Computational Statistics - Lab 03

Annalena Erhard (anner218) and Maximilian Pfundstein (maxpf364)

2019-01-30

Contents

1 Question 1: Cluster Sampling

An opinion pool is assumed to be performed in several locations of Sweden by sending interviewers to this location. Of course, it is unreasonable from the financial point of view to visit each city. Instead, a decision was done to use random sampling without replacement with the probabilities proportional to the number of inhabitants of the city to select 20 cities. Explore the file `population.xls`. Note that names in bold are counties, not cities.

Task: Import necessary information to R.

Municipality	Population
Botkyrka	81195
Danderyd	31150
Ekerö	25095
Haninge	76237
Huddinge	95798
Järfälla	65295

Task: Use a uniform random number generator to create a function that selects 1 city from the whole list by the probability scheme offered above (do not use standard sampling functions present in R).

```
get_city_by_urn_wo = function(city_pool) {

  # We take the cumulative sum and then runif from 1 to max(cumulative sum).
  # This way we respect the proportions. As we need every intermediate result,
  # we use a loop
  cumulative_pop_sum = 0

  for (i in 1:nrow(city_pool)) {
    cumulative_pop_sum = cumulative_pop_sum + city_pool$Population[i]
    city_pool$CumSum[i] = cumulative_pop_sum
  }

  # Now we get a random value between 1 to max(cumulative sum). As larger muni-
  # cipalities have larger ranges, this works as expected
  selection =
    floor(runif(n = 1, min = 1, max = city_pool$CumSum[nrow(city_pool)]))
```

```
  # Return the first city which has a greater CumSum than the selection
  return(city_pool[city_pool$CumSum > selection,][1, c(1, 2)])
}
```

Task: Use the function you have created in step 2 as follows:

a. Apply it to the list of all cities and select one city
b. Remove this city from the list
c. Apply this function again to the updated list of the cities
d. Remove this city from the list
e. ... and so on until you get exactly 20 cities.

Answer: We will combine all of these steps in one function. We're lazy.

```
get_n_cities = function(data, n) {

  # Create a copy to not touch the original data.
  city_pool = data
  selected_cities = data.frame()

  # As long as we don't have n samples, get one and remove it from the pool,
  # as we sample without replacement
  while(nrow(selected_cities) < n) {
    selected_city = get_city_by_urn_wo(city_pool)
    selected_cities = rbind(selected_cities, selected_city)
    city_pool = city_pool[!rownames(city_pool) %in% rownames(selected_cities),]
  }

  return(selected_cities)
}

sample = get_n_cities(data, 20)
```

Task: Run the program. Which cities were selected? What can you say about the size of the selected cities?

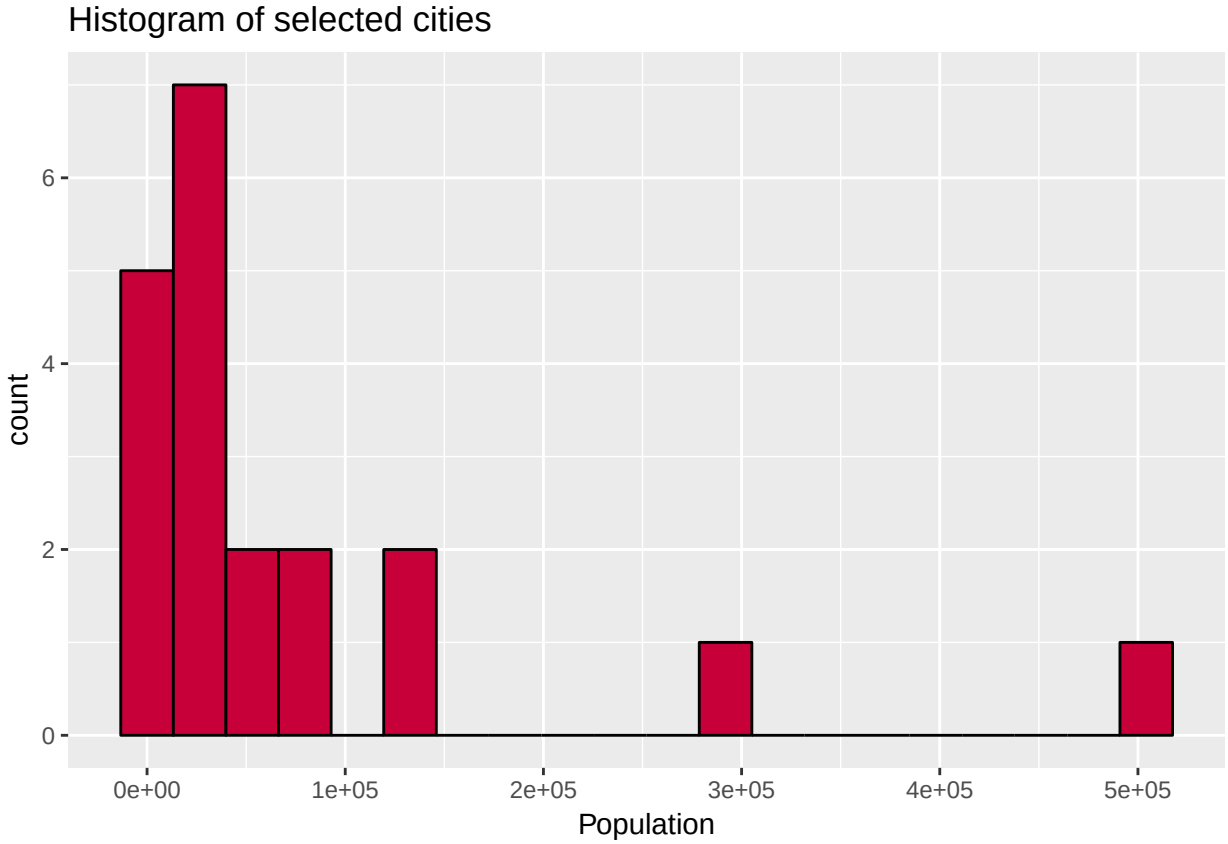
Answer: The following cities were selected:

