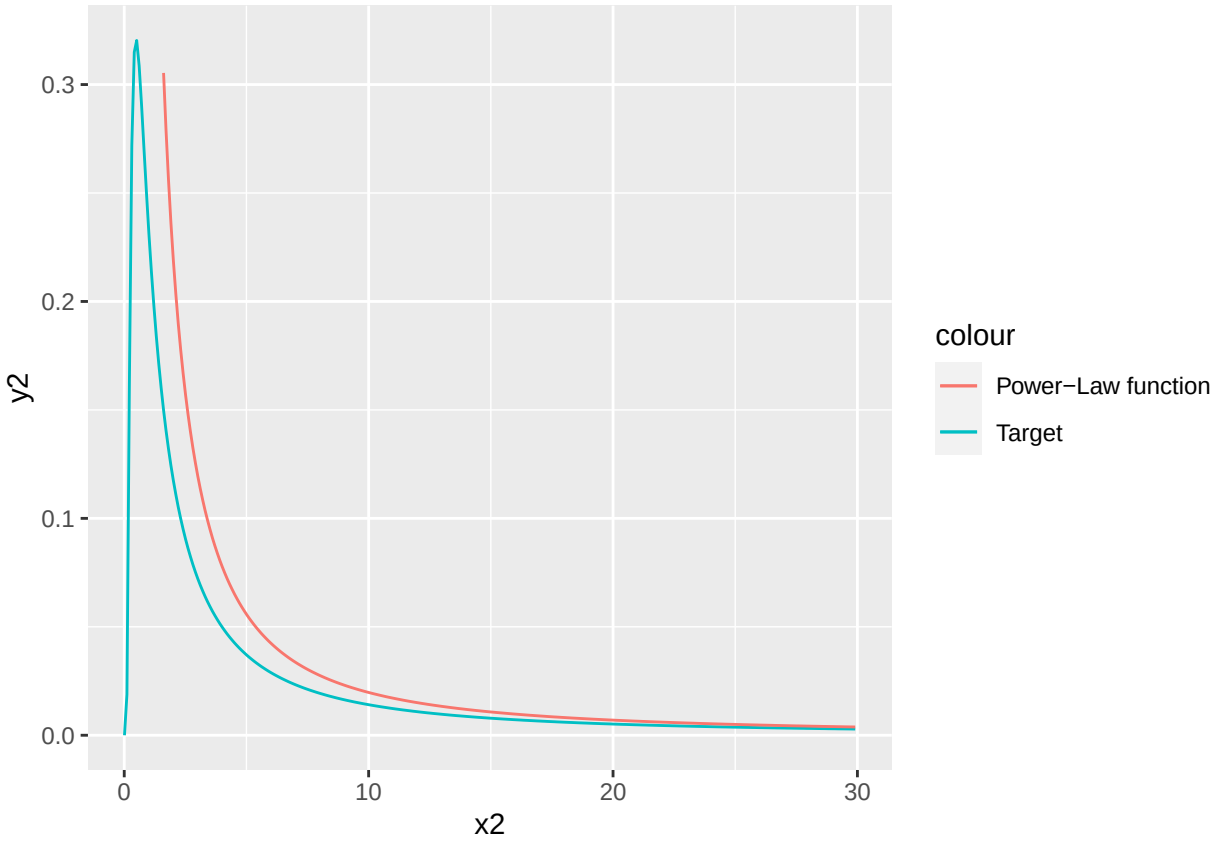
Computational Statistic Lab 3 Rev.2

YiHung Chen, Jonathan Dorairaj

2022-2-9

Question 1: Stable Distribution



By observing the graph, the power law distribution does envelope the target function, however it only has a support of (T_{min},∞). Since the target function operates in the interval (0,∞), we consider sampling from an uniform distribution in the region (0,T_{min}). This way, by having a mixture of the power law and uniform distribution as the majorizing distribution, we ensure that the proposal have the same support as the target distribution.

In this assignment, we use $\alpha = 1.5$ as values of the power-law distribution to be used in the Acceptance Rejection Algorithm(A/R algorithm). We also initialized c=1.2(constant c in target function) for demonstration, this will be changed in the last part as we experiment with different values of c.

Choosing alpha

$\alpha = 1.5$ is chosen because we should pick α less than (equal to) 1.5 by taking the limit of $\frac{M*f_p}{f}$ (M as majorizing constant) with following steps.
Since in Acceptance Rejection Algorithm, the proposal function should always be larger than (equal to) the target function.

$$\lim_{x\to\infty} \frac{M * f_p(x)}{f(x)} \geq 1$$

$$\lim_{x\to\infty} \frac{\frac{M*\alpha-1}{T_{min}} * (\frac{x}{T_{min}})^{-\alpha}}{c(\sqrt{2\pi})^{-1} * e^{-c/2x} * x^{-3/2}} \geq 1$$

$$lim_{x\to\infty} \frac{\frac{M*\alpha-1}{T_{min}^{1-\alpha}} * x^{-\alpha}}{c(\sqrt{2\pi})^{-1} * e^{-c/2x} * x^{-3/2}} \geq 1$$

Since $T_{min} > 0, \alpha > 1, c > 0, M > 1$, the constant part will not affect the limit because the combination is positive

$$lim_{x\to\infty} \frac{x^{-\alpha}}{e^{-c/2x} * x^{-3/2}} \geq 1$$

as $lim_{x\to\infty} e^{-c/2x} = 1$, which leaves

$$lim_{x\to\infty} \frac{x^{-\alpha}}{x^{-3/2}} \geq 1$$

Hence $\alpha \leq 1.5$, and the value 1.5 is arbitrarily chosen to fit this criteria.

Calculating

$$T_{min}$$

To find T_{min} (in this case, from which 'x' value the A/R algorithm will start to sample from the power law distribution.)
First, we find the maximum of the target function. This is done by differentiating the target function with respect to `x`.

$$\frac{df(x)}{dx} = \frac{\left(c^3 - 3cx\right) \mathrm{e}^{-\frac{c^2}{2x}}}{2^{\frac{3}{2}} \sqrt{\pi}\, x^{\frac{7}{2}}} = 0$$

For c>0, the root of this expression is found at :

$$x = \frac{c^2}{3}$$

Which means, at $x = \frac{c^2}{3}$, the target function has the maximum value.

By calculating at which x value(x^*), the power law function has the y value($f_p(x^*)$) equal to the maximum of target function $f(\frac{c^2}{3})$, We then can use this x^* value as the starting point($x^* = T_{min}$) to sample from power law distribution.

$$f(\frac{c^2}{3}) = f_p(x^*) = f_p(T_{min}) = \frac{\alpha-1}{T_{min}}(\frac{x^*}{T_{min}})^{-\alpha} = \frac{\alpha-1}{T_{min}}(\frac{T_{min}}{T_{min}})^{-\alpha} = \frac{\alpha-1}{T_{min}}$$

$$T_{min} = \frac{\alpha - 1}{f(\frac{c^2}{3})}$$

The reason of calculating the maximum value of target distribution is to use it as the upper bound(the horizontal line) of uniform part in majorizing distribution (which can be seen after multiplying the majorizing constant in later steps). Then proceed to find where the horizontal line connected with power-law distribution, and then take the connecting point as Tmin. In this assignment, as we set c=1.2. for demonstration, we obtain the $T_{min} = 1.5566188$
Note: we do include the calculation of Tmin for different c in our code.

