Exercise 2 - Random Number Generation through CDF and acceptance-rejection sampling

107.330 - Statistical Simulation and Computerintensive Methods, WS24

12433688 - Yash Lucas

22.10.2024

Contents

Task 1 2
Task 2 8
Task 3 12

Task 1

Summarise the concept of pseudo-random number generation with Linear Congruential Random Number Generation Algorithm using the available code examples from the course to create a working code example and simulate one pseudo-random sample.

Try out and compare different values of m, a to illustrate the behavior of the method as described on slide 18.

The LCG (Linear congruential Generator) algorithm is a simple and very well known PRNG (pseudo-random number generator). It works by recursively applying a modulo to some large integer m. The algorithm is defined by the recurrence relation $x_{n+1} = (a \cdot x_n + c) \mod m$.

Where:

- m is the modulus (a large integer)
- a is the multiplier ($\approx \sqrt{m}$)
- c is the increment ($0 \leq c < m$)
- d is the seed or starting value

Below, the Linear Congruential Random Number Generation Algorithm is presented. - The function takes as initial values the sample n, - the m and a parameter, which defines the range of numbers - the increment c - the starting value d and the - a1 where uniform pseudo random numbers are generated.

```r
# function for Linear Congruential Random Number Generator

lc_gen <- function(n, m, a, c=0, d){
  a1 <- numeric(n)
  for (i in 1:n){
    d <- (a * d + c) %% m
    a1[i] <- d / m
  }
  return(a1)
}
```

I have used the function below to test 3 samples

Sample -1

```r
n <- 500
m <-  12433688
a <- round(sqrt(m)-2,2)
c <-2
d<- 25
sm1<-lc_gen(n,m,a,c,d)
round((sm1),3)
```

```
##   [1] 0.007 0.972 0.049 0.035 0.956 0.532 0.653 0.138 0.262 0.600 0.164 0.383
##  [13] 0.008 0.858 0.692 0.404 0.436 0.835 0.050 0.230 0.921 0.250 0.803 0.655
##  [25] 0.298 0.961 0.489 0.531 0.563 0.602 0.972 0.992 0.498 0.527 0.503 0.265
##  [37] 0.916 0.804 0.876 0.703 0.204 0.939 0.908 0.464 0.425 0.991 0.792 0.683
##  [49] 0.604 0.147 0.608 0.416 0.191 0.386 0.505 0.864 0.481 0.609 0.102 0.569
##  [61] 0.521 0.989 0.511 0.478 0.326 0.913 0.611 0.441 0.552 0.884 0.815 0.674
##  [73] 0.126 0.676 0.174 0.661 0.431 0.887 0.359 0.394 0.429 0.375 0.612 0.580
##  [85] 0.538 0.165 0.685 0.716 0.008 0.166 0.589 0.263 0.564 0.423 0.988 0.098
##  [97] 0.909 0.239 0.340 0.967 0.124 0.653 0.275 0.854 0.737 0.314 0.043 0.501
## [109] 0.075 0.048 0.251 0.563 0.488 0.853 0.913 0.740 0.843 0.748 0.117 0.417
## [121] 0.792 0.559 0.790 0.343 0.243 0.101 0.827 0.114 0.324 0.660 0.182 0.786
## [133] 0.546 0.922 0.307 0.615 0.984 0.544 0.584 0.003 0.341 0.463 0.200 0.845
## [145] 0.746 0.446 0.178 0.364 0.039 0.679 0.094 0.372 0.046 0.099 0.429 0.961
## [157] 0.893 0.372 0.158 0.476 0.348 0.894 0.093 0.779 0.754 0.378 0.852 0.221
## [169] 0.172 0.890 0.341 0.420 0.573 0.787 0.170 0.422 0.486 0.510 0.390 0.073
## [181] 0.503 0.794 0.514 0.335 0.254 0.750 0.951 0.515 0.182 0.491 0.243 0.587
## [193] 0.166 0.791 0.958 0.195 0.097 0.150 0.464 0.947 0.570 0.901 0.228 0.763
## [205] 0.711 0.933 0.468 0.436 0.554 0.318 0.975 0.712 0.214 0.185 0.711 0.452
## [217] 0.829 0.370 0.790 0.954 0.223 0.950 0.916 0.328 0.737 0.212 0.545 0.800
## [229] 0.897 0.632 0.658 0.473 0.268 0.630 0.328 0.186 0.304 0.909 0.089 0.933
## [241] 0.259 0.703 0.902 0.418 0.380 0.936 0.962 0.788 0.772 0.966 0.257 0.183
## [253] 0.430 0.009 0.288 0.106 0.952 0.314 0.777 0.655 0.560 0.757 0.645 0.927
## [265] 0.730 0.723 0.406 0.661 0.048 0.725 0.691 0.004 0.357 0.403 0.618 0.927
## [277] 0.829 0.772 0.751 0.554 0.572 0.610 0.783 0.056 0.526 0.390 0.986 0.961
## [289] 0.013 0.010 0.581 0.125 0.989 0.491 0.618 0.421 0.873 0.115 0.553 0.126
## [301] 0.556 0.128 0.791 0.534 0.303 0.572 0.701 0.525 0.900 0.706 0.352 0.057
## [313] 0.728 0.566 0.501 0.498 0.942 0.963 0.627 0.401 0.962 0.073 0.286 0.007
## [325] 0.260 0.663 0.164 0.667 0.195 0.606 0.056 0.986 0.389 0.460 0.119 0.057
## [337] 0.334 0.598 0.176 0.017 0.758 0.440 0.792 0.418 0.012 0.708 0.437 0.490
## [349] 0.304 0.094 0.787 0.756 0.685 0.803 0.439 0.553 0.428 0.827 0.589 0.164
## [361] 0.604 0.374 0.871 0.509 0.001 0.418 0.972 0.763 0.450 0.983 0.924 0.870
## [373] 0.940 0.760 0.927 0.289 0.549 0.098 0.944 0.188 0.704 0.700 0.527 0.261
## [385] 0.251 0.061 0.726 0.229 0.865 0.695 0.102 0.954 0.096 0.480 0.899 0.904
## [397] 0.159 0.449 0.211 0.464 0.010 0.971 0.167 0.113 0.966 0.495 0.615 0.842
## [409] 0.657 0.056 0.764 0.408 0.007 0.331 0.152 0.796 0.525 0.178 0.315 0.556
## [421] 0.366 0.348 0.888 0.155 0.868 0.388 0.093 0.551 0.943 0.433 0.567 0.479
## [433] 0.598 0.517 0.750 0.516 0.212 0.124 0.518 0.955 0.704 0.384 0.664 0.251
## [445] 0.702 0.861 0.906 0.491 0.025 0.370 0.507 0.625 0.978 0.898 0.738 0.409
## [457] 0.819 0.352 0.196 0.445 0.879 0.954 0.798 0.974 0.241 0.159 0.436 0.172
## [469] 0.450 0.116 0.892 0.857 0.264 0.509 0.495 0.107 0.751 0.418 0.059 0.166
## [481] 0.524 0.064 0.352 0.208 0.120 0.130 0.355 0.479 0.053 0.856 0.899 0.841
## [493] 0.242 0.351 0.111 0.774 0.872 0.378 0.382 0.735
```

