Lab 1

Daniel Herzegh (danhe178) and Richard Friberg (rical803)

2017-09-26

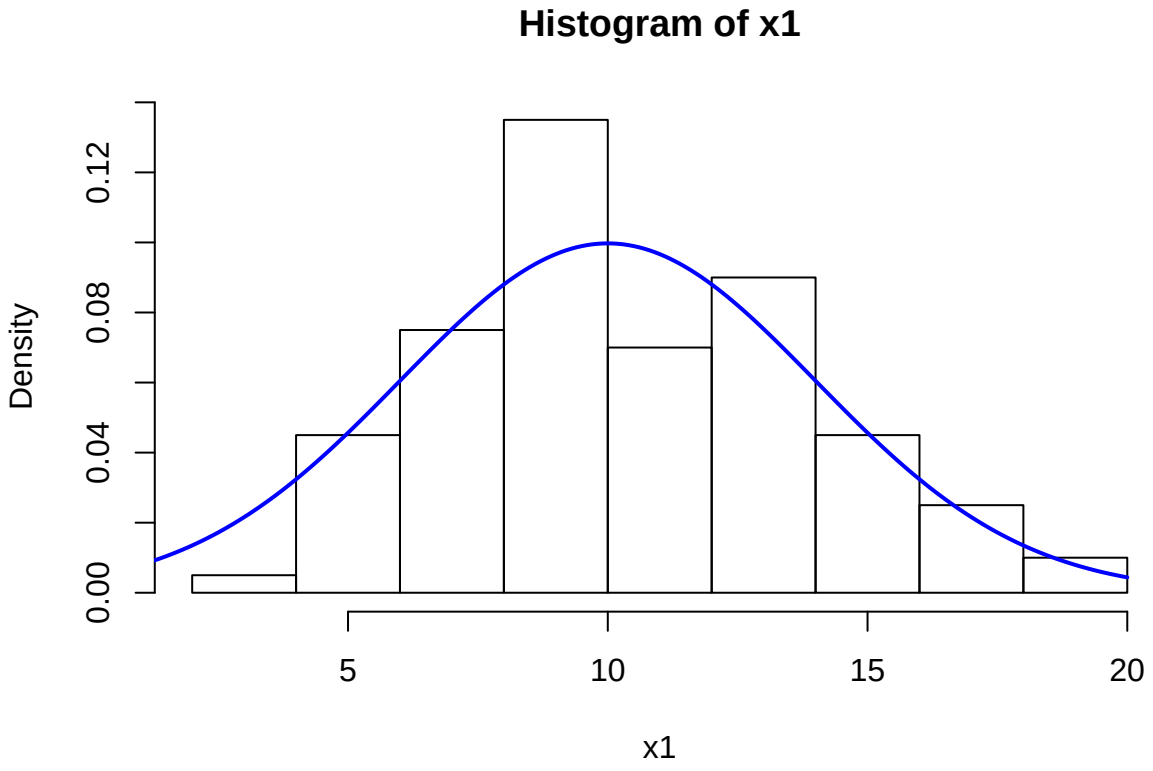
Uppgift 1 Simulering av normalfordelning

