# Computer Lab 3

Tim Yuki Washio & Nicolas Taba

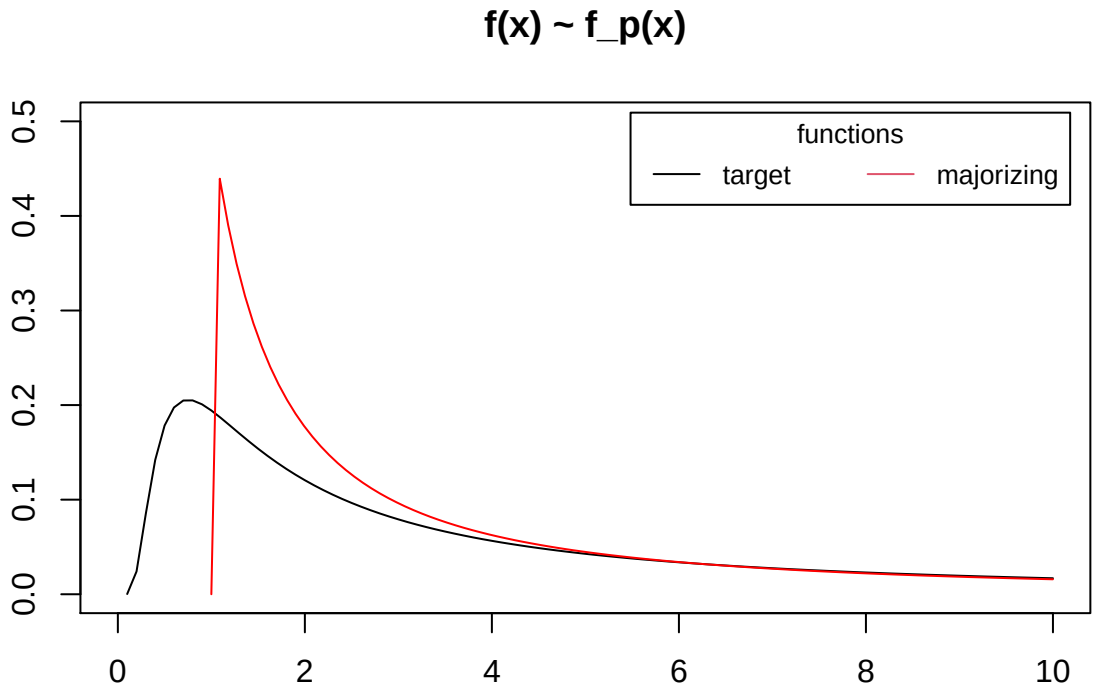
11/19/2020

## Question 1: Stable Distribution

### 1.

Plotting $f(x)$ with $c = 1.5$ and $f_p(x)$ with $\alpha = 1.5$ and $T_{min} = 1$. The value of c was chosen heuristically to illustrate the following argument.

```
c = 1.5
t_min = 1
alpha = 1.5

plot(1, xlim = c(0,10), ylim = c(0, 0.5), type = "n", xlab = "", ylab = "", main = "f(x) ~ f_p(x)")
curve(eval(c) * (sqrt(2 * pi)^(-1)) * exp(-eval(c)^(2) / (2 * x)) * x^(-3/2), from=0, to=10, add=TRUE,
curve(ifelse(x > t_min, (eval(alpha) - 1 / eval(t_min)) * (x / eval(t_min))^(-eval(alpha)), 0), from=t_

legend("topright", inset=.02, title="functions",
       c("target","majorizing"), horiz=TRUE, cex=0.8, col = 1:2, lty = 1)
```




The power-law distribution should not be used by itself because the two function do not have the same support. It might be better to combine the power-law distribution with another distribution that majorizes our target function in that area. We could use a uniform distribution with support $(0, T_{min}$ in addition to our given distribution with support $(T_{min}, \infty)$.

In order to find a good value for $T_{min}$, we should try to understand how the target function behaves. We try to identify the location of the maximum and relate this value to $c$.

$$\frac{df(x)}{dx} = \frac{ce^{(-c^2)/2x}(c^2 - 3x)}{2\sqrt{(2\pi)}x^{7/2}}$$

This has one root for $c \neq 0, x = \frac{c^2}{3}$.

$$max(f(x)) = f(\frac{c^2}{3})$$

We will set $T_{min}$ to be the value of $x$ for which $f_p(x)$ is the following:

$$f_p(x_{cutoff}) = f(\frac{c^2}{3})$$

with $x_{cutoff} = t_{min}$ from here on out.