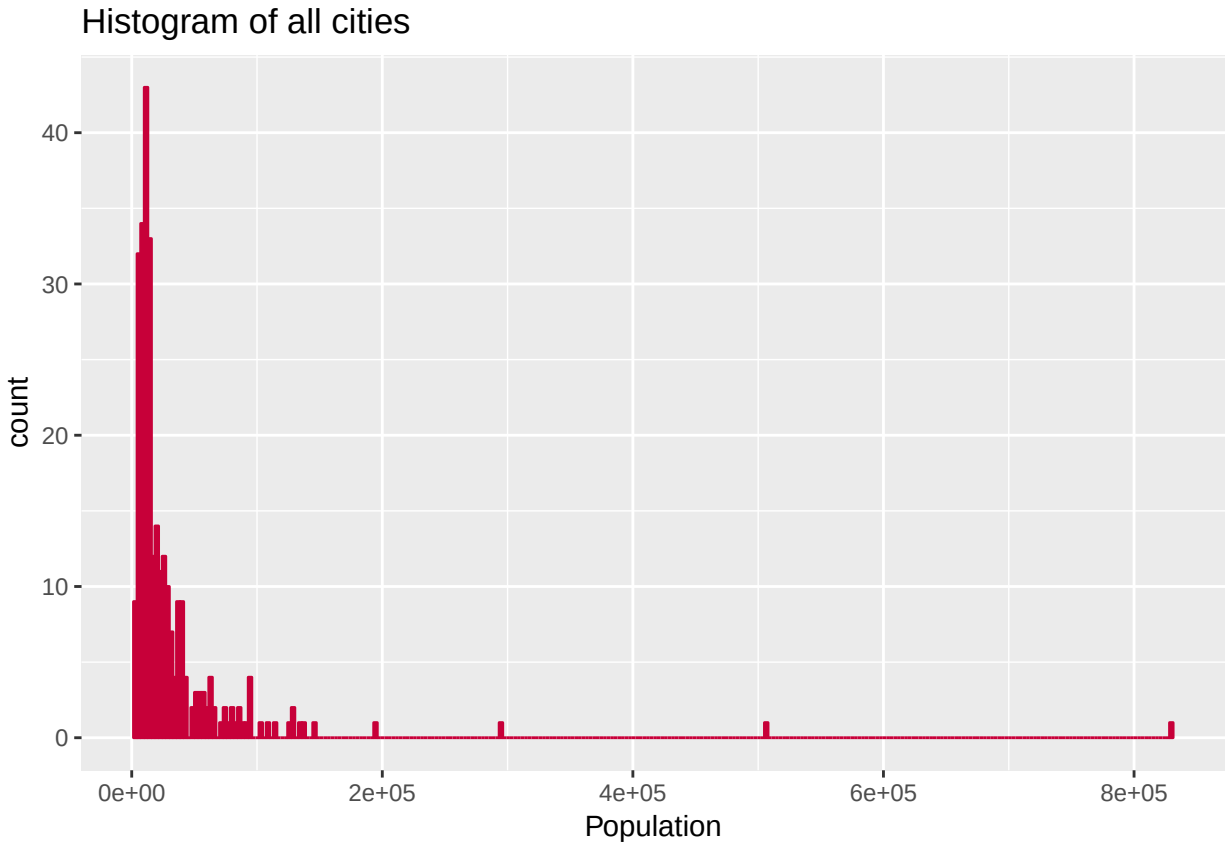
```
##     Municipality Population
## 248     Kramfors      19214
## 200    Askersund      11307
## 30       Knivsta      14477
## 50    Norrköping     129254
## 112        Malmö     293909
## 59      Gislaved      29212
## 19        Tyresö      42602
## 128 Östra Göinge      13526
## 146     Göteborg     507330
## 26     Österåker      39173
## 127  Örkelljunga       9639
## 150      Härryda      34007
## 134      Varberg      57439
## 281    Haparanda      10112
## 62     Jönköping     126331
## 31         Tierp      20044
## 277     Arjeplog       3143
## 285        Luleå      73950
```

```
## 191     Karlstad      84736
## 40     Oxelösund      11126
```

It can be seen, that mostly cities with a population greater than the mean (`32009.25`) were drawn.

Task : Plot one histogram showing the size of all cities of the country. Plot another histogram showing the size of the 20 selected cities. Conclusions?





Answer: We see that both histograms have a similar shape. This means that our sampling function is taking the size of the city into account and we created a similar distribution with a smaller subset. That is exaclty what we needed to perform the opinion pool.

2 Question 2: Different Distributions

The double exponential (Laplace) distribution is given by formula

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

Task: Write a code generating double exponential distribution $DE(0,1)$ from $Unif(0,1)$ by using the inverse CDF method. Explain how you obtained that code step by step. Generate `10000` random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

Answer: What we are going to do is we sample from `unif(0,1)` and take the results and put it into the quantile function of $DE(0,1)$. As we take the definition from Laplace Distribution (Wikipedia) for the quantile function we have to keep in mind that $\alpha = \frac{1}{b}$. Let's first define our quantile function with $\mu = 0$, $\alpha = 1$ and thus $b = \frac{1}{1} = 1$ as well:

$$Q(p) = F^{-1}(p) = \mu - b * sgn(p - 0.5) * ln(1 - 2|p - 0.5|)$$

With the above defined variables we obtain:

$$Q(p|\mu, b) \Rightarrow Q(p|0,1) = -sgn(p - 0.5) * ln(1 - 2|p - 0.5|)$$

Where `sgn` is the Sign Function (Wikipedia).