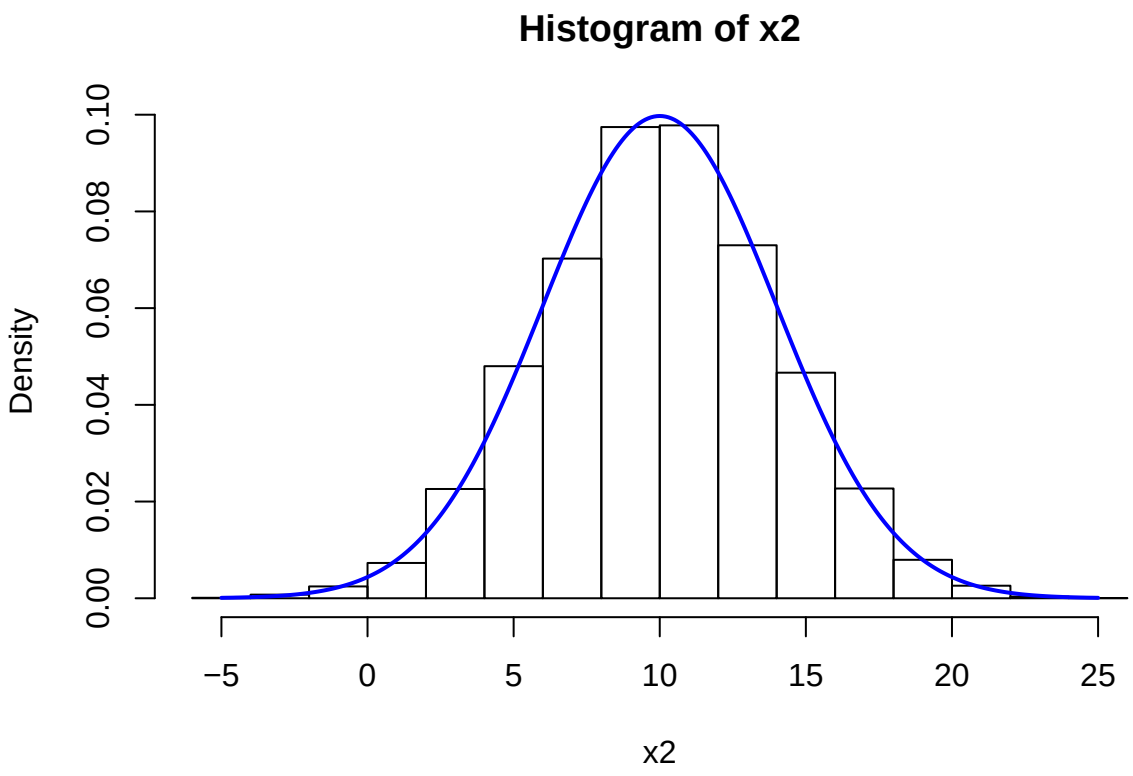
a) Visualisera fördelningarna i två histogram. Visualisera fordelningens pdf i samma graf.

Nedan simuleras normalfördelningen med olika antal dragningar.

```
x1 <- rnorm(100, mean = 10, sd = 4)
x2 <- rnorm(10000, mean = 10, sd = 4)
```

I figurerna nedan visas resultatet av dragningarna som ett histogram tillsammans med tathetsfunktionen.





b) Beskriv skillnaden mellan de olika graferna.

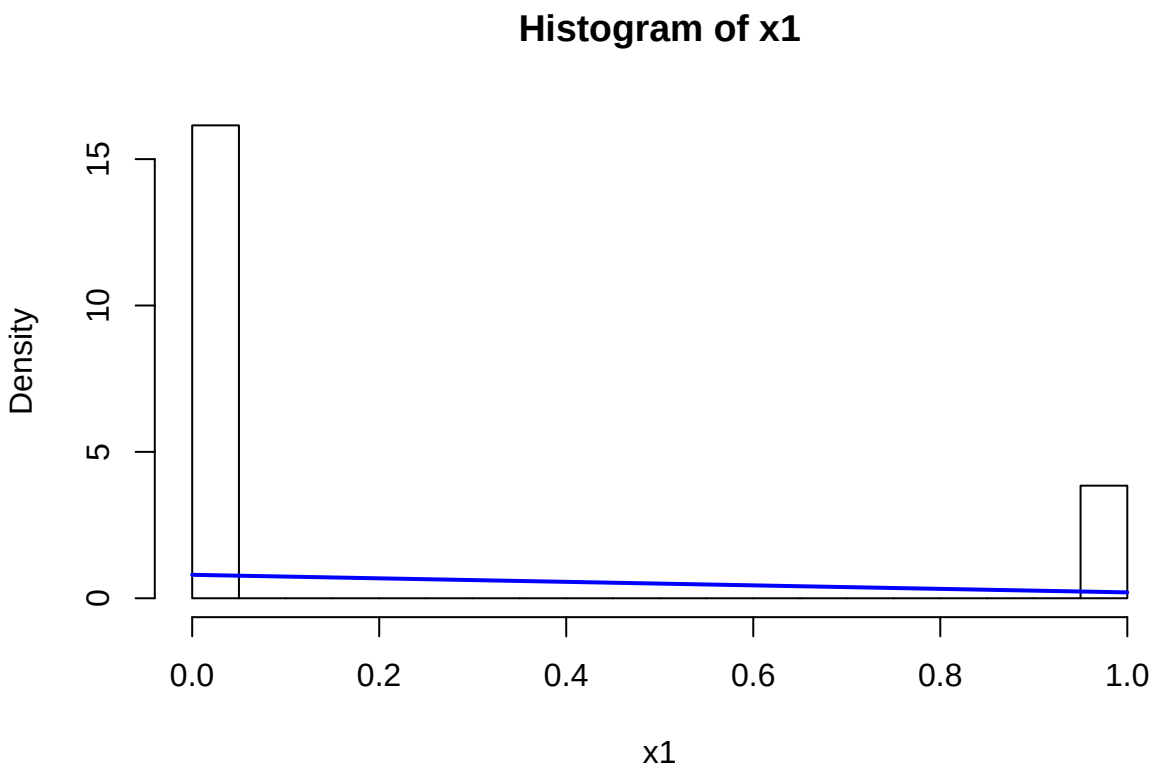
Vi kan tydligare se normalfordelningsformen om vi gor fler dragningar/simuleringar.

