A-2.R

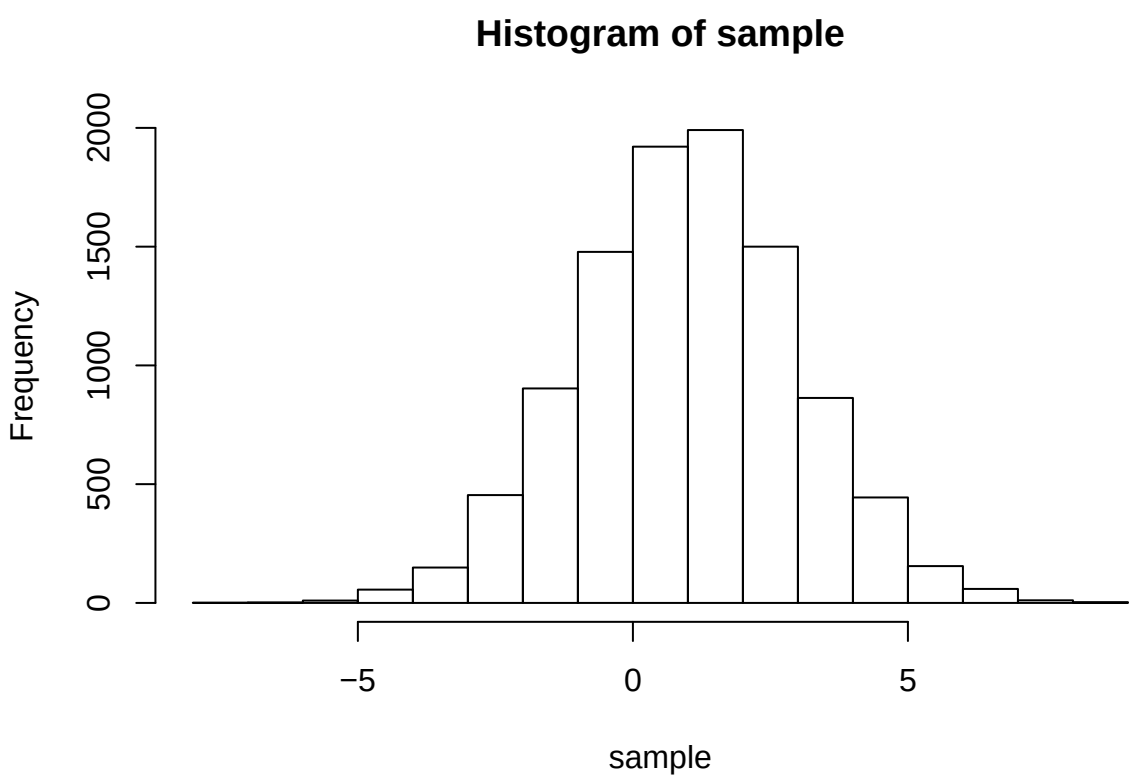
pradyuth

Thu Feb 08 18:44:29 2018

```
#Name : Pradyuth Vangur
#Assignment 2

#Problem 1: Rejection sampling from a truncated Gaussian

#Question 1
#Creating a sample as described in the question
sample <- rnorm(n = 10000, mean = 1, sd = 2)
head(sample)
```

```
## [1] -0.5691900  0.7243454 -0.3536508 -0.4714778 -4.1392611  0.1541246
```

```
#Generating histogram of the sample
hist(sample)
```

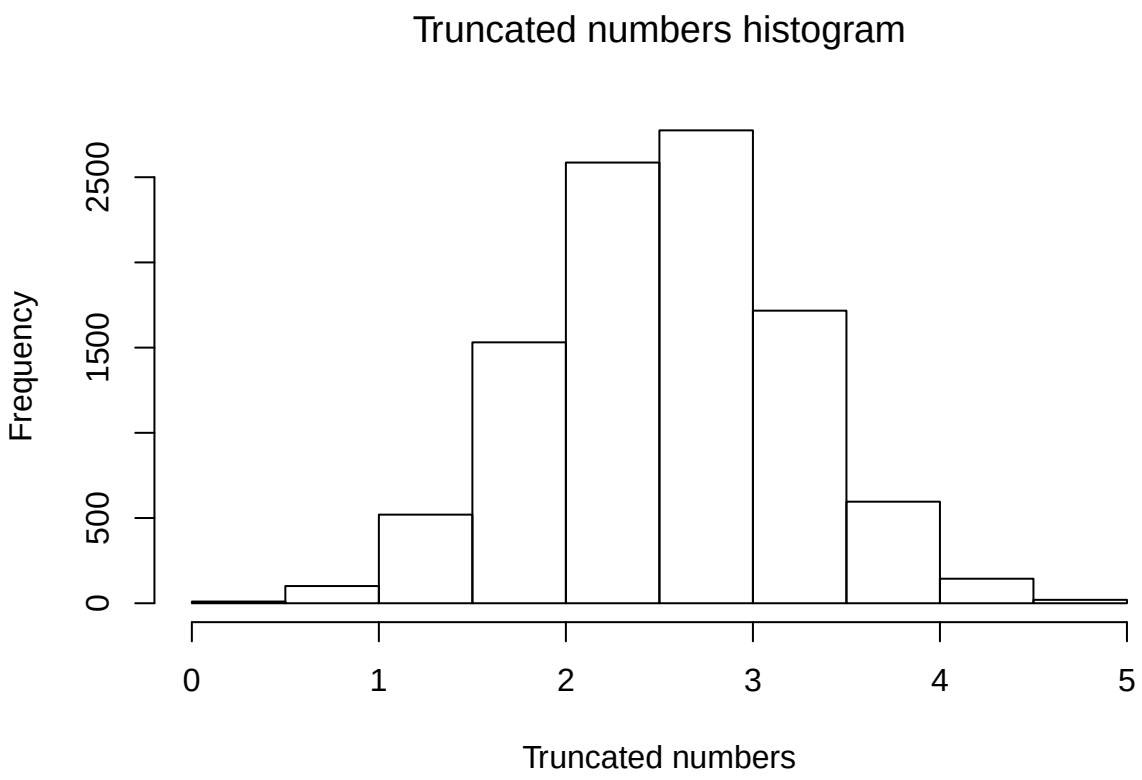



```
#Question 2
#Creating trunc_norm in order to generate number between (lower, upper)
k <- 1

#Creating function with the required inputs
```

```r
trunc_norm <- function(mean, sd, lower = -Inf, upper = Inf, num_samp = 1){
#Creating dataframe to store the data in the rows and columns represent the sample number
  trunc_sample <- data.frame()
#Generating sample sets of truncated gaussian data
  for(k in 1:num_samp){
    i <- 1
    j <- 1
#Generating random normalised data
    sample <- rnorm(n = 10000, mean = mean, sd = sd)
    while(i <= length(sample)){
#Converting each number from the normal Gaussian distribution to truncated distribution
      trunc_sample[j, k] <- (sample[i] - range(sample)[1]) * ((upper - lower) / (diff(range(sample)))) +