Sample - 2

```r
n <- 500
m <- 50
a <- 4
c <- 1
d <- 3
sm2<-lc_gen(n,m,a,c,d)
round((sm2),3)
```

```
##   [1] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
##  [16] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
##  [31] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
##  [46] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
##  [61] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
##  [76] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
##  [91] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [106] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [121] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [136] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [151] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [166] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [181] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [196] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [211] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [226] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [241] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [256] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [271] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [286] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [301] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [316] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [331] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [346] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [361] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [376] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [391] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [406] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [421] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [436] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [451] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [466] 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06
## [481] 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26 0.06 0.26
## [496] 0.06 0.26 0.06 0.26 0.06
```

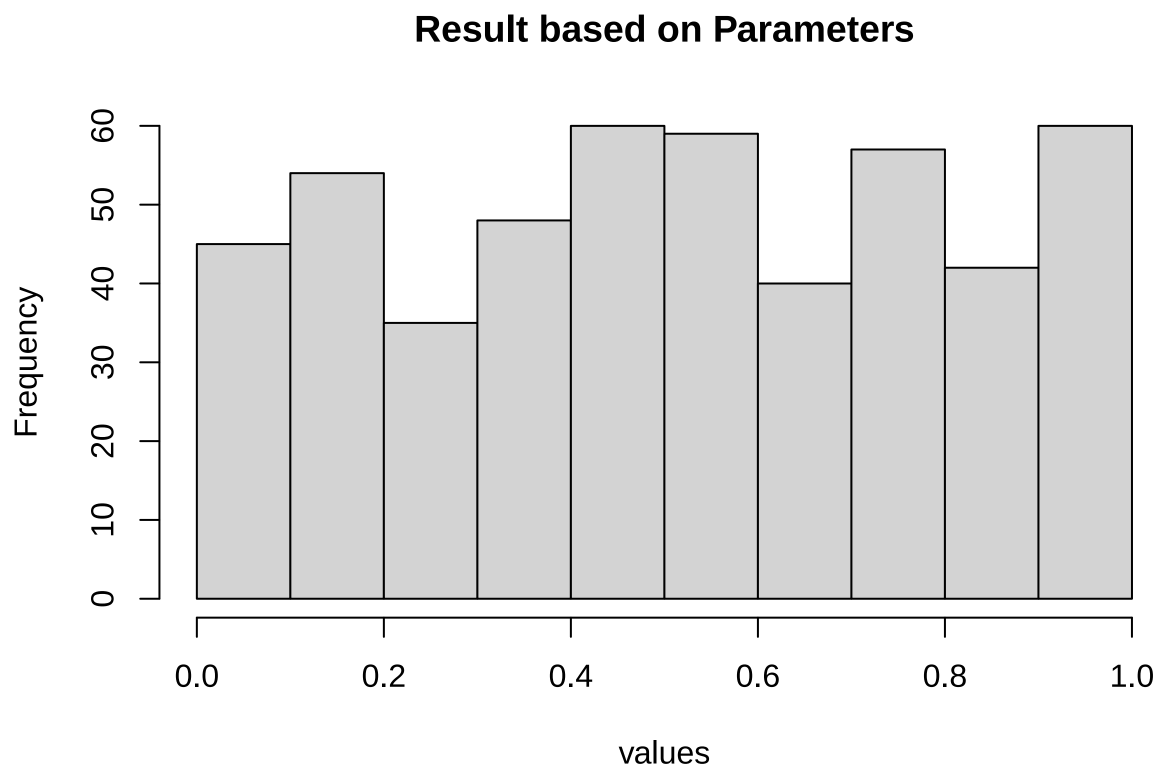
Sample - 3

```r
n <- 500
m <- 37
a <- 19
c <- 1
d <- 3
sm3<-lc_gen(n,m,a,c,d)
round((sm3),3)
```

