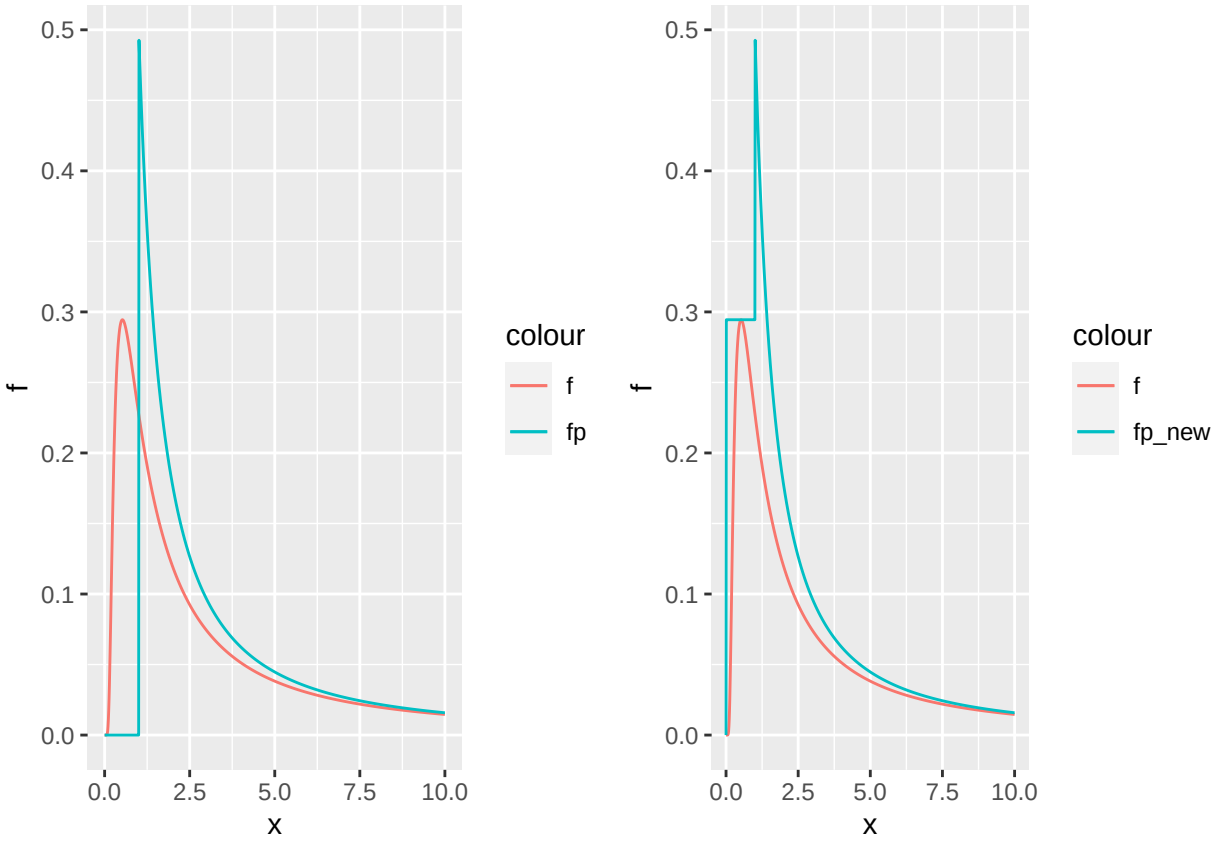
# Computational Statistics (732A90) Lab03

Christophoros Spyretos, Marc Braun, Marketos Damigos, Patrick Siegfried Hiemsch & Prakhar

2021-12-03

## Question 1

### Task 1




The power law distribution can not be used by itself, as the function is defined to be zero for $x \leq T_{min}$. Therefore, there exists no constant m such that the condition $m \cdot f_p(x) \geq f(x) \forall x$, as $f(x) > 0$ for $x > 0$. The problem is only in the region of $0 < x \leq T_{min}$ (which is called problematic interval from now on), as for $x \leq 0$ both functions evaluate to zero, so the condition $m \cdot f_p(x) \geq f(x)$ is fulfilled for $x \leq 0$. The problematic interval can not just be reduced to length zero as $T_{min} > 0$ by definition of the function. The maximum value of the function f calculates to

$$
\begin{aligned}
f'(x) &= 0 \\
\frac{c}{\sqrt{2\pi}} \frac{-3 \cdot e^{-c^2/2x}(c^2 - 3x)}{2 \cdot x^{3.5}} &= 0 \\
-c \cdot 3 \cdot e^{-c^2/2x}(c^2 - 3x) &= 0 \\
-c \cdot 3 \cdot e^{-c^2/2x} &\neq 0 \\
(c^2 - 3x) &= 0 \\
x &= \frac{c^2}{3} \\
f(\frac{c^2}{3}) &= \frac{c^{-2}}{\sqrt{2\pi}} \cdot \left(\frac{3}{e}\right)^{1.5}
\end{aligned}
$$

Therefore we can define a new function $f_{p,new} = \frac{c^{-2}}{\sqrt{2\pi}} \cdot \left(\frac{3}{e}\right)^{1.5}$ for $0 < x \leq T_{min}$ and $f_{p,new} = f_p$ elsewhere. This function now is larger or equal than $f(x)$ in the problematic interval by definition. The new function $f_{p,new}$ does not represent a density anymore, as its integral is larger than 1. This is not a problem though, because the integral is still finite (because the integral over the constant function in the problematic interval is finite as well as the integral over the power law distribution) and the function is scaled in the acceptance-rejection-method anyway. For the majoring constant m, the following condition must be fulfilled.

$$
m \cdot f_{p,new}(x) \geq f(x)
$$

First, the region where $f_{p,new}(x) = f_p(x)$ is examined.

$$
\begin{aligned}
m \cdot f_p(x) &\geq f(x) \\
m &\geq \frac{f(x)}{f_p(x)} \\
&= T_{min} \cdot \frac{\frac{c}{\sqrt{2\pi}} e^{-\frac{c}{2x}} x^{-\frac{3}{2}}}{(\alpha - 1)\left(\frac{x}{T_{min}}\right)^{-\alpha}}
\end{aligned}
$$

