Statistics 1 Unit 3

Group 8

December 29, 2017

Contents

1 Task 41

It is given that: $Y = AX + b$, with:
- Mean $m_Y = Am_X + b$
- Covariance matrix: $\sum_Y = A \sum_X A^t$.
A multivariate normal distribution has parameters:

μ, mean, and $\sum$, the covariance matrix.
The goal is to find a real matrix A such that:

$$AA^t = \sum$$

Let's apply case a), the eigendecomposition of $\sum$.
The eigenvalue decomposition of a general matrix A is:

$$A = TDT^t$$

But in our case: $A = T\sqrt{D}$. So:

$$\sum = T\sqrt{D}$$

To generate n random points from the multivariate normal distribution with the above parameters (μ and $\sum$), we obtain :

```
rmvnorm <- function(n, mu, sigma) {
  a <- ncol(sigma)
  mu <- rep(mu, n)
  X<-matrix(rnorm(n * a),n,a) %*% eigen(sigma)$vectors
     %*%sqrt(diag(eigen(sigma)$values))+mu
  return(X)
}
```

2 Task 42

2.1 a

2.1.1 Mean

F mean:
Let's prove that F has mean = 0

$$E(F) = E(\frac{\sqrt{\rho}}{1+\rho(d-1)}\sum X_j + \sqrt{\frac{1-\rho}{1+\rho(d-1)}}Y)$$
$$= \frac{\sqrt{\rho}}{1+\rho(d-1)}\sum E(X_j) + \sqrt{\frac{1-\rho}{1+\rho(d-1)}}E(Y),$$

$X_i, Y \sim N(0,1)$ this gives us $E(F) = 0$.

ϵ_i means:

Now prove that ϵ_i has the mean 0.

$$\epsilon_i = X_i - \sqrt{\rho} F$$

And we know that:

1. $X_i \sim N(0,1)$
2. $E(F) = 0$
3. $E(\alpha F) = \alpha E(F)$

From this 3 facts follows that all ϵ_i have means 0

2.1.2 Variance

F variance:

$$\operatorname{var}\left(\sum_{i=1}^{n} X_i\right) = E\left(\left[\sum_{i=1}^{n} X_i\right]^2\right) - \left[E\left(\sum_{i=1}^{n} X_i\right)\right]^2$$

From this after several steps follows:

$$\operatorname{var}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n}\sum_{j=1}^{n}\big(E(X_iX_j) - E(X_i)E(X_j)\big) = \sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{cov}(X_i, X_j)$$

Let's prove that $var(F) = 1$.

$\sigma^2(F) = \sigma^2\left(\frac{\sqrt{\rho}}{1+\rho(d-1)}\sum X_j + \sqrt{\frac{1-\rho}{1+\rho(d-1)}}Y\right)$
$= \left(\frac{\sqrt{\rho}}{1+\rho(d-1)}\right)^2 \sigma^2(\sum(X_j)) + \left(\sqrt{\frac{1-\rho}{1+\rho(d-1)}}\right)^2 \sigma^2(Y)$
$= \left(\frac{\sqrt{\rho}}{1+\rho(d-1)}\right)^2 \left(\sum Cov(X_i, X_i) + \sum\sum_{i\neq j} Cov(X_i, X_j)\right) + \left(\sqrt{\frac{1-\rho}{1+\rho(d-1)}}\right)^2 \sigma^2(Y)$
$= \left(\frac{\sqrt{\rho}}{1+\rho(d-1)}\right)^2 (d + d(d-1)\rho) + \left(\sqrt{\frac{1-\rho}{1+\rho(d-1)}}\right)^2 1$
$= \frac{\rho d(1+(d-1)\rho)}{(1+\rho(d-1))^2} + \left(\frac{1-\rho}{1+\rho(d-1)}\right) = \frac{\rho d+1-\rho}{(1+\rho(d-1))}$
$= \frac{1+\rho(d-1))}{(1+\rho(d-1))} = 1.$

Uncorrelation proof:

We need to prove that

$$Cov(F, \epsilon_i) = 0$$

$Cov(F, \epsilon_i) = Cov(F, X_i - \sqrt{\rho}F) = Cov(F, X_i) - Cov(F, \sqrt{\rho}F)$
$= Cov(F, X_i) - \sqrt{\rho}\sigma^2(F)$
$= Cov(\frac{\sqrt{\rho}}{1+\rho(d-1)} \sum_{j=1}^{d} X_j + \sqrt{\frac{1-\rho}{1+\rho(d-1)}}Y, X_i) - \sqrt{\rho}$
$=^{X,Y uncorr} \frac{\sqrt{\rho}}{1+\rho(d-1)} Cov(\sum_{j=1}^{d} X_j, X_i) - \sqrt{\rho}$
$= \frac{\sqrt{\rho}}{1+\rho(d-1)} \left(\sigma^2(X_i) + \sum_{j\neq 1}^{d} Cov(X_j, X_i)\right) - \sqrt{\rho}$
$= \frac{\sqrt{\rho}}{1+\rho(d-1)} (1 + \rho(d-1)) - \sqrt{\rho} = 0$

ϵ_i **variance:**

$\sigma^2(\epsilon_i) = \sigma^2(X_i - \sqrt{\rho}F) = \sigma^2(X_i) + \rho\sigma^2(F) - 2\sqrt{\rho}Cov(X_i, F)$
$= 1 + \rho - 2\sqrt{\rho}\left(Cov(\epsilon_i, F) + Cov(\sqrt{p}F, F)\right)$
$= 1 + \rho - 2\sqrt{\rho}\sqrt{\rho}$
$= 1 - \rho.$

3 Task 43

3.1 Find mean of X

$\mathbb{E}(X) = \mathbb{E}(m + \sqrt{W}AZ) = \mathbb{E}(m) + A\mathbb{E}(\sqrt{W}Z).$
Since

1. W and Z are independent
2. $\mathbb{E}(Z) = 0$

$$\mathbb{E}(\sqrt{W}Z) = \mathbb{E}(\sqrt{W}) * \mathbb{E}(Z) = 0$$

3.2 Find cov X

$$Cov(m + \sqrt{W}AZ) = Cov(\sqrt{W}AZ) = ACov(\sqrt{W}Z)A^T$$

Let's calculate

$$Cov(\sqrt{W}Z) = Cov(\sqrt{W}Z, \sqrt{W}Z) =$$
$$\mathbb{E}((\sqrt{W}Z - \mathbb{E}(\sqrt{W}Z))(\sqrt{W}Z - \mathbb{E}(\sqrt{W}Z))^T) =$$
$$\mathbb{E}((\sqrt{W}Z)(\sqrt{W}Z)^T) = \mathbb{E}(W)\mathbb{E}(ZZ^T).$$

The last expected value $\mathbb{E}(ZZ^T)$ is $Cov(Z)$.
$Cov(Z) = \mathbb{E}((Z-\mathbb{E})(Z-\mathbb{E}(Z)^T)) = \mathbb{E}(ZZ^T)$.
That's why it's equal to I. So, we have $Cov(m+\sqrt{W}AZ) = \mathbb{E}(W)\sum$.
For other questions we would be thankful for your explanation.