We must now set a value for $\alpha$. With no additional information, other than $\alpha > 1$, we must choose an appropriate alpha for which the majorizing function is greater than our target function for every x on the support $(T_{min}, \infty)$. We choose alpha as small as possible in order to reduce rejections in the acceptance-rejection algorithm. The chosen value of $\alpha$ after visual inspection is 1.3.

## 2.

To implement the acceptance-rejection algorithm, we must choose a majorizing constant such that:

$$f_{target}(x) = Cf_p(x)$$

However, we must take into account the mixture of distributions of our majorizing distribution and should study the majorizing constant on both intervals where those distributions are supported. We start with the majorizing constant on the interval defined by the power-law distribution:

$$C \geq \frac{f_{target}(x)}{f_p(x)} = \frac{2c}{\sqrt{2\pi}} \cdot exp\left(\frac{-c^2}{2x}\right)$$

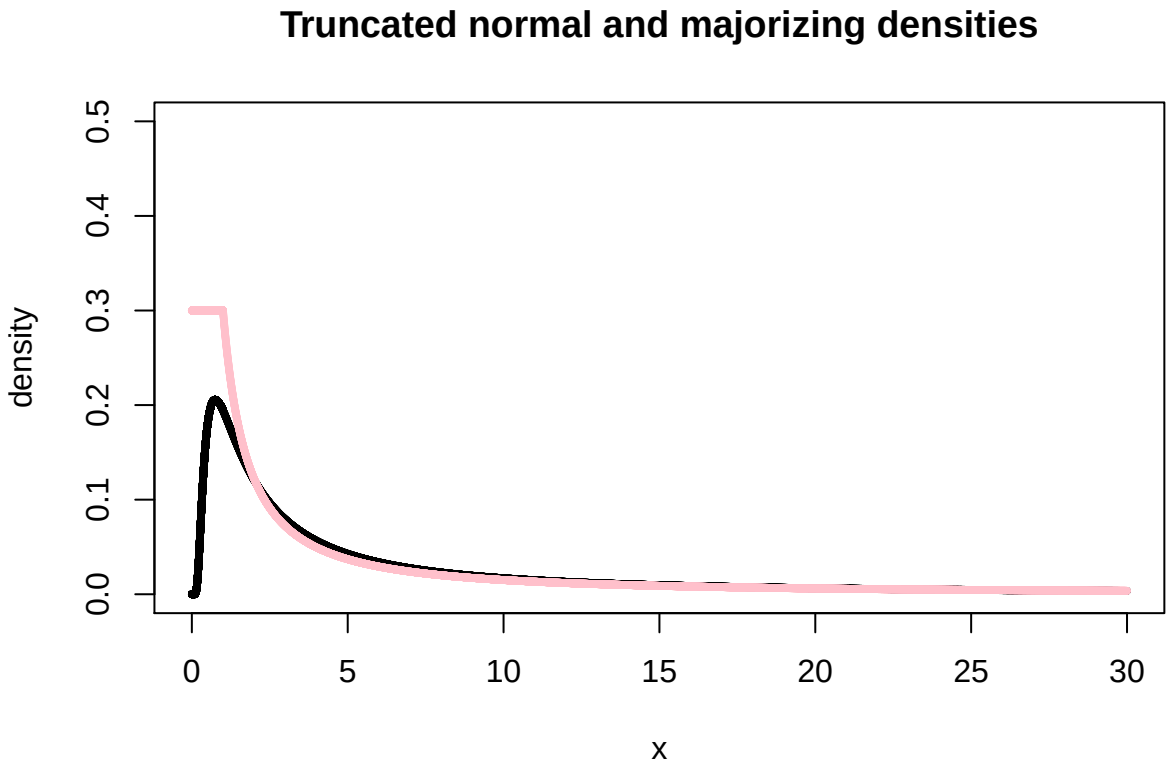
As $x$ increases, the majorizing constant goes to $\frac{2c}{\sqrt{2\pi}}$ so we choose this as our majorizing constant on the interval $[T_{min}, \infty[$. On the other interval, we choose our majorizing constant to have the same value as the power-law distribution at $T_{min}$, we have $f_p(x) = f_p(T_{min})$ on $[0, T_{min}]$. We thus have:

$$C \geq \frac{f_{target}(x)}{f_p(T_{min})} = \frac{2c}{\sqrt{2\pi}} exp\left(\frac{-c^2}{2x}\right) x^{-3/2}$$

