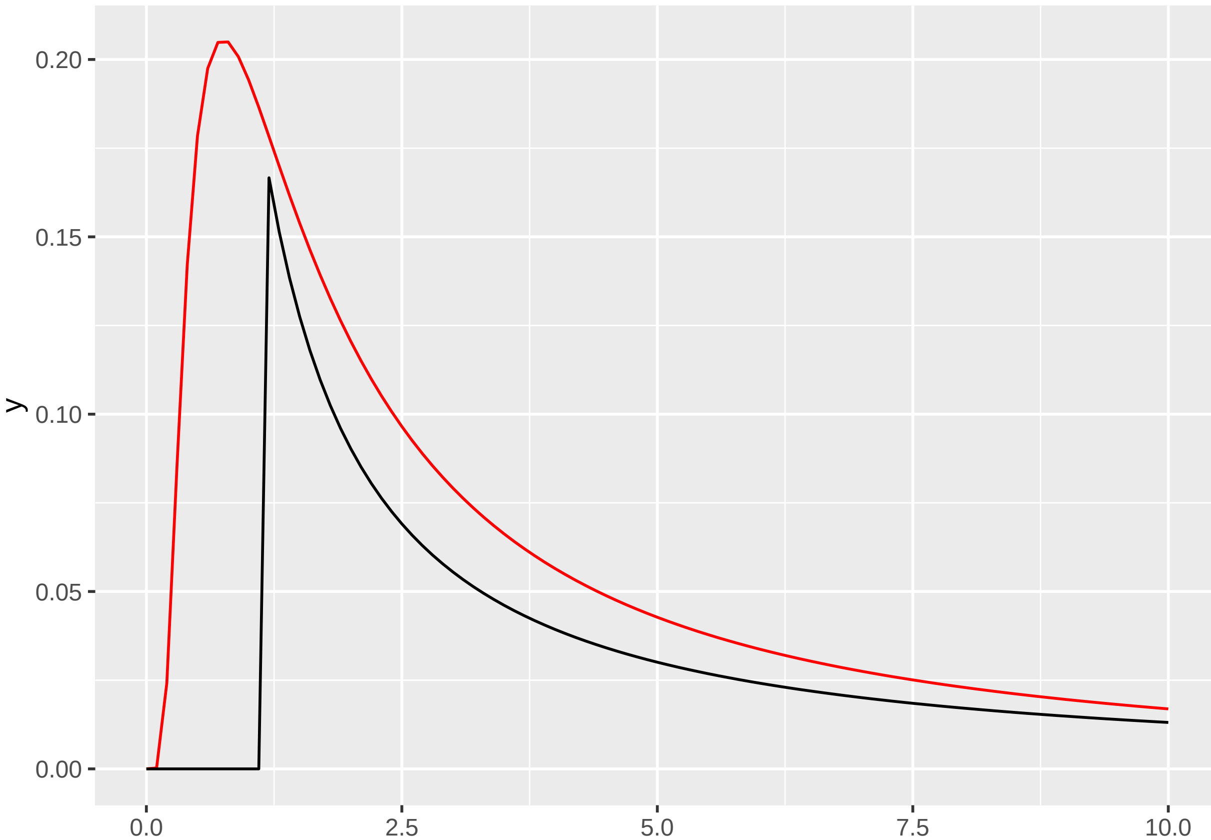
Lab 3

Group 4

Question 1

1.1

```r
f_x <- function(x, c) {
  return(c*(sqrt(2*pi))^-1 * exp(1)^(-(c^2)/(2*x)) * x^(-3/2))
}

f_y <- function(x) {
  y <- c()

  for(i in 1:length(x)) {
    if(x[i] < Tmin) y[i] <- 0
    else y[i] <- ((alpha-1)/Tmin) * (x[i]/Tmin)^-alpha
  }

  return(y)
}

c <- 1.5
alpha <- 1.2
Tmin <- 1.2

ggplot() +
  xlim(1e-04, 10) +
  geom_function(fun = f_x, args = list(c = c), colour = "red") +
  geom_function(fun = f_y, colour = "black")
```



The *one-sided strictly stable distribution of order 1/2* have the support $(0, \infty)$, while the power-law distribution have the support (T_{min}, ∞). This means that random values greater than 0, but smaller than T_{min} can not be generated. This problem can be solved by using another distribution for values greater than 0, but smaller than T_{min}.