Deriving a Majorizing Density

We calculate the mixture weight first:
As previously stated, we propose a mixture of the Uniform Distribution ~ Unif(0,T_{min}) and the powerlaw distribution on support (T_{min}, ∞). By taking the area under curve of the target density function, we can obtain the proportions or weights for combining the two distributions.

For c = 1.2 and $T_{min} = 1.56$

$$weight_1 = \int_0^{T_{min}} f(x)dx = \int_0^{T_{min}} c(\sqrt{2\pi})^{-1} e^{\frac{-c^2}{2x}} x^{-\frac{3}{2}} dx$$
$$weight_1 = 0.3361$$

$$weight_2 = \int_{T_{min}}^{\infty} f(x)dx = \int_{T_{min}}^{\infty} c(\sqrt{2\pi})^{-1} e^{\frac{-c^2}{2x}} x^{-\frac{3}{2}} dx$$
$$weight_2 = 0.6639$$

Where $weight_1$ & $weight_2$ are our mixture weights.

The majorizing density can be written as follow

$$g(x) = weight_1 \frac{1}{T_{min}} 1_{(0,T_{min})}(x) + weight_2 \frac{\alpha - 1}{T_{min}} (\frac{x}{T_{min}})^{-\alpha} 1_{(T_{min},\infty)}(x)$$

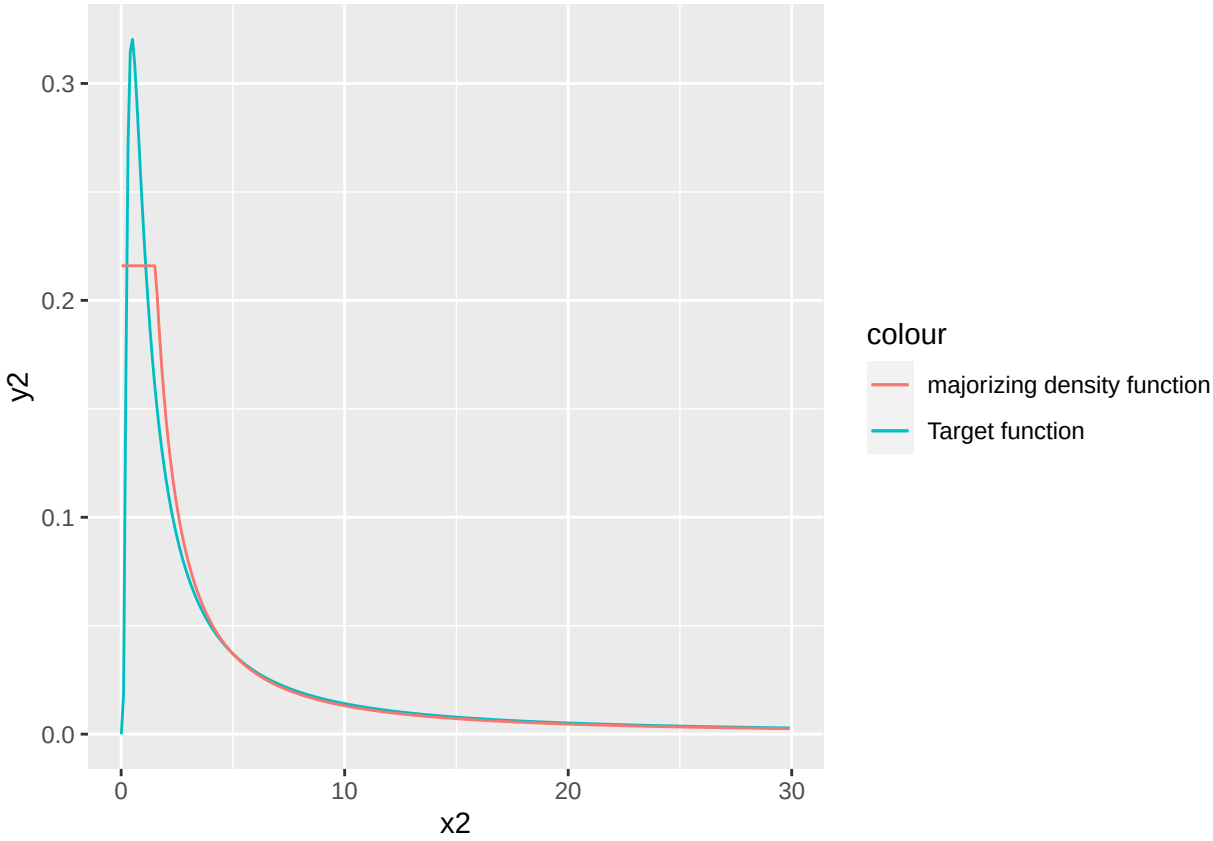
Where $\frac{1}{T_{min}}$ is the PDF of Uniform distribution and $\frac{\alpha-1}{T_{min}}(\frac{x}{T_{min}})^{-\alpha}$ is the PDF of power law distribution.

As a density function, the integration of g(x) should be equal to 1

$$\int_0^{\infty} g(x)dx = weight_1 \int_0^{T_{min}} \frac{1}{T_{min}} dx + weight_2 \int_{T_{min}}^{\infty} \frac{\alpha - 1}{T_{min}} (\frac{x}{T_{min}})^{-\alpha} dx$$

$$\int_0^{\infty} g(x)dx = 0.3361 \int_0^{1.56} \frac{1}{1.56} dx + 0.6639 \int_{1.56}^{\infty} \frac{1.5 - 1}{1.56} (\frac{x}{1.56})^{-1.5} dx = 1$$

Hence, g(x) is the majorizing density function.
Note: we do include the calculation of mixture weights for different c in our code, but for all the c we tested, the weight stays the same.



In the graph above, the target function and the majorizing density function are plotted. Since the majorizing density function does not envelope the target function. In order to use Accept/Rejection method,we introduce M(majorizing constant) is needed to ensure $M * g(x) \geq f(x)$ for all values of `x`.

To prove that we need a majorizing constant, we run the following code to check if there are any values where the majorizing density is smaller than the target function.

```
any((majorizing_vec/target_val_vec < 1))
```

```
## [1] TRUE
```

This returns `TRUE` showing that for some values of x, the majorizing function is not greater than the target function.Hence, we require a majorizing constant M to ensure that $M * g(x) \geq f(x)$ for all values of `x`.