One can see that setting $T_{min} = 1$ and $\alpha = \frac{3}{2}$ vastly reduces the complexity of the expression. This leads to:

$$
m \geq \frac{2c \cdot e^{-\frac{c}{2x}}}{\sqrt{2\pi}}
$$

As:

$$
\frac{2c \cdot e^{-\frac{c}{2x}}}{\sqrt{2\pi}} \overset{x>0}{\leq} \frac{2c}{\sqrt{2\pi}}
$$

One can conclude that if:

$$
m \geq \frac{2c}{\sqrt{2\pi}}
$$

The desired condition is fulfilled for the function $f_p(x)$.
Now one can examine the problematic interval where $f_{p,new}(x) \neq f_p(x)$. If $m \geq 1$ the condition is also fulfilled in the problematic interval for $f_{p,new}$, because the unscaled function itself fulfils the desired condition.

Therefore, $m = max\left(1, \frac{2c}{\sqrt{2\pi}}\right)$ fulfils the condition.

### Task 2

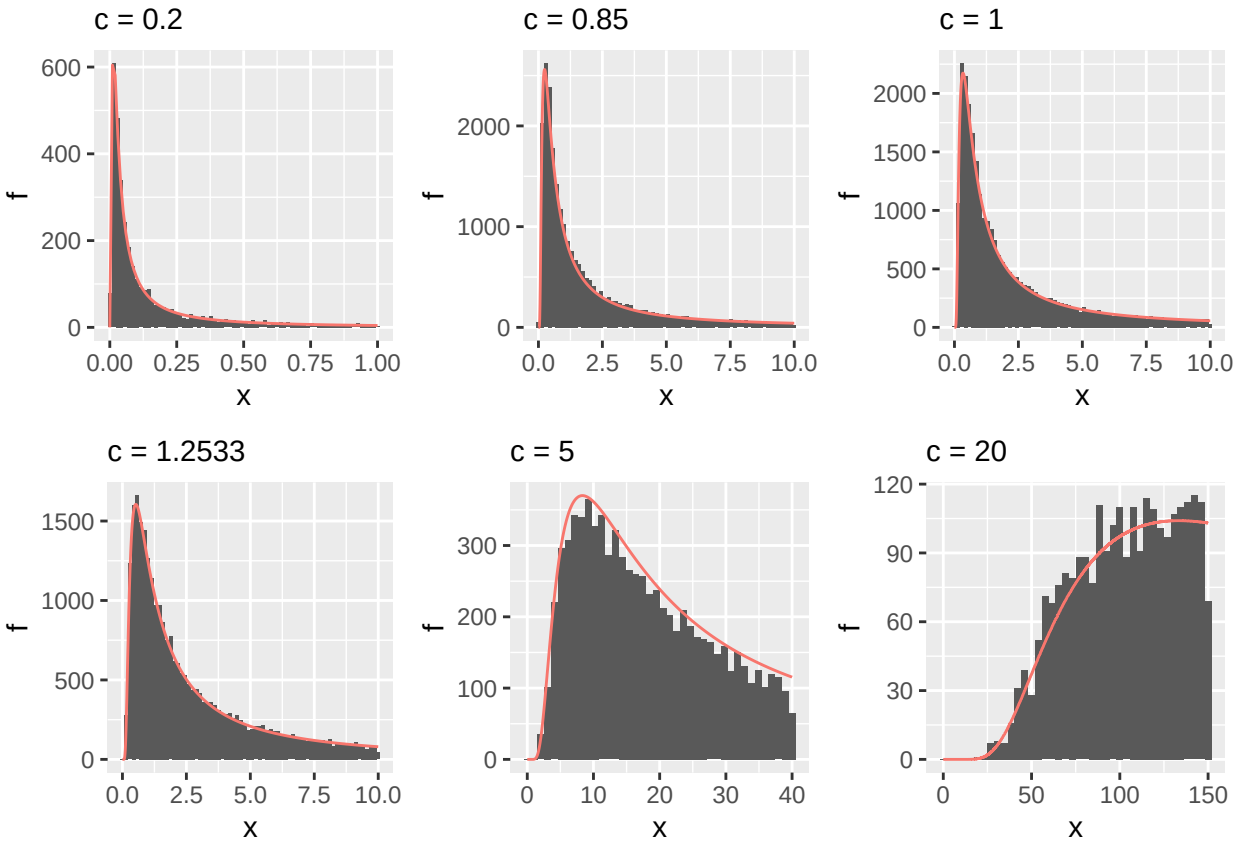
The acceptance-rejection algorithm was implemented according to the framework described in Task 1.

```
acceptance_rejection_sampler <- function(n, c_value){
  constant_for_uni_distr <- (c_value^-2) * (2 * pi)^(-(1/2)) * (3/exp(1))^(3/2)
  # The integral of fp_new in the probl. interval is equal to the constant, as the
  # interval has length 1.
  # The integral of the power law distribution equals 1 by definition.
  # According to this ratio between the two integrals, the samples are drawn from
  # either the uniform distribution or the power law distribution.
  ratio <- 1 / (1 + constant_for_uni_distr)
  number_fp <- rbinom(1, n, ratio)
  number_unif <- n - number_fp
  m <- 2 * c_value / (sqrt(2 * pi))
  if (m < 1){m <- 1}
  sample_fp <- poweRlaw::rplcon(number_fp, xmin = 1, alpha = 1.5)
  sample_unif <- runif(number_unif, 0, 1)
  sample_fp_new <- c(sample_fp, sample_unif)
  U <- runif(n, 0, 1)
  X <- c()
  for (i in 1:n) {
    if (U[i] <= (f(sample_fp_new[i], c = c_value) /
                 (m * fp_new(sample_fp_new[i], Tmin = 1, alpha = 1.5,
                             c = c_value)))){
      X <- append(X, sample_fp_new[i])
    }
  }
  return(X)
}
```