      i = i + 1
      j = j + 1
      }
  }
  return(trunc_sample)
}

#Question 3
#Generating the truncated numbers between 0 and 5
num <- trunc_norm(mean = 1, sd = 2, lower = 0, upper = 5)
head(num)
```

```
##         V1
## 1 2.006227
## 2 2.609308
## 3 3.129078
## 4 3.014070
## 5 3.093091
## 6 4.363424
```

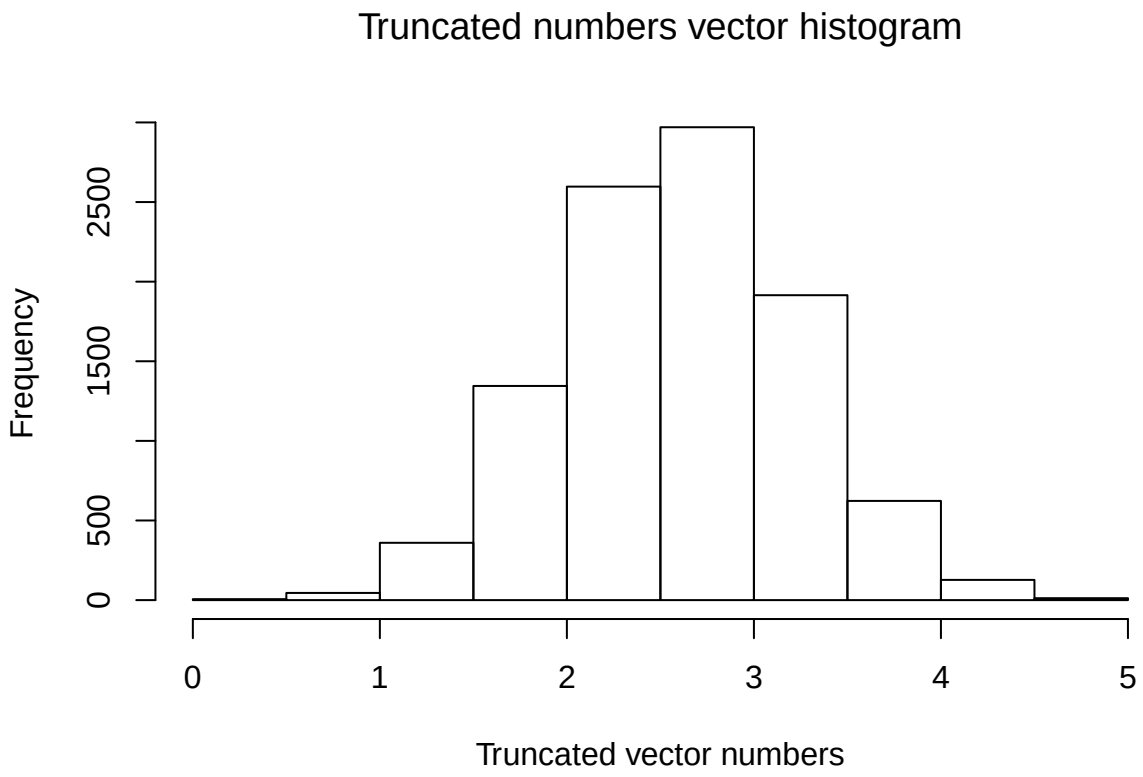
```r
#We obtain the data in a list form which has to be unlisted to be histogrammed
hist(unlist(num), xlab = "Truncated numbers", main = expression("Truncated numbers histogram"))
```




```r
#Question 4
#Creating function with the required inputs
trunc_norm_vec <- function(mean, sd, lower = -Inf, upper = Inf, num_samp = 1){
  sample_vec <- rnorm(10000, mean = mean, sd = sd)
  z <- 1
#Matrix represents the data samples with columns refering to the sample numbers
  trunc_sample_vec <- matrix(nrow = 10000, ncol = num_samp)
  while(z <= num_samp){
    trunc_sample_vec[,z] <- (sample_vec - range(sample_vec)[1]) * ((upper - lower) / diff(range(sample_v
    z = z + 1
  }
  return(trunc_sample_vec)
}

#Question 5
#Generating truncated numbers between 0 and 5
num_vec <- trunc_norm_vec(mean = 1, sd = 2, lower = 0, upper = 5)