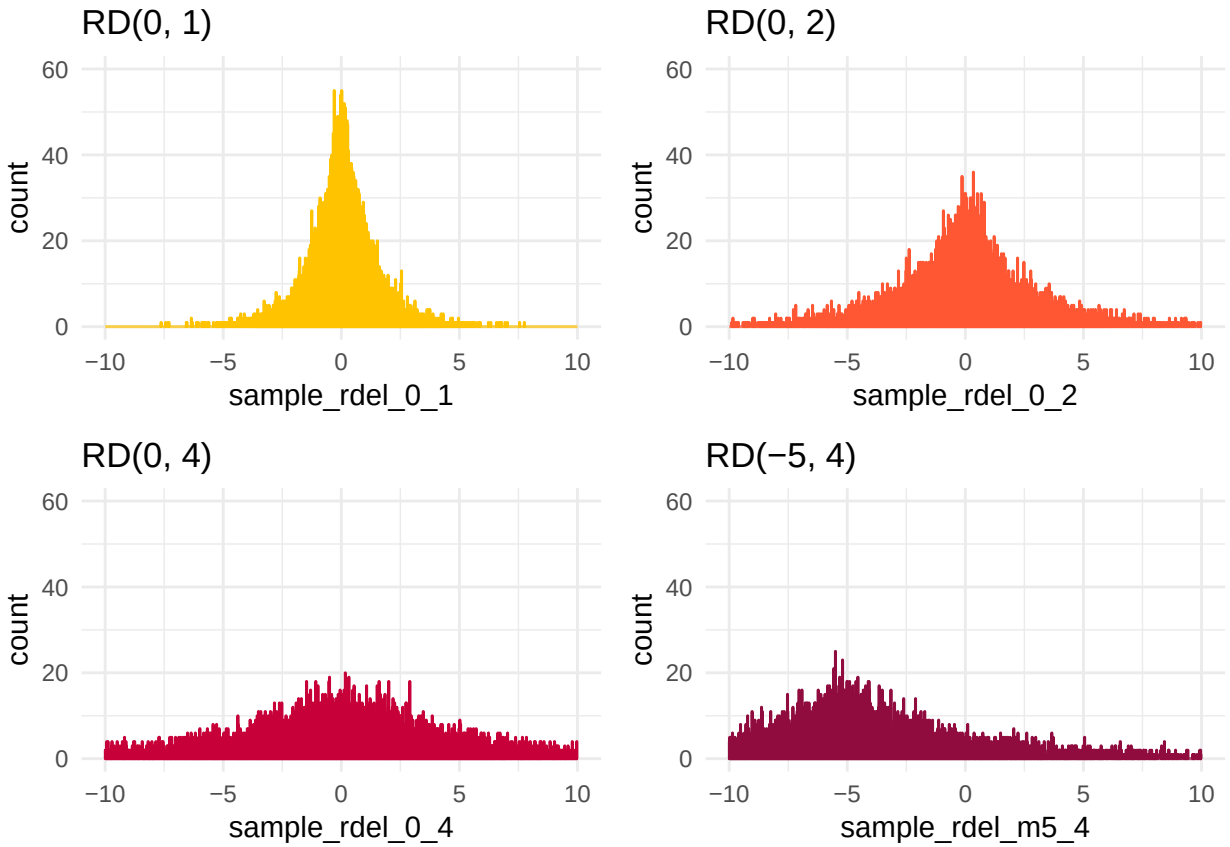
Let's implement those two functions as a start:

```
sgn = function(x) {
  if (x < 0) return(-1)
  if (x > 0) return( 1)
  return(0)
}

qdel = function(p, mu = 0, b = 1) {
  if (p < 0 | p > 1) stop("p must be in range (0, 1)")
  return(mu - b * sgn(p-0.5) * log(1 - 2 * abs(p - 0.5)))
}
```

Next we implement a function for drawing n times.

```
rdel = function(n = 1, mu = 0, b = 1) {
  quantiles = runif(n = n, min = 0, max = 1)
  rdels = sapply(X = quantiles, FUN = qdel, mu = mu, b = b)
  return(rdels)
}
```

Let's look how the plot for `10000` random numbers from this distribution looks like:



The results look reasonable when compared to the original density function as the shape is what we would expect.

Task: Use the Acceptance/rejection method with $DE(0,1)$ as a majorizing density to generate $\mathcal{N}(0,1)$ variables. Explain step by step how this was done. How did you choose constant `c` in this method? Generate

2000 random numbers $\mathcal{N}(0,1)$ using your code and plot the histogram. Compute the average rejection rate `R` in the acceptance/rejection procedure. What is the expected rejection rate `ER` and how close is it to `R`? Generate 2000 numbers from $\mathcal{N}(0,1)$ using standard `rnorm()` procedure, plot the histogram and compare the obtained two histograms.

Answer: As we have not yet created a function for the PDF and CDF $DE(\mu,\alpha)$ we will do this now and compare it to the normal distribution. We well swap α with b to ensure consitency.

PDF:

$$f(X|\mu,\alpha) = \frac{1}{2b} e^{\frac{-|x-\mu|}{b}}$$

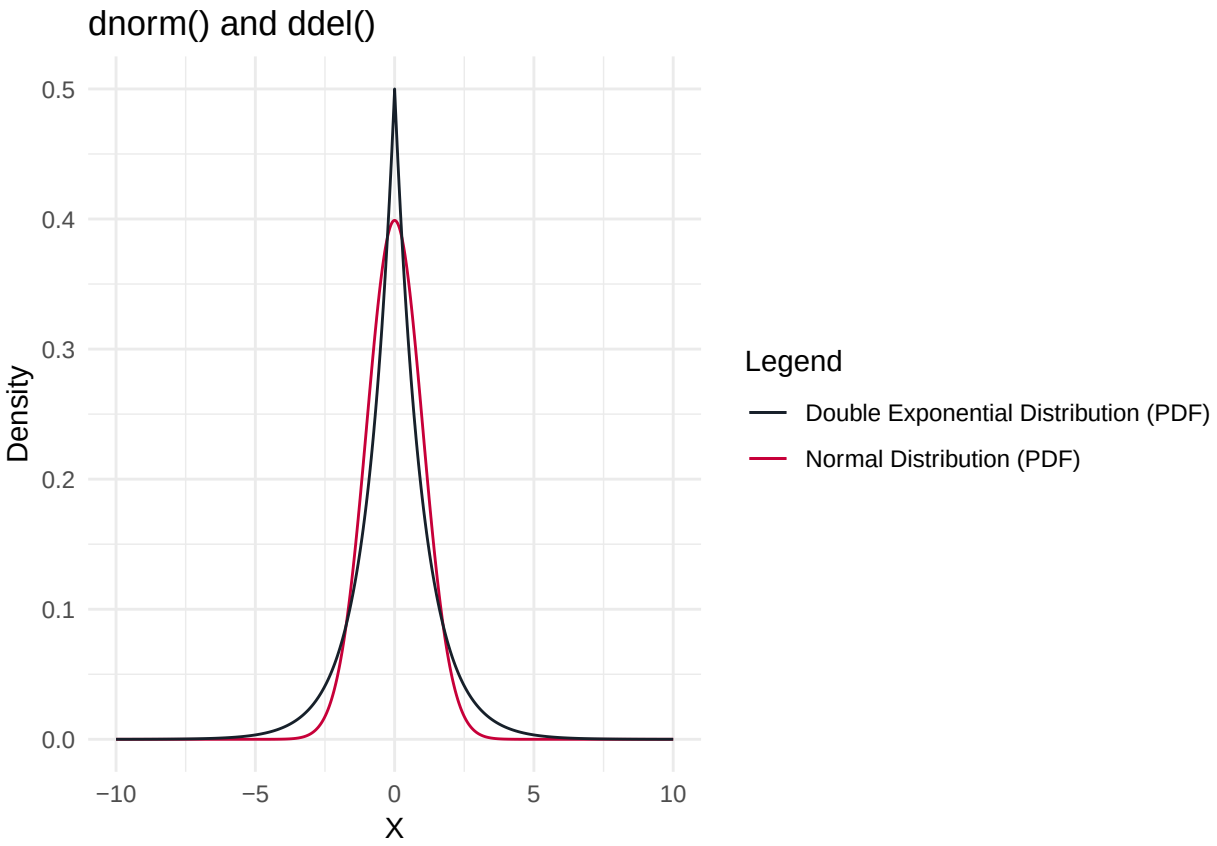
```
ddel = function(x = 1, mu = 0, b = 1) {
  return(1/(2*b) * exp(-abs(x-mu)/(b)))
}
```