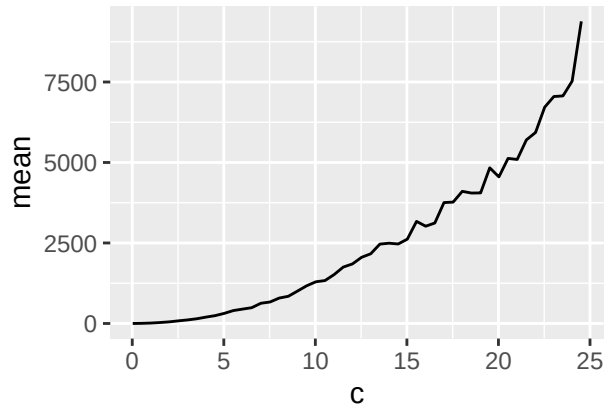
### Task 3

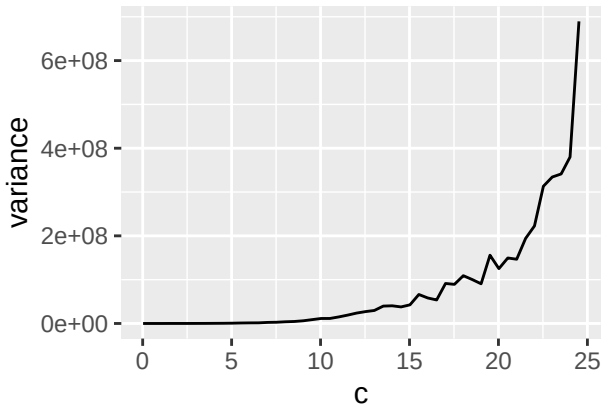
Shown below are the histograms of numbers generated by the algorithm for different values of c. Furthermore, the function $f$ is plotted for the respective c values but scaled to fit the height of the histograms. The means and variances for the plotted c values are also calculated. One can see that they do not fit the means and variances of the 95-percentile, which are shown for different c values further below.

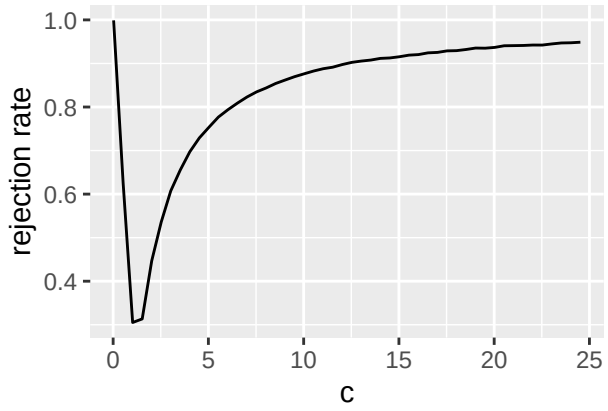
```
## [1] "the mean for c = 0.2 is 5545.8076141503"

## [1] "the variance for c = 0.2 is 57352204735.0939"

## [1] "the mean for c = 0.85 is 3459930.27592789"

## [1] "the variance for c = 0.85 is 183331378355954432"

## [1] "the mean for c = 1 is 8167.98122607227"

## [1] "the variance for c = 1 is 147240497442.983"

## [1] "the mean for c = 1.2533 is 43105.5782952238"

## [1] "the variance for c = 1.2533 is 10789050105399.1"

## [1] "the mean for c = 5 is 2051128.27326108"

## [1] "the variance for c = 5 is 58846201433900936"

## [1] "the mean for c = 20 is 5345386.60783429"

## [1] "the variance for c = 20 is 188884955699294592"
```




The means, variances and rejection rate for different c values are plotted below. For the means and variances, only the 95-percentile of a sample from the rejection-acceptance algorithm is plotted for each c value, as the outliers can vastly impact the outcome of the two metrics. The rejection rate decreases to a minimum of about 30% at a value of $c = \frac{1}{2}\sqrt{2\pi}$ because for this value, the majoring constant m is minimized for both the problematic interval as well as the power-law function $f_p$. This means that the function $f_{p,new}$ fits $f_p$ most closely under the restrictions and therefore maximizes the probability of a sample drawn from $f_{p,new}$ falling in the acceptance region. For larger values of c, the rejection rate increases again. It is not trivial to predict the mean because the expected value of the power-law the function is not well-defined for $\alpha \leq 2$ and in the proposed algorithm $\alpha = 1.5$ was chosen. Nonetheless, from the plots below, it seems like the mean and the variance are both increasing faster than proportional to c.


mean
7500
5000
2500
0
0
5
10
15
20
25
c


variance
6e+08
4e+08
2e+08
0e+00
0
5
10
15
20
25
c


rejection rate
1.0
0.8
0.6
0.4
0
5
10
15
20
25
c

## Question 2

### Task 1

The distribution function of the double exponential (Laplace) distribution is given by the formula:

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

$$= \begin{cases} \frac{1}{2}\exp(a(x-\mu_{DE}), x < \mu_{DE} \\ 1 - \frac{1}{2}\exp(-a(x-\mu_{DE})), x \geq \mu_{DE} \end{cases}$$

$$= \frac{1}{2} + \frac{1}{2} sgn(x-\mu_{DE})(1 - \exp(-\alpha|x-\mu_{DE}|)$$

We invert each part of the function separately.