4 Task 44

Bivariate Normal Distribution has the density:

$$f(x_1,x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} exp\{-\frac{z_1^2 - 2\rho z_1 z_2 + z_2^2}{2(1-\rho^2)}\}$$

where $z_1 = \frac{x_1-\mu}{\sigma_1}$ and $z_2 = \frac{x_2-\mu}{\sigma_2}$ and $\sigma_1^2\sigma_2^2(1-\rho^2)$ is just the determinant of the covariance matrix.

Using substitutions we can get standard normal distributions: $X_1 \sim N(0,1)$ and $X_2 \sim N(0,1)$.

The density now is:

$$f(x_1,x_2) = \frac{1}{2\pi\sqrt{1-\rho^2}} exp\{-\frac{x_1^2 - 2\rho x_1 x_2 + x_2^2}{2(1-\rho^2)}\}$$

where $z = \frac{x-\mu}{\sigma} = x$.
Let's rewrite our density as:

$$f(x_1,x_2) = \frac{1}{\sqrt{2\pi}} exp\{-\frac{x_1^2}{2}\} \cdot \frac{1}{\sqrt{2\pi(1-\rho^2)}} exp\{-\frac{(\rho x_1 - x_2)^2}{2(1-\rho^2)}\} \tag{1}$$

Using conditional distribution formula :

$$f(x_1,x_2) = f_{X_1}(x_1) \cdot f_{X_2|X_1}(x_1|x_2) \tag{2}$$

The first part of in (1) is density function of a standard normal distribution. The second part represents normal distribution too.

$$f_{X_2|X_1}(x_2|x_1) = \frac{1}{\sqrt{2\pi(1-\rho^2)}} exp\{-\frac{(\rho x_1 - x_2)^2}{2(1-\rho^2)}\}.$$

with parameters $\mu = \rho x$, $\sigma^2 = 1-\rho^2$

5 Task 47

```
U <- runif(19908)
```

5.1 a)

```
mean(U)

## [1] 0.4997907

var(U)

## [1] 0.08413155

sd(U)

## [1] 0.2900544
```

5.2 b)

Uniformly distributed random variable $X \sim U[0, 1]$ has these properties:
$E(X) = \frac{b-a}{2}$,
$\sigma^2(X) = \frac{(b-a)^2}{12}$,
$\sigma(X) = \sqrt{(Var(X))}$.
Given $a = 0$ and $b = 1$ it is possible to calculate:
$E(X) = \frac{1}{2}$,
$\sigma^2(X) = \frac{1}{12} = 0.083333...$,
$\sigma(X) = \sqrt{(Var(X))} = 0.28867...$

Thus, the result from randomly generated numbers is quite close to theoretical one.

5.3 c)

```
length(U[U<0.6])/length(U)

## [1] 0.6002612
```

$P(X < 0.6) = F(x^-) - F(y)$, where $x = 0.6$ and $y = 0$:

$$F(x) = \frac{x-a}{b-a}.$$

Once again, we have that $a = 0$, $b = 1$. Thus, $F(0.6) = 0.6$.

6 Task 48

```
U1 <- runif(10000,0,1)

U2 <- runif(10000,0,1)
```

6.1 a)

We know $\mathbb{E}(U_1+U_2) = \mathbb{E}(U_1)+\mathbb{E}(U_2)$ since the expectation operator is linear.

```
mean(U1 + U2)

## [1] 0.9975505

mean(U1) + mean(U2)

## [1] 0.9975505
```

We can see that our theoretical expectation holds true. Now let's calculate the true value:

$$\mathbb{E}(X) = \int_a^b xf(x)dx = \int_a^b x\frac{1}{b-a}dx = \frac{1}{b-a}\int_a^b xdx = \frac{1}{b-a}\frac{x^2}{2}\Big|_a^b = \frac{1}{b-a}\frac{b^2-a^2}{2} = \frac{a+b}{2}.$$
$$\mathbb{E}(X) = \frac{a+b}{2} = \frac{0+1}{2} = \frac{1}{2}.$$

Therefore, the expectation of the sum of two random variables with $\mathbb{E}(X) = 0.5$ must be 1. Thus, we are quite close to the theoretical value of the expectation.

6.2 b)

```
var(U1 + U2)

## [1] 0.1668067

var(U1)+ var(U2)

## [1] 0.1671039
```

They are not exactly equal. We know that $Var(X + Y) = Var(X) + Var(Y) + 2Cov(X, Y)$. If X and Y were really stochastically independent, the $Cov(X, Y)$ would be equal to 0. Thus, it can be seen that our two randomly drawn variables are not that independent as one would expect in a random sample.

6.3 c)

```
U <- U1 + U2

length(U[U<=1.5])/length(U)

## [1] 0.8794
```

6.4 d)

```
K <- sqrt(U1) + sqrt(U2)

length(K[K<=1.5])/length(K)

## [1] 0.6573
```

7 Task 49

```
# a

n <- 1000000

U1_vector <- runif(n, 0, 1)

U2_vector <- runif(n, 0, 1)

U3_vector <- runif(n, 0, 1)

sum_vector <- rep(0, n)

for(i in 1:n)
{
  sum_vector[i] <- U1_vector[i] + U2_vector[i] +
     U3_vector[i]
}


expectation_1 <- mean(sum_vector)

print(expectation_1)

## [1] 1.50003

# b1

var_1 <- var(sum_vector)

print(var_1)

## [1] 0.250416

# b2

var_2 <- var(U1_vector) + var(U2_vector) + var(
   U3_vector)

print(var_2)

## [1] 0.250135
```

```
# c

sqrt_sum_vector <- rep(0, n)

for(i in 1:n)
{
  sqrt_sum_vector[i] <- sqrt(U1_vector[i] + U2_vector[
     i] + U3_vector[i])
}

expectation_2 <- mean(sqrt_sum_vector)

print(expectation_2)

## [1] 1.205634

# d

frequency <- 0

for(i in 1:n)
{
  if ( (sqrt(U1_vector[i]) + sqrt(U2_vector[i]) + sqrt
     (U3_vector[i])) >= 0.8 )
  {
    frequency <- frequency + 1
  }
}


relative_frequency <- frequency / n

print(relative_frequency)

## [1] 0.997078
```

8 Task 50

```
n <- 100

size <- 20

prob <- 0.5

student_results <- rbinom(n, size, prob)

student_results

##   [1] 12  8 10  8  8 12 12  8 13 11 13  9  9 14 15
    7 10 10
##  [19]  6  9  9  9 12 12 11 12  7 10  6 10 12  9 11
   11  9 10
##  [37]  9 10 11  8 12  9 10  8  6 12 10 11  5  4 10
    9 11 10
##  [55] 11 10  9  9 11 11 10  9  9 14 11  9 10 12 12
   10  8  9
##  [73]  7 11 11  8 10 10  9  7 11 14  9 15 12  9 14
    7  5 16
##  [91] 11  9 12 11 12  6  6 12 11 10

# a

mean(student_results)

## [1] 9.98

sd(student_results)

## [1] 2.265151

# b

proportion <- sum(student_results >= 0.3*size) / n

proportion

## [1] 0.97
```