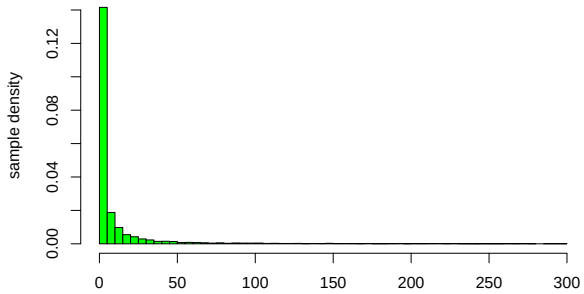
To get a better value for our rejection algorithm, we choose C to be as small as possible. We must choose our majorizing constant to be as large as the maximum value of our target function on this interval. We know that this happens when $x = c^2/3$ and we replace this value of x in the previously calculated majorizing constant and obtain:

$$C = \frac{2c}{\sqrt{2\pi}} e^{3/2} x^{-3/2}$$

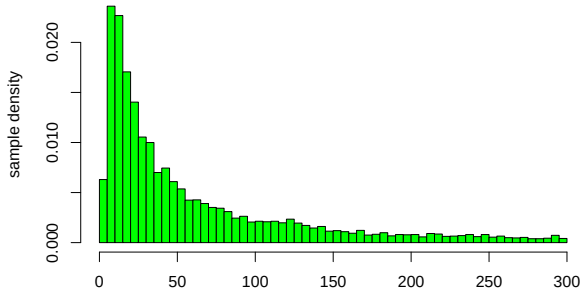
We can see here that the majorizing constant depends on the value of C and in the implementation of the algorithm, we will have to decide when to sample from either of the mixed distribution by integrating the target function from $[0, T_{min}]$.



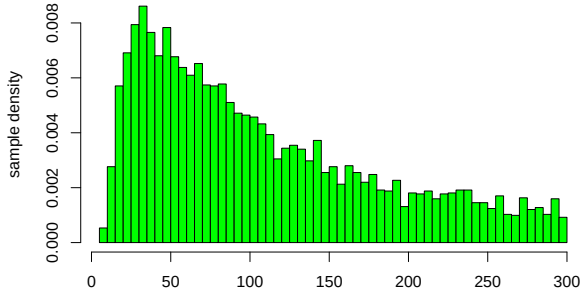

**3.**




```
##
## Mean:  20.03188
## Var:  9834.156
## Rejection rate:  0.2075442
```




```
##
## Mean:  38596.73
## Var:  1.079174e+12
## Rejection rate:  0.7474046
```




```
##
## Mean:  1119955
## Var:  7.328964e+15
## Rejection rate:  0.8754841
```

Both mean and variance increase as $c$ increases. The rejection rate is very high for large values of $c$. In the density graph we also observe that with rising $c$ the maximum of the sample density decreases. Therefore we conclude that the larger our $c$ the less we can trust the sampling algorithm as the only values that are not rejected will be very small values (machine precision level). Finally, the high rejection rate proves this.

# Question 2: Laplace Distribution

## Write a code generating random numbers from the double exponential distribution

The double exponential distribution is given by the followinf formula:

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

To get to the inverse CDF, $F^{-1}(p)$, of this function we first need to get the CDF $F(x)$

$$F(x) = \int_{-\infty}^{x} f(u)du = \begin{cases} \frac{1}{2}\mathrm{e}^{(\frac{\mathrm{x}-\mu}{\alpha})} & \text{if } \mathrm{x} < \mu \\ 1 - \frac{1}{2}\mathrm{e}^{(\frac{\mathrm{x}-\mu}{\alpha})} & \text{if } \mathrm{x} \geq \mu \end{cases}$$

After transformation, we get

$$F(x) = \frac{1}{2} + \frac{1}{2} sgn(x - \mu)\left(1 - exp\left(-\frac{|x - \mu|}{\alpha}\right)\right)$$

Now we calculate the inverse CDF $F^{-1}(p)$ by using the CDF $F(x)$. We set:

$$y = \frac{1}{2} + \frac{1}{2} sgn(x - \mu)\big(1 - \exp(-\alpha \mid \mathrm{x} - \mu \mid)\big)$$