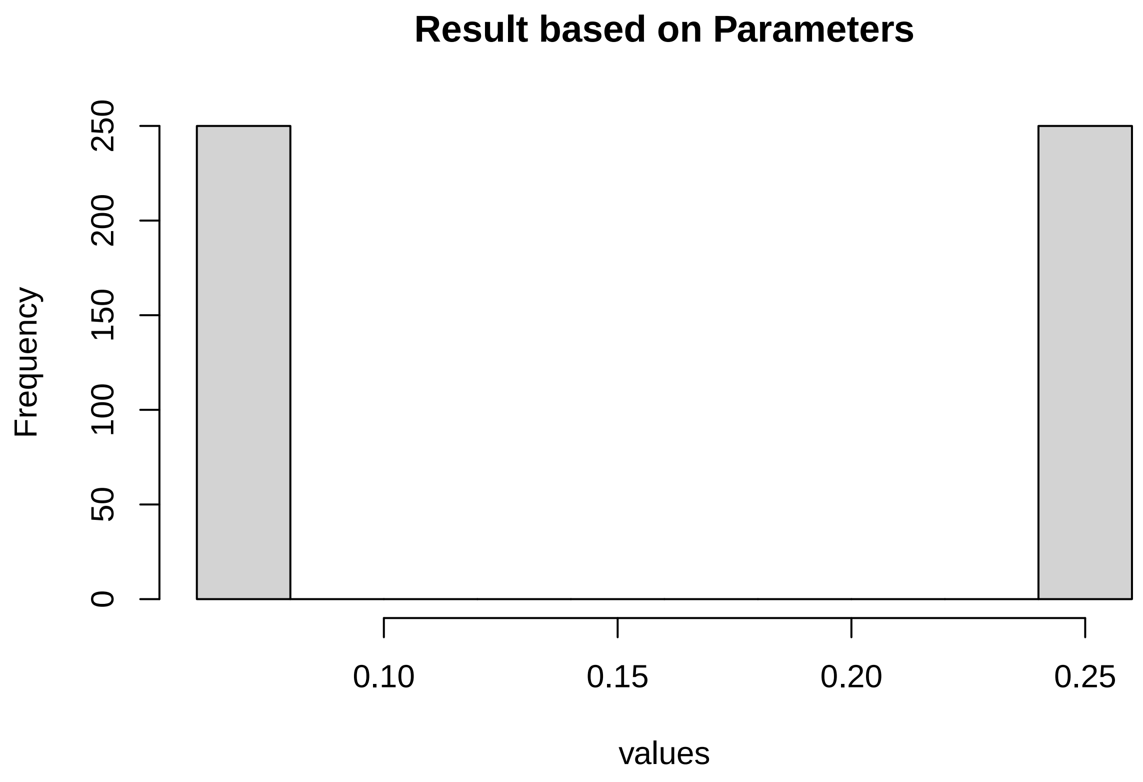
```
##   [1] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
##  [13] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
##  [25] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
##  [37] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
##  [49] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
##  [61] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
##  [73] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
##  [85] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
##  [97] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [109] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [121] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [133] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [145] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [157] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [169] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [181] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [193] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [205] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [217] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [229] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [241] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [253] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [265] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [277] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [289] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [301] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [313] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [325] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [337] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [349] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [361] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [373] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [385] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [397] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [409] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [421] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [433] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [445] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [457] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486 0.270 0.162 0.108 0.081
## [469] 0.568 0.811 0.432 0.243 0.649 0.351 0.703 0.378 0.216 0.135 0.595 0.324
## [481] 0.189 0.622 0.838 0.946 0.000 0.027 0.541 0.297 0.676 0.865 0.459 0.757
## [493] 0.405 0.730 0.892 0.973 0.514 0.784 0.919 0.486
```