head(num_vec)
```

```
##          [,1]
## [1,] 3.839555
## [2,] 3.163373
## [3,] 3.407290
## [4,] 2.863774
## [5,] 2.099622
## [6,] 2.258477
```

```
#Generating histogram
hist(unlist(num_vec), xlab = "Truncated vector numbers", main = expression("Truncated numbers vector his
```




```
#Question 6
#System time for normal function with loops
system.time(trunc_norm(mean = 1, sd = 2, lower = 0, upper = 5, num_samp = 4))
```

```
##    user  system elapsed
##   43.31    0.00   43.48
```

```
#System time for normal function with vectors
system.time(trunc_norm_vec(mean = 1, sd = 2, lower = 0, upper = 5, num_samp = 4))
```

```
##    user  system elapsed
##    0.01    0.00    0.02
```

```
#Problem 2: Calculating Entropy

#Question 1
#Generating variables of mean 1 and sd 1
ent_vec <- matrix(rnorm(60000, mean = 1, sd = 1), ncol = 6)
head(ent_vec)
```

```
##              [,1]       [,2]      [,3]        [,4]       [,5]      [,6]
## [1,]  1.251630577 0.59248479 0.4834980 -0.02863188  1.7322321 0.7495302
## [2,]  0.007297025 1.44932859 1.3858776  1.21731701 -0.5802739 0.2917395
## [3,]  0.097189152 0.03335058 0.2318255  1.93886709  0.4601490 2.4347292
## [4,] -0.301477546 0.34857039 0.7484387  1.57540289  2.3592471 0.5722327
```