And we now solve for x to obtain the inverse CDF. We have two cases. When $\mathrm{x} \geqslant \mu$ where we then have:

$$x = \mu - \frac{1}{\alpha}\ln(2 - 2\mathrm{y})$$

and the case when $\mathrm{x} < \mu$ when we have:

$$x = \mu + \frac{1}{\alpha}\ln(2\mathrm{y})$$

We combine both expression and express them with respect to the sign of de difference between x and $\mu$. We obtain the following:

$$F^{-1}(y) = \mu . sgn(x - \mu)\frac{1}{\alpha} ln(1 + sgn(x - \mu) - sgn(x - \mu)2y)$$

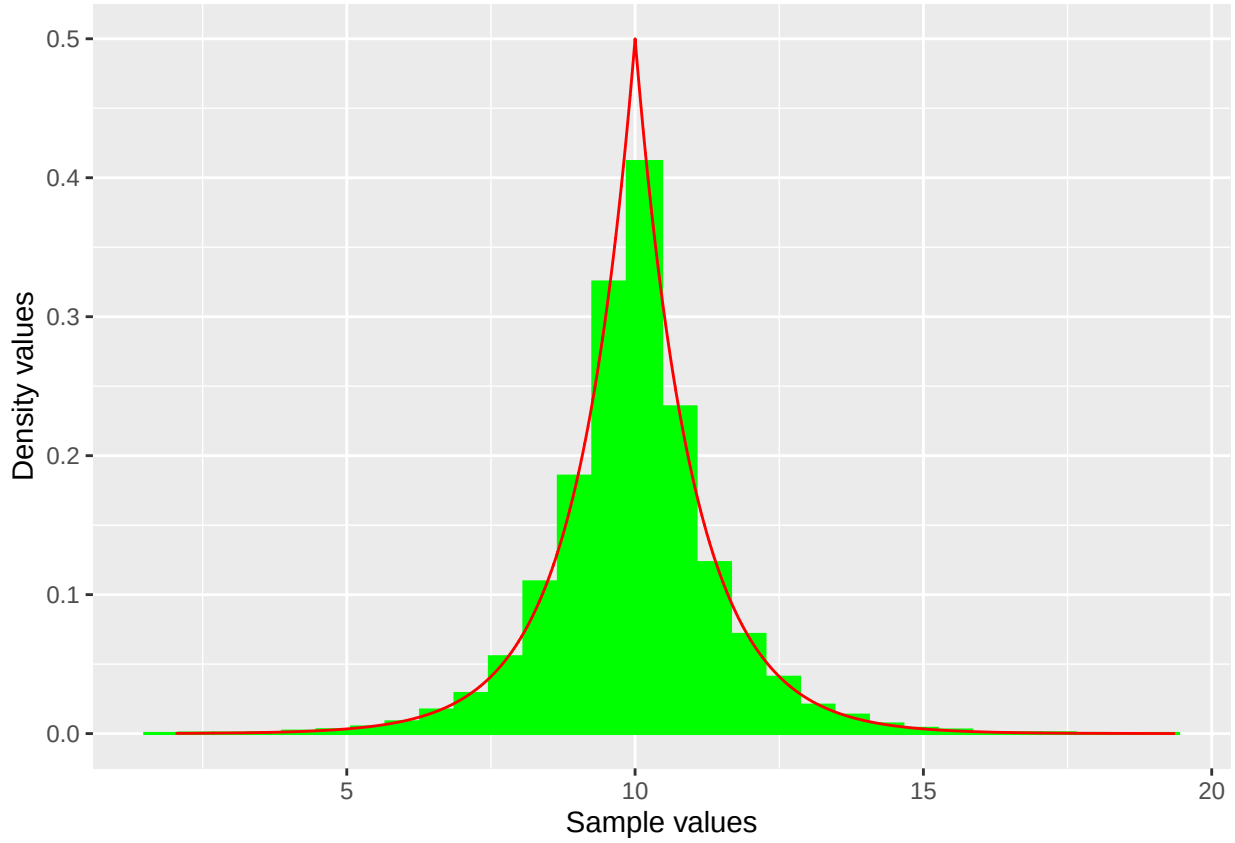
However, we would like an expression that does not depend on the sign of $x - \mu$. We try to find a quantity related to it but with respect to $y$. We investigate the critical value of $y$ when the sign of $x - \mu$ changes (i.e. we look for $sgn(x - \mu)\frac{1}{\alpha} ln(1 + sgn(x - \mu) - sgn(x - \mu)2y) = x - \mu$ when x is greater and smaller than $\mu$.)

