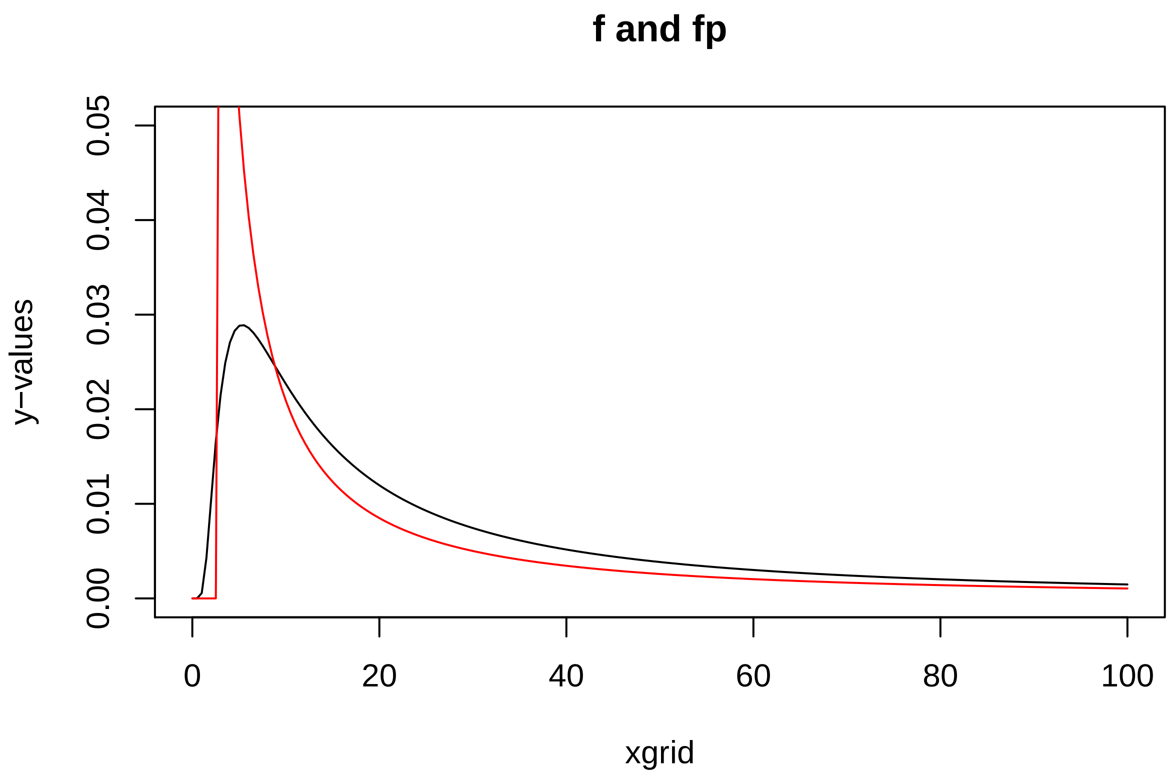
732A90: Lab 3

David Björelind, davbj395

11/18/2020

Question 1: Stable distribution

1.



One problem is that the two functions has different support. The power-law is supported on (Tmin, inf) while f(x) has support on (0, inf). This means that support for (0, Tmin) is missing from power-law. to account for this, a proposal density from a uniform distribution is used for (0, Tmin). Now, the proposal density has the support (0, Inf). The used proposal density is incorporated in the the acceptance/rejection algorithm and the weight between the two is calculated from the amount of probability mass that occurs for f(x) to the left of Tmin. When c=4 it is 2,2%, which seems reasonable.

Alpha and **Tmin** is picked so that the shape of **fp** looks similar to **f** for the values it is defined, see plot above. **Alpha = 1.3** seems to provide a good fit, and **Tmin = 3** puts the peaks at the same x-value.

2.

3.