A power-law distribution with parameters $\alpha = 1.2$ and $T_{min} = 1.2$ seem to result in a distribution that is quite similar to a one-sided strictly stable distribution of order 1/2 with parameter $c = 1.5$ for values greater than $T_{min} = 1.2$, and can therefore be used in the acceptance-rejection algorithm. For values less than $T_{min} = 1.2$, a uniform distribution with parameters $a = 0$ and $b = 1.2$ will be used to appproximate the target distribution.

The majorizing constants c is chosen by generating a grid of probable x values, and then calculating:

$$c = \max\left(\frac{f_x(x)}{f_y(x)}\right)$$

1.2

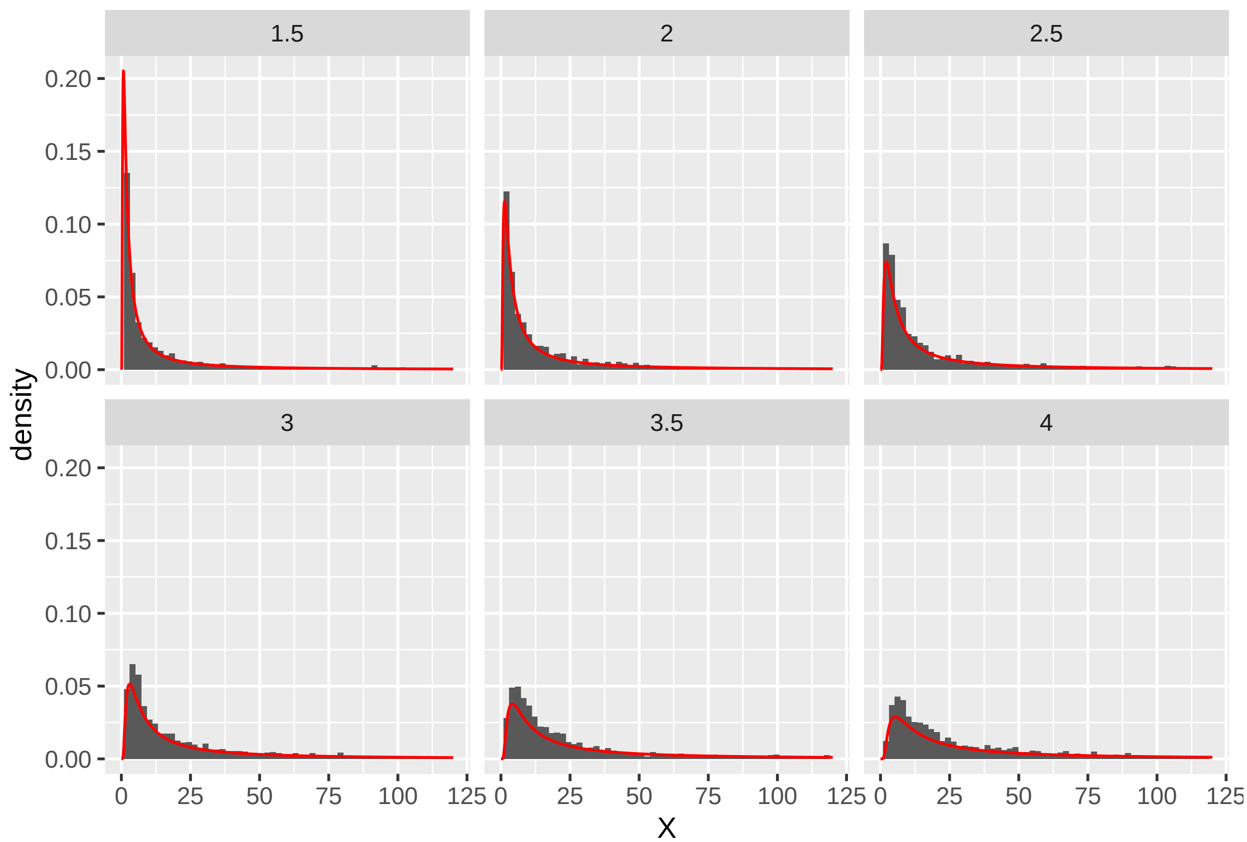
```
sampler <- function(n, c) {
  samples <- data.frame(X = NA, Accept = NA, Dist = NA)
  counter <- 1
  majorizing_constant <- max(f_x(seq(Tmin, 100, by = 0.01), c = c) /
                               f_y(seq(Tmin, 100, by = 0.01)))
  majorizing_constant_unif <- max(f_x(seq(0.001, Tmin, by = 0.01), c) /
                                    dunif(seq(0.001, Tmin, by = 0.01), 0, Tmin))
```

```
  tot <- f_x(seq(0.001, 1000, by = 0.001), c)
  b_Tmin <- tot[which(seq(0.01, 100, by = 0.001) <= Tmin)]
  prob <- sum(b_Tmin)/sum(tot)
  for(i in 1:n) {
    X <- NA
    while(is.na(X)) {
      U <- runif(1)
      r_num <- sample(0:1, 1, prob = c(1-prob, prob))
      if(r_num == 1) {
        Y <- runif(1, 0, Tmin)
        if (U <= f_x(Y, c) / majorizing_constant_unif) {
          X <- Y
          samples[counter,1] <- X
          samples[counter,2] <- 1
          samples[counter,3] <- "U"
        }
        else {
          samples[counter,1] <- Y
          samples[counter,2] <- 0
          samples[counter,3] <- "U"
        }
      }
      else {
        Y <- rplcon(1, xmin = Tmin, alpha = alpha)
        if(U <= f_x(Y, c)/(majorizing_constant*f_y(Y))) {
          X <- Y
          samples[counter,1] <- X
          samples[counter,2] <- 1
          samples[counter,3] <- "PL"
        }
        else {
          samples[counter,1] <- Y
          samples[counter,2] <- 0
          samples[counter,3] <- "PL"
        }
      }
      counter <- counter+1
    }
  }
  return(samples)
}

samples_1_2 <- sampler(1000, c)
```