We find that $sgn(x - \mu) = sgn(y - \frac{1}{2})$. We can simplify the expression of $F^{-1}(u)$ and finally obtain:

$$F^{-1}(u) = \mu - \alpha . sgn(u - \frac{1}{2}) ln(1 - 2|u - \frac{1}{2}|)$$

```
# Given density function
laplace_density = function(x, mu, alpha) {
  result = (alpha/2)*exp(-alpha*abs(x-mu))
  return(result)
}

# Calculated inverse CDF for Laplace function
```

```r
laplace_inv_cdf = function(p, mu, alpha){
  result = mu-alpha*sign(p-0.5)*log(1-2*abs(p-0.5))
  return(result)
}

# Random number generation with n=10000, mu=10, alpha=1
n = 10000
unif_sample = runif(n)
sample = laplace_inv_cdf(unif_sample, 10, 1)
sample_density = laplace_density(sample, 10, 1)
```




From this graph, we can see that our sampled numbers follow the distribution.

### Acceptance rejection method using DE(0,1) as a majorizing density for N(0,1).

The main task to solve this exercise is to find the majorizing constant $c$ such that

$$c \cdot f_M(x) \geq f_T(x)$$

Where $f_M(x)$ is the majorizing density and $f_T(x)$ is the density we wish to sample from (target density). This inequality must hold true for all x that are on the support of the target function. We must choose $c$ to be large enough for the inequality to hold true, but not so large that the rejection rate for the acceptance/rejection algorithm becomes too great. Setting the parameters in both densities to be (1,0) and setting our inequality, we have:

$$c \cdot \frac{1}{2} \exp -|x| \geq \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

We solve for $c$ and obtain:

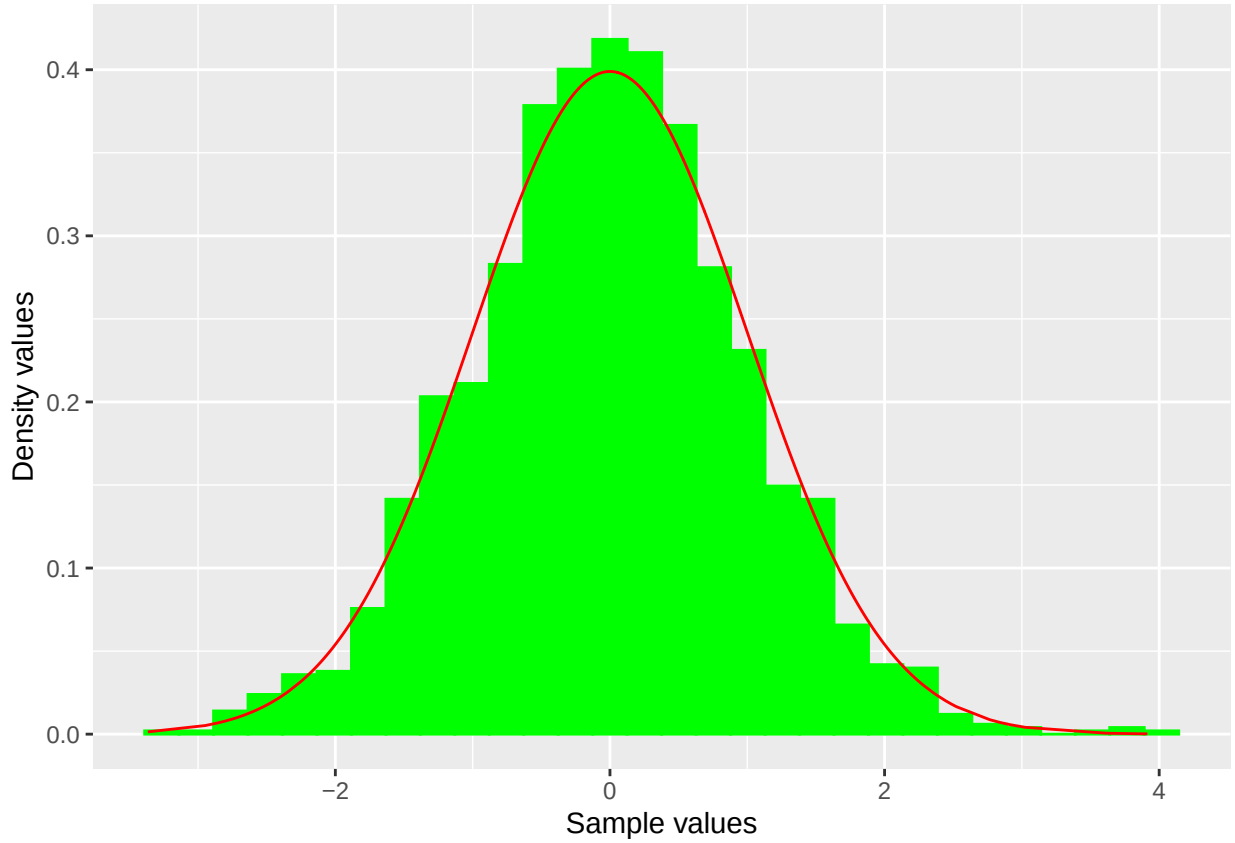
$$c \geq \sqrt{\frac{2}{\pi}} \exp\left(|x| - x^2/2\right)$$