If we compare the above 3 samples, we get the follwoing histogram

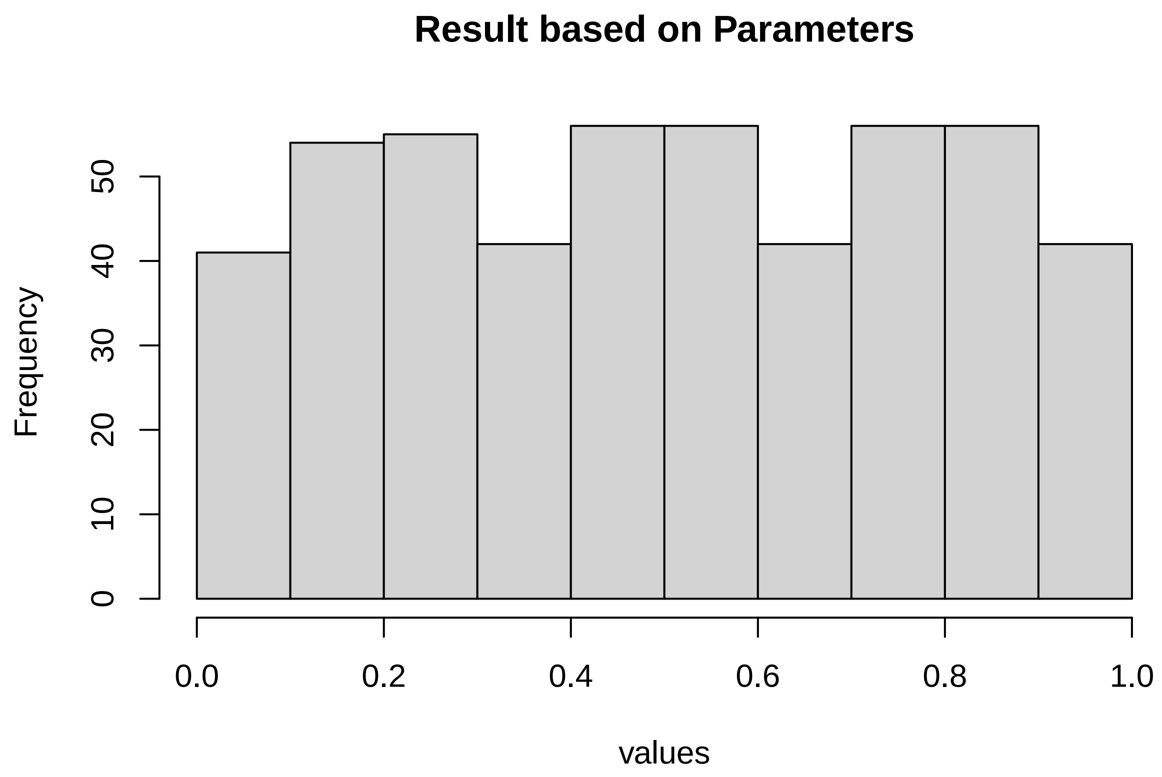
```r
hist(round((sm1),3),main="Result based on Parameters",xlab="values",ylab="Frequency")
```



```r
hist(round((sm2),3),main="Result based on Parameters",xlab="values",ylab="Frequency")
```



```r
hist(round((sm3),3),main="Result based on Parameters",xlab="values",ylab="Frequency")
```



If we compare the above 3 samples, the 1st and 3rd sample has good parameters and the 2nd has poor parameters.

We can observe that the choice of parameters significantly impacts the quality of the generated pseudo-random numbers.

- m: A large prime number for m generally produces better results, maximizing the cycle length of the sequence.
- a: The multiplier a should be carefully chosen, with values close to $\sqrt{m}$ often working well.
- c: With poor parameters, the sequence may repeat quickly, leading to non-random behavior.

Task 2

The exponential distribution has the following cdf: $F(x) = 1 - e^{-\lambda x}$

Assume you can generate easily uniform random variables. How can you obtain then a random sample from the exponential distribution?

- Write down the mathematical expression for this.
- Write an R function which takes as input the number of samples required and the parameter λ. Use the function runif to create the uniform random variables.
- For three different values of λ create 1000 random samples and evaluate the quality of your random number generator using qq-plots which compare against a real exponential distribution with the specified parameter λ.

In order to obtain a random sample from the exponential distribution, we are going to use the inversion method. We have the cdf F(x) and we proceed to the next 3 steps:

1. Compute the quantile function F_x^{-1}
2. Generate a u ~ unif[0, 1]
3. Make the transformation $x = F_x^{-1}$

Therefore, the mathematical expression for the cdf F(x) will be: $x = -\frac{1}{\lambda}\ln(1 - F)$