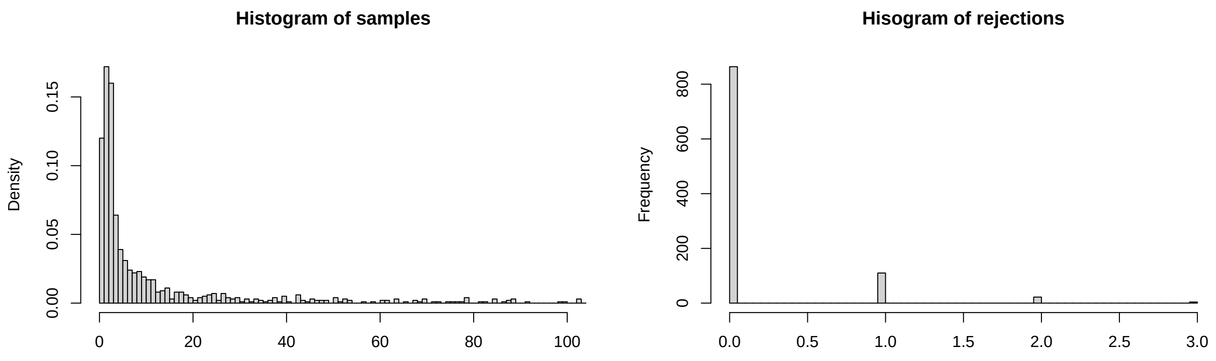
```
## [1] 1.5
##
## c =  1.5 gives:
## Mean rejection rate:  0.166
## Mean:  1051.606
## Variance:  470015043
```

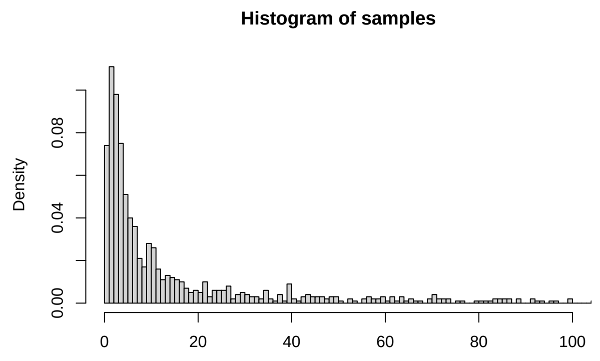
```
## [1] 2
##
## c =  2 gives:
## Mean rejection rate:  0.226
## Mean:  1033.531
## Variance:  70860144
```

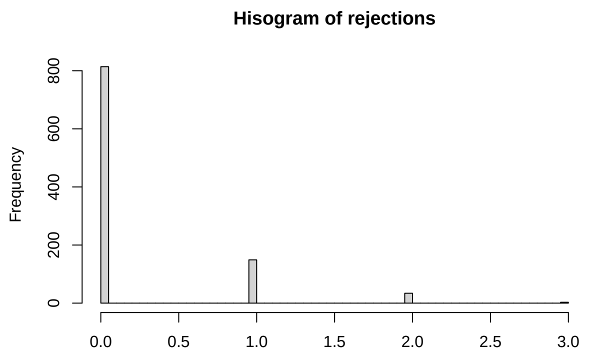
```
## [1] 3
##
## c =  3 gives:
## Mean rejection rate:  0.39
## Mean:  8510.195
## Variance:  22755733558
```

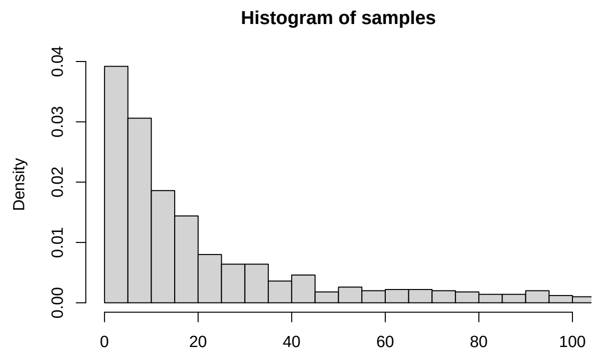
```
## [1] 4
##
## c =  4 gives:
## Mean rejection rate:  0.556
## Mean:  1786.966
## Variance:  271779135
```

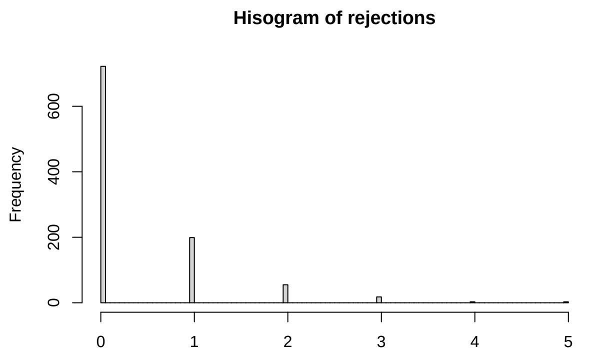
```
## [1] 5
##
## c =  5 gives:
## Mean rejection rate:  0.709
## Mean:  46431.05
## Variance:  713384063283
```