- If $x \geq \mu_{DE} \Leftrightarrow x - \mu_{DE} \geq 0$ (relation 1):

$$\begin{aligned} y &= 1 - \frac{1}{2}\exp(-\alpha(x-\mu_{DE})) \\ y - 1 &= -\frac{1}{2}\exp(-\alpha(x-\mu_{DE})) \\ 2 - 2y &= \exp(-\alpha(x-\mu_{DE})) \\ \ln(2-2y) &= \ln(\exp(-\alpha(x-\mu_{DE}))) \\ \ln(2-2y) &= -\alpha(x-\mu_{DE}) \\ -\frac{1}{\alpha}\ln(2-2y) &= \mu_{DE} - x(\text{ relation 2}) \\ x &= \mu_{DE} - \frac{1}{\alpha}\ln(2-2y) \end{aligned}$$

From relations 1 & 2:

$$\begin{aligned} -\frac{1}{a}\ln(2-2y) &\geq 0 \\ \ln(2-2y) &\leq 0 \\ \ln(2-2y) &\leq \ln 1 \\ 2-2y &\leq 1 \\ 2-1 &\leq 2y \\ 1 &\leq 2y \\ y &\geq \frac{1}{2} \end{aligned}$$

- If $x < \mu_{DE} \Leftrightarrow x - \mu_{DE} < 0$ (relation 3):

$$\begin{aligned} y &= \frac{1}{2}\exp(\alpha(x-\mu_{DE})) \\ 2y &= \exp(\alpha(x-\mu_{DE})) \\ \ln(2y) &= \ln(\exp(\alpha(x-\mu_{DE}))) \\ \ln(2y) &= \alpha(x-\mu_{DE}) \\ \frac{1}{\alpha}\ln(2y) &= x - \mu_{DE}(\text{ relation 4}) \\ x &= \mu_{DE} + \frac{1}{\alpha}\ln(2y) \end{aligned}$$

From relations 3 & 4:

$$\begin{aligned}
\frac{1}{a}\ln(2y) &< 0 \\
\ln(2y) &< 0 \\
\ln(2y) &< \ln 1 \\
2y &< 1 \\
y &< \frac{1}{2}
\end{aligned}$$

The sign function is described below and it will be used to combine the two inverted parts.

$$sgn(x) = \begin{cases} -1, x < 0 \\ 0, x = 0 \\ 1, x > 0 \end{cases}$$

We combine the inverted parts to generate the inverted distribution function of the double exponential (Laplace) distribution.

$$\begin{aligned}
F^{-1}(x) &= \begin{cases} \mu_{DE} - \frac{1}{\alpha}\ln(2-2x), x \geq \frac{1}{2} \\ \mu_{DE} + \frac{1}{\alpha}\ln(2x), x < \frac{1}{2} \end{cases} \\
&= \mu_{DE} - \frac{1}{2}sgn(x-\frac{1}{2})ln(1 + sgn(x-\frac{1}{2}) - sgn(x-\frac{1}{2})2x) \\
&= \mu_{DE} - \frac{1}{2}sgn(x-\frac{1}{2})ln(1-2|x-\frac{1}{2}|) \\
&= \mu_{DE} - \frac{1}{2}sgn(x-\frac{1}{2})ln(1-|2x-1|)
\end{aligned}$$

The variable u follows the Uniform distribution $u \sim U(0,1)$.

The *cumulative distribution function*(CDF) of u is given by:

$$F_U(u) = P(U \leq u) = \begin{cases} 0, u < 0 \\ u, 0 \leq u \leq 1 \\ 1, u > 1 \end{cases}$$

Afterwards we use the inverse CDF method.

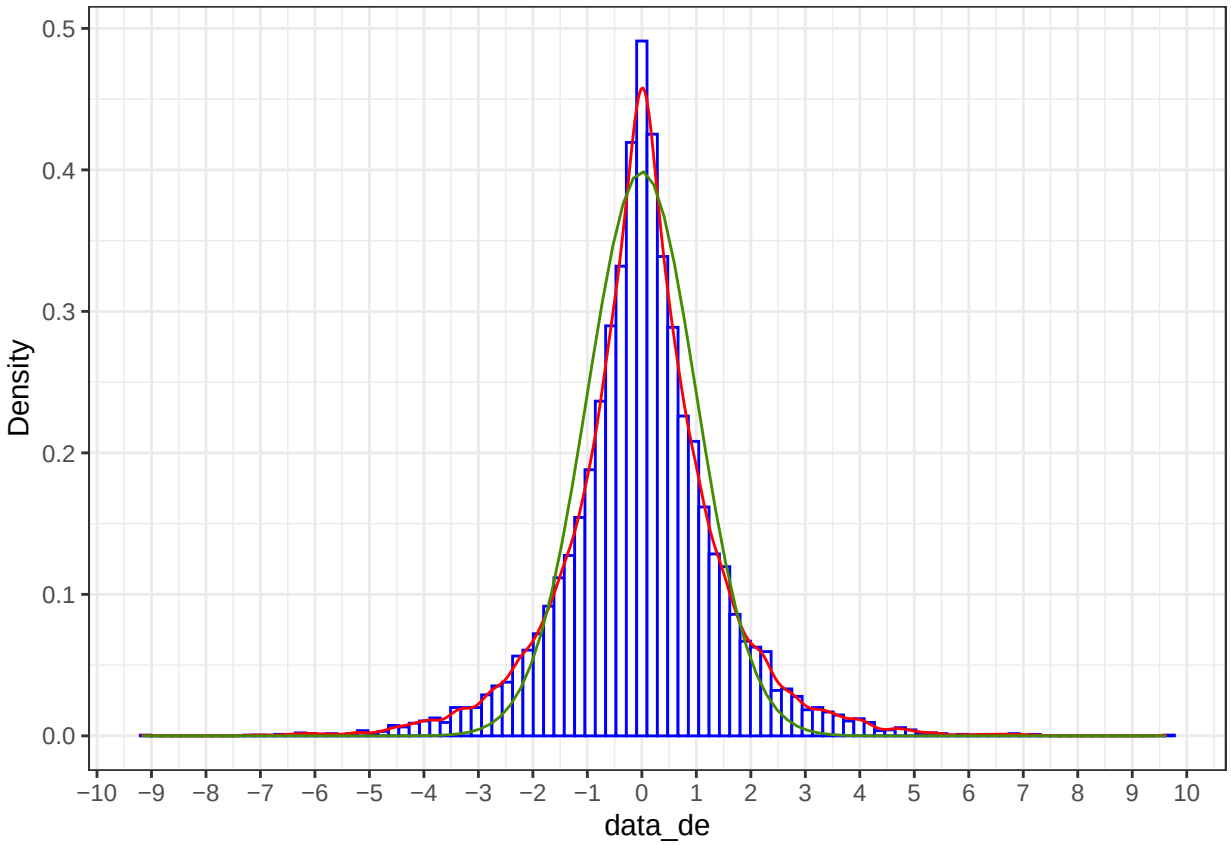
Consider $Y = F_X^{-1}$.