```
## [5,] -0.735103412 1.28625458 2.9958694  0.29512583  0.2551383 1.3641438
## [6,]  1.515473783 1.37064301 1.3890277  1.74929940  0.8254816 1.6714642
```

```
#Setting all negative elements to zero
ent_vec[which(ent_vec < 0)] <- 0

#Rescaling rows of matrix to one
ent_vec_sum <- rowSums(ent_vec)
head(ent_vec_sum)
```

```
## [1] 4.809376 4.351560 5.196111 5.603892 6.196532 8.521390
```

```
#Reducing the sum of vectors in each row to one
ent_vec <- ent_vec / ent_vec_sum
head(ent_vec)
```

```
##             [,1]        [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] 0.260248038 0.123193703 0.10053238 0.00000000 0.36017816 0.15584771
## [2,] 0.001676876 0.333059565 0.31847836 0.27974269 0.00000000 0.06704251
## [3,] 0.018704212 0.006418373 0.04461519 0.37313815 0.08855644 0.46856763
## [4,] 0.000000000 0.062201485 0.13355695 0.28112657 0.42100155 0.10211345
## [5,] 0.000000000 0.207576528 0.48347518 0.04762758 0.04117438 0.22014633
## [6,] 0.177843500 0.160847355 0.16300483 0.20528334 0.09687171 0.19614925
```

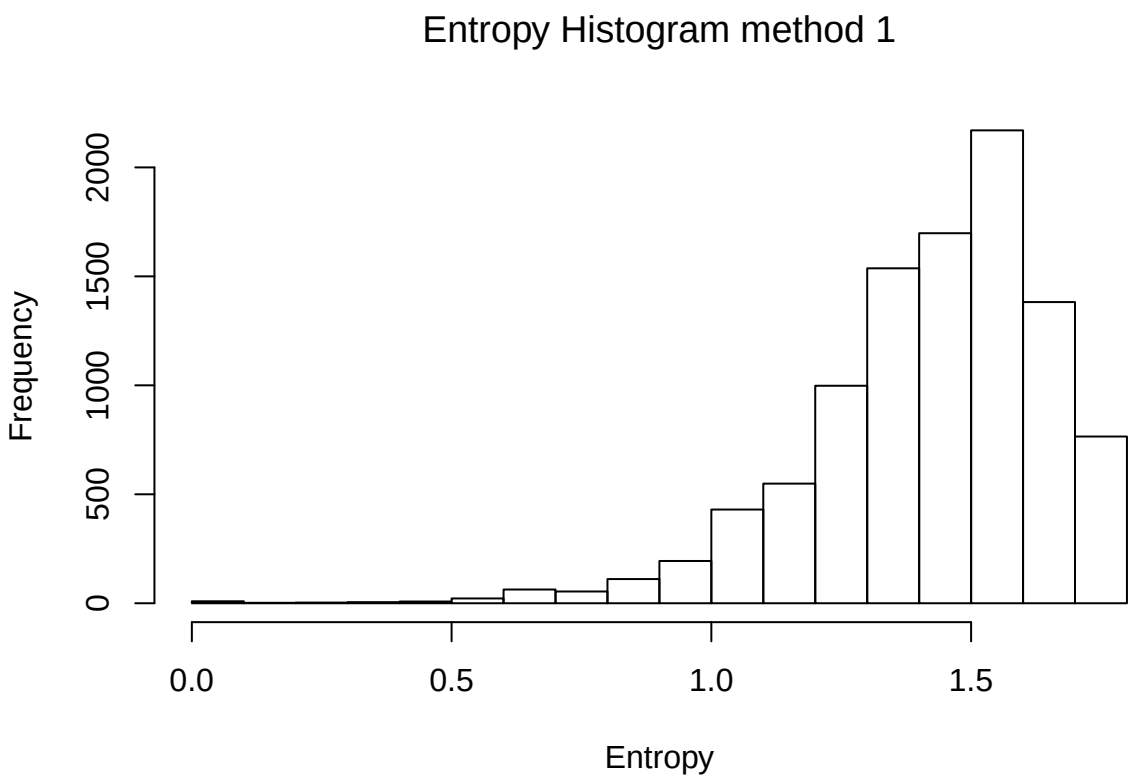
```
#Question 2
#Calculation of entropy of each element