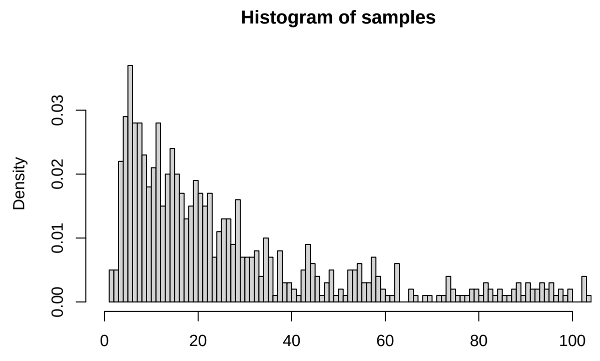


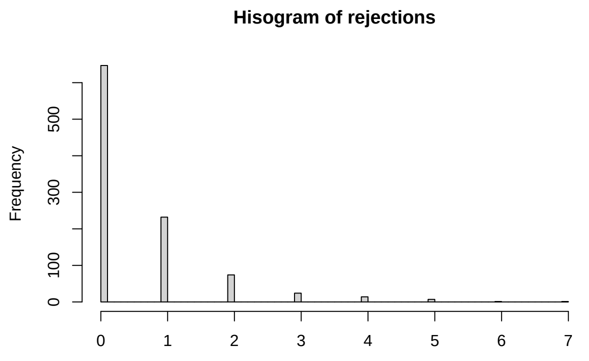

Histogram of samples
Density
0.00
0.04
0.08
0
20
40
60
80
100

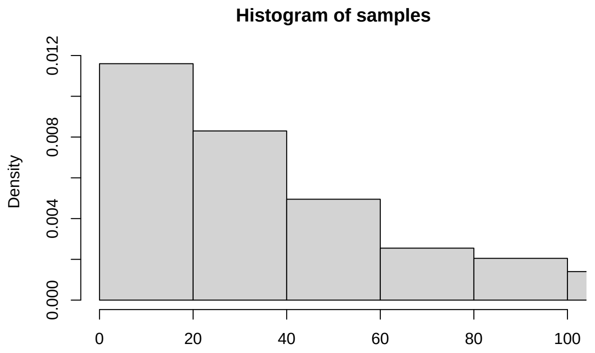

Hisogram of rejections
Frequency
0
200
400
600
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0