Uppgift 2 Simulera och visualisera andra fordelningar

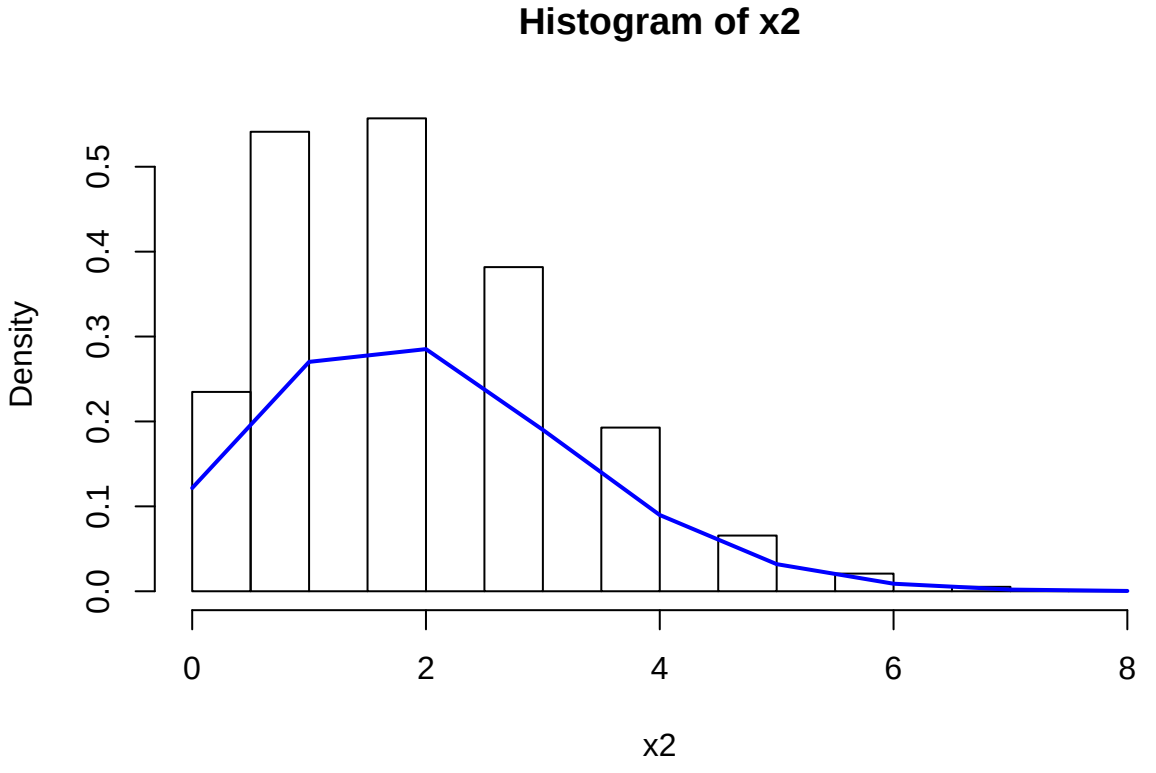
a) Simulera och visualisera foljande fordelningar med 10000 dragningar fran varje fordelning samt fordelningens tathetsfunktioner.

```
x1 <- rbern(10000, 0.2)
x2 <- rbinom(n = 10000, size = 20, prob = 0.1)
x3 <- rbinom(n = 10000, size = 20, prob = 0.5)
x4 <- rgeom(10000, 0.1)
x5 <- rpois(10000, 10)
```