1.3

```
c_sampler <- function(c_vec) {
  c_list <- list()
  for(i in 1:length(c_vec)) {
    c_list[[i]] <- sampler(n = 2e3, c = c_vec[i])
  }
```

```r
  return(c_list)
}

c_vec <- c(1.5, 2, 2.5, 3, 3.5, 4)

c_list <- c_sampler(c_vec)
c_list <- mapply(cbind, c_list, "c" = c_vec, SIMPLIFY = FALSE)
c_df <- do.call("rbind", c_list)

dens_func <- data.frame(x = rep(seq(1e-15, 120, length.out = 1000),
                                length(unique(c_df$c))), c = rep(unique(c_df$c),
                                                                  each = 1000))
dens_func$val <- f_x(dens_func$x, dens_func$c)

ggplot() +
  geom_histogram(data = c_df[which(c_df$Accept == 1),], aes(x = X, y = stat(density)),
                 bins = 60) +
  geom_line(data = dens_func, aes(x = x, y = val), col = "red") +
  xlim(c(0,120)) +
  facet_wrap(c~.)
```




```r
# Mean
kable(setNames(aggregate(X ~ c, data = c_df[which(c_df$Accept == 1),], FUN = mean),
               c("c", "Mean")))
```

c	Mean
1.5	5880.397
2.0	22033.166
2.5	551971.395
3.0	1925.064
3.5	9999.087
4.0	230042.397

```r
# Variance
kable(setNames(aggregate(X ~ c, data = c_df[which(c_df$Accept == 1),], FUN = var),
               c("c", "Variance")))
```

c	Variance
1.5	2.810432e+10
2.0	2.505816e+11
2.5	5.929488e+14
3.0	4.758368e+08
3.5	3.589490e+10
4.0	5.197548e+13

```r
# Rejection rate
kable(setNames(aggregate(as.numeric(Accept) ~ c, data = c_df, function(x) {
  1-(sum(x)/length(x))
}), c("c", "Rejection rate")))
```

c	Rejection rate
1.5	0.4295493
2.0	0.4410285
2.5	0.4557823
3.0	0.4720169
3.5	0.5054402
4.0	0.5312866

Outlying numbers can have a great effect on the mean and variance which makes it hard to investigate how these parameters depends on c. If outlying numbers are excluded it appears as if both the mean and the variance increases as the value of c increases.