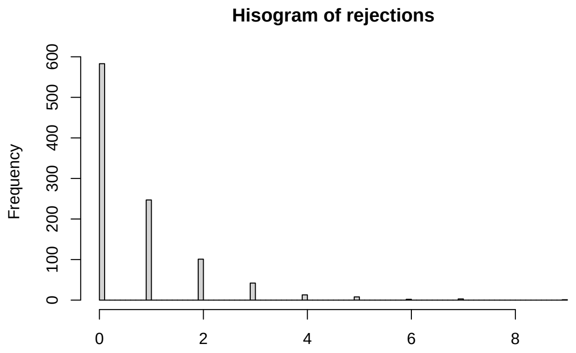

Histogram of samples
Density
0.00
0.01
0.02
0.03
0.04
0
20
40
60
80
100


Hisogram of rejections
Frequency
0
200
400
600
0
1
2
3
4
5


Histogram of samples
Density
0.00
0.01
0.02
0.03
0
20
40
60
80
100


Hisogram of rejections
Frequency
0
100
300
500
0
1
2
3
4
5
6
7





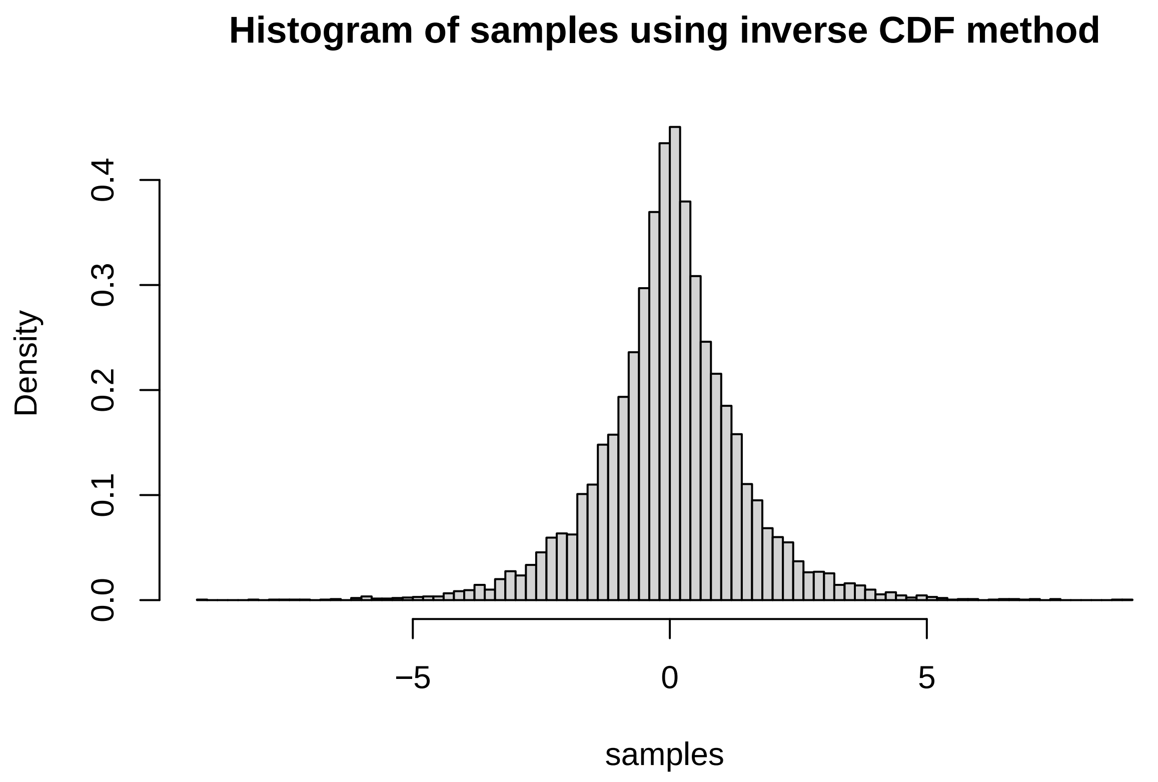
It can be seen that when **c** increases, the mean and variance increases with it. Judging from the behavior of **c** on **f(x)**, as **c** decreases, the higher the peak at the start becomes. A high **c**-value creates a low peak and a more smooth graph. A more smooth function will be easier to sample from using acceptance-rejection sampling than a jaggy function. This is because is it difficult to capture sharp peaks from the random nature of the sampling.

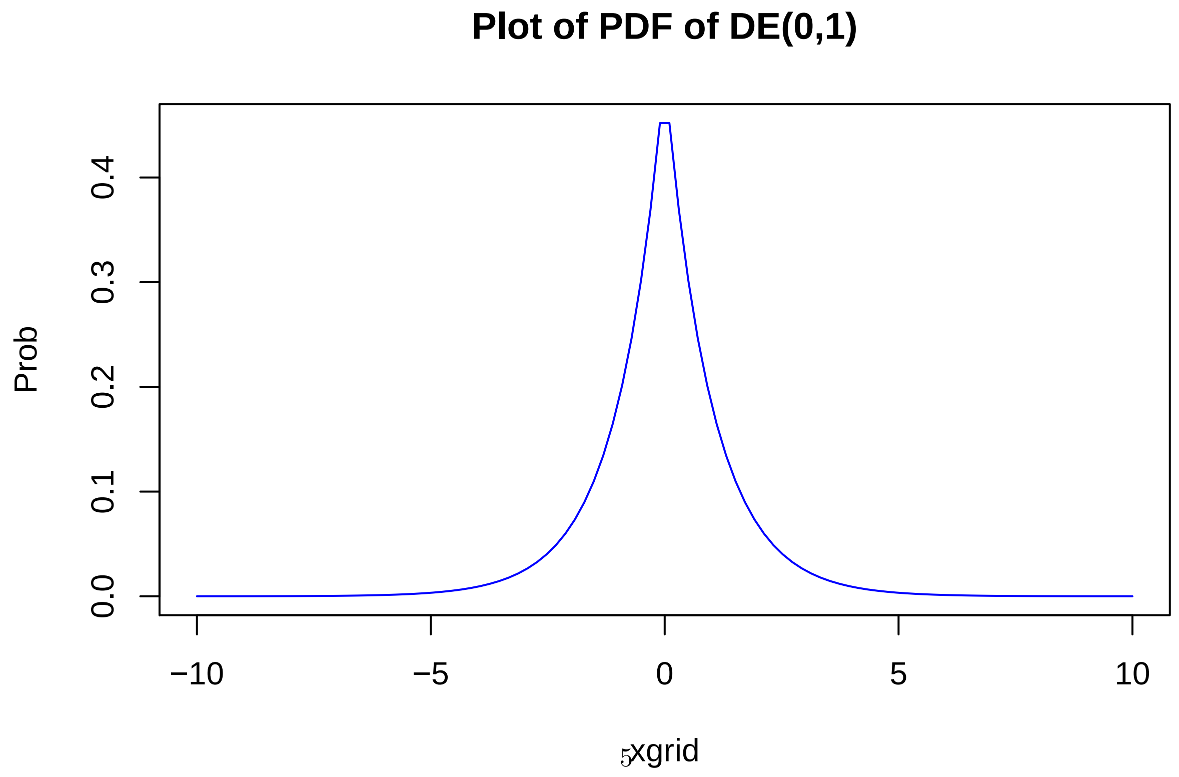
The reason why the mean and variance increases with a higher **c** is that larger values are being samples since the function becomes more smooth.