Calculating M:

As the maximum of target function is calculate as $f(\frac{c^2}{3}) = f(\frac{1.2^2}{3}) = 0.321209$

$$M * g(x) \geq max(f(x)) = 0.3212$$

where max(f(x)) is within $0 \leq x < T_{min}$. Therefore,

$$M * 0.3361 \frac{1}{T_{min}} \geq 0.3212$$

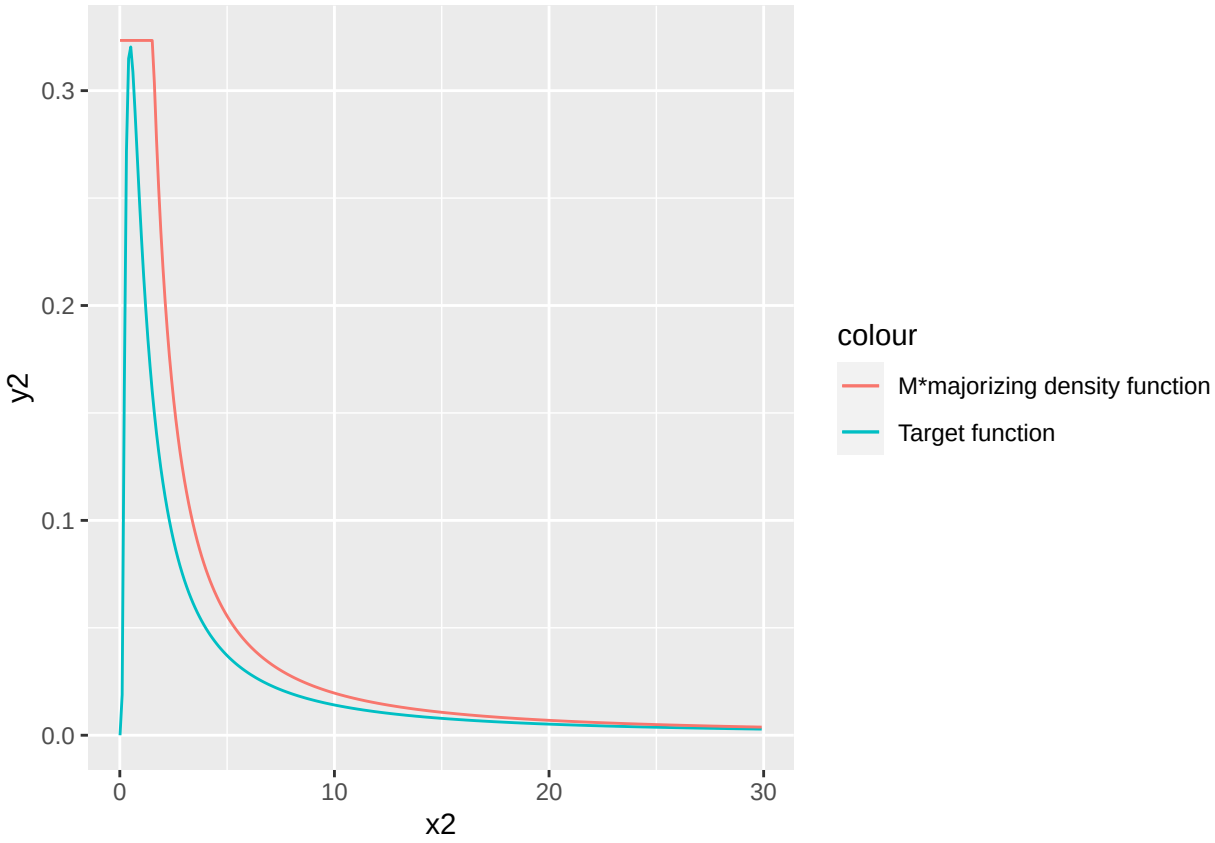
$$M * 0.3361 \frac{1}{1.56} \geq 0.3212$$
$$M \geq 1.49$$

Note: we do include the calculation of the minimum M for different c in our code. Also, in order to avoid rounding error, we add 0.01 to our calculation of minimum M.

```
any(((1.49*majorizing_vec)/target_val_vec < 1))
```

```
## [1] FALSE
```

Running the check again, this time multiplying the Majorizing function values by M, we see that no values are smaller than the target function.



As the M*majorizing density function envelope the whole target function, we can bow use it to do A/R algorithm.

1.2

Acceptance Rejection Algorithm

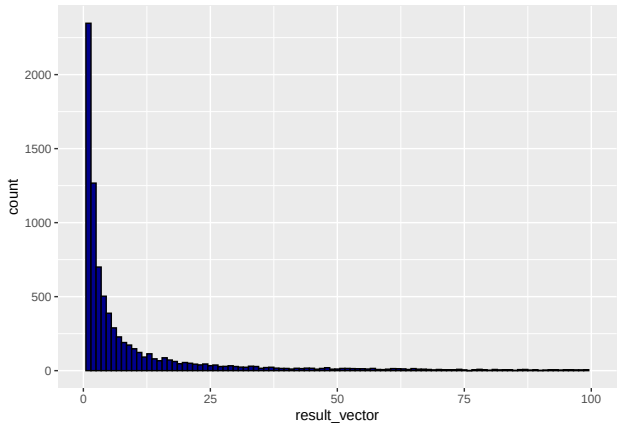
```
AccRejdf <- data.frame()
Accept_Rejection <- function(C){
  #changed tmin inside this function to tmin1
```

```
counter <- 0
reject_counter <- 0
alpha <- 1.5
constant <- C
res <-NA

#initial parameters, here the parameters below will depend on different
#c (constant in target function)
constant <- C
max_target <-  constant * (sqrt(2 * pi)^(-1)) * exp(-constant^(2) / (2 * (
    constant^2/3))) * (constant^2/3)^(-3/2)
tmin1 <- get_tmin(constant)


weight_uni_for_c <- calculate_major_weight(constant,tmin1)[1]
weight_pl_for_c <- calculate_major_weight(constant,tmin1)[2]
major_c <- get_minM(tmin1,max_target,weight_uni_for_c)

#print(tmin1)
#print(max_target)
#print(major_c)


result_vector <-c()

while (length(result_vector)<10000) {
    counter <- counter+1
    u <- runif(1)
    sample_from=sample(1:2,1,prob=c(weight_pl_for_c,weight_uni_for_c))
    if(sample_from==1) {
      z <- rplcon(1,tmin1,alpha)#get the x value,
      #the naming z is to differentiate from x we use for plotting lines
    }
    if(sample_from==2){
      z <- runif(1,0,tmin1)#get the x value
    }
    if(u <= target(z,constant)/(major_c*majorizing_den(z,alpha,tmin1,weight_uni_for_c,
                                                       weight_pl_for_c)))
    {
     result_vector <- c(result_vector,z)
    }
    else{
        reject_counter <- reject_counter + 1
    }

}



df <- as.data.frame(result_vector)
assign(x = "df1",value = df,envir = .GlobalEnv )
title <- paste0("Plot of C =",C)
plot_output<-ggplot(df, aes(result_vector)) +
```