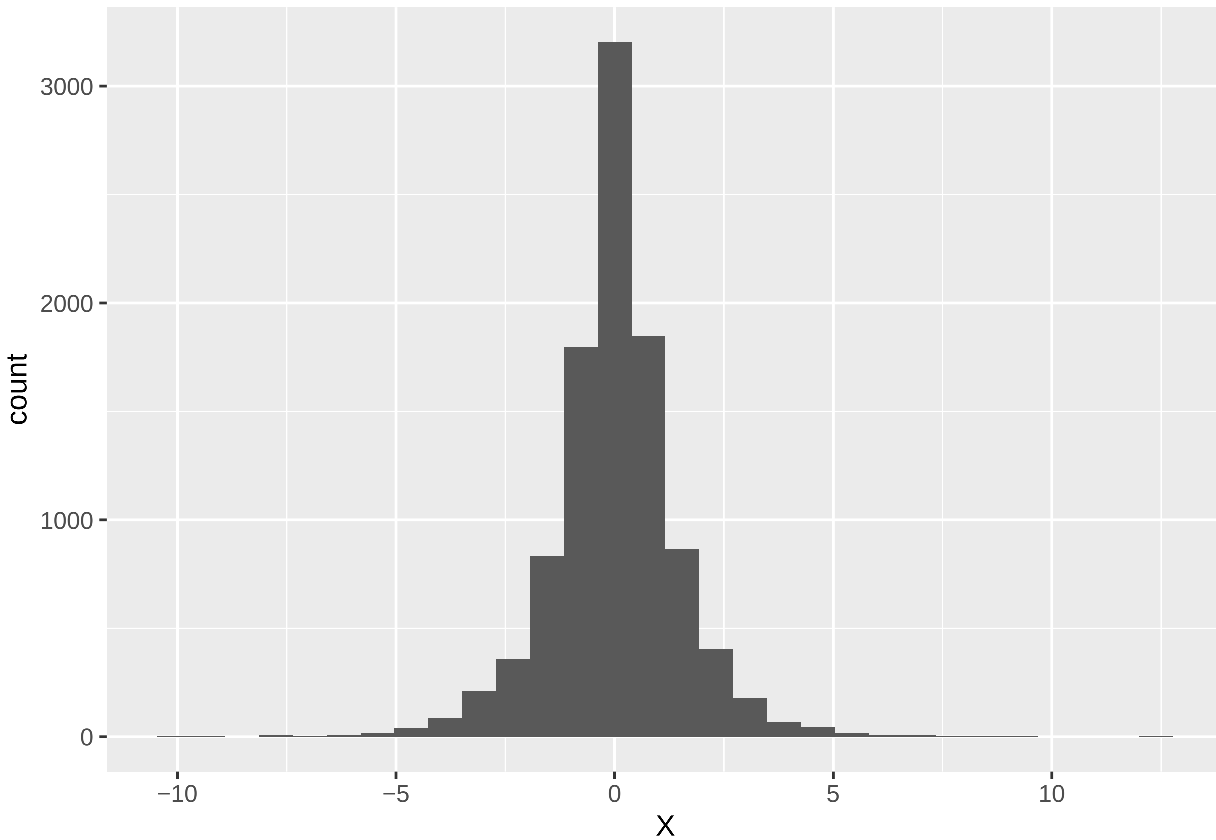
The rejection rate increases as the value of c increases. The larger c gets, the less similar the power-law distribution with parameters $\alpha = 1.05$ and $T_{min} = 1.2$ will be to the one-sided strictly stable distribution of order 1/2, and more samples will fall outside of the target function.