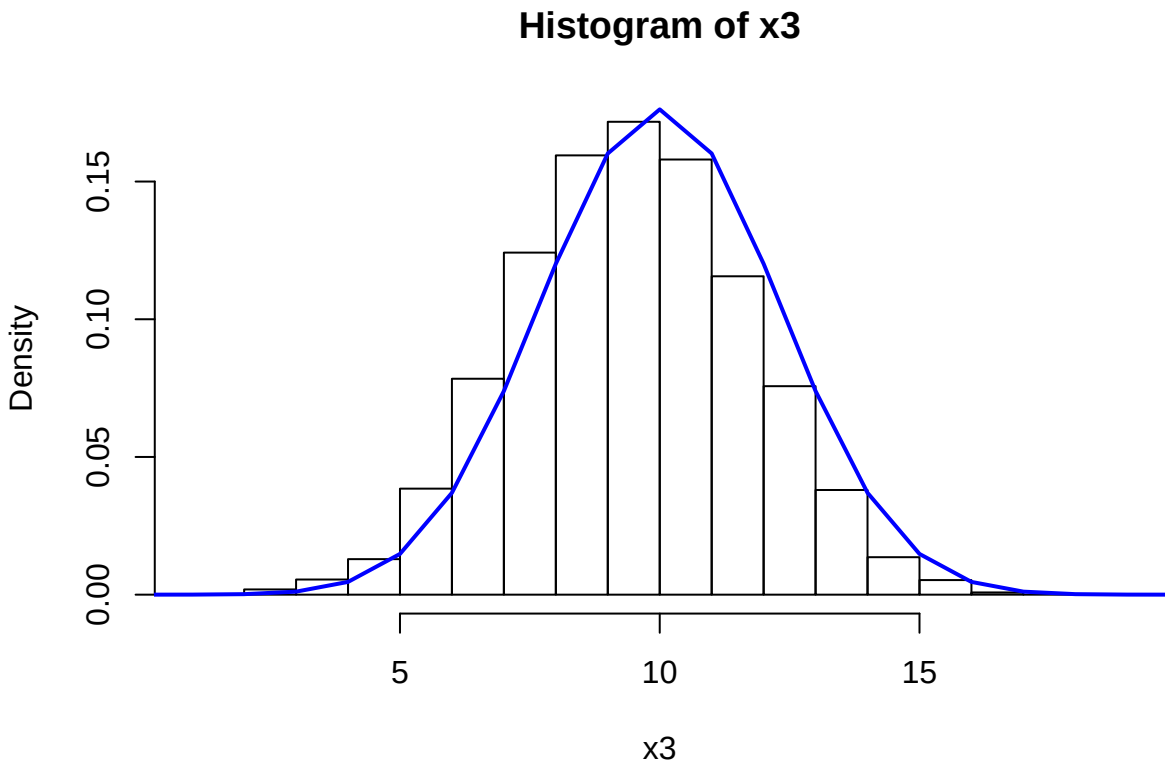
```
hist(x1, probability = TRUE)
xfit <- seq(0, 1, 1)
yfit <- dbern(xfit, 0.2, log = FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```



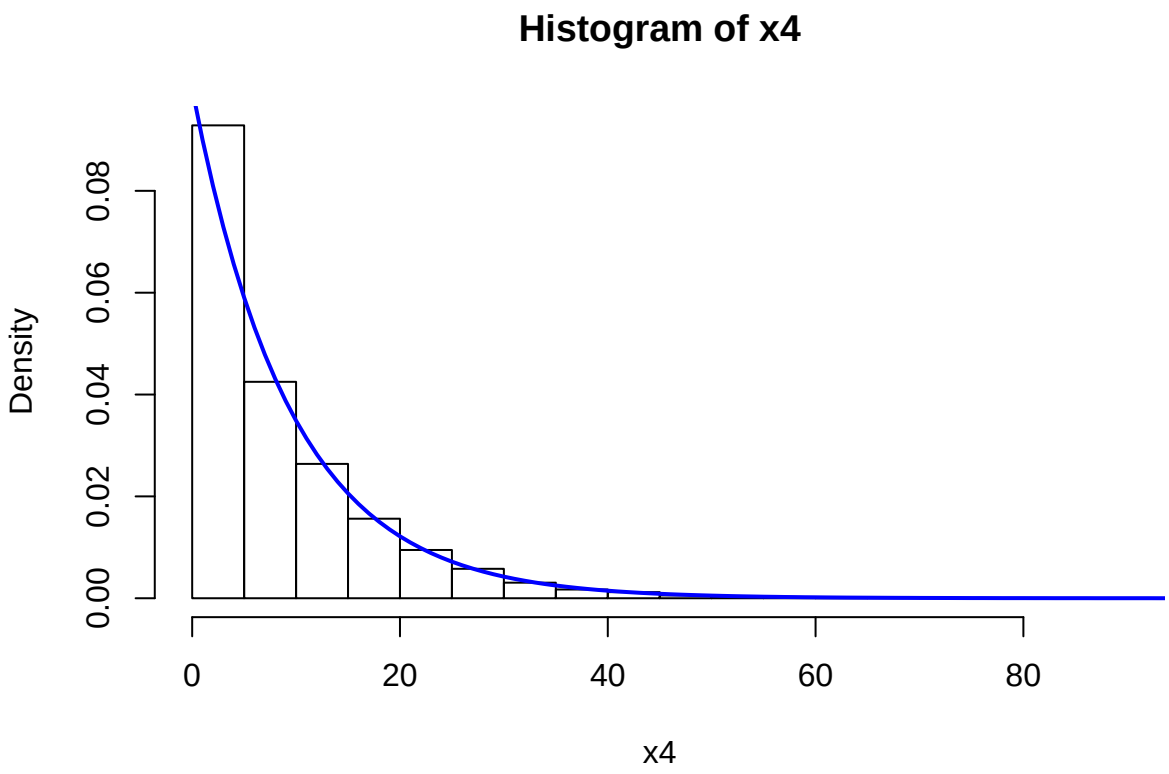