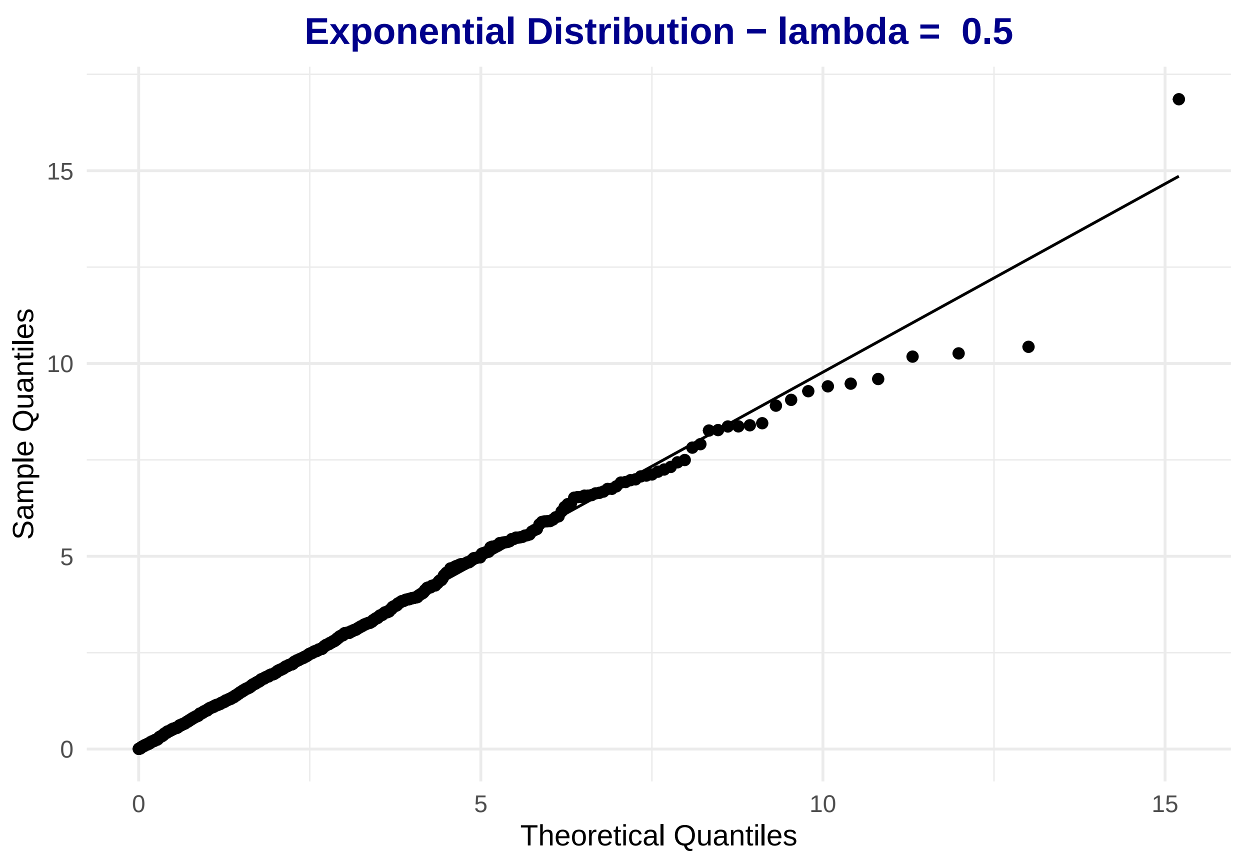
```
exponential_samp <- function(n, lambda) {
    u <- runif(n)
    x <- -1/lambda * log(1 - u)
    return(x)
}
```

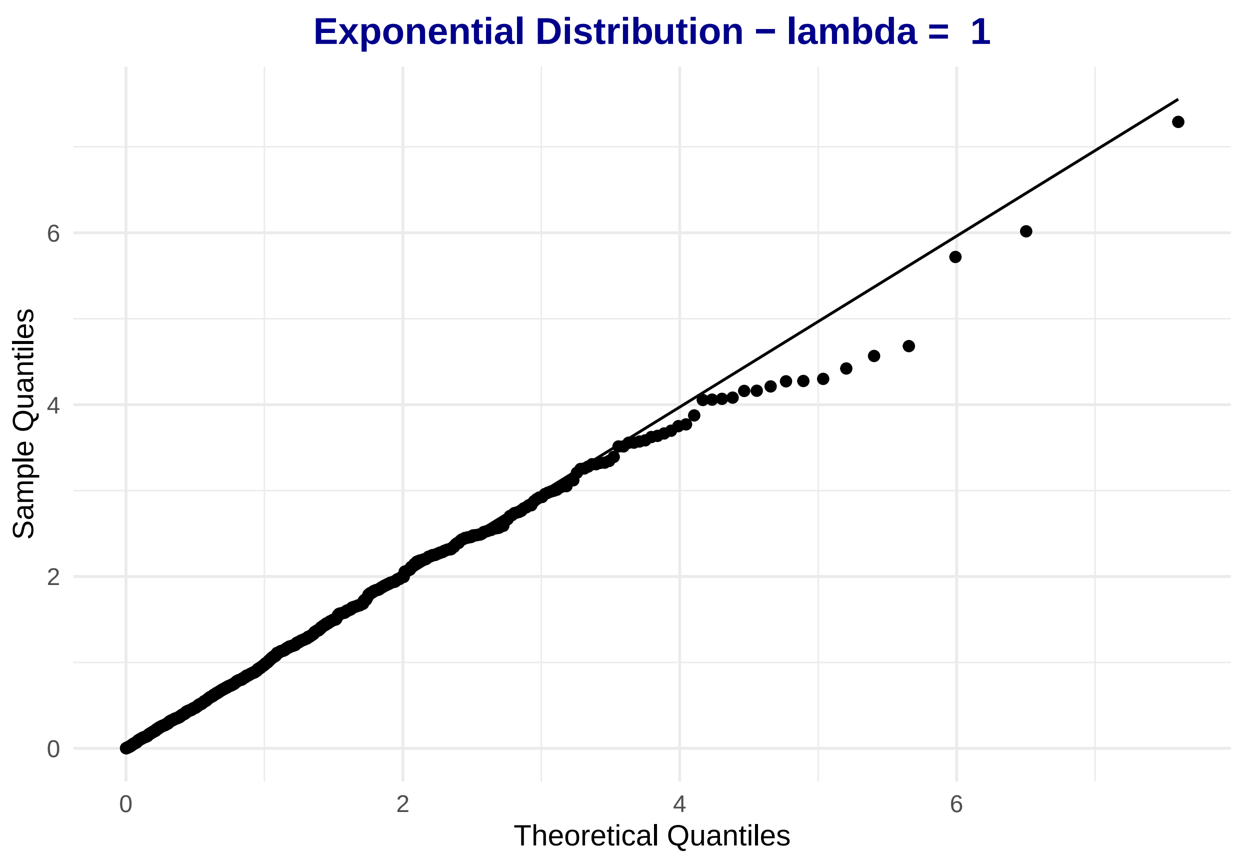
After, I have taken 3 different values for the λ and 1000 random samples. For the λ, the values I have used are 0.5, 1 and 1.5 are used.