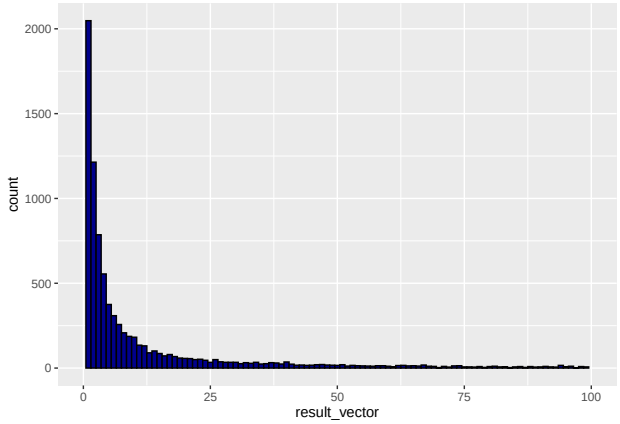
```
    geom_histogram(color="black", fill="darkblue",binwidth = 1)+
    xlim(0,100)
    theme_bw() + labs(x = "x",y = "Count")+
    ggtitle(title)+
    theme(plot.title = element_text(hjust = 2))

  m <- mean(result_vector)
  trunc_mean <- mean(result_vector[1:100])
  v <- var(result_vector)
  trunc_var <- var(result_vector[1:100])
  RR <- (reject_counter/counter)*100
  ER <- (1-(1/major_c))*100
  temp <- cbind(C,major_c,m,v,trunc_mean,trunc_var,RR,ER)
  AccRejdf <<- rbind(AccRejdf,temp)


  return(plot_output)

}
```

1.3

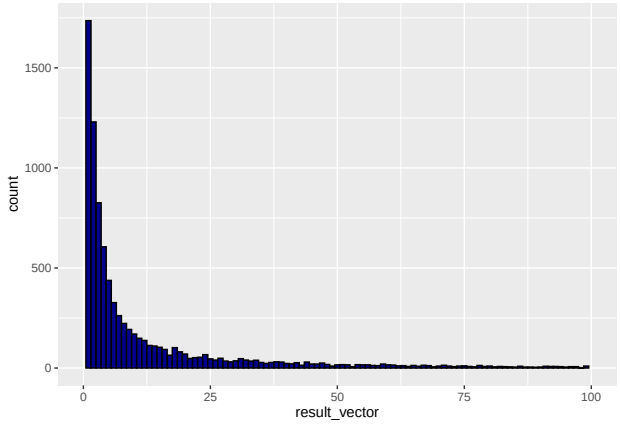
```
set.seed(12345)
Accept_Rejection(1.2)
```



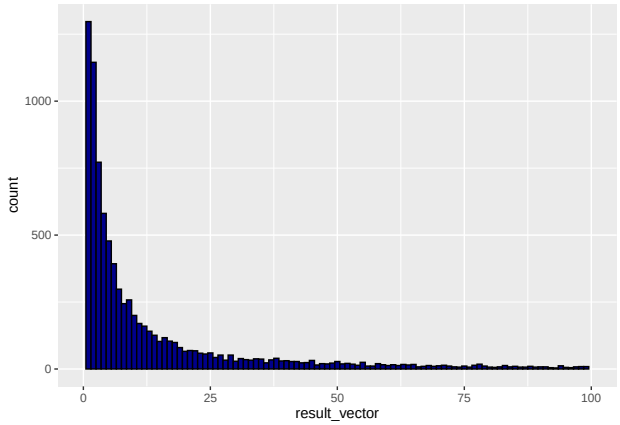

```
Accept_Rejection(1.4)
```



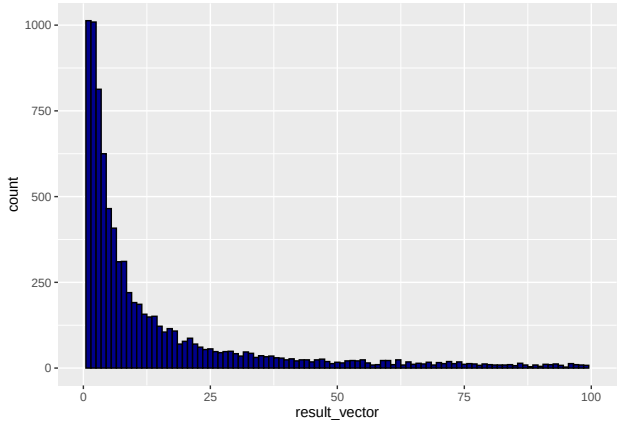

```
Accept_Rejection(1.6)
```



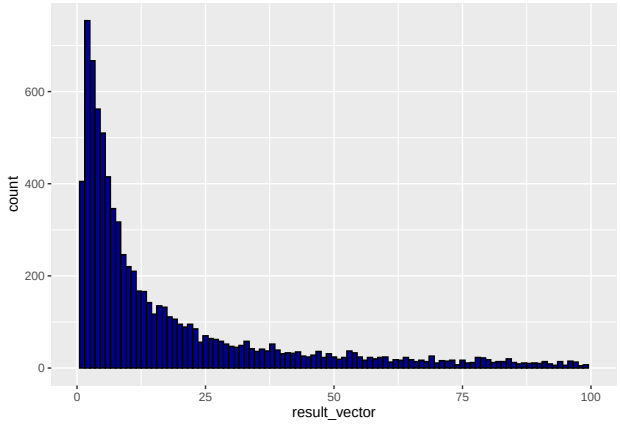

```
Accept_Rejection(1.8)
```




```
Accept_Rejection(2.0)
```




```
Accept_Rejection(2.5)
```




```
Accept_Rejection(3)
```

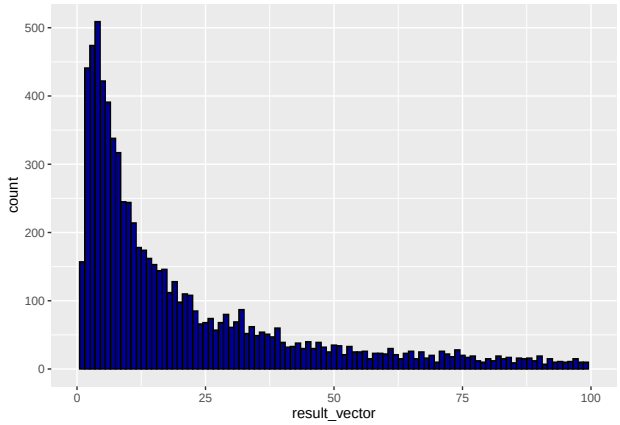


Table 1: Summary of Rejection Rate for Different C

C	Majorizing Constant	Rejection Rate	Expected Rejection Rate
1.2	1.497457	32.62817	33.22013
1.4	1.497457	33.44869	33.22013
1.6	1.497457	32.78666	33.22013
1.8	1.497457	33.02973	33.22013
2.0	1.497457	33.06112	33.22013
2.5	1.497457	33.59012	33.22013
3.0	1.497457	33.12826	33.22013

Above are the results from different `c` values (the constant of target function). It should be noted that changing the c will change the shape of target function. It can be observed that as `c` increase the majorizing constant stays the same. We suggest that the reason is because when changing the c, the Tmin will increase and the maximum of target function will decrease in the same scale. Also, with the calculation above, the mixture weight for different c will stays the same. Since the optimal majorizing constant(M) is based on Tmin, maximum of target function and mixture weight, M stays the same for different c.