9 Task 51

```
n <- 10000
size <- 20
prob <- 0.3
simulation_results <- rbinom(n, size, prob)

# a
frequency <- sum(simulation_results <= 5)
empirical_probability_1 <- frequency / n
theoretical_probability_1 <- pbinom(5, size, prob)

# b
frequency <- sum(simulation_results == 5)
empirical_probability_2 <- frequency / n
theoretical_probability_2 <- dbinom(5, size, prob)

# c
empirical_expectation <- mean(simulation_results)
theoretical_expectation <- size*prob
```

```
# d

empirical_variance <- var(simulation_results)

theoretical_variance <- size*prob*(1-prob)


# e

empirical_percentile_95 <- quantile(simulation_results
   , 0.95)

print(empirical_percentile_95)

## 95%
##   9

theoretical_percentile_95 <-qbinom(0.95, size, prob)

print(theoretical_percentile_95)

## [1] 9

# f

empirical_percentile_99 <- quantile(simulation_results
   , 0.99)

print(empirical_percentile_99)

## 99%
##  11

theoretical_percentile_99 <-qbinom(0.99, size, prob)

print(theoretical_percentile_99)

## [1] 11

# g

empirical_percentile_99.9999 <- quantile(
   simulation_results, 0.999999)
```

```
print(empirical_percentile_99.9999)

## 99.9999%
##       14

theoretical_percentile_99.9999 <-qbinom(0.999999, size
   , prob)

print(theoretical_percentile_99.9999)

## [1] 16

# To estimate extreme quantities accurately, the
   sample size should be increased
```

10 Task 52

```
ranbin1 <- function(n, size, prob){
  cumbins <- pbinom(0: (size -1), size, prob)
  singlenumber <- function(){
    x <- runif(1)
    sum(x > cumbins)
  }
  replicate(n, singlenumber())
}
```

10.1 a)

The function "ranbin1" can simulate binomial pseudorandom variates using the inversion method.
First, we need to generate one number from $U[0,1]$. In the function it is x. Moreover, it is necessary to generate all values of CDF, excluding the last value (CDF = 1). Values of CDF will be reduced (size -1), in order to compare $F(x_{i-1}) < u \leq F(x_i)$.
The "sum(x > cumbins) means that the function sums up TRUE-values of the logical vector when $u > F(x_{i-1})$, it gives us the number of experiences in which we had success with the given probability p.

10.2 b)

```
system.time(ranbin1(1000, 10, 0.4))

##    user  system elapsed
##       0       0       0

system.time(rbinom(1000,10,0.4))

##    user  system elapsed
##       0       0       0
```

```
system.time(ranbin1(10000,10,0.4))

##    user  system elapsed
##    0.03    0.03    0.06

system.time(rbinom(10000,10,0.4))

##    user  system elapsed
##       0       0       0
```

```
system.time(ranbin1(100000,10,0.4))

##    user  system elapsed
##    0.31    0.02    0.32

system.time(rbinom(100000,10,0.4))

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

The standard function ”rbinom” is faster for generating binomial distributed random variables.

11 Task 53

```
ranbin2 <- function(n, size, prob){
  singlenumber <- function(size, prob){
    x <- runif(size)
    sum(x < prob)
  }
  replicate(n, singlenumber(size, prob))
}
```

11.1 a)

The function consists of 3 arguments: n, size and prob.
n indicates the number of random variables generated in the process.
size gives the maximal bound for a binomial distributed random variable.
prob is the probability of the random variables.
Inner function ”singlenumber” consists of 2 arguments: size and prob. The function generates uniform distributed random variables corresponding to the argument size.
”sum(x < prob)”. The summation of all variables < the given probability allows to count all succesfull realizations. This procedure is replicated n-times in order to get the vector with binomial random variables.
Also, observe a binomial distribution is the sum of independent and identically distributed Bernoulli random variables. By replicating the Bernoulli process n-times, we get the binomial distribution.

11.2 b)

```
system.time(ranbin2(10000,10,0.4))

##    user  system elapsed
##    0.03    0.00    0.03

system.time(rbinom(10000,10,0.4))

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

```
system.time(ranbin1(10000,10,0.4))

##    user  system elapsed 
##    0.03    0.00    0.03
```

```
system.time(ranbin2(10000,100,0.4))

##    user  system elapsed 
##    0.08    0.00    0.08

system.time(rbinom(10000,100,0.4))

##    user  system elapsed 
##       0       0       0

system.time(ranbin1(10000,100,0.4))

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

```
system.time(ranbin2(10000,1000,0.4))

##    user  system elapsed 
##    0.37    0.00    0.38

system.time(rbinom(10000,1000,0.4))

##    user  system elapsed 
##       0       0       0

system.time(ranbin1(10000,1000,0.4))

##    user  system elapsed 
##    0.07    0.00    0.06
```

”ranbin2” does not enhance the efficiency of code considering system time. The fastest is still ”rbin”, but ”ranbin1” is a bit faster than ”ranbin2”.

12 Task 54

```
ranbin3 <- function(n, size, prob) {
  singlenumber <- function(size, prob) {
    k <- 0
    U <- runif(1)
    X <- numeric(size)
    while (k < size) {
      k <- k + 1
      if (U <= prob) {
        X[k] <- 1
        U <- U / prob
      } else {
        X[k] <- 0
        U <- (U - prob) / (1 - prob)
      }
    }
    sum(X)
  }
  replicate(n, singlenumber(size, prob))
}


ranbin3(100,20,0.4)

##   [1]  5 10 10  7  9  6  9  7  5  6  9 10  9  8  7
   10  7 11
##  [19]  8  7  9  9 11  6  7 11 11  8  8 11  5  5  9
    7  6  8
##  [37]  6  8  4  7 11  8  7 13 10  9  9  9  6  8  8
    2  5  7
##  [55]  7  9 10 10 11  5  7  7 12  7 15  8  6  7  7
    9  7 10
##  [73]  5 11  8 10  7 11  6 10  7  7  7 12  7  4  8
    7  5  9
##  [91]  9  7 10  8  5  8  6 10 10  9

ranbin3(100,500,0.7)
```

```
##   [1] 356 350 354 359 352 346 339 348 339 340 350
   363 357
##  [14] 345 349 347 354 342 346 344 344 355 344 334
   354 353
##  [27] 349 340 341 339 344 346 352 342 375 346 361
   342 358
##  [40] 364 354 357 361 360 343 349 367 337 342 366
   347 368
##  [53] 347 345 337 357 351 343 331 350 356 345 369
   355 364
##  [66] 365 359 335 342 343 357 347 356 344 347 343
   332 342
##  [79] 372 360 359 347 371 345 353 350 330 353 346
   326 347
##  [92] 360 356 356 346 364 348 341 348 346
```

12.1 b)

General formula: $P(X_2 \mid X_1) = \frac{P(X_2 \cap X_1)}{P(X_1)}$
$\Rightarrow P(\frac{U}{p} < x \cap U < p) = \frac{px}{x} = p$

12.2 c)

Analogous: $P(\frac{U-p}{1-p} < x \cap U > p) = \frac{x-px}{1-p} = x$

12.3 d)

"ranbin3" generates random numbers from the uniform distribution by dividing the random variable U by the probability (if $U \leq$ prob) or (U - prob) / (1 - prob). Then the vector X is constructed consisting of the values 0 and 1. It represents the sequence of the Bernoulli experiment. Summation and replication create a binomial random variable for each number.

13 Task 55

First, let's find the probability that k people have different birthdays is: (firsly we considered some simple cases, and it seems that the pattern preserves).
The total of all the possible outcomes is 365^k.

Therefore: $P(k=0) = \frac{(365)(364)...(365-k+1)}{365^k}$,where $P(k=0)$ means noone has the same birthday
Now, let's use the following rule of probability:
$P(A) = 1 - P(A^c)$
$P(k \geq 1) = 1 - P(k=0) = 1 - \frac{(365)(364)...(365-k+1)}{365^k} = 1 - \prod_{i=1}^{k}(1 - \frac{i-1}{365})$