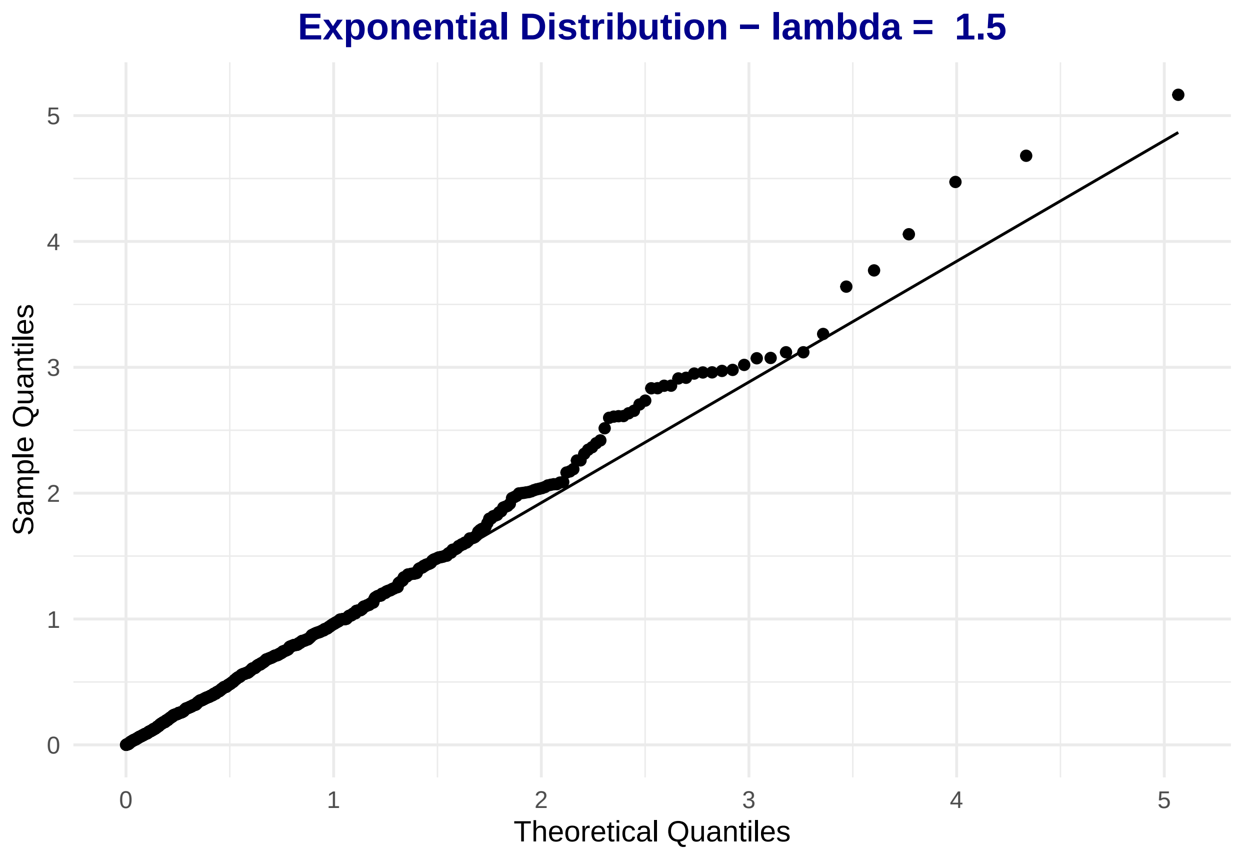
We have also used rexp(n,rate)$^{-1}$ where the rate is the rate parameter λ and n is the number of samples. This function is prebulit in R.