```
hist(x2, probability = TRUE)
xfit <- seq(0, 8, 1)
yfit <- dbinom(xfit, size = 20, prob = 0.1)
lines(xfit, yfit, col="blue", lwd=2)
```



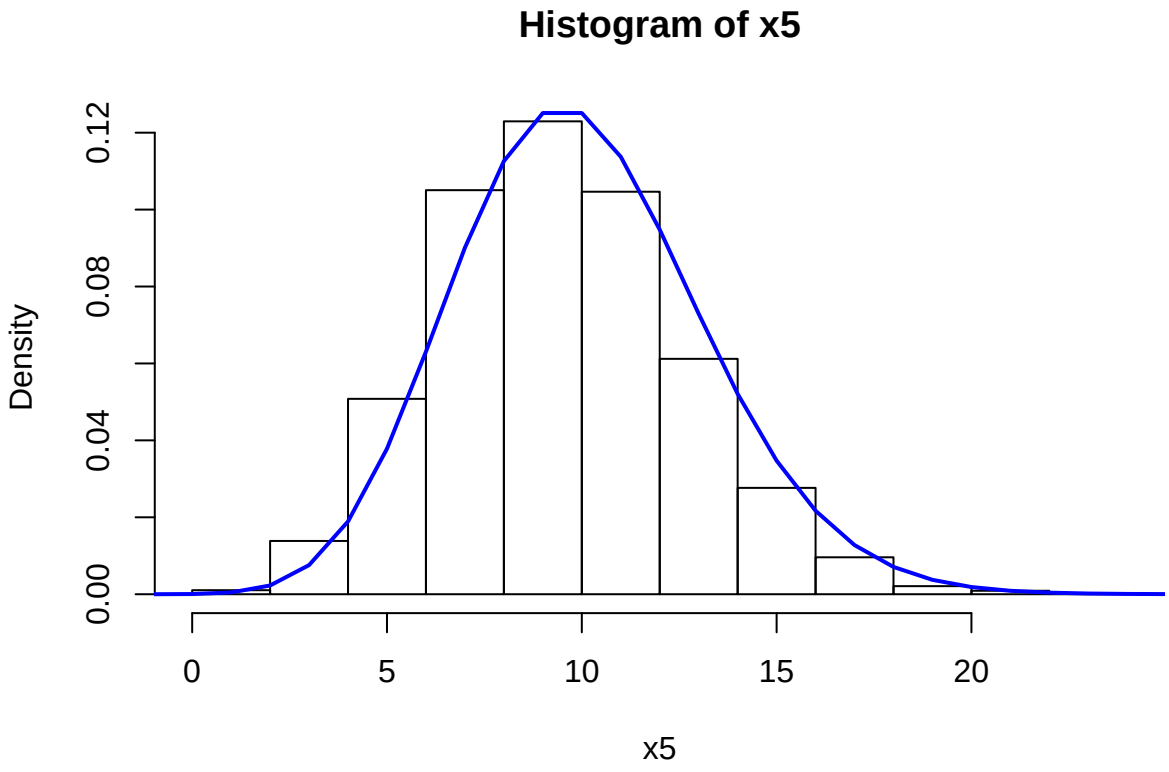
```r
hist(x3, probability = TRUE)
xfit <- seq(0, 20, 1)
yfit <- dbinom(xfit, size = 20, prob = 0.5)
lines(xfit, yfit, col="blue", lwd=2)
```