Also, as the majorizing constant stays the same, the rejection rate also stay roughly the same, and the rejection rate we obtain are close to the expected rejection rate $(1 - \frac{1}{M})$.

Below are the two plots with different `c` value, These plots are use to demonstrate the shape of target and majorizing function has change. When c increase, the maximum of the target function decrease and the Tmin increases and per our implementation, the majorizing density function also matches the change.

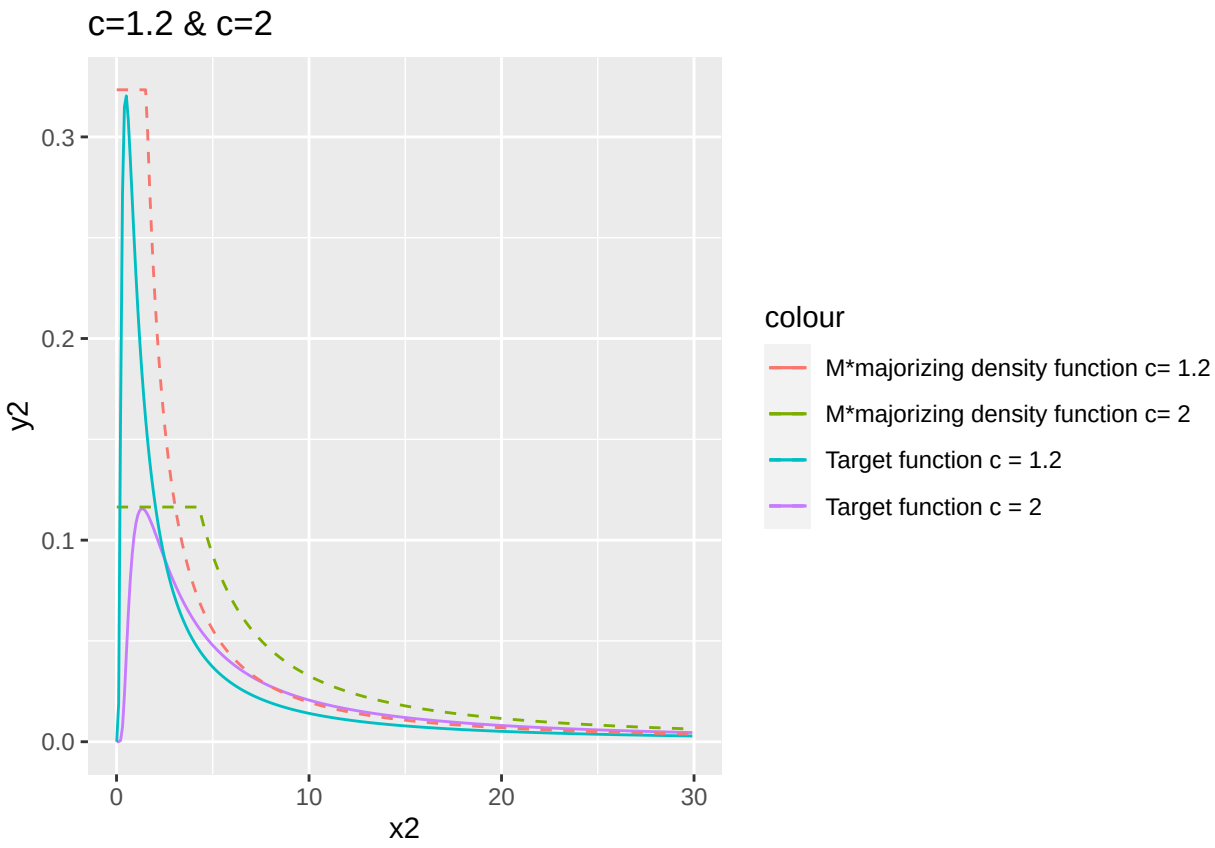


Table 2: Summary of statistics

C	Majorizing Constant	Mean	Variance
1.2	1.497457	68310.45	2.687160e+13
1.4	1.497457	28807.12	3.600276e+12
1.6	1.497457	32465.98	2.451091e+12
1.8	1.497457	480595.15	1.874112e+15
2.0	1.497457	7179390.16	4.937902e+17
2.5	1.497457	194488.20	2.788624e+14
3.0	1.497457	102631.79	7.161730e+13

We also observe that the value of mean and variances are high and does not have clear pattern due to the random samples from `rplcon()`. Using `rplcon()` will produce some high values, and these large values can be accepted by the algorithm. This distorts and highly affect the values of the mean and variance.

Question 2: Laplace distribution

Q1.

We obtain the inverse CDF formula by doing calculation as below The pdf of Laplace distribution is

$$DE(\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha|x - \mu|)$$

$$DE(\mu, \alpha) = \begin{cases} \frac{\alpha}{2} e^{-\alpha(\mu - x)} & if\ x < \mu \\ \frac{\alpha}{2} e^{-\alpha(x - \mu)} & if\ x \geq \mu \end{cases}$$

Calculate CDF
CDF is calculated by integration of pdf

$$F(x) = \int_{-\infty}^{x} f(u)\, du$$

$$= \begin{cases} \frac{1}{2} e^{-\alpha(\mu - x)} & if\ x < \mu \\ 1 - \frac{1}{2} e^{-\alpha(x - \mu)} & if\ x \geq \mu \end{cases}$$

$$\begin{cases} F(x) = \frac{1}{2} e^{-\alpha(\mu - x)} & if\ x < \mu \\ F(x) = 1 - \frac{1}{2} e^{-\alpha(x - \mu)} & if\ x \geq \mu \end{cases}$$

Find $F(x)^{-1}$, by following lecture slide p.13.
Let y = F(x)

$$\begin{cases} y = \frac{1}{2} e^{-\alpha(\mu - x)} & if\ x < \mu \\ y = 1 - \frac{1}{2} e^{-\alpha(x - \mu)} & if\ x \geq \mu \end{cases}$$

$$\begin{cases} 2y = e^{-\alpha(\mu - x)} & if\ x < \mu \\ 2y - 2 = e^{-\alpha(x - \mu)} & if\ x \geq \mu \end{cases}$$

$$\begin{cases} ln(2y) = -\alpha(\mu - x) & if\ x < \mu \\ ln(2y - 2) = -\alpha(x - \mu) & if\ x \geq \mu \end{cases}$$