$$\begin{aligned}
F_Y(y) &= P(Y \leq y) \\
&= P(F_X^{-1}(u) \leq y) \\
&= P(F_X(F_X^{-1}(u)) \leq F_X(y)) \\
&= P(u \leq F_{X(y)}) \\
&= F_U(F_X(y)) \\
&= F_X(y)
\end{aligned}$$

We combine the inverse function of the double exponential distribution (Laplace) DE(0, 1) and the inverse CDF method of u~U(0,1).

$$F^{-1}(u) = \mu_{DE} - \frac{1}{2} sgn(u - \frac{1}{2}) ln(1 - 2|u - 1|)$$

```
set.seed(12345)
de_distribution = function(m = 0, a = 1){
  u = runif(1)
  result = m-(1/a)*sign(u-0.5)*log(1-abs(2*u-1))
  return(result)
}

data_de = c()
for(i in 1:10000){
  data_de[i] <- de_distribution()
}

data_de = as.data.frame(data_de)

library(ggplot2)

ggplot(data = data_de, aes(x = data_de)) +
  geom_histogram(bins = 100, color = "blue", fill = "white", aes(y=..density..))+
  geom_density(colour = "red")+
  stat_function(fun = dnorm, color = "chartreuse4")+
  ylab("Density")+
  scale_x_continuous(breaks = -10:10)+
  theme_bw()
```




The Laplace distribution, which looks like the normal distribution, is unimodal (one peak), and it is also a symmetrical distribution. However, it has a sharper peak than the normal distribution. The normal distribution has very thin tails. The probability density rapidly drops as it moves further from the middle, like $exp(x^2)$. The Laplace distribution has moderate tails, going to zero like $exp(- \mid x \mid)$. From the above plots, we could assume that the result looks reasonable because the density of the normal distribution (green line) and the double exponential distribution (red line) clearly illustrate the above description.

**Task 2**

The probability density function of Y~DE(0,1) is given by the formula $f_Y(x) = \frac{1}{2} exp^{-|x|}$.

The probability density function of N~(0,1) is given by the formula $f_X(x) = \frac{1}{\sqrt{2\pi}} exp^{-\frac{x^2}{2}}$.

The ratio boundary of $\frac{f_X(x)}{cf_Y(x)}$ is between 0 and 1 ,thus $0 < \frac{f_X(x)}{cf_Y(x)} \leq 1$.

Because c>0 we have the below relations.

$$0 < \frac{f_X(x)}{cf_Y(x)} \leq 1$$

$$0 < \frac{f_X(x)}{f_Y(x)} \leq c$$

We want to maximise c and by replacing $f_X(x)$ and $f_Y(x)$ we get the below relations.

$$c \geq \frac{f_X(x)}{f_Y(x)}$$
$$c \geq \frac{\frac{1}{\sqrt{2\pi}} exp^{-\frac{x^2}{2}}}{\frac{1}{2} exp^{-|x|}}$$
$$c \geq \frac{2}{\sqrt{2\pi}} exp^{|x|-\frac{x^2}{2}}$$

Thus, we need to maximise $exp^{|x|-\frac{x^2}{2}}$.

We want $x > 0$ and we set $g(x) = x - \frac{x^2}{2}$ and we calculate the derivative.

$g'(x) = 1 - \frac{2x}{2} = 1 - x$

Setting $g'(x) = 0$ to find x.

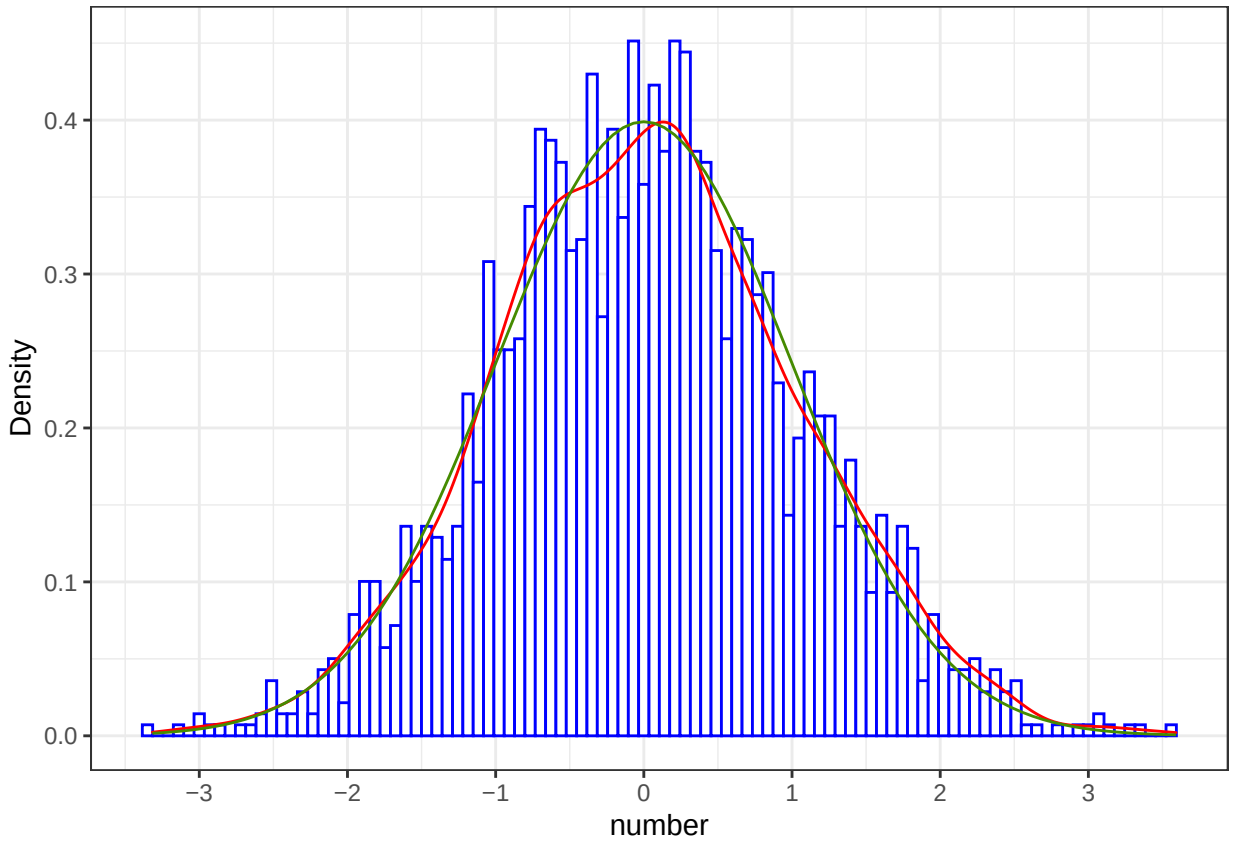
$g'(x) = 0 \Leftrightarrow 1 - x = 0 \Leftrightarrow x = 1$