```r
hist(x4, probability = TRUE)
xfit <- seq(0, 100, 1)
yfit <- dgeom(xfit, prob = 0.1)
lines(xfit, yfit, col="blue", lwd=2)
```

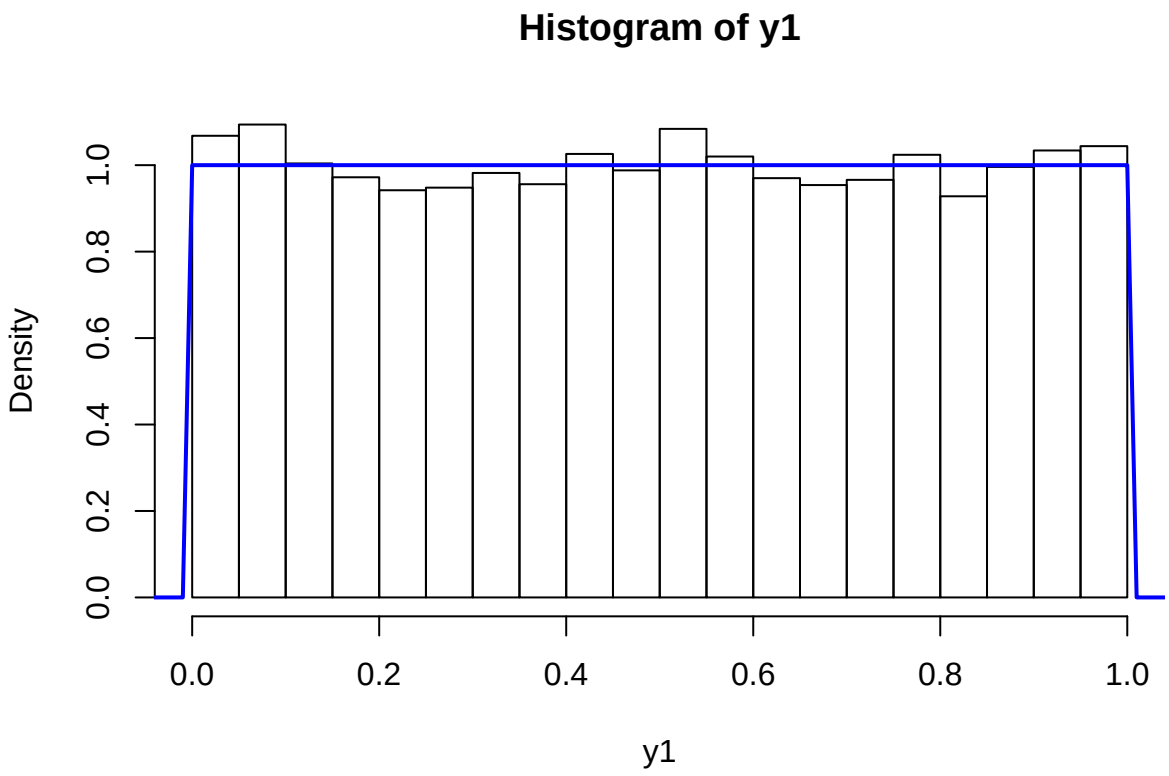