#Method 1: Calculating by using na.rm inside the function
ent <- -ent_vec * log(ent_vec)
#Calculation of entropy of each row
ent_row_1 <- rowSums(ent, na.rm = TRUE)
head(ent_row_1)
```

```
## [1] 1.496743 1.278834 1.183289 1.395579 1.287253 1.767388
```

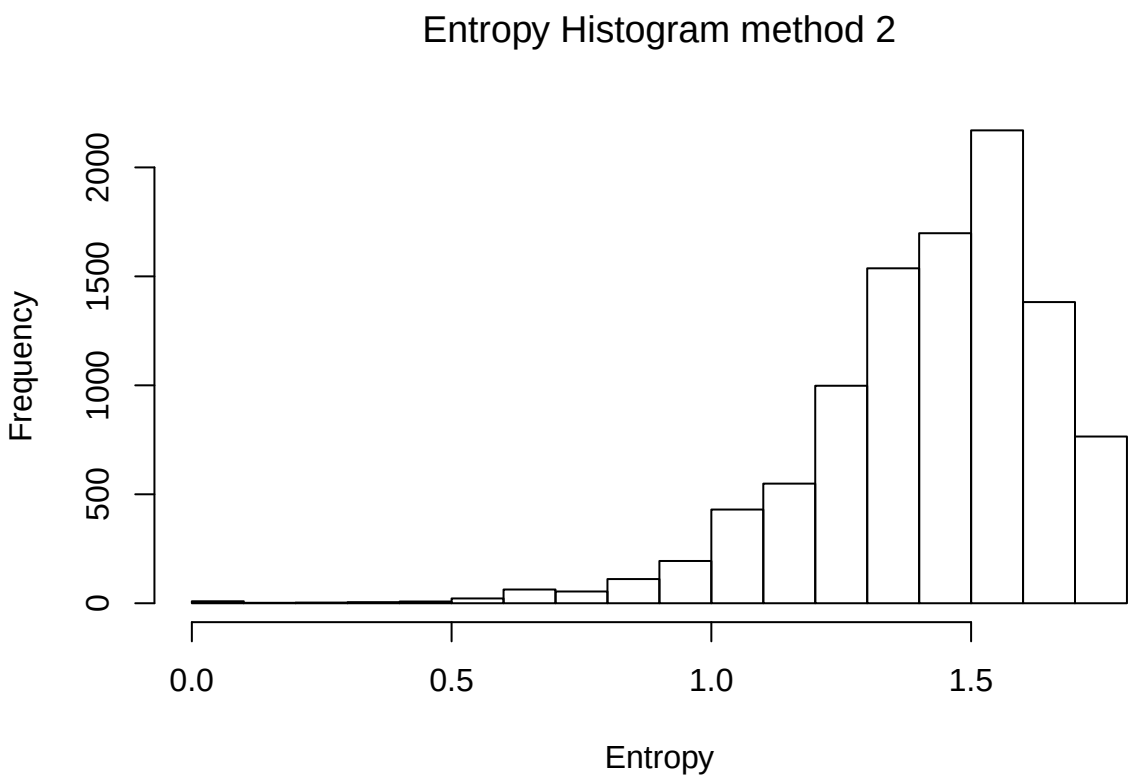
```
#Method 2: Calculating by assigning 0 to NaN values
ent2 <- -ent_vec * log(ent_vec)
ent2[which(ent2 == "NaN")] <- 0
ent_row_2 <- rowSums(ent2)
head(ent_row_2)
```

```
## [1] 1.496743 1.278834 1.183289 1.395579 1.287253 1.767388
```

```
#We observe the output generated by both methods are same.

#Question 3
#Method 1
hist(ent_row_1, xlab = "Entropy", main = expression("Entropy Histogram method 1"))
```




```r
#Method 2
hist(ent_row_2, xlab = "Entropy", main = expression("Entropy Histogram method 2"))
```




```
#Question 4
#Largest entropy observed
ent_row_1[which(ent_row_1 == max(ent_row_1))]
```

```
## [1] 1.791325
```

```
#Corresponding vector
which(ent_row_1 == max(ent_row_1))
```

```
## [1] 1610
```

```
#Question 5
#Entropy refers to the measure of information in the probability information.
#Higher entropy refers to the less information and lower entropy refers to large
#amount of information.
```