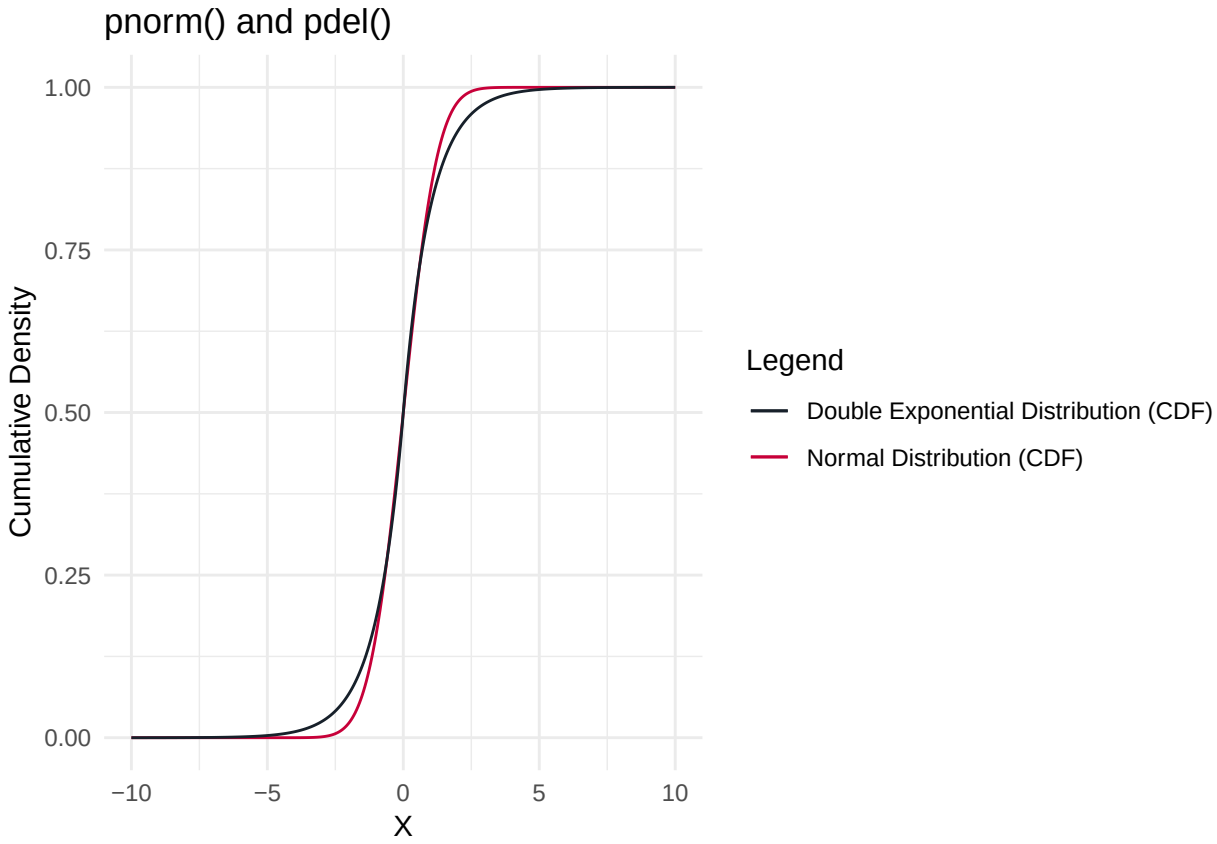
CDF:

$$DE(\mu,b) = F(X|\mu,\alpha) = \frac{1}{2} + \frac{1}{2} * sgn(x-\mu) * (1 - e^{-\frac{|x-\mu|}{b}})$$

```
pdel = function(x = 1, mu = 0, b = 1) {
  return(1/2 + 1/2 * sgn(x-mu) * (1 - exp(-(abs(x-mu))/(b))))
}
```

Let's plot `ddel()` with `dnorm()` and `pdel()` with `dnorm()`:



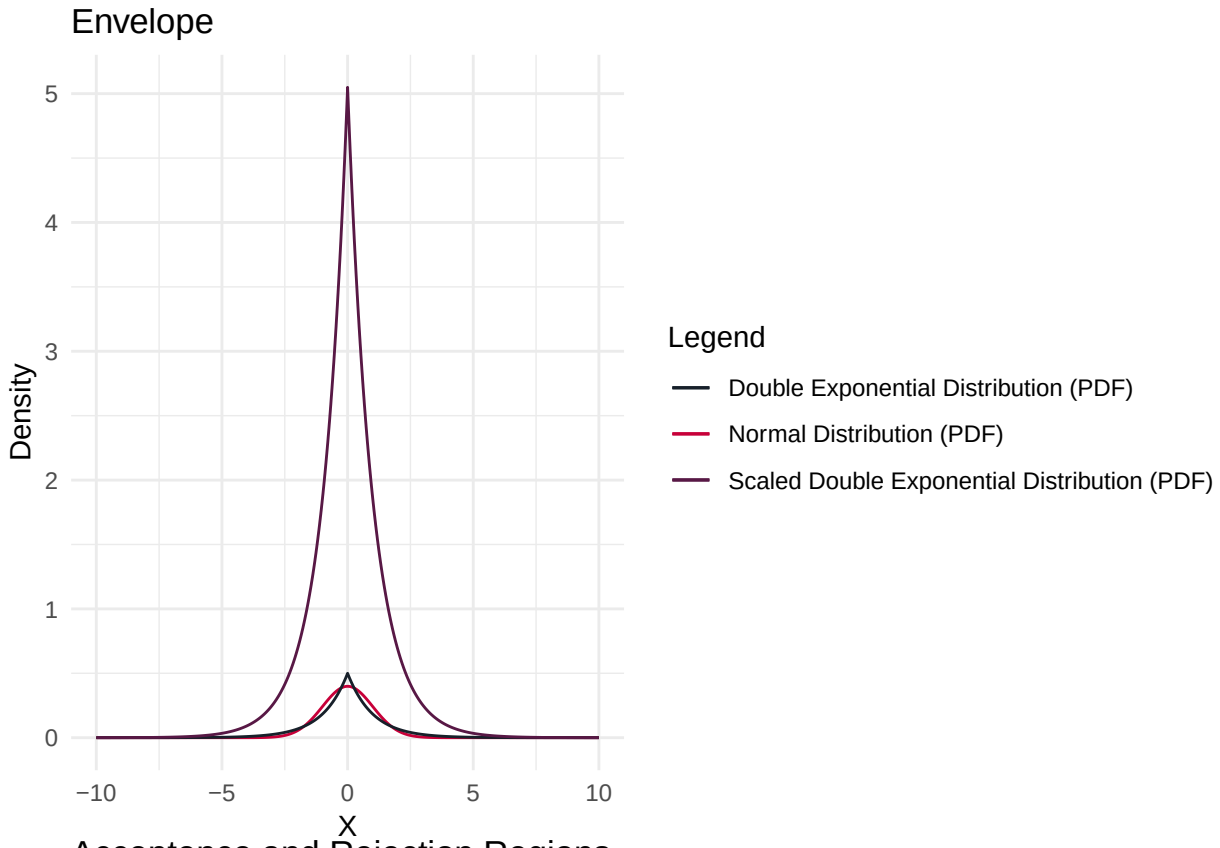


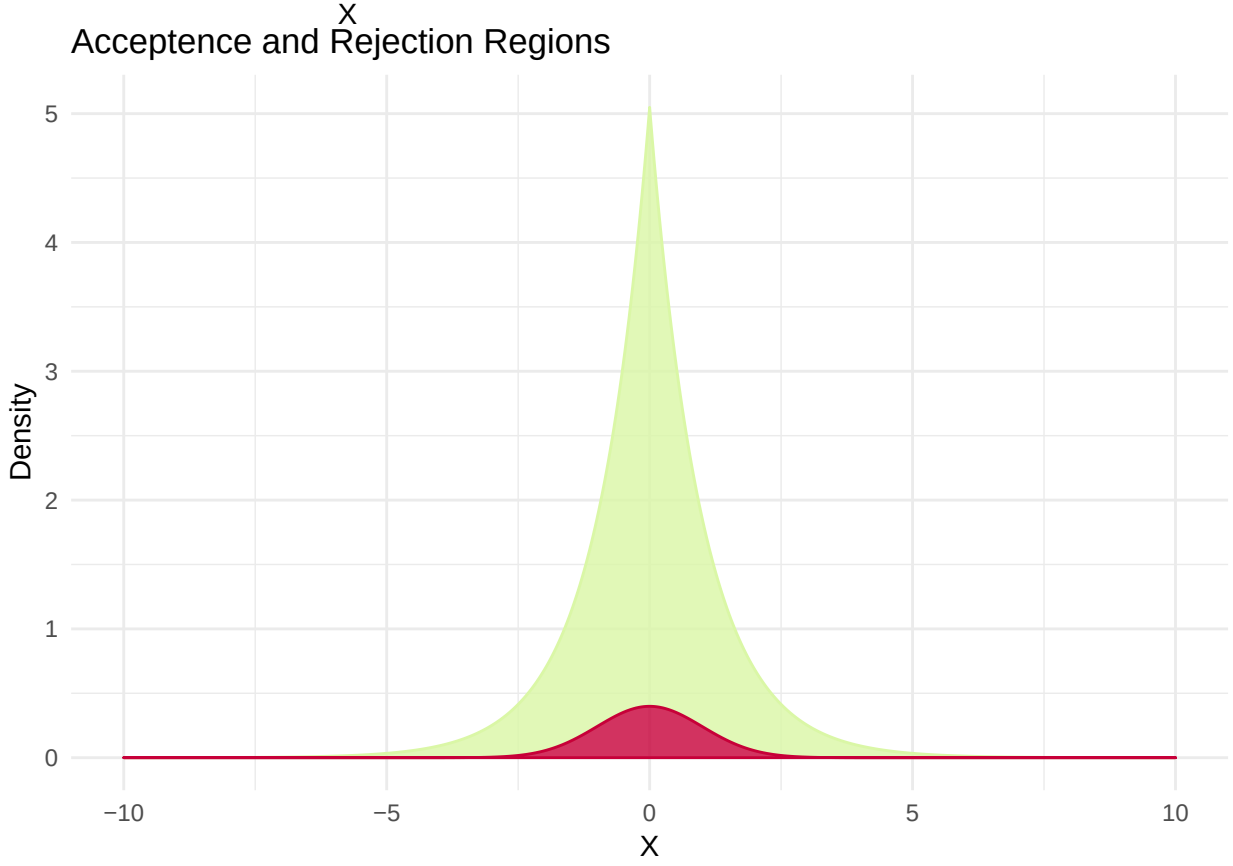
We need our sampling distribution to envelope our function to generate. Therefore we calcualte the fraction of $\frac{q(x)}{f(x)}$. This ensures our $\mathcal{N}(0,1)$ is covered and that $c \geq 1$ as defined by the rejection method.

```
c = max(dnorm_samples / pnorm_samples)
c
```