I have also also generated 3 QQ-plots, for each λ. The plots helps us compare the quality of our custom-generated exponential samples against the theoretical exponential distributions.

```
library(ggplot2)

generate_qqplot <- function(rate_param) {
    # Generate samples using the exponential_samp function
    df <- data.frame(custom_samples = exponential_samp(1000, rate_param), theoretical_samples = rexp(10
    ggplot(df, aes(sample = custom_samples)) +
        stat_qq(distribution = qexp, dparams = list(rate = rate_param)) +
        stat_qq_line(distribution = qexp, dparams = list(rate = rate_param)) +
        ggtitle(paste("Exponential Distribution - lambda = ", rate_param)) +
        xlab("Theoretical Quantiles") +
        ylab("Sample Quantiles") +
        theme_minimal() +  # Apply a clean theme
        theme(
            plot.title = element_text(color = "darkblue", size = 14, face = "bold", hjust = 0.5)
        )
}

# New variable names for the lambda values
rate_params <- c(0.5, 1, 1.5)  # Changed from lambda_values to rate_params

# Generate and display QQ plots with the new variable names
plots <- lapply(rate_params, generate_qqplot)
for (plot in plots) {
    print(plot)
}
```


Exponential Distribution − lambda = 0.5
Sample Quantiles
Theoretical Quantiles
0
5
10
15
0
5
10
15


Exponential Distribution − lambda = 1
Sample Quantiles
0
2
4
6
Theoretical Quantiles
0
2
4
6



From the above QQ-plots, it can be directly analyzed that our custom random number generator is producing the samples that closely matches the theoretical exponential distribution(repx) for all three values of λ that we have tested.

The points in QQ-plots lies in the same area where the line exist.

Task 3

The Beta distribution has the following pdf: $f(x;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$

Write a function which uses an acceptance-rejection approach to sample from a beta distribution. Argue what is a natural candidate for a proposal distribution.

- Focus first on the case where both parameters of the beta distribution are 2. Find what is a good constant to keep the rejection proportion small.
- Write a function based on this approach to sample from an arbitrary beta distribution and which adapts the constant to the user specified parameters assuming that both parameters are larger than 1.

All functions above should take as input the number of samples to be created and if applicable also the parameters of the beta distribution.

Evaluate the quality of your different functions visually.

The f(x) is known, according to the instructions of the exercise, and it is a beta distribution with the following pdf: $f(x;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$. Now, we have to find a density function g(x) so that the following inequality is fulfilled ($f(x) \leq cg(x)$). Therefore, we are able to perform the acceptance-rejection method and generate the proper values. In our implementation, the uniform distribution has been chosen. Thus, the g(x) will be: $g(x) = \frac{1}{b-a}$ where $x \in [a, b]$.

The uniform distribution, despite not being as efficient as other distributions, is characterized for its simplicity and its flexibility and versatility. In our case, we know the boundaries of the x variable, which are [0,1], since we have the pdf of a beta distribution. Thus, we can easily apply g(x). In terms of flexibility, by slightly changing the parameters a and b from $f(x; \alpha, \beta)$, our curve will be different every time which means that we can easily adapt g(x) by choosing an appropriate value for the parameter c.

Below, the acceptance rejection function is presented.

These functions provide a flexible way to sample from Beta distributions using the acceptance-rejection method. The efficiency of the method depends on how close the proposal distribution is to the target distribution. For Beta distributions with parameters far from 1, the rejection rate may become high, and other methods (like inverse transform sampling) might be more efficient.

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

# Function to sample from Beta distribution using rejection sampling
sample_beta <- function(n, alpha, beta) {
    if (alpha <= 1 || beta <= 1) {
        stop("Invalid parameters: alpha and beta must be greater than 1.")
    }

    samples <- numeric(n)
    accepted <- 0

    mode <- (alpha - 1) / (alpha + beta - 2)

    c <- dbeta(mode, alpha, beta)

    while (accepted < n) {
        y <- runif(1)
        u <- runif(1)

        if (u <= dbeta(y, alpha, beta) / c) {
            accepted <- accepted + 1
            samples[accepted] <- y
        }
    }

    return(samples)
}
```

This function adjusts the constant c to the user-specified parameters α and β of the Beta distribution. It calculates the mode of the Beta distribution and uses the density at that point as the value for c. The $Uniform(0, 1)$ distribution remains a natural choice for the proposal distribution for any $Beta(\alpha, \beta)$ with $\alpha > 1$ and $\beta > 1$, as the Beta distribution is always defined on $[0, 1]$.

Generating samples

```r
s1_2_2<-round(s1 <- sample_beta(500, 2, 2),3)
s2_5_6<-round(s2<- sample_beta(500, 5, 6),3)
```

```r
# Generate theoretical samples for comparison
be_2_2<-round(be1 <- rbeta(500, 2, 2),3)
be_5_6<-round(be2 <- rbeta(500, 5, 6),3)
```

Creating QQ-Plots and Histograms for Beta samples