The constant is given by the below relation.

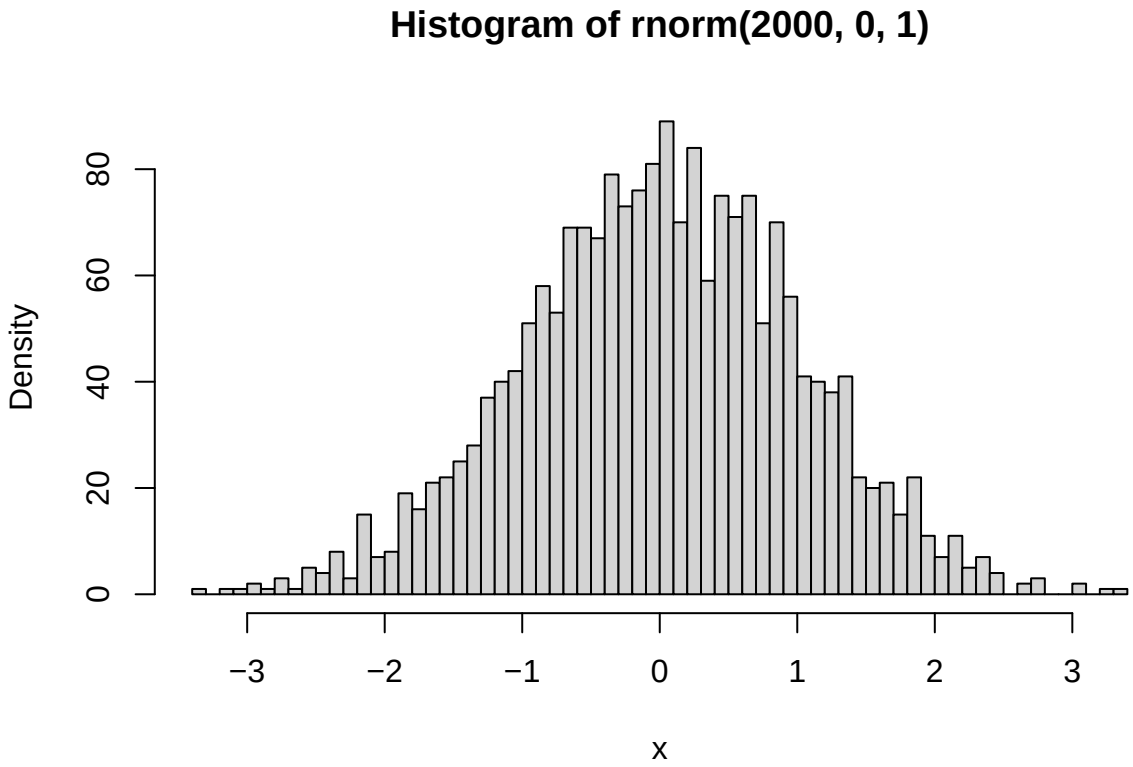
$$c \geq \frac{2}{\sqrt{2\pi}} e^{1-\frac{1}{2}} = \frac{2}{\sqrt{2\pi}} e^{\frac{1}{2}} = \frac{2\sqrt{e}}{\sqrt{2\pi}} \simeq 1.315$$

```
de_pdf = function(x){
  (1/2)*exp(-abs(x))
}

c = (2*sqrt(exp(1))) / (sqrt(2*pi))

ar_method = function(c){
  x=NA
  rej_counter = 0
  while(is.na(x))
  {
    y = de_distribution()
    U = runif(1)
    f_y = dnorm(y, mean = 0, sd = 1)
    g_y = de_pdf(y)
    if(U < f_y / (c * g_y))
    {
      x=y
    }
    else
    {
        rej_counter=rej_counter+1
    }
  }
  return(c(y, rej_counter))
}

ar_data <- data.frame(number = 1, rejections = 1)

for(i in 1:2000){
  ar_data[i,] <- ar_method(c=c)
}
```

```r
set.seed(12345)
ar_data$norm = rnorm(2000,0,1)

ggplot(data = ar_data, aes( x = number))+
  geom_histogram(bins = 100, color = "blue", fill = "white", aes(y=..density..))+
  geom_density(colour = "red")+
  stat_function(fun = dnorm, color = "chartreuse4")+
  ylab("Density")+
  scale_x_continuous(breaks = -10:10)+
  theme_bw()
```




```r
set.seed(12345)
hist(rnorm(2000,0,1), xlab = "x", ylab = "Density", breaks = 50 )
```




Comparing both histograms and densities of the standardised normal distribution N(0,1) (green line) and the Acceptance/rejection method with DE(0,1) as a major density to generate N(0,1) variables (red line), we could assume that they almost look similar. Thus our code could responsibly generate 2000 random numbers N(0,1).