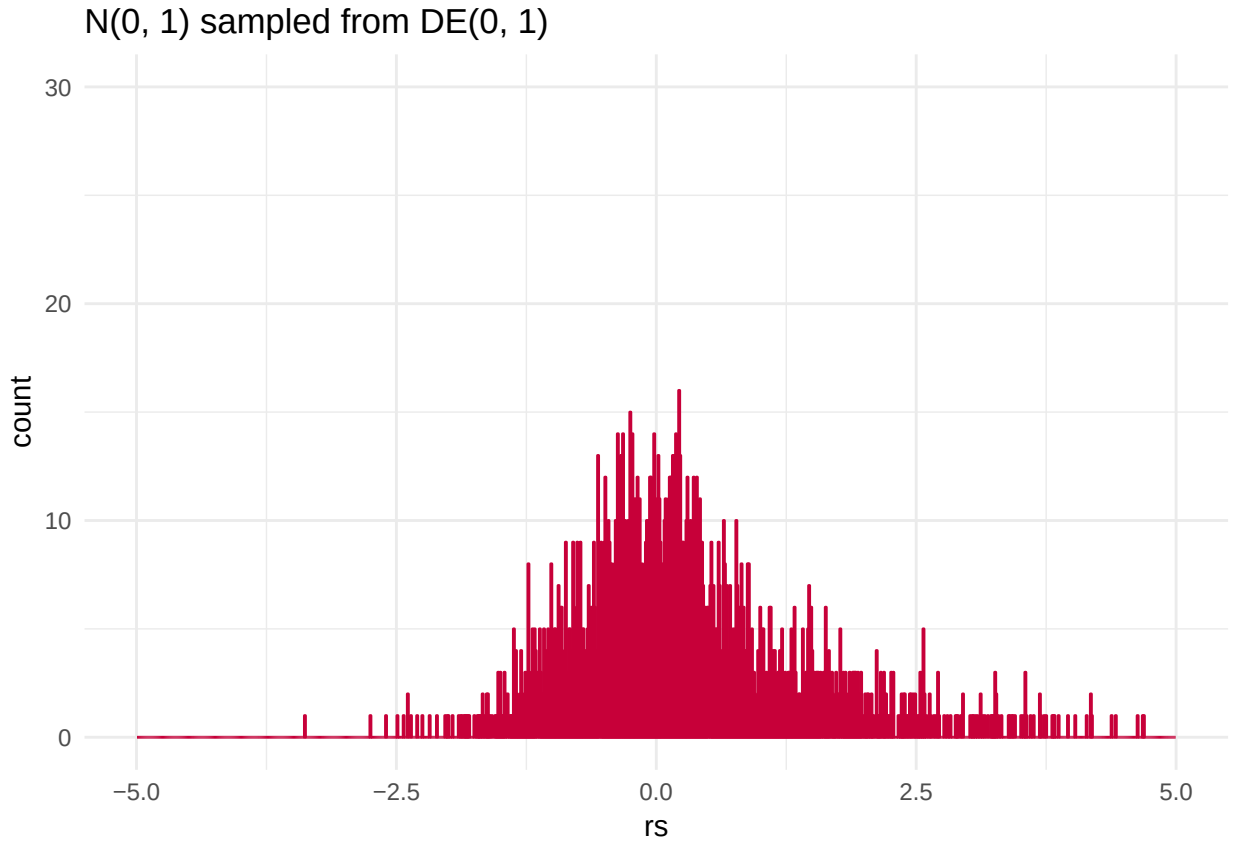
```
## [1] 10.09809
```

The first plot shows the distributions and the new envelope. The second plot shows the acceptance and rejection region. We can see that the function successfully envelops our distribution.


Envelope
Density
X
Legend
Double Exponential Distribution (PDF)
Normal Distribution (PDF)
Scaled Double Exponential Distribution (PDF)


Acceptence and Rejection Regions
Density
X

```r
# We draw from DE(0, 1) (proposal)

rs = c()
rs_rejected = c()

while (length(rs) < 2000) {
  # Take a random sample from our proposal (x-axis)
  z = rdel(n = 1, mu = 0, b = 1)

  # Take a unform, thus a random y value
  u = runif(n = 1, min = 0, max = c * pdel(z))

  # Check in which region this on lies
  if (u <= pnorm(z)) {
    rs = c(rs, z)
  }
  else {
    rs_rejected = c(rs_rejected, z)
  }
}
```



We can see that this is close to the desired $\mathcal{N}(0, 1)$. The (average) rejection rate `R` is:

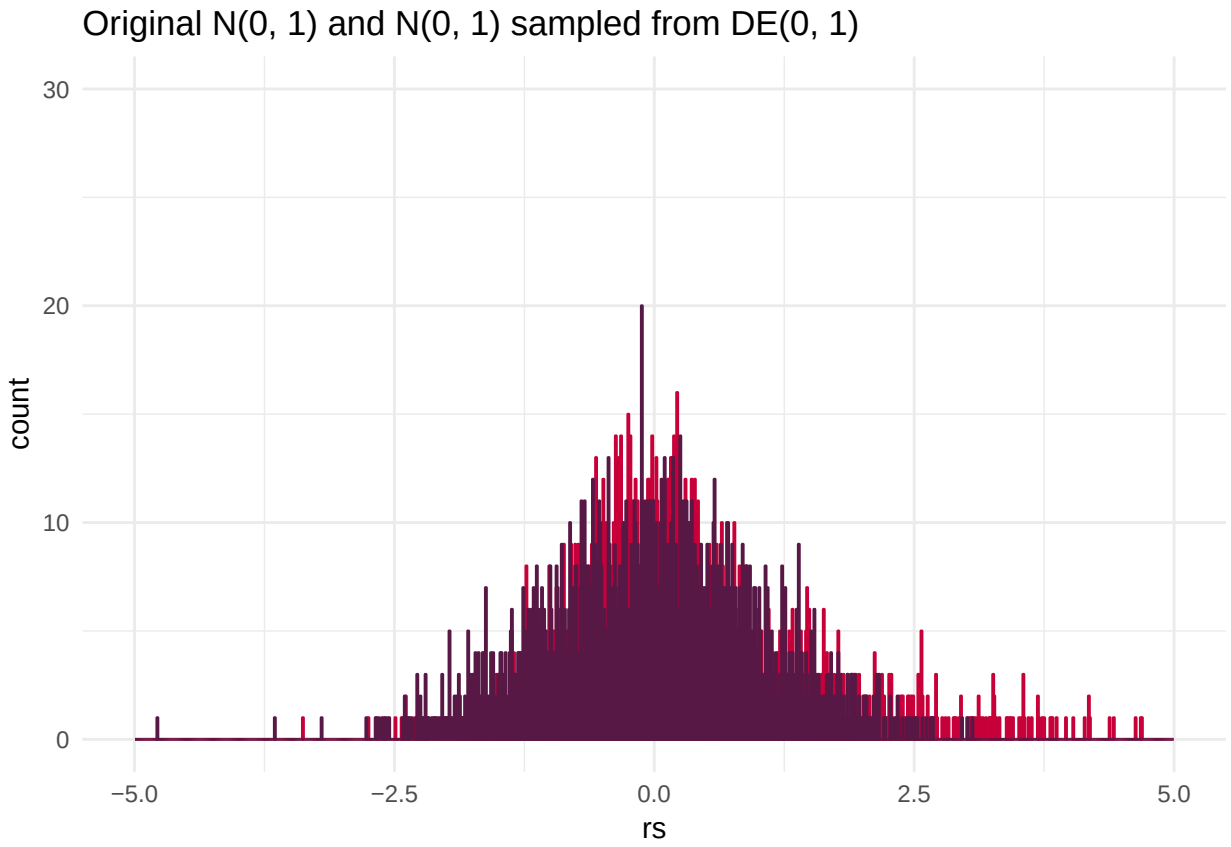
```
## [1] 0.9067425
```

`c` defines the expected error rate. Note that we changed α and b, sso we have to take the inverse of `c`. The expected error rate therefore is:

```
## [1] 0.9009714
```

Which is very close to what we observed in our example.

Last but not least let us draw from the real $\mathcal{N}(0, 1)$ and compare the plots.



We observe that both histograms are almost the same and conclude that our method worked.

```
                .--.    _
            .---|__| .((\=.
         .--|===|--|/     ,(,
         |   |===|   |\       y
         |%%|    |   | `.__,'
         |%%|    |   | /  \\\
         |   |    |   |/|   | \`----.
         |   |    |   ||\   \   |___.'_
       _|   |    |__||,\   \-+-._.' )_
      / |   |===|--|\   \   \       /  \
     /  `--^---'--'  `--`-'---^-'     \
    '=================================`

    This is teddy bear. Be nice to teddy bear.
```

3 Source Code

```
knitr::opts_chunk$set(echo = TRUE, cache = FALSE, include = TRUE, eval = TRUE)
library(knitr)
library(readxl)
```