We can find a solution for $c$ for $x > 0$ because we have an even function on the right-hand side and the other maximum will be attained at $x = -x_{positive}$ with the same value for $c$. The expression maximizes for $x = 1$ and yields:

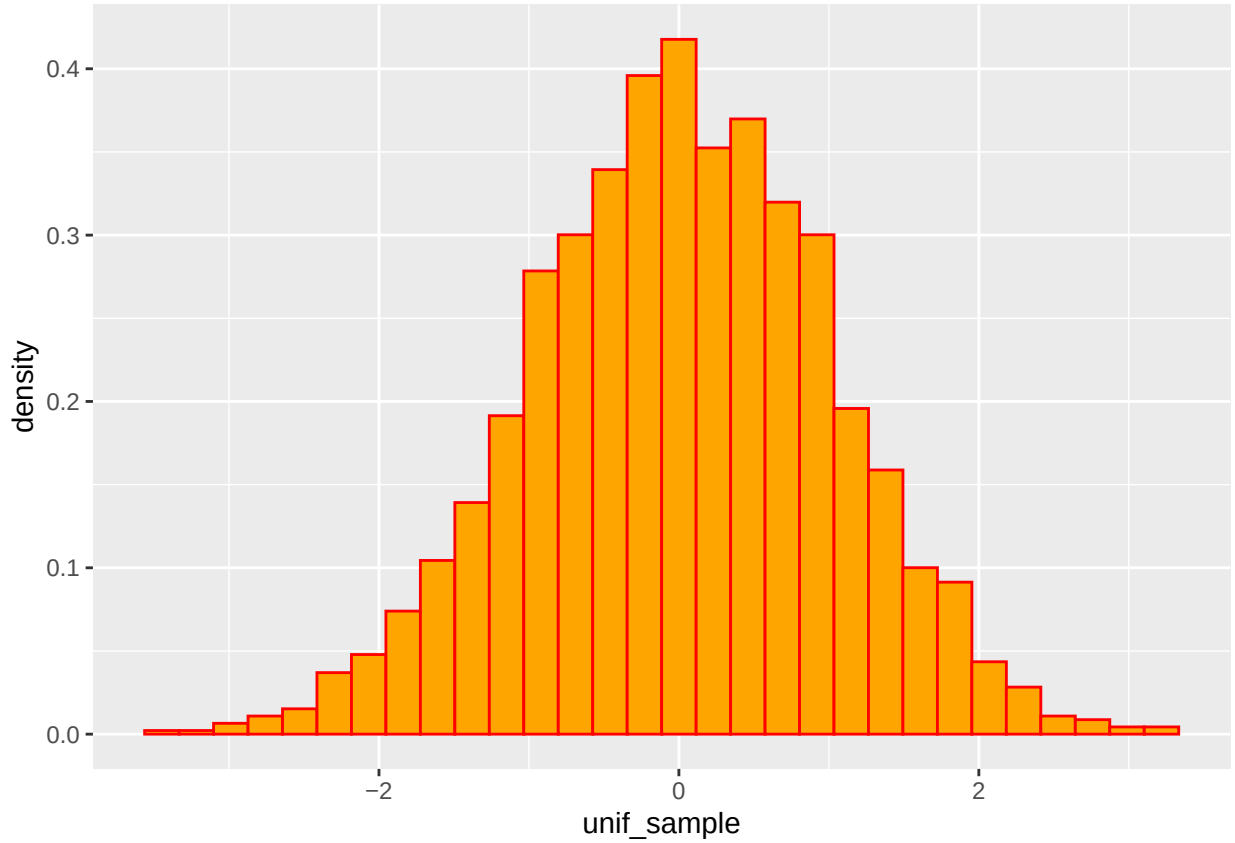
$$c_{major} = \sqrt{\frac{2e}{\pi}}$$

```
set.seed(12345)
c_major = sqrt( (2*exp(1)) / pi )


accept_reject = function(n){
  results = rep(0,times = n)
  counter = 0

  for(i in 1:n){
    reject = TRUE
    while(reject == TRUE){
      counter = counter + 1
      random_major = laplace_inv_cdf(runif(1), 0, 1)
      random_uniform = runif(1)
      if(random_uniform <= dnorm(random_major)/(c_major*laplace_density(random_major, 0, 1))){
        results[i] = random_major
        reject = FALSE
      }
    }
  }
  return(list(rn = results, draws = counter))
}
```

```
n = 2000
unif_sample = rnorm(n)
sample = accept_reject(n)


# Plotting the histogram
p1 = ggplot() +
geom_histogram(aes(x = sample$rn, y = ..density..), col = "green", fill="green") +
geom_line(aes(x = sample$rn, y = dnorm(sample$rn)), col = "red") +
xlab("Sample values") + ylab("Density values")

p1
```




```
# Histogram using rnorm
p2 = ggplot() +
  geom_histogram(aes(x= unif_sample, y= ..density..), col = "red", fill = "orange")

p2
```