```r
p_qq_1 <- function(samples, theoretical, alpha, beta) {
    ggplot(data.frame(samples = samples, theoretical = theoretical), aes(sample = samples)) +
        stat_qq(distribution = qbeta, dparams = list(shape1 = alpha, shape2 = beta)) +
        stat_qq_line(distribution = qbeta, dparams = list(shape1 = alpha, shape2 = beta), color = "yell
        ggtitle(paste("QQ-Plot for Beta(", alpha, ", ", beta, ")", sep = "")) +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")
}

p_hist_1 <- function(samples, alpha, beta) {
    ggplot(data.frame(samples = samples), aes(x = samples)) +
        geom_histogram(aes(y = ..density..), bins = 20, color = "black") +
        stat_function(fun = dbeta, args = list(shape1 = alpha, shape2 = beta), color = "blue", size = 1
        ggtitle(paste("Histogram of Samples from Beta(", alpha, ", ", beta, ")", sep = "")) +
        xlab("Samples") +
        ylab("Density")
}
```

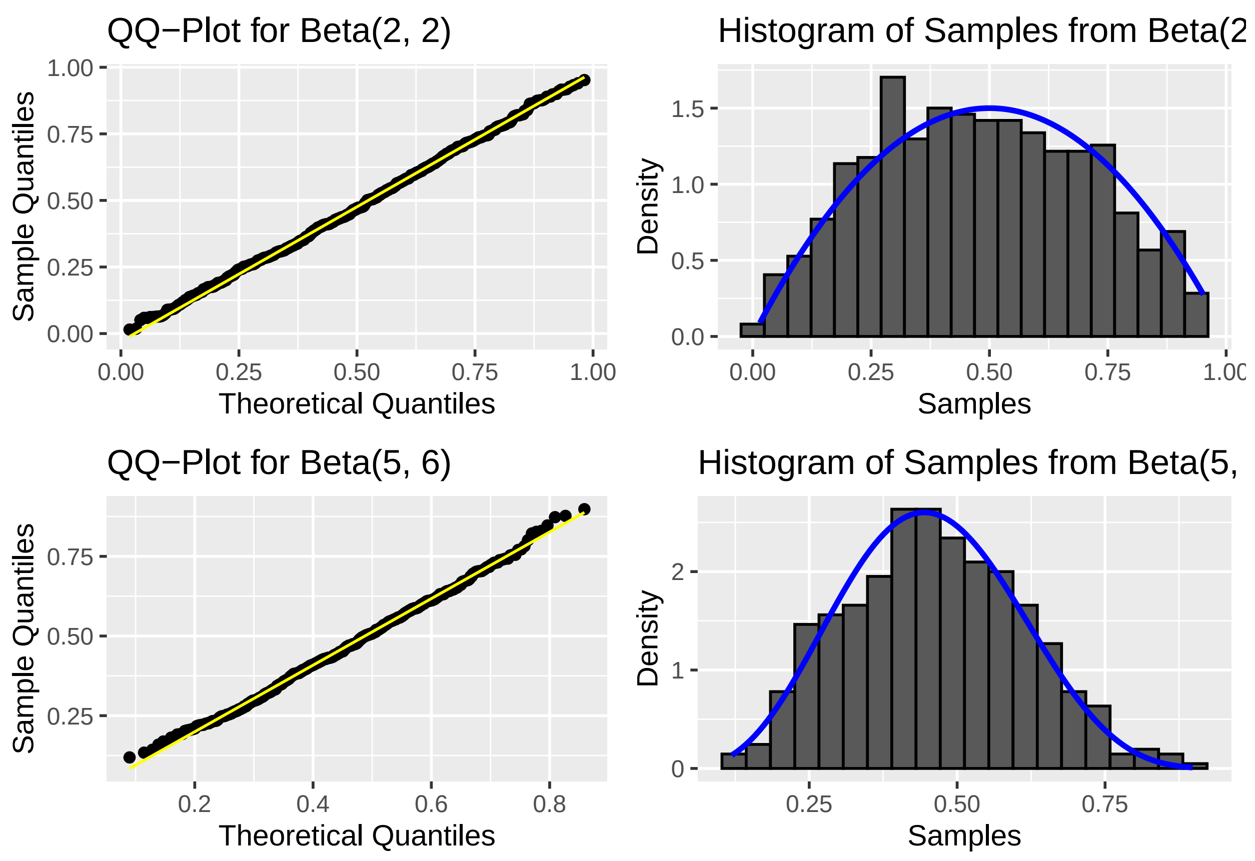
Visualizing plots for Beta (2,2) and (5,6)

```r
plot1_qq <- p_qq_1(s1_2_2, be_2_2, 2, 2)
plot2_qq <- p_qq_1(s2_5_6, be_5_6, 5, 6)

plot1_hist <- p_hist_1(s1_2_2, 2, 2)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
plot2_hist <- p_hist_1(s2_5_6, 5, 6)

grid.arrange(plot1_qq, plot1_hist, plot2_qq, plot2_hist, nrow = 2)
```

```
## Warning: The dot-dot notation (‘..density..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(density)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



- QQ Plot for Beta Distribution (2,2): The points in the QQ plot lie very close to a straight line, indicating a strong linear relationship between the theoretical quantiles of the Beta distribution and the sample quantiles. This suggests that the sample data likely comes from a Beta distribution.

- QQ Plot for Beta Distribution (5,6): The points in the QQ plot also align closely with a straight line, although there is a slight deviation in the upper tail. This tail divergence may suggest some asymmetry or heavier tails in the actual data compared to the theoretical Beta distribution, despite the overall pattern suggesting a Beta distribution.