Question 2

2.1

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

```
}

laplace_sampling <- function(n, mu, alpha) {
  u <- runif(n, min = 0, max = 1)
  return(laplace_inv_cdf(u, mu, alpha))
}

samples <- data.frame(X = laplace_sampling(1e4, 0, 1))

ggplot(samples, aes(x = X)) +
  geom_histogram(bins = 30)
```



A CDF $F(x)$ gives the probability that a random variable X is smaller or equal to a specific value x:

$F(x) = Pr(X \leq x)$

A CDF is derived by integrating the PDF:

$F(x) = \int_{-\infty}^{x} f(s)ds$

$= \int_{-\infty}^{x} \frac{\alpha}{2} \exp(-\alpha|s-\mu|)ds$

$= \frac{1}{2} + \frac{1}{2}\text{sign}(x-\mu)(1-\exp(-\alpha|x-\mu|)$

An inverse CDF gives the value of x for which $F(x)$ will return a certain probability p:

$F^{-1}(p) = x$

As the name suggest, the inverse CDF is obtained by taking the inverse of the CDF:

$F^{-1}(p) = \mu - \frac{1}{\alpha}\text{sign}(p - 0.5)\ln(1 - 2|p - 0.5|)$

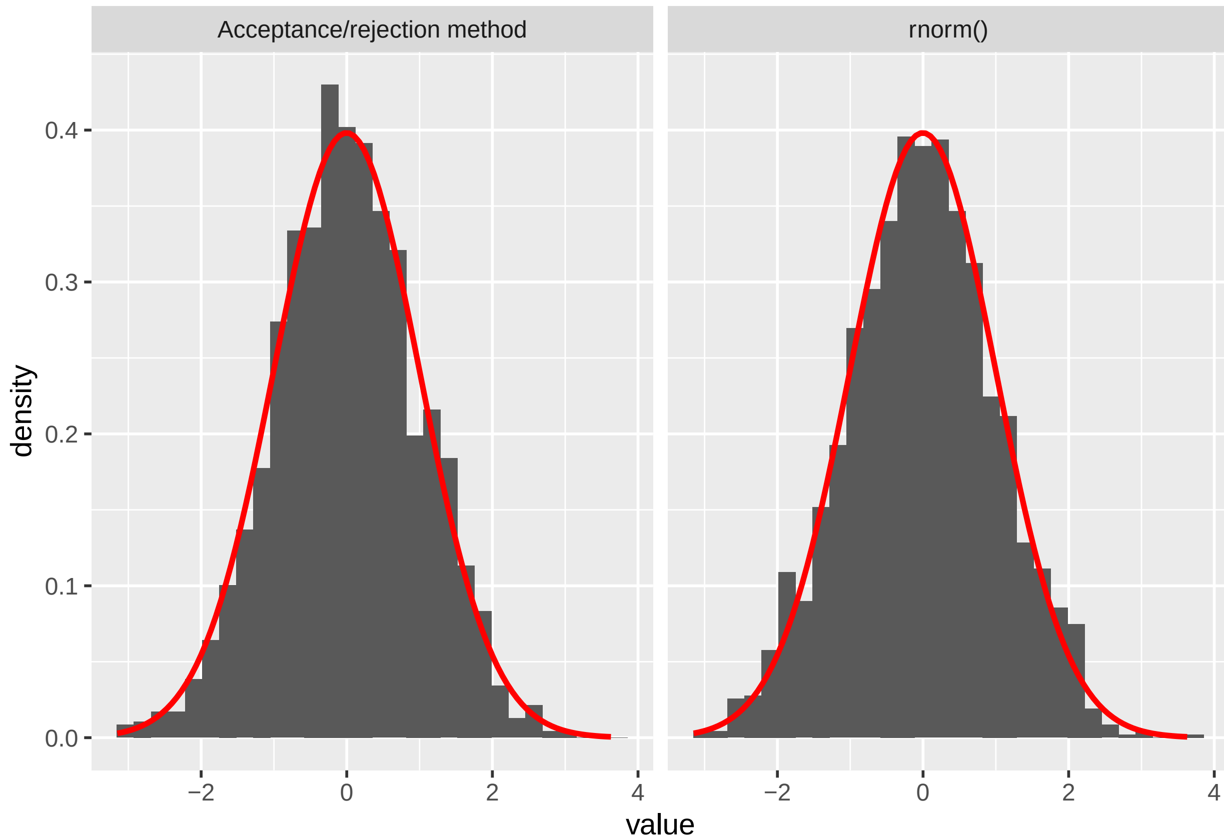
Random numbers can be generated from the Laplace distribution by generating a random number U from $Unif(0, 1)$ and then inserting these into the inverse CDF function for the Laplace distribution.