Both historgrams have a similar shape.We now compute the rejection rate and the expected rejection rate.

```
reject_rate = 2000/sample$draws
expected_rejection = 1/c_major
difference = abs(reject_rate-expected_rejection)
relative = reject_rate/expected_rejection

cat("The rejection rate is: ", round(reject_rate, 4),"\nThe expected rejection rate is: ", round(expecte
```

```
## The rejection rate is:  0.7637
## The expected rejection rate is:  0.7602
## The difference in rejection rate is:  0.003476798
```

The rejection rate is very similar to the expected rejection rate. They differ by just 0.003476798.

# Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(poweRlaw)
c = 1.5
t_min = 1
alpha = 1.5

plot(1, xlim = c(0,10), ylim = c(0, 0.5), type = "n", xlab = "", ylab = "", main = "f(x) ~ f_p(x)")
curve(eval(c) * (sqrt(2 * pi)^(-1)) * exp(-eval(c)^(2) / (2 * x)) * x^(-3/2), from=0, to=10, add=TRUE,
curve(ifelse(x > t_min, (eval(alpha) - 1 / eval(t_min)) * (x / eval(t_min))^(-eval(alpha)), 0), from=t_

legend("topright", inset=.02, title="functions",
       c("target","majorizing"), horiz=TRUE, cex=0.8, col = 1:2, lty = 1)
target_fun = function(x, c) {
  res = 0
  res = ((c * (sqrt(2 * pi)^(-1)) * exp((-c^(2)) / (2 * x)) * x^(-3/2)))
  res[x<0] = 0
  return (res)
}

power_law = function(x, alpha, t_min) {
  return(((alpha - 1) / t_min) * (x / t_min)^(-alpha))
}


majorizing_fun = function(x, alpha, t_min) {
  sapply(x, function(y) {
    res = NA
    if (y<0) {
      res = 0
    }
    if ((y>=0) && (y<=t_min)) {
      res = power_law(t_min, alpha, t_min)
    }
    if (y>t_min) {
      res = power_law(y, alpha, t_min)
    }
    res
  }, simplify = TRUE)
}

alpha = 1.3
c = 1.5
t_min = 1

# Find out maximum value max(f(x)) for given parameters
x_max_target = c * (sqrt(2 * pi)^(-1)) * exp(-c^(2) / (2 * (c^2/3))) * (c^2/3)^(-3/2)

# Plot both the target function and the combined majorizing function
vx = c(seq(0, t_min, t_min/10000), seq(t_min, 30, 30/10000))
plot(vx, ylim=c(0, 0.5), (target_fun(vx, c)), pch=19, cex=0.4, xlab="x", ylab="density", main="Truncate
```