Rejection rates increase as **c** increases, see histograms of rejections rates. This can be explained by that the majorizing constant is increasing, making the proposal density function appear higher above the target density. Thus, increasing the number of rejections.

Question 2: Laplace distribution

1.

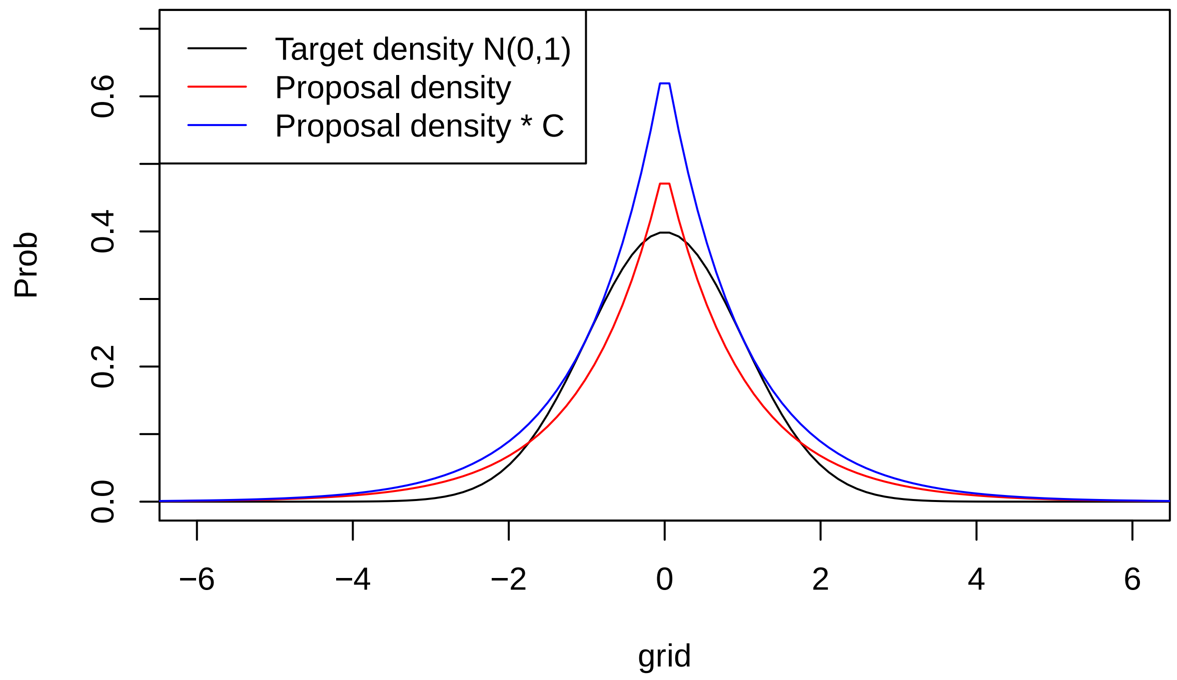




The PDF of DE(0,1) is **1/2 * e^(-|x|)**. When calculating the integral separation in two cases are necessary when x>0 and x<0. Integrating these from (-Inf,x) and (0,x) respectively I get the CDFs **1/2 * (1-e^(-x))** for x>0 and **1/2 * e^(x)** for x<0. When inverting these functions I get: **x=-ln(1-2y)** for x>0 and **ln(2y)** for x<0. When using the inverse CDF method, draws from **runif** > 0.5 will be calculated using **x=-ln(1-2y)** and < 0.5 calculated using **ln(2y)**. This is because half of the probability mass is at x>0 and the other half at x<0.