Inverse CDF
To define the range of y , we took the first equation of the previous section, and since α is postive in this assignment. We get,

$$\begin{cases} 0 \leq y < \frac{1}{2} & when\ x < \mu \\ \frac{1}{2} \geq y \geq 1 & when\ x \geq \mu \end{cases}$$

So,

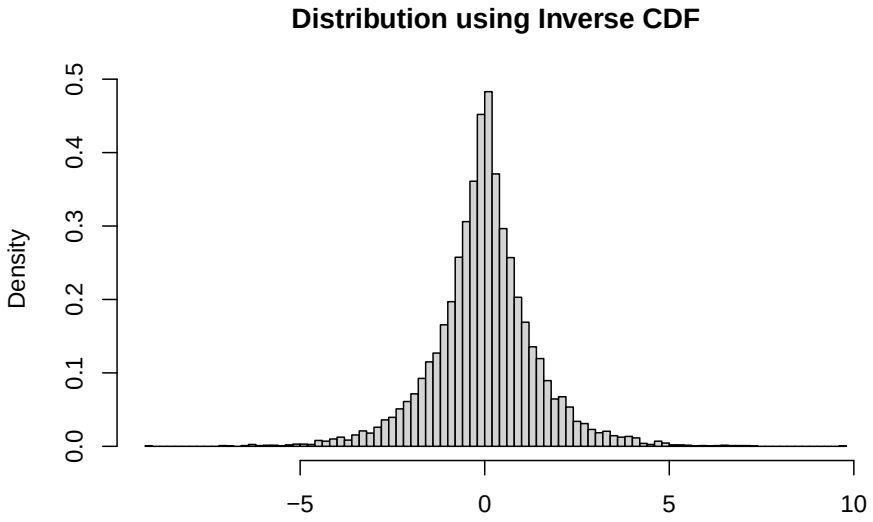
$$\begin{cases} x = \frac{ln(2y)}{\alpha} + \mu & if\ 0 \leq y < \frac{1}{2} \\ x = \frac{ln(2-2y)}{-\alpha} + \mu & if\ \frac{1}{2} \geq y \geq 1 \end{cases}$$

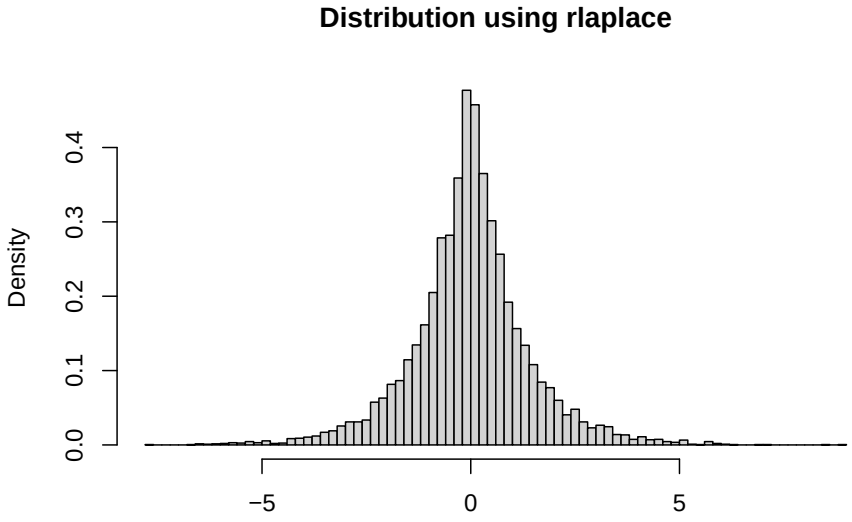
For$U \sim U(0, 1)$

$$\begin{cases} x = \frac{ln(2U)}{\alpha} + \mu & if\ 0 \leq U < \frac{1}{2} \\ x = \frac{ln(2-2U)}{-\alpha} + \mu & if\ \frac{1}{2} \geq U \geq 1 \end{cases}$$

```r
inverse_cdf_laplace <- function(u, mu, alpha){
  inverse_cdf_laplace <- c()

  for (i in 1:length(u)){
    if (u[i] < 0.5){
      inverse_cdf_laplace[i] <- (log(2*u[i])/alpha)+mu
      }
    else {
      inverse_cdf_laplace[i] <- mu+log(2-2*u[i])/(-alpha)
      }
  }

  return(inverse_cdf_laplace)
}
```

Then we implement the formula into a function, and get the sample form it.





Comparing the histogram using inverse CDF method and the one created using rlaplace from VGAM package. It can be observed that the Inverse CDF method produce similar graph, hence the result looks resonable.

Q2. Acceptance/rejection method

We starting with creating the target function(the pdf of Normal distribution) and the majorizing density (DE(0,1))

```
target <- function(x){
  1*exp(-x^2/2)/sqrt(2*pi)
}

majorizing  <-  function(x) {
   mu <- 0
   alpha <- 1
  result = (alpha/2)*exp(-alpha*abs(x-mu))
  return(result)
 }
```

Then we proceed to choose constant c. This is done by following steps.

$$\textit{The pdf of normal distribution is } f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

So, since $\sigma = 1$ and $\mu = 0$ and the density is symmetric, we can use x > 0 to get the value of c

$$c * \frac{1}{2}e^{-x} \geq \frac{e^{\left(-\frac{x^2}{2}\right)}}{\sqrt{2\pi}}$$

$$c \geq \sqrt{\frac{2}{\pi}} e^{(x-\frac{x^2}{2})}$$

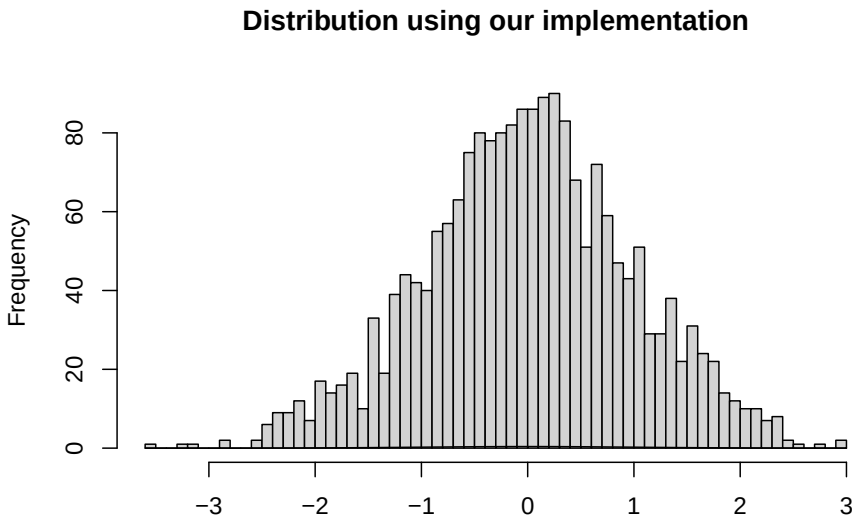
We then take the derivative on the right hand side and set it to 0, as we want to find the maximum of it (the minimum of c).
We get

$$\frac{d\sqrt{\frac{2}{\pi}}\exp\left(x-x^2/2\right)}{dx}=\frac{\sqrt{2}\left(1-x\right)\mathrm{e}^{x-\frac{x^2}{2}}}{\sqrt{\pi}}=0$$