The observed rejection rate is 0.2472714, and the expected rejection rate is 0.2398265. Both numbers are very close to each other; thus, our algorithm provides sensible results.

```
reject_rate = 1- 2000/(2000+sum(ar_data$rejections))
reject_rate
```

```
## [1] 0.2472714
```

```
exp_reject_rate = 1-1/c
exp_reject_rate
```

```
## [1] 0.2398265
```

## Appendix

```
knitr::opts_chunk$set(echo = TRUE)
f <- function(x, c){
  if (x > 0){
    res <- c * (sqrt(2 * pi))^(-1) * exp(1)^(-c^2/(2*x)) * x^(-3/2)
  }
  else {res <- 0}
  return(res)
}

fp <- function(x, alpha, Tmin){
  if (x > Tmin){
    res <- (alpha - 1) / Tmin * (x / Tmin)^-alpha
  }
  else{res <- 0}
  return(res)
}

fp_new <- function(x, alpha, Tmin, c){
  if (x > Tmin){
    res <- (alpha - 1) / Tmin * (x / Tmin)^-alpha
  }
  else if (x <= 0) {
    res <- 0
  }
  else{
    res <- c^-2 * (2 * pi)^-(1/2) * (3/exp(1))^(3/2)
  }
  return(res)
}

fp_new2 <- function(x, alpha, Tmin){
  if (x > Tmin){
    res <- (alpha - 1) / Tmin * (x / Tmin)^-alpha
  }
  else if (x <= 0) {
    res <- 0
  }
  else{
    res <- 0.5
  }
  return(res)
}

c <- sqrt(2 * pi) / 2
x <- seq(0, 10, by=0.01)
plot_data <- data.frame("x" = x, "f" = sapply(x, f, c = c),
                        "fp" = sapply(x, fp,
                                      alpha=1.5,
                                      Tmin=1),
                        "fp_new" = sapply(x, fp_new,
                                      alpha=1.5,
                                      Tmin=1,
```

```
                                    c = c),
                        "fp_new2" = sapply(x, fp_new2,
                                    alpha=1.5,
                                    Tmin=1))

library(ggplot2)
library(gridExtra)
plot_fp <- ggplot(data=plot_data) +
  geom_line(mapping = aes(x, f, color="f")) +
  geom_line(mapping = aes(x, fp, color="fp"))

plot_fp_new <- ggplot(data=plot_data) +
  geom_line(mapping = aes(x, f, color="f")) +
  geom_line(mapping = aes(x, fp_new, color="fp_new"))

grid.arrange(plot_fp, plot_fp_new, nrow=1, ncol=2)

acceptance_rejection_sampler <- function(n, c_value){
  constant_for_uni_distr <- (c_value^-2) * (2 * pi)^(-(1/2)) * (3/exp(1))^(3/2)
  # The integral of fp_new in the probl. interval is equal to the constant, as the
  # interval has length 1.
  # The integral of the power law distribution equals 1 by definition.
  # According to this ratio between the two integrals, the samples are drawn from
  # either the uniform distribution or the power law distribution.
  ratio <- 1 / (1 + constant_for_uni_distr)
  number_fp <- rbinom(1, n, ratio)
  number_unif <- n - number_fp
  m <- 2 * c_value / (sqrt(2 * pi))
  if (m < 1){m <- 1}
  sample_fp <- poweRlaw::rplcon(number_fp, xmin = 1, alpha = 1.5)
  sample_unif <- runif(number_unif, 0, 1)
  sample_fp_new <- c(sample_fp, sample_unif)
  U <- runif(n, 0, 1)
  X <- c()
  for (i in 1:n) {
    if (U[i] <= (f(sample_fp_new[i], c = c_value) /
                 (m * fp_new(sample_fp_new[i], Tmin = 1, alpha = 1.5,
                             c = c_value)))){
      X <- append(X, sample_fp_new[i])
    }
  }
  return(X)
}


# Plots
plot_sample <- list()
length(plot_sample) <- 6

# Plot 1
x <- seq(0, 1, by=0.01)
plot_data_scaled_funtion <- data.frame("x" = x, "f" = 56 * sapply(x, f, c = 0.2))
plot_data_sample <- acceptance_rejection_sampler(50000, 0.2)
```

```r
print(paste("the mean for c = 0.2 is", as.character(mean(plot_data_sample))))
print(paste("the variance for c = 0.2 is", as.character(var(plot_data_sample))))
plot_data_sample <- data.frame("x" = plot_data_sample[which(plot_data_sample < 1)])

plot_sample[[1]] <- ggplot() +
geom_histogram(data = plot_data_sample, aes(x), bins = 70) +
geom_line(data=plot_data_scaled_funtion, mapping = aes(x, f, color="f")) +
labs(subtitle="c = 0.2") +
theme(legend.position = "none")

# Plot 2
x <- seq(0, 10, by=0.01)
plot_data_scaled_funtion <- data.frame("x" = x, "f" = 4000 * sapply(x, f, c = 0.85))
plot_data_sample <- acceptance_rejection_sampler(50000, 0.85)
print(paste("the mean for c = 0.85 is", as.character(mean(plot_data_sample))))
print(paste("the variance for c = 0.85 is", as.character(var(plot_data_sample))))
plot_data_sample <- data.frame("x" = plot_data_sample[which(plot_data_sample < 10)])

plot_sample[[2]] <- ggplot() +
geom_histogram(data = plot_data_sample, aes(x), bins = 70) +
geom_line(data=plot_data_scaled_funtion, mapping = aes(x, f, color="f")) +
labs(subtitle="c = 0.85") +
theme(legend.position = "none")

# Plot 3
x <- seq(0, 10, by=0.01)
plot_data_scaled_funtion <- data.frame("x" = x, "f" = 4700 * sapply(x, f, c = 1))
plot_data_sample <- acceptance_rejection_sampler(50000, 1)
print(paste("the mean for c = 1 is", as.character(mean(plot_data_sample))))
print(paste("the variance for c = 1 is", as.character(var(plot_data_sample))))