Judging from the produced graphs, the results looks very reasonable. The samples achieve the same peak at x=0 and flats out at around the same values. The inverse CDF method works great.

2.

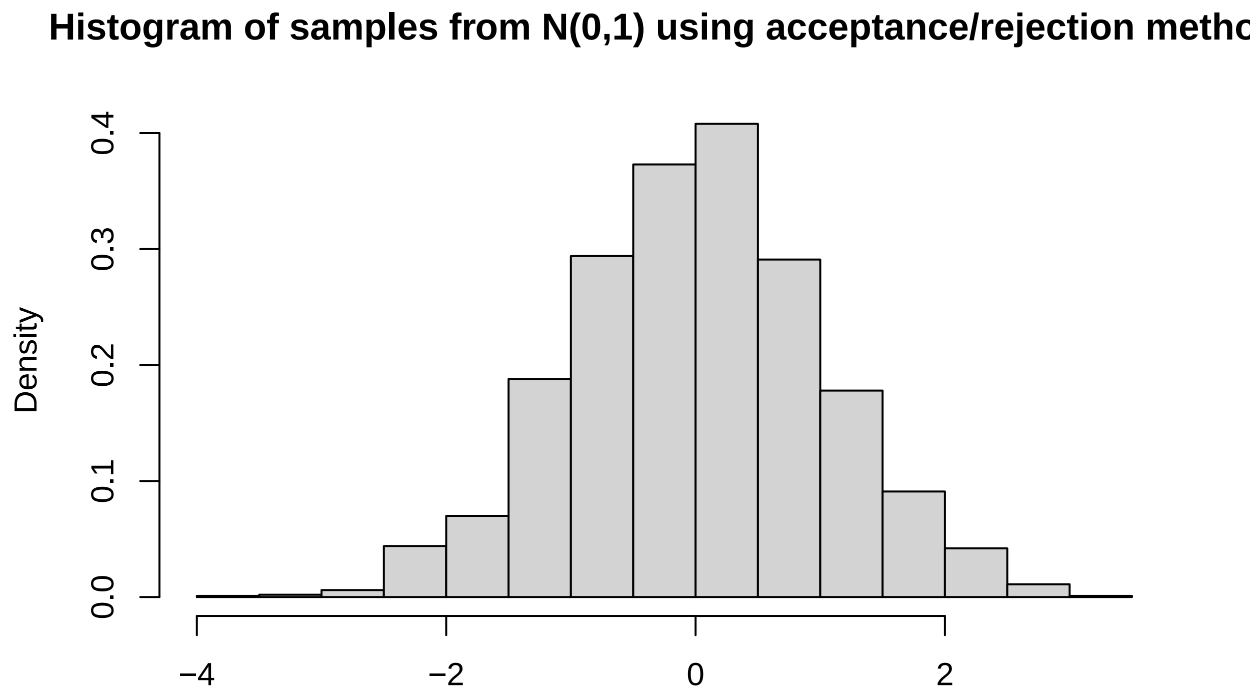


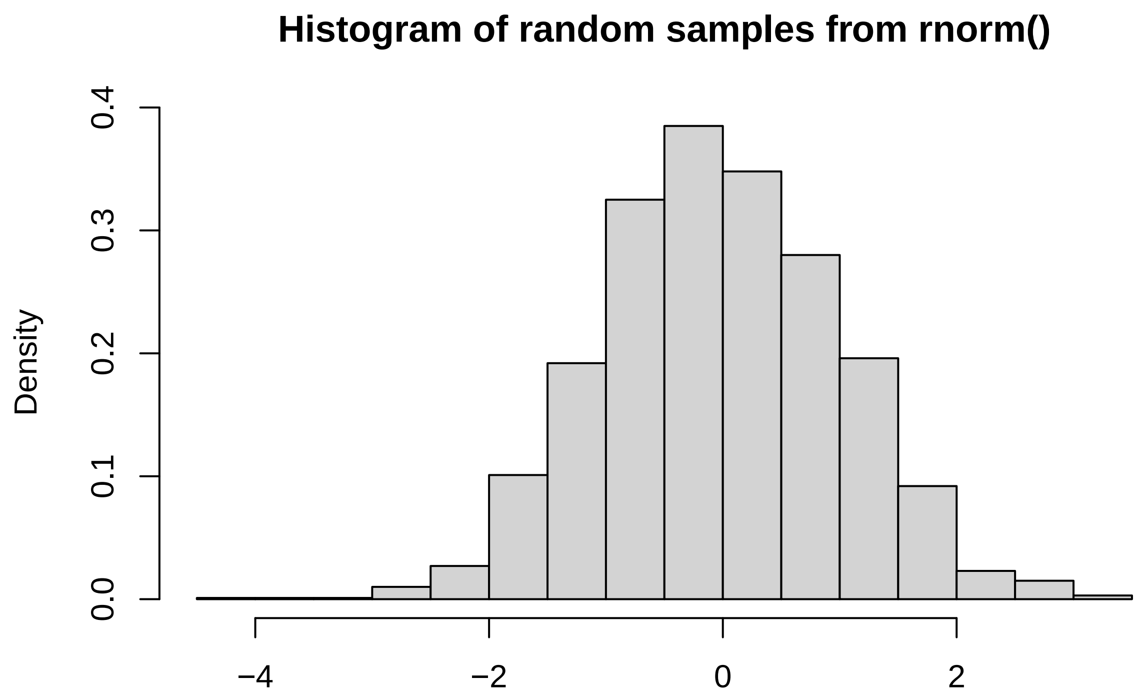
### Majorizing constant. The majorizing constant is calculated by calculating values from both the target density, N(0,1), and the proposal density, DE(0,1) over a reasonable grid. In this case x=[-30,30]. These values as then divided by each other. the majorizing constant is then the maximum value that this produces. This ensures that every value from the proposal density is above or equal to the target density. This is displayed in the graph, where every part of the blue line (target density * c) is above the line of the target density (black).

Acceptance/rejection method.

Here, the target density is N(0,1) and target density is DE(0,1). The calculate random samples from N(0,1) a random draw (y) from DE(0,1) was made using inverse CDF method. This draw was then used as an input to the target density and the majorizing density (itself). A random draw (u) from unif(0,1) was also made. If **u*DE(0,1)(y) < N(0,1)(y)**, then** y** is accepted as a draw from N(0,1).

```
## Mean rejection rate:  0.2965
## ER:  0.2396418
## Mean:  0.01586899
## Variance:  1.011437
```





The expected rejection rate is calculated by **1-(1/major_const)** and is lower than the measured rejection rate. The mean and variance is very close to the desired 0 and 1 that the target density has.