When x=1, we will have the smallest c $=\sqrt{\frac{2}{\pi}}e^{\frac{1}{2}}$

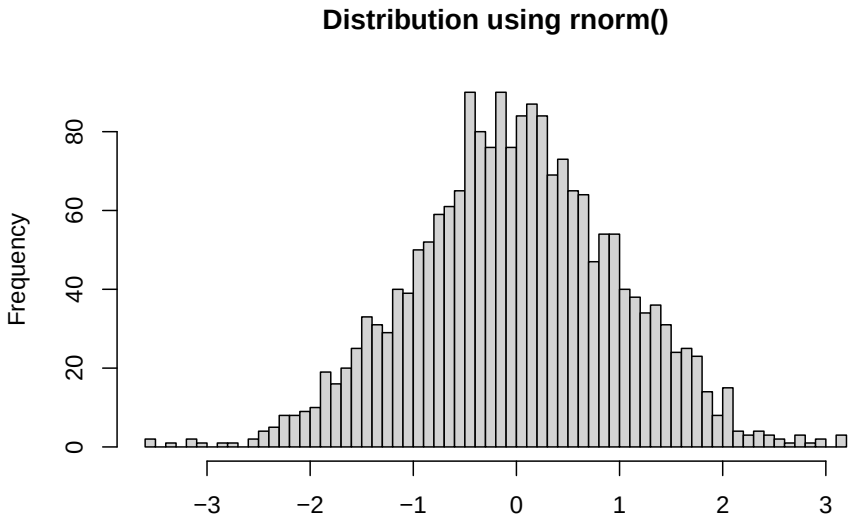
```
#get c
c <- sqrt( (2*exp(1)) / pi )
```

We then generate 2000 number using our implementation.



We get the rejection rate R=0.2386753 where the expect rejection rate ER= 0.2398265. The difference between is -0.0011513 which is very small.

Finally, we plot a histogram using rnorm() with 2000 samples. Apart from the binwidth difference, we can conclude that our implementation is good.



Reference

For Accept/Rejection algorithm in this lab, we referred to the video from Allen Kei
https://www.youtube.com/watch?v=WFswb-zLe4Y