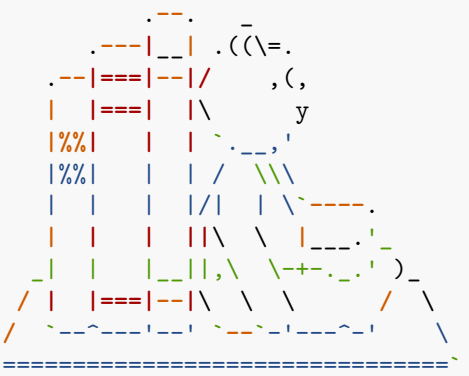
```r
library(ggplot2)
library(gridExtra)

################################################################################
# Question 1: Cluster Sampling
################################################################################

# reading the data in ----------------------------------
data = read.csv("population.csv", sep = ";", encoding = "latin1")

kable(head(data))


get_city_by_urn_wo = function(city_pool) {

  # We take the cumulative sum and then runif from 1 to max(cumulative sum).
  # This way we respect the proportions. As we need every intermediate result,
  # we use a loop
  cumulative_pop_sum = 0

  for (i in 1:nrow(city_pool)) {
    cumulative_pop_sum = cumulative_pop_sum + city_pool$Population[i]
    city_pool$CumSum[i] = cumulative_pop_sum
  }

  # Now we get a random value between 1 to max(cumulative sum). As larger muni-
  # cipalities have larger ranges, this works as expected
  selection =
    floor(runif(n = 1, min = 1, max = city_pool$CumSum[nrow(city_pool)]))

  # Return the first city which has a greater CumSum than the selection
  return(city_pool[city_pool$CumSum > selection,][1, c(1, 2)])
}


get_n_cities = function(data, n) {

  # Create a copy to not touch the original data.
  city_pool = data
  selected_cities = data.frame()

  # As long as we don't have n samples, get one and remove it from the pool,
  # as we sample without replacement
  while(nrow(selected_cities) < n) {
    selected_city = get_city_by_urn_wo(city_pool)
    selected_cities = rbind(selected_cities, selected_city)
    city_pool = city_pool[!rownames(city_pool) %in% rownames(selected_cities),]
  }

  return(selected_cities)
}

sample = get_n_cities(data, 20)
```

```r
sample


ggplot(sample)+
  geom_histogram(aes(x = Population), bins = nrow(sample), color = "black", fill = "#C70039") +
  ggtitle("Histogram of selected cities")

ggplot(data)+
  geom_histogram(aes(x = Population), bins = nrow(data), color = "#C70039", fill = "#C70039") +
  ggtitle("Histogram of all cities")


################################################################################
# Question 2: Different Distributions
################################################################################


sgn = function(x) {
  if (x < 0) return(-1)
  if (x > 0) return( 1)
  return(0)
}

qdel = function(p, mu = 0, b = 1) {
  if (p < 0 | p > 1) stop("p must be in range (0, 1)")
  return(mu - b * sgn(p-0.5) * log(1 - 2 * abs(p - 0.5)))
}


rdel = function(n = 1, mu = 0, b = 1) {
  quantiles = runif(n = n, min = 0, max = 1)
  rdels = sapply(X = quantiles, FUN = qdel, mu = mu, b = b)
  return(rdels)
}


sample_rdel_0_1 = rdel(10000, mu = 0, b = 1)
sample_rdel_0_2 = rdel(10000, mu = 0, b = 2)
sample_rdel_0_4 = rdel(10000, mu = 0, b = 4)
sample_rdel_m5_4 = rdel(10000, mu = -5, b = 4)

df = data.frame(sample_rdel_0_1, sample_rdel_0_2,
                sample_rdel_0_4, sample_rdel_m5_4)


p1 = ggplot(df) +
  geom_histogram(aes(x = sample_rdel_0_1),
                 color = "#FFC300", fill = "#FFC300", binwidth = 0.01) +
  xlim(-10, 10) +
  ylim(0, 60) +
  ggtitle("RD(0, 1)") +
  theme_minimal()
```

```r
p2 = ggplot(df) +
  geom_histogram(aes(x = sample_rdel_0_2),
                 color = "#FF5733", fill = "#FF5733", binwidth = 0.01) +
  xlim(-10, 10) +
  ylim(0, 60) +
  ggtitle("RD(0, 2)") +
  theme_minimal()

p3 = ggplot(df) +
  geom_histogram(aes(x = sample_rdel_0_4),
                 color = "#C70039", fill = "#C70039", binwidth = 0.01) +
  xlim(-10, 10) +
  ylim(0, 60) +
  ggtitle("RD(0, 4)") +
  theme_minimal()

p4 = ggplot(df) +
  geom_histogram(aes(x = sample_rdel_m5_4),
                 color = "#900C3F", fill = "#900C3F", binwidth = 0.01) +
  xlim(-10, 10) +
  ylim(0, 60) +
  ggtitle("RD(-5, 4)") +
  theme_minimal()

grid.arrange(p1, p2, p3, p4, nrow = 2)


ddel = function(x = 1, mu = 0, b = 1) {
  return(1/(2*b) * exp(-abs(x-mu)/(b)))
}


pdel = function(x = 1, mu = 0, b = 1) {
  return(1/2 + 1/2 * sgn(x-mu) * (1 - exp(-(abs(x-mu))/(b))))
}


sequence = seq(from = -10, to = 10, by = 0.01)

dnorm_samples = sapply(X = sequence, FUN = dnorm)
ddel_samples = sapply(X = sequence, FUN = ddel)
pnorm_samples = sapply(X = sequence, FUN = pnorm)
pdel_samples = sapply(X = sequence, FUN = pdel)

df = data.frame(dnorm_samples, ddel_samples, pnorm_samples, pdel_samples)

ggplot(df) +
  geom_line(aes(x = sequence, y = dnorm_samples,
                colour = "Normal Distribution (PDF)")) +
  geom_line(aes(x = sequence, y = ddel_samples,
                colour = "Double Exponential Distribution (PDF)")) +
  labs(title = "dnorm() and ddel()", y = "Density",
  x = "X", color = "Legend") +
```

```
  scale_color_manual(values = c("#17202A", "#C70039")) +
  theme_minimal()

ggplot(df) +
  geom_line(aes(x = sequence, y = pnorm_samples,
                colour = "Normal Distribution (CDF)")) +
  geom_line(aes(x = sequence, y = pdel_samples,
                colour = "Double Exponential Distribution (CDF)")) +
  labs(title = "pnorm() and pdel()", y = "Cumulative Density",
  x = "X", color = "Legend") +
  scale_color_manual(values = c("#17202A", "#C70039")) +
  theme_minimal()


c = max(dnorm_samples / pnorm_samples)
c


df$scaled_envelop = c * df$ddel_samples

ggplot(df) +
  geom_line(aes(x = sequence, y = dnorm_samples,
                colour = "Normal Distribution (PDF)")) +
  geom_line(aes(x = sequence, y = ddel_samples,
                colour = "Double Exponential Distribution (PDF)")) +
  geom_line(aes(x = sequence, y = scaled_envelop,
                colour = "Scaled Double Exponential Distribution (PDF)")) +
  labs(title = "Envelope", y = "Density",
  x = "X", color = "Legend") +
  scale_color_manual(values = c("#17202A", "#C70039", "#581845")) +
  theme_minimal()

ggplot(df) +
  geom_ribbon(aes(x = sequence, ymin = df$dnorm_samples, ymax = df$scaled_envelop),
              alpha = 0.8, fill = "#DAF7A6", color = "#DAF7A6") +
  geom_ribbon(aes(x = sequence, ymin = 0, ymax = df$dnorm_samples),
              alpha = 0.8, fill = "#C70039", color = "#C70039") +
  labs(title = "Acceptence and Rejection Regions", y = "Density",
  x = "X", color = "Legend") +
  scale_color_manual(values = c("#17202A", "#C70039", "#581845")) +
  theme_minimal()


# We draw from DE(0, 1) (proposal)

rs = c()
rs_rejected = c()

while (length(rs) < 2000) {
  # Take a random sample from our proposal (x-axis)
  z = rdel(n = 1, mu = 0, b = 1)

  # Take a unform, thus a random y value
```

```
  u = runif(n = 1, min = 0, max = c * pdel(z))

  # Check in which region this on lies
  if (u <= pnorm(z)) {
    rs = c(rs, z)
  }
  else {
    rs_rejected = c(rs_rejected, z)
  }
}


df2 = as.data.frame(rs)

ggplot(df2) +
  geom_histogram(aes(x = rs),
                 color = "#C70039", fill = "#C70039", binwidth = 0.01) +
  xlim(-5, 5) +
  ylim(0, 30) +
  ggtitle("N(0, 1) sampled from DE(0, 1)") +
  theme_minimal()


length(rs_rejected) / (length(rs)+length(rs_rejected))


1 - 1/c


df2$norm_samples = rnorm(n=2000)

ggplot(df2) +
  geom_histogram(aes(x = rs),
                 color = "#C70039", fill = "#C70039", binwidth = 0.01) +
  geom_histogram(aes(norm_samples),
                color = "#581845", fill = "#581845", binwidth = 0.01) +
  xlim(-5, 5) +
  ylim(0, 30) +
  ggtitle("Original N(0, 1) and N(0, 1) sampled from DE(0, 1)") +
  theme_minimal()
```



```
    This is teddy bear. Be nice to teddy bear.
```