Comparing samples from acceptance/rejection and random samples from N(0,1), the histograms look very similar. The peak it at the same value and at x=0 and drops out at the same values. It can be concluded that the method works well!

Include all code for this report

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, message=FALSE)
# Include packages here
library(poweRlaw)


func = function(x, c){
  res = c/(sqrt(2*pi))*exp(-c^2/(2*x))*x^(-3/2)*1
  return(res)
}

power_func = function(x, alpha, tmin){
  res = c()
  for(i in 1:length(x)){
    ones = if(x[i] >= tmin) 1 else 0
    f = (alpha-1)/tmin*(x[i]/tmin)^(-alpha)* ones
    res = c(res, f)
  }
  return(res)
}

xgrid = seq(0.001, 100, length.out = 200)

c = 4 # Will vary throughout the lab

# Majorizing constant
y = func(seq(0,100, length.out = 200), c)
major_const = max(y)

alpha = 1.3 # Pick a good value
tmin = 3 # Fixed for entire lab
f_val = func(xgrid, c)
power_val = power_func(xgrid, alpha, tmin)

grid = seq(0.001, 1000, length.out = 10000)
vals = func(grid, c)
pos = sum(vals[1:(10000*(tmin/1000))])/sum(vals) # Density between 0 to 1, meaning probability for valu

plot(x = xgrid, y = f_val, type='l', col="black", ylab = "y-values", ylim=c(0, 0.05), main="f and fp")
lines(x = xgrid, y = power_val, type='l', col="red")


acc_rej = function(c, tmin, alpha, p){
  fx = func(seq(tmin,10000, length.out = 1000), c)
  fy = power_func(seq(tmin,10000, length.out = 1000), alpha, tmin)
  major_const2 = max(fx/fy)
  major_const1 = max(func(seq(0.001,10000, length.out = 1000), c))
  reject = 0

  while (1<2) {
    choice = rbinom(1,1,p)
    if (choice == 1){
      y = runif(1, 0, tmin)
      major_dens = 1
      major_const = major_const1