plot_data_sample <- data.frame("x" = plot_data_sample[which(plot_data_sample < 10)])

plot_sample[[3]] <- ggplot() +
geom_histogram(data = plot_data_sample, aes(x), bins = 70) +
geom_line(data=plot_data_scaled_funtion, mapping = aes(x, f, color="f")) +
labs(subtitle="c = 1") +
theme(legend.position = "none")

# Plot 4
x <- seq(0, 10, by=0.01)
plot_data_scaled_funtion <- data.frame("x" = x, "f" = 5450 * sapply(x, f, c = sqrt(2 * pi) / 2))
plot_data_sample <- acceptance_rejection_sampler(50000, sqrt(2 * pi) / 2)
print(paste("the mean for c = 1.2533 is", as.character(mean(plot_data_sample))))
print(paste("the variance for c = 1.2533 is", as.character(var(plot_data_sample))))
plot_data_sample <- data.frame("x" = plot_data_sample[which(plot_data_sample < 10)])

plot_sample[[4]] <- ggplot() +
geom_histogram(data = plot_data_sample, aes(x), bins = 70) +
geom_line(data=plot_data_scaled_funtion, mapping = aes(x, f, color="f")) +
labs(subtitle="c = 1.2533") +
theme(legend.position = "none")

# Plot 5
```

```r
x <- seq(0, 40, by=0.01)
plot_data_scaled_funtion <- data.frame("x" = x, "f" = 20000 * sapply(x, f, c = 5))
plot_data_sample <- acceptance_rejection_sampler(75000, 5)
print(paste("the mean for c = 5 is", as.character(mean(plot_data_sample))))
print(paste("the variance for c = 5 is", as.character(var(plot_data_sample))))
plot_data_sample <- data.frame("x" = plot_data_sample[which(plot_data_sample < 40)])

plot_sample[[5]] <- ggplot() +
geom_histogram(data = plot_data_sample, aes(x), bins = 40) +
geom_line(data=plot_data_scaled_funtion, mapping = aes(x, f, color="f")) +
labs(subtitle="c = 5") +
theme(legend.position = "none")

# Plot 6
x <- seq(0, 150, by=0.01)
plot_data_scaled_funtion <- data.frame("x" = x, "f" = 90000 * sapply(x, f, c = 20))
plot_data_sample <- acceptance_rejection_sampler(400000, 20)
print(paste("the mean for c = 20 is", as.character(mean(plot_data_sample))))
print(paste("the variance for c = 20 is", as.character(var(plot_data_sample))))
plot_data_sample <- data.frame("x" = plot_data_sample[which(plot_data_sample < 150)])

plot_sample[[6]] <- ggplot() +
geom_histogram(data = plot_data_sample, aes(x), bins = 40) +
geom_line(data=plot_data_scaled_funtion, mapping = aes(x, f, color="f")) +
labs(subtitle="c = 20") +
theme(legend.position = "none")


do.call(grid.arrange, c(plot_sample, ncol = 3, nrow=2))
means <- c()
variances <- c()
rejections <- c()
c <- seq(0.025,25,0.5)
for (i in c) {
  sample <- acceptance_rejection_sampler(100000, c_value = i)
  means <- append(means, mean(sample[which(sample < quantile(sample, probs = 0.95))]))
  variances <- append(variances, var(sample[which(sample < quantile(sample, probs = 0.95))]))
  rejections <- append(rejections, 1 - (length(sample) / 100000))
}
means_vars <- data.frame("means" = means, "vars" = variances,
                         "rejections" = rejections)

means_plot <- ggplot(means_vars, aes(x = c)) +
  geom_line(aes(y = means)) +
  labs(y = "mean")

vars_plot <- ggplot(means_vars, aes(x = c)) +
  geom_line(aes(y = vars)) +
  labs(y = "variance")

rej_plot <- ggplot(means_vars, aes(x = c)) +
  geom_line(aes(y = rejections)) +
  labs(y = "rejection rate")
```

```
grid.arrange(means_plot, vars_plot, rej_plot, ncol=2)
set.seed(12345)
de_distribution = function(m = 0, a = 1){
  u = runif(1)
  result = m-(1/a)*sign(u-0.5)*log(1-abs(2*u-1))
  return(result)
}

data_de = c()
for(i in 1:10000){
  data_de[i] <- de_distribution()
}

data_de = as.data.frame(data_de)

library(ggplot2)

ggplot(data = data_de, aes(x = data_de)) +
  geom_histogram(bins = 100, color = "blue", fill = "white", aes(y=..density..))+
  geom_density(colour = "red")+
  stat_function(fun = dnorm, color = "chartreuse4")+
  ylab("Density")+
  scale_x_continuous(breaks = -10:10)+
  theme_bw()
de_pdf = function(x){
  (1/2)*exp(-abs(x))
}

c = (2*sqrt(exp(1))) / (sqrt(2*pi))

ar_method = function(c){
  x=NA
  rej_counter = 0
  while(is.na(x))
  {
    y = de_distribution()
    U = runif(1)
    f_y = dnorm(y, mean = 0, sd = 1)
    g_y = de_pdf(y)
    if(U < f_y / (c * g_y))
    {
      x=y
    }
    else
    {
        rej_counter=rej_counter+1
    }
  }
  return(c(y, rej_counter))
}

ar_data <- data.frame(number = 1, rejections = 1)
```

```r
for(i in 1:2000){
  ar_data[i,] <- ar_method(c=c)
}
set.seed(12345)
ar_data$norm = rnorm(2000,0,1)

ggplot(data = ar_data, aes( x = number))+
  geom_histogram(bins = 100, color = "blue", fill = "white", aes(y=..density..))+
  geom_density(colour = "red")+
  stat_function(fun = dnorm, color = "chartreuse4")+
  ylab("Density")+
  scale_x_continuous(breaks = -10:10)+
  theme_bw()
set.seed(12345)
hist(rnorm(2000,0,1), xlab = "x", ylab = "Density", breaks = 50 )

reject_rate = 1- 2000/(2000+sum(ar_data$rejections))
reject_rate

exp_reject_rate = 1-1/c
exp_reject_rate
```