```
f55<- function(n){
  p <- c()
  for(i in 1:n)
    p <- prod(c((1 - (i -1)/365), p))
  1-p
}

#The min sample size is:

i <- 1

while (f55(i) < 1/2) {
  i <- i+1
}

i

## [1] 23

#TEST

f55(i)

## [1] 0.5072972

f55(i) == pbirthday(i)

## [1] TRUE
```

14 Task 56

In order to quadratic equation has real roots, we need to find
$P(D = b^2 - 4ac \geq 0)$:

```
a<-runif(100, min=-1, max=1)

b<-runif(100, min=-1, max=1)

c<-runif(100, min=-1, max=1)


D <- function(a, b, c){
  ifelse((b^2-4*a*c) >= 0, 1, 0)
  }


sum(D(a,b,c)) / length(D(a,b,c))

## [1] 0.65
```

Firstly, we need to generate 100 numbers which are U-distributed on the interval $[-1, 1]$ for a, b, c. Then, we trying to find the cases when $D \geq 0$ and create the vector which has 1 or 0 values, depending on the discriminant. After, we calculate the probability by summing up the cases when $D \geq 0$ and divide by numbers of all possible outcomes of D. This gives us the probability when quadratic equation has real roots.

15 Task 57

$F^{-1}(u) \leq x \Leftrightarrow u \leq F(x)$
Proof:
$F^{-1}(u) \leq x \Rightarrow x \in \{z : F(z) \geq u\} \Rightarrow u \leq F(x)$
$u \leq F(x) \Rightarrow x \in \{z : F(z) \geq u\} \Rightarrow F^{-1}(u) \leq x$. Because $F^{-1}(u) = \inf\{x : F(x) \geq u\}$

F(F$^{-1}(u)) \geq u$
Proof:
Let $x_n \in \{x : F(x) \geq u\}$ s.t. $\lim_{n\to\infty} x_n = x_0 \Rightarrow \liminf_n F(x_n) \geq u$ (but since F is monotone nondecreasing and continuous from the right) $\Rightarrow \liminf_n F(x_n) \leq F(x_0) \Rightarrow x_0 \in \{x : F(x) \geq u\} \Rightarrow \{x : F(x) \geq u\}$ contains its infimum (since closed) $\Rightarrow F(F^{-1}(u)) \geq u$

$F^{-1}(F(x)) \leq x$
Proof:

$F^{-1}(F(x)) = \inf\{z : F(z) \geq F(x)\} \Rightarrow x \in \{z : F(z) \geq F(x)\}$
$\Rightarrow F^{-1}(F(x)) \leq x$

Taking into account the propositions about CDFs and quantile functions, it is possible that $x \notin$ range(F)$\cup\{\inf$ range(F)$, \sup$ range(F)$\}$,e.g.$x < \inf$ range(F) and a CDF does not have to be strictly increasing (it can be flat). Thus, there are two cases when the inequalities are strict respectively:
$F(F^{-1}(u)) \neq u \wedge F^{-1}(F(x)) \neq x$

16 Task 58

Let F be a CDF and $X \sim F$.
Prove that: If F is continious, then $F\,(X) \sim U\,[0, 1]$

Proof:
$P(F(X) \leq x) \stackrel{1}{=} P(F^{-1}(F(X)) \leq F^{-1}(x)) \stackrel{2}{=} P(X \leq F^{-1}(x)) =$
$= F(F^{-1}(x)) \stackrel{3}{=} x \;\; \forall x \in (0, 1)$
Thus, $P(F(X) \leq x) = x \;\; \forall x \in (0, 1) \Rightarrow F\,(X) \sim U\,[0, 1]$

The proof is based on the following 3 propositions of quantile functions and CDF:
1) F is continious $\Leftrightarrow F^{-1}$ is strictly increasing on $[0, 1]$
2) $F^{-1}(F(x)) \leq x$. If F is strictly increasing, then $F^{-1}(F(x)) = x$
3) Since $x \in$ range(F) $\cup \{\inf$ range(F)$, \sup$ range(F)$\} \Rightarrow F(F^{-1}(x)) = x$

17 Task 59

Based on given data we can find distribution function F(X) =

$$\begin{aligned}0, (-\infty, x_1)\\ p_1, [x_1, x_2)\\ p_1 + p_2, [x_2, x_3)\\ \ldots\\ p_1 + \cdots + p_{n-1}, [x_{n-1}, x_n)\\ 1, [x_n, +\infty)\end{aligned}$$

Now F(x) is a discrete random variable with the set of values:0, $p_1, p_1 + p_2 ..., 1$

Let's find its distribution function F(F(X)) =

$$\begin{aligned} 0, (-inf, 0) \\ \frac{x_1 - m}{p - m}, [0, p_1) \\ \frac{x_2 - m}{p - m}, [p_1, p_2) \\ \dots \\ \frac{x_{n-1} - m}{p - m}, [p_{n-1}, p_n) \\ 1, [p_n, +\infty] \end{aligned}$$

where $p \to +\infty, m \to -\infty$

18 Task 60

The Pareto(a,b) distribution has cdf

$$F(x) = 1 - \left(\tfrac{b}{x}\right)^a,$$

where $x \geq 0$ and $a > 0$.

Let's derive $F^{-1}(U)$:

$$F(x) = 1 - \left(\frac{b}{x}\right)^a$$

$$\frac{b}{x} = (1 - F(x))^{1/a}$$

$$x = b(1 - F(x))^{-1/a}$$

$$x = F^{-1}(U) = b(1 - u)^{-1/a}$$

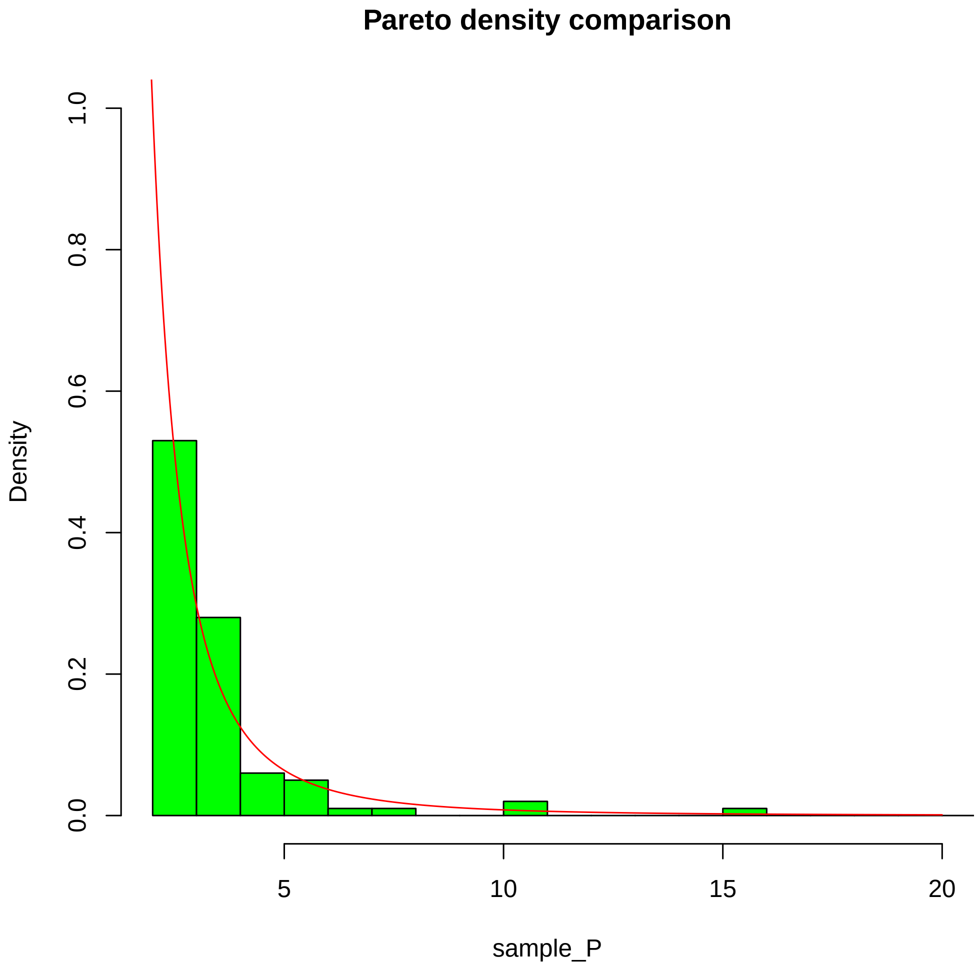
It is possible to simplify it further, because for U~$U(0,1)$, U is uniform if $1-U$ is uniform. So the shorter version:

$$x = F^{-1}(U) = bu^{-1/a}$$

Since we need to generate sample Pareto(2,2), $x \geq 2$.