    } else{
      y = rplcon(n=1, xmin=tmin, alpha=alpha)
      major_dens = power_func(y, alpha, tmin)
      major_const = major_const2

    }
    u = runif(1, 0,1)
    if(u*major_dens*major_const <= func(y, c)){
      return(c(y, reject))

    } else {
      # Rejected!
      reject = reject+1
    }
  }
}

n_samples = 1000

cs = c(1.5, 2, 3, 4, 5)
#cs = c(1.5, 2)
grid = seq(0.001, 1000, length.out = 10000)
for (j in cs){
  Sys.sleep(3)
  print(j)
  vals = func(grid, j)
  pos = sum(vals[1:(10000*(tmin/1000))])/sum(vals)

  samples = matrix(data = 0, nrow = n_samples, ncol = 2)
  for(i in 1:n_samples){
    sam = acc_rej(j, tmin, alpha, pos)
    samples[i,] = sam
  }

  mean = mean(samples[,1])
  var = var(samples[,1])
  rrate = sum(samples[,2])/n_samples
  cat('\nc = ', j, 'gives: \nMean rejection rate: ', rrate, '\nMean: ', mean, '\nVariance: ', var)
  hist(samples[,1],breaks=(max(samples[,1])-min(samples[,1]))/1, freq = FALSE, xlim=c(0,100), main="His
  hist(samples[,2],breaks=100, freq = TRUE, main="Hisogram of rejections", xlab="")
}
# Inverse CDF method
inv_de_pos = function(y){
  return(-log(2*(1-y)))
}
inv_de_neg = function(y){
  return(log(2*y))
}

de_01 = function(x){
  return((1/2)*exp(-abs(x)))
}

rde01 = function(n){
  uni = runif(n, 0, 1)
  samples = c()
  for(i in 1:n){
    if(uni[i]>0.5){
      samples = c(samples, inv_de_pos(uni[i]))
    } else {
      samples = c(samples, inv_de_neg(uni[i]))
    }
  }
  return(samples)
}

n_iter = 10000
```

```r
samples = rde01(n_iter)
hist(samples, freq=FALSE, breaks = 100, main='Histogram of samples using inverse CDF method')

xgrid = seq(-10,10, length.out = 100)
plot(x = xgrid, y = de_01(xgrid), type='l', col='blue', main='Plot of PDF of DE(0,1)', ylab='Prob')
# Target density: N(0,1)
# Majorizing density: DE(0,1)

# Calculating majorizing constant
grid = seq(-30,30, length.out = 500)
fx = dnorm(grid, 0, 1)
fy = de_01(grid)
major_const = max(fx/fy)

plot(x=grid, y=fx, type='l', xlim=c(-6, 6), ylim=c(0, 0.7), ylab='Prob')
lines(x=grid, y=fy, type='l', col='red')
lines(x=grid, y=fy*major_const, type='l', col='blue')
legend("topleft", c("Target density N(0,1)","Proposal density", "Proposal density * C"), col=c("black",

acc_rej_n01 = function(major_const){

  reject = 0
  er = (1-(1/major_const))

  while (1<2) {
    # Draw from DE(0,1) (Majorizing density)
    y = rde01(1)
    major_dens = de_01(y)

    u = runif(1, 0,1)
    if(u*major_dens*major_const <= dnorm(y)){
      return(c(y, reject, er))
    } else {
      # Rejected!
      reject = reject+1
    }
  }
}

n_samples = 2000

samples = matrix(nrow = n_samples, ncol = 3)
for(i in 1:n_samples){
  sam = acc_rej_n01(major_const)
  samples[i,] = sam
}

mean = mean(samples[,1])
var = var(samples[,1])
rrate = sum(samples[,2])/n_samples
cat('Mean rejection rate: ', rrate, '\nER: ', mean(samples[,3]), '\nMean: ', mean, '\nVariance: ', var)
hist(samples[,1], freq = FALSE, main="Histogram of samples from N(0,1) using acceptance/rejection metho
#hist(samples[,2], freq = TRUE, main="Hisogram of rejections", xlab="")
```

```r
# Generating random numbers from N(0,1)
rand_samples = rnorm(n_samples, 0, 1)
hist(rand_samples, freq = FALSE, main="Histogram of random samples from rnorm()", xlab="")
```