The histogram looks obtained by plotting 1000 random numbers from the Laplace distribution looks reasonable. A Laplace distribution is expected to have it's highest density around 0, and then decrease exponentially in both direction.

2.2

```
laplace_pdf <- function(x, mu, alpha) {
  return((alpha/2) * exp(-alpha*abs(x-mu)))
}

norm_sampler <- function(n, majorizing_constant) {
  samples <- data.frame(X = NA, Accept = NA)
  counter <- 1
  for(i in 1:n) {
    X <- NA
    while(is.na(X)) {
      Y <- laplace_sampling(1, 0, 1)
      U <- runif(1, min = 0, max = 1)
      if(U <= dnorm(Y, mean = 0, sd = 1)/(majorizing_constant*laplace_pdf(Y, 0, 1))) {
        X <- Y
        samples[counter,1] <- X
        samples[counter,2] <- 1
      }
      else {
        samples[counter,1] <- Y
        samples[counter,2] <- 0
      }
      counter <- counter+1
    }
  }
  return(samples)
}

majorizing_constant <- max(dnorm(seq(-10, 10, by = 0.0001), 0, 1) /
                             laplace_pdf(seq(-10, 10, by = 0.0001), 0, 1))

norm_samples <- norm_sampler(n = 2000, majorizing_constant = majorizing_constant)

df2 <- data.frame(AR = norm_samples[which(norm_samples$Accept == 1),1],
                  rnorm = rnorm(2000, mean = 0, sd = 1))

df2_long <- pivot_longer(df2, cols = c(1,2))

ggplot(df2_long, aes(x = value)) +
  geom_histogram(aes(y = stat(density)), bins = 30) +
  facet_wrap(name ~ ., labeller = labeller(name = c("AR" = "Acceptance/rejection method",
                                                      "rnorm" = "rnorm()"))) +
  stat_function(fun = dnorm,
```

```r
              args = list(mean = mean(df2_long$value), sd = sd(df2_long$value)),
              col = "red", lwd = 1)
```




```r
# Average rejection rate
1-(sum(norm_samples$Accept)/nrow(norm_samples))
```

```
## [1] 0.2409867
```

```r
# Expected rejection rate
1-(1/c)
```

```
## [1] 0.3333333
```

To generate values from a random variable $X \sim N(0,1)$, the following steps were performed:

1. The PDF of the Laplace distribution is implemented as a function with three parameters - x, mu and $alpha$.
2. A random number Y is drawn from the Laplace distribution using the inverse CDF method implemented in step 1.
3. A random number U is drawn from a uniform distribution $Unif(0,1)$.
4. The value of the random variable Y is put into the built-in function *dnorm()* and into the probability density function of the Laplace distribution.
5. If $U \leq \frac{f_x(Y)}{cf_y(Y)}$, the random variable X is set to Y, if not step 2-4 is re-run.

6. Step 2-5 is repeated until enough random numbers have been generated.

The constant c can be chosen by generating a grid of probable x values, and then calculating:

$$c = \max\left(\frac{f_x(x)}{f_y(x)}\right)$$

In this case this gives the value $c = 1.315489$.

The expected rejection rate ER is:

$$1 - \frac{1}{c} = 1 - \frac{1}{1.315489} = 0.2398265$$

The expected rejection rate ER is very close to the obtained average rejection rate R which is 0.2239038.