```
pareto_random <- function(a,b,n){
  b* (runif(n))^(-1/a)
}


sample_P <- pareto_random(2,2,100)

sample_P

##   [1]  3.325486  3.699011  2.385296  3.726264
   2.257667
##   [6]  2.720053  2.441455  2.331075  2.664290
   2.648200
##  [11]  2.036477  2.197717  2.090563  2.102388
   3.653324
##  [16] 33.497095 10.317984  6.635197  3.043013
   3.562123
##  [21]  5.689599  2.694089  4.887840  2.391745
   2.321450
##  [26]  2.398294  2.735983  3.367176  2.300671
   5.124057
##  [31]  4.073326  3.229266  2.091941  4.693725
   3.264095
##  [36]  3.075510  3.284092  2.536293  3.543949
   2.617719
##  [41]  2.661073  2.940340  2.099374  3.315215
   2.131667
##  [46]  2.433068  2.736705  3.465463  3.228794
   2.890593
##  [51]  2.364830  4.610000  3.367984  2.046186
   5.560560
##  [56]  2.038646  3.691938  2.276004  2.739327
   2.234115
```

```
##  [61]  3.333617  3.495355  2.170560  3.953408
   10.055524
##  [66] 56.887410  2.097016  2.512217  2.165988
   5.698593
##  [71] 33.773782  2.824720  2.759635  2.884854
   2.406157
##  [76]  3.250843  2.184369  2.180328  2.057906
   2.061449
##  [81]  3.627290  5.727803  3.116235  2.385474
   2.658611
##  [86]  4.819117  2.187947  2.333142  2.080466
   7.955976
##  [91]  3.123840  3.195935  3.756834  2.524027
   4.482508
##  [96]  2.016394  3.398146 15.095531  3.247647
   2.059488

hist(sample_P, probability = TRUE, main="Pareto
   density comparison",
     col="green", xlim = c(2,20), ylim=c(0,1), breaks
        =50)

z <- seq(0,20,.01)

lines(z, 8/z^3, col="red")
```



The red line indicates the density of the Pareto distribution Pareto(2,2):

$$F'(x) = \left(1 - \left(\frac{2}{x}\right)^2\right)' = \frac{8}{x^3}$$

The pink histogram is the sample we have created.

19 Task 61

The generalized Pareto distribution has cdf

$$F(x) = 1 - (1 + \xi(x - \mu)/\sigma)^{-1/\xi}$$

for in the support of this distribution, where $\mu \in \mathbb{R}$ is the location parameter, $\sigma > 0$ is the scale parameter, and $\xi \in \mathbb{R}$ is the shape parameter.

19.1 a)

Let's express $(x-\mu)/\sigma$ as w.
Then:

$$\left(1+\frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}} = \frac{1}{(1+\xi w)^{\frac{1}{\xi}}}$$

It is known that:

$$\lim_{n\to 0}(1+n)^{\frac{1}{n}} = e$$

Therefore:

$$\lim_{\xi\to 0}\left(1-\left(1+\frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}}\right) = 1-\frac{1}{\lim\limits_{\xi\to 0}\left(1+\frac{\xi(x-\mu)}{\sigma}\right)^{\frac{1}{\xi}}}$$

$$= 1-\frac{1}{\lim\limits_{\xi\to 0}(1+\xi w)^{\frac{1}{\xi}}} = 1-\frac{1}{e^{w}} = 1-e^{-\frac{(x-\mu)}{\sigma}}$$

19.2 b)

Let's consider separated cases:

19.2.1 $\xi = 0$

As it is proved above, if ξ tends to zero, then Pareto cdf tends to $1-e^{-\frac{(x-\mu)}{\sigma}}$. Since this expression is cdf, we can conclude that:

$$0 \le 1-e^{-\frac{(x-\mu)}{\sigma}} \le 1$$

$$-1 \le -e^{\frac{(\mu-x)}{\sigma}} \le 0$$

$$0 \le e^{\frac{(\mu-x)}{\sigma}} \le 1$$

Since e in any power is always greater than zero, we can skip the left boundary of inequality:

$$e^{\frac{(\mu-x)}{\sigma}} \le 1$$

By taking logarithms we get:

$$\frac{(\mu - x)}{\sigma} \leq 0$$

$$\mu - x \leq 0$$

$$x \geq \mu$$

19.2.2 $\xi < 0$

Since the original expression is cdf, we get:

$$0 \leq 1 - \left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}} \leq 1$$

$$-1 \leq -\left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}} \leq 0$$

$$0 \leq \left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}} \leq 1$$

Since $\xi < 0$, if we raise inequality to the power $-\xi$, which is positive, we keep the signs:

$$0 \leq 1 + \frac{\xi(x-\mu)}{\sigma} \leq 1$$

$$-1 \leq \frac{\xi(x-\mu)}{\sigma} \leq 0$$

Since $\xi < 0$, if we multiply by ξ, we need to reserves signs ($\sigma > 0$, so it has no influence in this case):

$$-\frac{\sigma}{\xi} \geq x - \mu \geq 0$$

$$0 \leq x - \mu \leq -\frac{\sigma}{\xi}$$

$$\mu \leq x \leq \mu - \frac{\sigma}{\xi}$$

19.2.3 $\xi > 0$

Since the original expression is cdf, we get:

$$0 \leq 1 - \left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}} \leq 1$$

$$-1 \leq - \left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}} \leq 0$$

$$0 \leq \left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}} \leq 1$$

Since $\xi > 0$, if we raise inequality to the power $-\xi$, which is negative, we reverse the signs and skip zero boundary:

$$1 + \frac{\xi(x-\mu)}{\sigma} \geq 1$$

$$\frac{\xi(x-\mu)}{\sigma} \geq 0$$

Since $\xi > 0$ and $\sigma > 0$:

$$\xi(x-\mu) \geq 0$$

$$x - \mu \geq 0$$

$$x \geq \mu$$

20 Task 62

Let's derive $F^{-1}(U)$:

$$F(x) = 1 - \left(1 + \frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}}$$

$$\left(1+\frac{\xi(x-\mu)}{\sigma}\right)^{-\frac{1}{\xi}}=1-F(x))$$

$$1+\frac{\xi(x-\mu)}{\sigma}=(1-F(x))^{-\xi}$$

$$\frac{\xi(x-\mu)}{\sigma}=(1-F(x))^{-\xi}-1$$

$$x-\mu=\frac{\sigma}{\xi}((1-F(x))^{-\xi}-1)$$

$$x=\frac{\sigma}{\xi}((1-F(x))^{-\xi}-1)+\mu$$

$$x=F^{-1}(U)=\frac{\sigma}{\xi}((1-u)^{-\xi}-1)+\mu$$

It is possible to simplify it further, because for $U \sim U(0,1)$, U is uniform if $1-U$ is uniform. So the shorter version:

$$x=F^{-1}(U)=\tfrac{\sigma}{\xi}(u^{-\xi}-1)+\mu$$

Let's generate random sample where $\mu=2$, $\sigma=2$ and $\xi=-0.25$.
X is bounded as follows: $2 \leq x \leq 10$.