```r
points(vx, (majorizing_fun(vx, alpha, t_min)), pch=19, cex=0.4, col="pink")
Nsample=10000

rmajorizing=function(n){
  sapply(1:n,function(i){
    res=rplcon(1, t_min, alpha)
    return(res)
  })
}


fgentruncnormal=function(c_target){
  x=NA
  major_c_uniform = c_target^(-2) * 1 / sqrt(2 * pi) * 3^(3/2) * exp(-3/2)
  major_c_pow_law = 2 * c_target / sqrt(2 * pi)
  prob = integrate(target_fun, 0, t_min, c_target)$value
  num_reject=0
  num_tries=0
  while (is.na(x)){
    num_tries = num_tries + 1
    u=runif(1)
    if (u > prob) {
      y=rmajorizing(1)
      if (u<=target_fun(y, c_target)/(major_c_pow_law*majorizing_fun(y, alpha, t_min))){
        x=y
      } else{
        num_reject=num_reject+1
      }
    } else {
      y=runif(1)
      if (u<=target_fun(y, c_target)/(major_c_uniform*majorizing_fun(y, alpha, t_min))){
        x=y
      } else{
        num_reject=num_reject+1
      }
    }
  }
  c(x,num_reject,num_tries)
}

c_vals_target = c(1,5,10)
for (i in 1:length(c_vals_target)) {
  vtruncnormal_acceptreject = sapply(rep(c_vals_target[i],Nsample), fgentruncnormal)
  vtruncnormal_acceptreject_density_vals = vtruncnormal_acceptreject[1,]
  vtruncnormal_acceptreject_density_vals = vtruncnormal_acceptreject_density_vals[vtruncnormal_acceptre

  hist(vtruncnormal_acceptreject_density_vals, col="green", breaks=100, xlab="", ylab="sample density",

  vtruncnormal_acceptreject_mean = mean(vtruncnormal_acceptreject[1,])
  vtruncnormal_acceptreject_var = var(vtruncnormal_acceptreject[1,])

  cat("\nMean: ", vtruncnormal_acceptreject_mean)
  cat("\nVar: " , vtruncnormal_acceptreject_var)
```

```
  num_rejections = sum(vtruncnormal_acceptreject[2,])
  num_tries = sum(vtruncnormal_acceptreject[3,])
  rejection_rate = num_rejections/num_tries
  cat("\nRejection rate: ", rejection_rate)
}
# Given density function
laplace_density = function(x, mu, alpha) {
  result = (alpha/2)*exp(-alpha*abs(x-mu))
  return(result)
}

# Calculated inverse CDF for Laplace function
laplace_inv_cdf = function(p, mu, alpha){
  result = mu-alpha*sign(p-0.5)*log(1-2*abs(p-0.5))
  return(result)
}

# Random number generation with n=10000, mu=10, alpha=1
n = 10000
unif_sample = runif(n)
sample = laplace_inv_cdf(unif_sample, 10, 1)
sample_density = laplace_density(sample, 10, 1)
# Plotting the histogram
hist_plot = ggplot() +
geom_histogram(aes(x = sample, y = ..density..), col = "green", fill="green") +
geom_line(aes(x = sample, y = sample_density), col = "red") +
xlab("Sample values") + ylab("Density values")

hist_plot
set.seed(12345)
c_major = sqrt( (2*exp(1)) / pi )


accept_reject = function(n){
  results = rep(0,times = n)
  counter = 0

  for(i in 1:n){
    reject = TRUE
    while(reject == TRUE){
      counter = counter + 1
      random_major = laplace_inv_cdf(runif(1), 0, 1)
      random_uniform = runif(1)
      if(random_uniform <= dnorm(random_major)/(c_major*laplace_density(random_major, 0, 1))){
        results[i] = random_major
        reject = FALSE
      }
    }
  }
  return(list(rn = results, draws = counter))
}

n = 2000
```

```r
unif_sample = rnorm(n)
sample = accept_reject(n)


# Plotting the histogram
p1 = ggplot() +
geom_histogram(aes(x = sample$rn, y = ..density..), col = "green", fill="green") +
geom_line(aes(x = sample$rn, y = dnorm(sample$rn)), col = "red") +
xlab("Sample values") + ylab("Density values")

p1

# Histogram using rnorm
p2 = ggplot() +
  geom_histogram(aes(x= unif_sample, y= ..density..), col = "red", fill = "orange")

p2
reject_rate = 2000/sample$draws
expected_rejection = 1/c_major
difference = abs(reject_rate-expected_rejection)
relative = reject_rate/expected_rejection

cat("The rejection rate is: ", round(reject_rate, 4),"\nThe expected rejection rate is: ", round(expect
```