Appendix

Assignment 1

```
rm(list=ls())
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(warning = FALSE)
knitr::opts_chunk$set(fig.align = "center")
library(ggplot2)
library(dplyr)
library(poweRlaw)
library(kableExtra)
library(VGAM)

target <- function(x,constant){

  output <- c()
  element1 <- constant*((sqrt(2*pi))^(-1))
  element2 <- exp((-(constant^2))/(2*x))
  element3 <- x^(-3/2)
  value <- element1*element2*element3
  final <- ifelse(between(x,0,Inf),value,0)
  return(final)
}

power_law <- function(x,alpha,tmin){
  value <- ((alpha-1)/tmin) * ((x/tmin)^(-alpha))*1

  final <- ifelse(between(x,tmin,Inf),value,NaN)
  return(final)

}

get_tmin <- function(constant){ #This function is used to get the c value of Tmin so we know where we n
  x <- (constant^2)/3
  f_x_max <- target(constant = constant,x = x)
  tmin <- (alpha-1)/f_x_max
  return(tmin)
}

x <- seq(0.01,30,0.1)

constant <- 1.2 # this is the constant c in f(x) formula we intialize as 1.2 and change it later
alpha =1.5
tmin<-get_tmin(constant)

max_target <- constant * (sqrt(2 * pi)^(-1)) * exp(-constant^(2) / (2 * (
      constant^2/3))) * (constant^2/3)^(-3/2)


plot_df <- data.frame(x1 = numeric(),y1 = numeric())
for (i in 1:length(x)) {
```

```
  plot_df[i,] <- c(x[i],power_law(x[i],alpha,tmin))

}
plot_df2 <- data.frame(x2=numeric(),y2=numeric())
for(i in 1:length(x)){
  plot_df2[i,] <- c(x[i],target(x[i],constant))
}
ggplot() +
  geom_line(data = plot_df2,aes(x=x2,y=y2, colour = "Target")) +
  geom_line(data = plot_df[-c(1:20),],aes(x = x1,y = y1,colour  = "Power-Law function"))


calculate_major_weight <- function(constant, tmin){
  weight_uni <- integrate(target,upper=tmin,lower = 0,constant=constant)
  weight_pl <- integrate(target,upper=Inf,lower=tmin,constant=constant)
  weight_vector <- c(unlist(weight_uni[1]),unlist(weight_pl[1]))

  return(unname(weight_vector))
}

weight_uni <- calculate_major_weight(constant,tmin)[1]
weight_pl <- calculate_major_weight(constant,tmin)[2]

majorizing_den = function(x, alpha, tmin,weight_uni,weight_pl){

  if(x>=tmin){
    final <-  weight_pl*power_law(x,alpha,tmin)
  }
  else{
    final <- weight_uni*1/tmin #the horizontal line
  }
  return(final)
}

plot_df3 <- data.frame(x3=numeric(),y3=numeric())
for(i in 1:length(x)){
  plot_df3[i,] <- c(x[i],majorizing_den(x[i], alpha, tmin,weight_uni,weight_pl))
}

ggplot() +
  geom_line(data = plot_df2,aes(x=x2,y=y2, colour = "Target function"))+
  geom_line(data = plot_df3,aes(x = x3,y = y3, colour = "majorizing density function"),linetype = 1)

get_minM <- function(tmin,max_target,weight){

  return((max_target*(tmin/weight))+0.01)
}

min_major_c <- get_minM(tmin,max_target,weight_uni)

plot_df3 <- data.frame(x3=numeric(),y3=numeric())
for(i in 1:length(x)){
  plot_df3[i,] <- c(x[i],min_major_c*majorizing_den(x[i], alpha, tmin,weight_uni,weight_pl))
```

```
}

ggplot() +
  geom_line(data = plot_df2,aes(x=x2,y=y2, colour = "Target function"))+
  geom_line(data = plot_df3,aes(x = x3,y = y3, colour = "M*majorizing density function"),linetype = 1)

AccRejdf <- data.frame()
Accept_Rejection <- function(C){
  #changed tmin inside this function to tmin1

  counter <- 0
  reject_counter <- 0
  alpha <- 1.5
  constant <- C
  res <-NA

  #initial parameters, here the parameters below will depend on different c (constant in target functio
  constant <- C
  max_target <-  constant * (sqrt(2 * pi)^(-1)) * exp(-constant^(2) / (2 * (
      constant^2/3))) * (constant^2/3)^(-3/2)
  tmin1 <- get_tmin(constant)


  weight_uni_for_c <- calculate_major_weight(constant,tmin1)[1]
  weight_pl_for_c <- calculate_major_weight(constant,tmin1)[2]
  major_c <- get_minM(tmin1,max_target,weight_uni_for_c)

  #print(tmin1)
  #print(max_target)
  #print(major_c)


  result_vector <-c()

  while (length(result_vector)<10000) {
      counter <- counter+1
      u <- runif(1)
      sample_from=sample(1:2,1,prob=c(weight_pl_for_c,weight_uni_for_c))
      if(sample_from==1) {
        z <- rplcon(1,tmin1,alpha)#get the x value, the naming z is to differentiate from x we use for
      }
      if(sample_from==2){
        z <- runif(1,0,tmin1)#get the x value
      }
      if(u <= target(z,constant)/(major_c*majorizing_den(z, alpha,tmin1,weight_uni_for_c,weight_pl_for_
      {
       result_vector <- c(result_vector,z)
      }
      else{
          reject_counter <- reject_counter + 1
      }

  }
```

```
  df <- as.data.frame(result_vector)
  assign(x = "df1",value = df,envir = .GlobalEnv )
  title <- paste0("Plot of C =",C)
  plot_output<-ggplot(df, aes(result_vector)) +
    geom_histogram(color="black", fill="darkblue",binwidth = 1)+
    xlim(0,100)
    theme_bw() + labs(x = "x",y = "Count")+
    ggtitle(title)+
    theme(plot.title = element_text(hjust = 2))

  m <- mean(result_vector)
  trunc_mean <- mean(result_vector[1:100])
  v <- var(result_vector)
  trunc_var <- var(result_vector[1:100])
  RR <- (reject_counter/counter)*100
  ER <- (1-(1/major_c))*100
  temp <- cbind(C,major_c,m,v,trunc_mean,trunc_var,RR,ER)
  AccRejdf <<- rbind(AccRejdf,temp)


  return(plot_output)

}

set.seed(12345)
Accept_Rejection(1.2)
Accept_Rejection(1.4)
Accept_Rejection(1.6)
Accept_Rejection(1.8)
Accept_Rejection(2.0)
Accept_Rejection(2.5)
Accept_Rejection(3)
Accept_Rejection(3.5)


colnames(AccRejdf) <- c("C","Majorizing Constant","Mean","Variance","Truncated Mean","Truncated Variance
kbl(AccRejdf[,c(1,2,7,8)], caption = "Summary of Rejection Rate for Different C", booktabs = T) %>%
  kable_styling(latex_options = "hold_position",
                                  position = "center")

#This code chunck is to compare the shape of slope between c=1.2 and c=2
ggplot() +
  geom_line(data = plot_df2,aes(x=x2,y=y2, colour = "Target function"))+
  geom_line(data = plot_df3,aes(x = x3,y = y3, colour = "M*majorizing density function"),linetype = 1)+
  ggtitle("c=1.2")



plot_df4 <- data.frame(x2=numeric(),y2=numeric())
for(i in 1:length(x)){
  plot_df4[i,] <- c(x[i],target(x[i],2))
}
```

```
constant_2 <- 2
tmin_c2 <- get_tmin(constant_2)
weight_uni_c2 <- calculate_major_weight(constant_2,tmin_c2)[1]
weight_pl_c2 <- calculate_major_weight(constant_2,tmin_c2)[2]
max_target_C2 <-  constant_2 * (sqrt(2 * pi)^(-1)) * exp(-constant_2^(2) / (2 * (
      constant_2^2/3))) * (constant_2^2/3)^(-3/2)
min_major_c2 <- get_minM(tmin_c2,max_target_C2,weight_uni_c2)

plot_df5 <- data.frame(x3=numeric(),y3=numeric())
for(i in 1:length(x)){
  plot_df5[i,] <- c(x[i],min_major_c2*majorizing_den(x[i], alpha, tmin_c2,weight_uni_c2,weight_pl_c2))
}
ggplot()+
  geom_line(data = plot_df4,aes(x=x2,y=y2, colour = "Target function"))+
  geom_line(data = plot_df5,aes(x = x3,y = y3, colour = "M*majorizing density function"),linetype = 1)+
  ggtitle("c=2")

kbl(AccRejdf[,-c(5,6,7,8)], caption = "Summary of statistics", booktabs = T) %>%
  kable_styling(latex_options = "hold_position",
                                  position = "center")
```

Assignment 2

```
library(ggplot2)


set.seed(12345)
U <- runif(10000, 0, 1)


inverse_cdf_laplace <- function(u, mu, alpha){
  inverse_cdf_laplace <- c()

  for (i in 1:length(u)){
    if (u[i] < 0.5){
      inverse_cdf_laplace[i] <- (log(2*u[i])/alpha)+mu
      }
    else {
      inverse_cdf_laplace[i] <- mu+log(2-2*u[i])/(-alpha)
      }
  }

  return(inverse_cdf_laplace)
}


plot_data <- inverse_cdf_laplace(U, mu=0, alpha=1)
hist(plot_data,prob=T,breaks=100,main="Distribution using Inverse CDF",xlab="")
hist(rlaplace(10000, location = 0, scale = 1),
     breaks = 100,main="Distribution using rlaplace",prob=T,xlab="")
```

```r
target <- function(x){
  1*exp(-x^2/2)/sqrt(2*pi)
}

majorizing  <-  function(x) {
   mu <- 0
   alpha <- 1
  result = (alpha/2)*exp(-alpha*abs(x-mu))
  return(result)
}

 #get c
 c <- sqrt( (2*exp(1)) / pi )



 smaple_vector <- rep(0,2000)
  counter <- 0
  reject_times <- 0
  for(i in 1:length(smaple_vector)){

    repeat{
      counter <- counter+1
      x <-  inverse_cdf_laplace(runif(1) ,0,1)
      u <-  runif(1)

      # counter <- counter+1
      if(u < target(x)/(c*majorizing(x))) {
        break}
      else{reject_times <- reject_times+1}
    }

    if(runif(1)<0.5){
      accept <- x
    }
    else{accept <- -x}

    smaple_vector[i] <- accept

  }
  rejection_rate <- (reject_times/counter)
  expected_rejection <- 1-1/c

  hist(smaple_vector,breaks=50,main="Distribution using our implementation")
  curve(dnorm(x,0,1),add=T)

  hist(rnorm(2000,0,1),breaks=50,
     main="Distribution using rnorm")
```