```
gen_pareto<-function(n,mu,sigma,xi) {
    (sigma/xi)*(runif(n)^(-xi)-1)+mu
}

sample_GP <- gen_pareto(1000,2,2,-0.25)

sample_GP
```

```
##     [1] 3.474516 2.840548 2.641039 2.278604 5.577784
    2.660753
##     [7] 2.049451 3.347238 2.326887 4.228189 3.450094
    4.529226
##    [13] 5.698206 5.035148 5.547596 5.457823 3.162447
    3.292550
##    [19] 2.194025 6.100951 3.275370 2.790491 3.394311
    2.757134
##    [25] 2.529139 2.549399 4.947406 3.909432 3.179921
    3.750516
##    [31] 2.011111 2.710867 4.561823 2.890057 2.426015
    2.763446
##    [37] 2.630090 3.943340 3.153412 5.789792 4.658175
    3.151569
##    [43] 5.746735 4.061786 2.376133 2.501868 3.139030
    2.354562
##    [49] 3.271412 2.798927 2.763107 2.230551 2.099779
    3.455865
##    [55] 2.459743 5.637272 7.835046 4.789167 4.203484
    2.290112
##    [61] 2.238763 2.083324 5.679221 6.419927 4.205111
    2.078037
##    [67] 5.619535 5.070083 3.124925 4.223013 4.883019
    4.737639
##    [73] 6.471362 5.680091 3.570539 6.393316 2.849872
    2.399777
##    [79] 3.086943 3.550473 2.597024 4.739776 4.121758
    5.205338
##    [85] 2.075808 2.656378 4.542407 3.216477 2.509087
    3.143113
##    [91] 3.570404 3.129606 2.582711 5.371767 4.969260
    2.447256
##    [97] 4.149812 2.839933 2.065578 4.325245 2.069404
    4.320463
##   [103] 2.113369 3.162802 3.826512 4.431588 8.028040
    5.601460
##   [109] 3.644487 6.083674 2.393526 5.490798 2.460713
    2.993527
##   [115] 6.094140 3.053870 2.392500 2.409059 3.438112
    2.142296
```

```
## [121] 4.004539 5.765013 2.217625 6.718129 4.744416
    2.504007
## [127] 2.025603 6.084996 3.982270 5.578242 4.509114
    2.764790
## [133] 6.488964 2.209868 2.024968 3.996512 2.525516
    2.034964
## [139] 2.333607 3.857899 3.526734 2.587769 4.964776
    4.636710
## [145] 2.513840 2.592186 3.084974 3.780922 4.929705
    2.945532
## [151] 3.110537 4.659183 3.946345 6.229295 3.059290
    5.215173
## [157] 3.108676 3.267533 4.703784 4.110553 2.691690
    2.094738
## [163] 3.088940 3.510445 2.082357 3.700231 2.484642
    2.116737
## [169] 2.517209 2.847857 3.530654 3.795298 4.432678
    3.149590
## [175] 2.929134 2.861825 4.768954 3.088738 4.763238
    2.104684
## [181] 3.542919 4.092591 6.070874 2.993487 3.597780
    2.529059
## [187] 2.968075 6.809822 2.043781 5.224898 3.476090
    2.731062
## [193] 2.974616 5.157360 5.524511 4.461732 2.119070
    5.868425
## [199] 2.698698 3.742485 4.699114 5.184597 3.128820
    3.655267
## [205] 4.210748 4.639725 2.530085 2.248086 2.810894
    4.788348
## [211] 4.389884 3.069083 2.073296 3.728072 2.834462
    3.846622
## [217] 5.286113 2.981391 3.461495 5.351623 2.116836
    5.176091
## [223] 3.285811 3.411434 3.694027 2.280674 4.456003
    6.841417
## [229] 3.562356 2.538710 2.227310 2.345944 2.248836
    3.577203
## [235] 2.527536 4.068605 2.453593 2.507928 2.066696
    2.532432
```

```
## [241] 3.778813 3.173390 2.586137 2.657739 5.351505
    2.393524
## [247] 2.148376 2.415684 2.013698 2.307222 2.467700
    4.204082
## [253] 3.899471 3.053210 3.262430 2.340121 2.435975
    5.049109
## [259] 3.166255 3.648814 3.327569 3.560538 3.199177
    4.262987
## [265] 2.319671 2.339365 3.689628 2.076043 2.934758
    3.363602
## [271] 4.367411 5.555639 4.024409 3.343254 4.688932
    2.624928
## [277] 4.772303 5.321933 3.156116 3.562813 4.417482
    3.392157
## [283] 2.683597 2.832095 4.137274 3.904843 2.081649
    3.203306
## [289] 2.472884 3.305081 3.110891 4.915670 3.403149
    2.379943
## [295] 2.341370 2.806758 3.320971 2.446554 5.141714
    2.843033
## [301] 2.645217 5.706051 2.867109 4.331631 4.646159
    4.518555
## [307] 3.043585 4.019827 4.837313 2.822732 3.866208
    3.230614
## [313] 5.195453 4.811070 2.680120 2.412096 3.075998
    3.009208
## [319] 4.957256 5.340998 3.407496 2.888664 2.218973
    2.194990
## [325] 3.906722 3.444186 2.292578 2.190558 5.750919
    2.299855
## [331] 6.429191 3.868919 3.628265 2.077151 4.041821
    2.992216
## [337] 4.455296 4.312943 6.089408 3.815495 2.300476
    5.584566
## [343] 5.498687 3.939153 2.904888 2.820192 2.600758
    2.759194
## [349] 2.623692 7.888710 3.604692 3.755378 3.453077
    3.781360
## [355] 2.762530 4.428118 6.197315 3.089107 2.323430
    2.870984
```

```
## [361] 7.483636 2.147169 5.899507 3.038598 3.541859
    2.048046
## [367] 4.878522 3.299703 2.666903 3.728832 7.650147
    2.383296
## [373] 5.091524 2.156186 2.796673 5.332345 2.221392
    2.688102
## [379] 4.755316 2.311005 4.617189 3.669176 4.002228
    4.034577
## [385] 3.496474 2.276940 2.809389 6.606859 2.872125
    3.467530
## [391] 3.889234 4.770750 2.906745 2.124980 5.167600
    2.663603
## [397] 2.734928 3.210313 4.283643 3.752934 2.522307
    3.192784
## [403] 4.223094 5.919139 2.728196 2.210849 2.183052
    5.095270
## [409] 2.180184 2.503064 2.204374 2.951857 4.519768
    2.197548
## [415] 3.439057 3.592837 6.262072 2.514218 2.118169
    3.825239
## [421] 2.428393 2.922445 2.094368 2.976054 3.485155
    3.263421
## [427] 4.054121 2.675193 2.050075 5.360786 8.161815
    4.974612
## [433] 2.414616 4.188634 2.218346 3.755816 2.507579
    3.008298
## [439] 6.924115 3.961138 4.234153 4.632106 4.321440
    3.171541
## [445] 2.207549 2.853550 2.935193 3.115872 3.923962
    3.830607
## [451] 3.259593 2.418666 3.185830 2.898541 2.575216
    2.479264
## [457] 3.426714 3.659672 2.844135 2.066358 2.311145
    6.035658
## [463] 2.553308 3.898465 5.648439 2.427528 2.824336
    2.060382
## [469] 2.198842 4.171358 4.515965 2.849638 3.801073
    2.652821
## [475] 2.971925 2.704148 3.781389 3.539483 2.387690
    4.984256
```

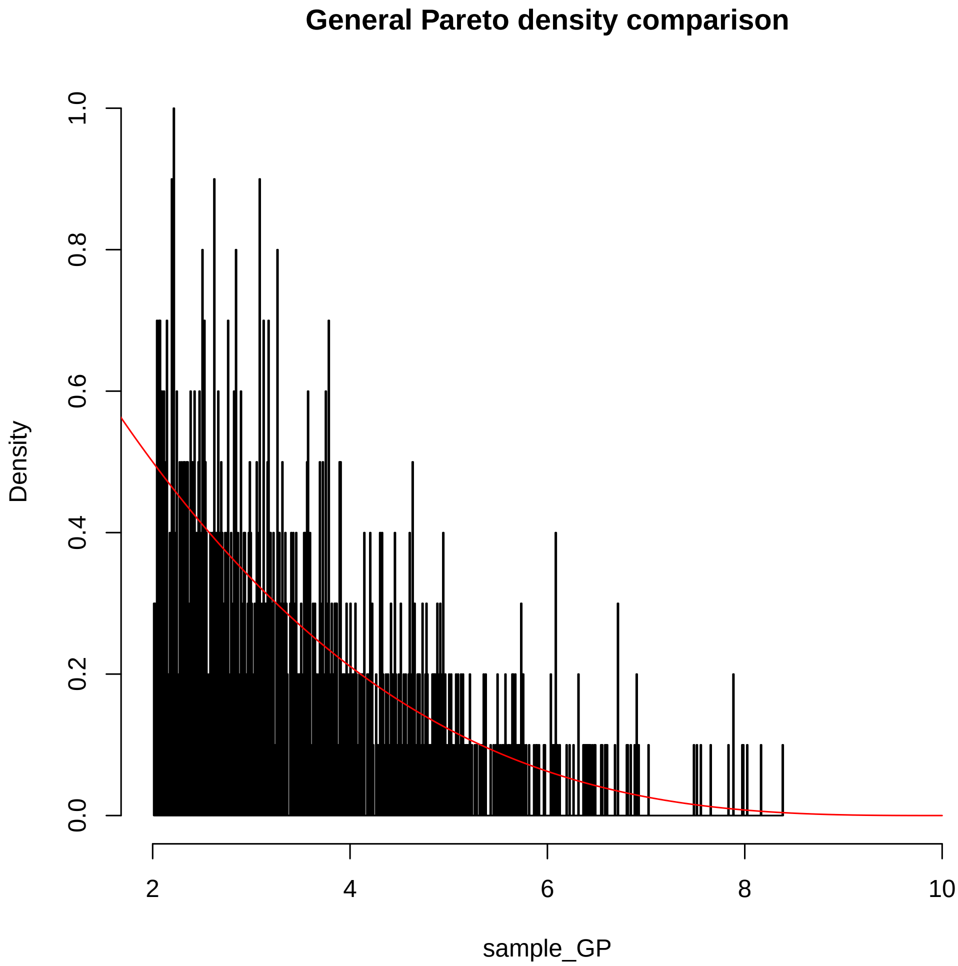
```
##  [481] 2.823733 2.616503 2.895500 4.734844 2.290617
    2.933593
##  [487] 3.699036 3.334990 2.434086 2.336736 2.443045
    4.147636
##  [493] 4.848407 4.625866 2.456049 4.493263 3.241880
    3.189571
##  [499] 2.542825 2.409104 2.083270 3.636278 3.623637
    5.237746
##  [505] 3.486311 5.114076 2.470490 8.382454 6.087986
    7.021947
##  [511] 3.894854 2.473218 2.694233 3.355555 3.900635
    5.670996
##  [517] 2.904699 3.262940 2.357447 3.836844 2.293893
    3.674843
##  [523] 4.445609 6.888558 4.461252 2.966909 4.308703
    2.897467
##  [529] 2.981127 4.410143 2.428303 3.493338 5.021129
    4.802489
##  [535] 2.620147 2.891682 3.061104 3.599633 2.615712
    2.464203
##  [541] 2.772749 3.848857 2.116911 2.981000 6.901438
    3.393946
##  [547] 2.628876 6.371317 2.618836 4.595465 4.263946
    2.128452
##  [553] 5.310001 2.358113 4.149694 4.837251 3.961436
    2.565105
##  [559] 2.251082 3.191104 3.053571 3.818805 5.738112
    4.607412
##  [565] 4.850636 4.566200 3.267230 6.437530 2.630478
    3.057103
##  [571] 2.336171 2.922412 3.284081 2.698834 4.487870
    2.501507
##  [577] 2.049975 2.276449 2.534250 2.545163 5.426397
    5.062428
##  [583] 6.069530 4.728996 2.369232 4.383519 7.555020
    2.366484
##  [589] 3.626952 6.033191 4.690368 2.613375 3.684984
    2.636450
##  [595] 2.108237 2.169241 2.703547 7.514569 2.265785
    6.719523
```

```
## [601] 3.124896 4.313649 2.464343 2.365217 3.556423
    3.252997
## [607] 3.422048 2.191615 3.583674 4.606778 2.961172
    6.055087
## [613] 5.217703 2.386229 4.655413 5.019824 2.709271
    3.277503
## [619] 3.342199 4.885938 3.138248 3.959043 2.212114
    3.331911
## [625] 3.070838 2.973351 2.270999 2.138219 4.942583
    3.632228
## [631] 2.797753 2.047017 5.073937 4.556156 3.563756
    2.625734
## [637] 5.655857 4.849777 4.108415 5.128670 3.083897
    3.077645
## [643] 2.266809 2.950610 5.819067 3.024602 2.328503
    2.080955
## [649] 3.265678 2.195774 4.118647 3.290271 3.533423
    3.087194
## [655] 3.710457 5.276995 2.181385 2.494674 4.271208
    2.747318
## [661] 2.725681 3.093808 3.358672 2.024345 3.593381
    2.207655
## [667] 5.122135 2.285499 4.900803 3.576567 4.408304
    2.244121
## [673] 5.645437 5.006655 5.755975 3.035966 4.916260
    2.539939
## [679] 5.375570 2.699017 2.105672 2.819907 3.532786
    4.042232
## [685] 2.485085 3.762315 4.944630 4.491970 3.275278
    4.123106
## [691] 2.056785 2.220471 4.394802 4.302733 2.848601
    2.765407
## [697] 4.200785 5.713109 3.558817 2.247998 5.966428
    2.217656
## [703] 4.908122 3.668553 2.445733 3.400185 3.574773
    2.280392
## [709] 2.624363 3.069609 3.543054 3.225106 4.637756
    5.656307
## [715] 6.387886 2.472565 2.640480 3.967604 6.583685
    3.260972
```

```
## [721] 2.517882 3.169731 3.002170 4.779129 2.869079
    2.525826
## [727] 6.902272 3.193257 2.392767 3.318910 2.758043
    3.364061
## [733] 2.384273 4.476880 4.946710 4.598710 4.181630
    2.771906
## [739] 3.027803 2.490825 3.989354 5.882260 3.722743
    4.099266
## [745] 2.910487 2.572891 2.171374 2.744731 6.409089
    3.312281
## [751] 3.815749 2.381260 3.172143 5.739924 2.069410
    4.540049
## [757] 4.351612 4.865522 3.427363 2.237709 4.029233
    2.194333
## [763] 2.676004 7.971411 4.347998 2.606807 3.223385
    3.178187
## [769] 4.827970 2.506844 4.321304 3.560690 2.247674
    2.925365
## [775] 2.620487 2.044629 3.784397 2.422712 3.078373
    4.926803
## [781] 3.088033 4.053873 2.356895 4.912841 2.155247
    4.001078
## [787] 2.086593 6.452034 3.171714 4.455485 2.489533
    3.647002
## [793] 4.532969 3.124224 3.414948 2.711851 4.307497
    3.708676
## [799] 2.733970 3.729437 3.186874 3.344152 2.827550
    3.445290
## [805] 4.084872 2.146815 2.172964 4.603346 4.700507
    2.315330
## [811] 2.982137 2.609036 2.271766 2.149518 4.052549
    3.265350
## [817] 5.473389 3.767698 5.008969 2.994517 2.492497
    4.412838
## [823] 5.669933 2.669998 3.583186 3.090920 6.129136
    3.317013
## [829] 2.913643 3.120007 3.698559 5.979882 3.085827
    3.789407
## [835] 4.036974 2.559131 2.075265 3.574036 2.464165
    5.746925
```

```
## [841] 3.422883 3.927699 5.483781 2.349166 3.506995
    2.667507
## [847] 5.739369 2.946074 4.078041 6.682190 2.131964
    2.821530
## [853] 6.368571 3.756492 6.815787 3.851675 3.690507
    2.529525
## [859] 3.690916 3.355539 3.214416 4.178132 4.797706
    3.177853
## [865] 4.626531 2.176988 2.847259 2.672105 3.163496
    7.980955
## [871] 4.637273 6.311352 3.097398 2.195810 3.259591
    2.215258
## [877] 3.852841 2.210725 3.933675 4.137387 2.854929
    3.891398
## [883] 4.679301 2.804783 3.788084 2.738801 4.606939
    3.419192
## [889] 3.401497 2.148303 3.541880 3.459067 2.626132
    2.782243
## [895] 2.869463 2.629886 3.764889 3.223609 2.075451
    6.315901
## [901] 4.304455 2.123006 3.783348 6.596709 4.142522
    4.689998
## [907] 2.609466 2.310700 3.037466 3.145868 2.212813
    2.348573
## [913] 3.797558 2.045394 6.718346 3.280627 2.424404
    4.864891
## [919] 4.887465 4.421141 5.509102 2.939275 3.436881
    4.572605
## [925] 2.478387 5.725133 2.247915 2.894986 3.305997
    3.712818
## [931] 3.752095 2.764880 3.187808 3.848971 4.938983
    3.123689
## [937] 2.068243 2.013631 2.322454 2.407918 2.350934
    2.408640
## [943] 2.095450 3.894453 2.582202 3.874310 3.733524
    7.881466
## [949] 2.707262 2.229114 2.754233 4.365650 5.269048
    2.160979
## [955] 2.726110 2.211211 2.127026 4.854564 2.095767
    3.735812
```

```
## [961] 2.599588 4.310953 3.500606 2.142004 3.907642
   4.330616
## [967] 2.673515 5.021816 2.642106 2.096222 3.707857
   2.982106
## [973] 2.329659 3.287801 4.450996 2.177266 3.866513
   3.505692
## [979] 2.711207 3.517094 5.147169 3.047757 3.319550
   2.543482
## [985] 3.316101 2.666433 2.488083 3.330571 3.228433
   2.402822
## [991] 2.766744 3.306097 3.724382 5.086582 6.543060
   6.556793
## [997] 4.109006 2.126232 4.663105 2.383717

hist(sample_GP, probability = TRUE, main="General
  Pareto density comparison",
     col="green", xlim = c(2,10), ylim=c(0,1), breaks
        =500)

z <- seq(0,10,.01)

lines(z, 0.5*(1.25-0.125*z)^3, col="red")
```



The red line indicates the density of the General Pareto distribution GPD(2,2,-0.25):

$$F'(x) = \left(1 - \left(1 - \frac{0.25}{2}(x-2)\right)^{\frac{1}{0.25}}\right)' = (1 - (1.25 - 0.125x)^4)' = 0.5 * (1.25 - 0.125x)^3$$

21 Task 63

```
discrete_inverse_transform <- function(
   probability_vector)
```

```
{
  U  <- runif(1)
  if(U <= probability_vector[1]){
    return(1)
  }
  for(k in 2:length(probability_vector)) {
    if(sum(probability_vector[1:(k-1)]) < U && U <=
       sum(probability_vector[1:k]))
    {
      return(k)
    }
  }
}


probability_vector <- c(0.1, 0.2, 0.2, 0.2, 0.3)

x_vector <- c(0, 1, 2, 3, 4)

n <- 1000


sample_vector_1 <- rep(0,n)

frequency_vector_1 <- rep(0,length(x_vector))

for (i in 1:n)
{
  k <- discrete_inverse_transform(probability_vector)
  sample_vector_1[i] <- x_vector[k]
  frequency_vector_1[k] <- frequency_vector_1[k] + 1
}


relative_frequency_vector_1 <- frequency_vector_1 / n


cat("Theoretical probability vector: ",
   probability_vector, '\n')

## Theoretical probability vector:  0.1 0.2 0.2 0.2
   0.3
```

```
cat("Empirical probability vector: ",
   relative_frequency_vector_1, '\n')

## Empirical probability vector:  0.105 0.186 0.195
   0.203 0.311

sample_vector_2 <- sample(x_vector, 1000, replace =
   TRUE)

frequency_vector_2 <- rep(0,length(x_vector))

for (i in 1:n)
{
  k <- sample_vector_2[i] + 1
  frequency_vector_2[k] <- frequency_vector_2[k] + 1
}

relative_frequency_vector_2 <- frequency_vector_2 / n

cat("Relative frequency vector with sample function:
   ", relative_frequency_vector_2, '\n')

## Relative frequency vector with sample function:
   0.209 0.203 0.172 0.205 0.211
```