```
hist(x5, probability = TRUE)
xfit <- seq(-5, 25, 1)
yfit <- dpois(xfit, lambda = 10)
lines(xfit, yfit, col="blue", lwd=2)
```

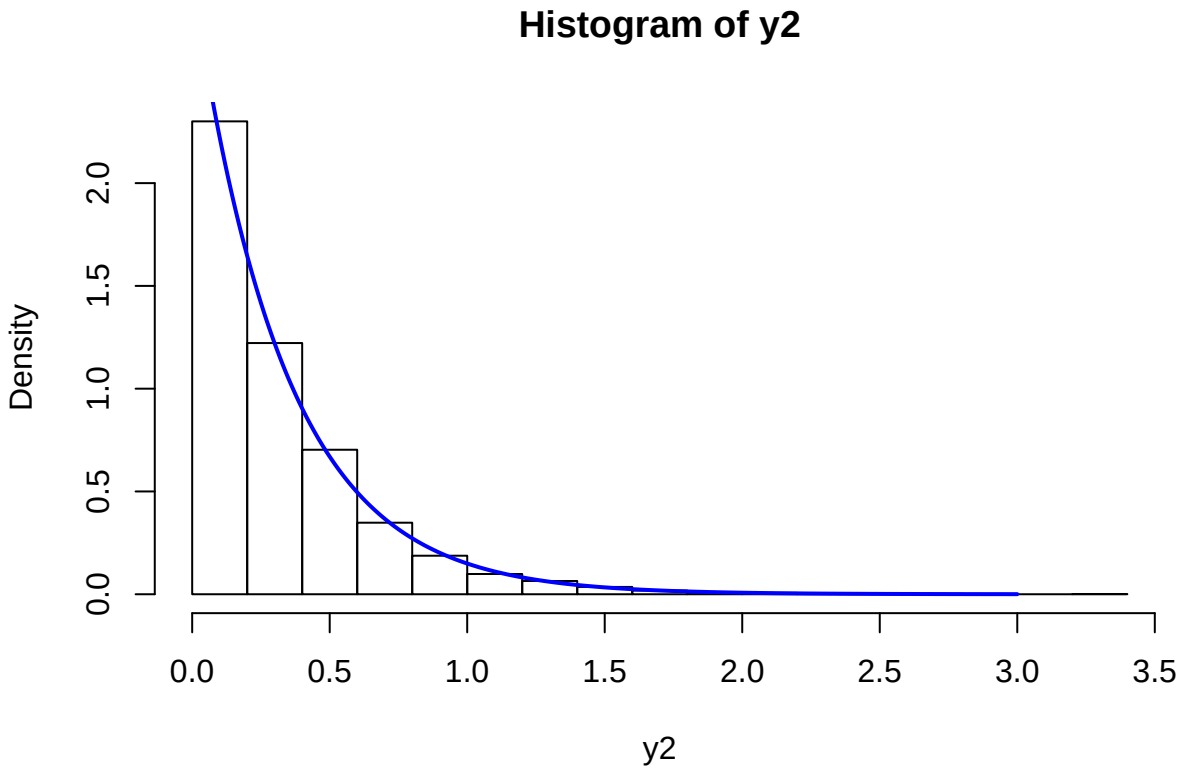


```r
y1 <- runif(10000, min = 0, max = 1)
y2 <- rexp(10000, rate = 3, beta = 1/3)
y3 <- rgamma(10000, shape=2, rate = 1, scale = 1, alpha = 2, beta = 1)
y4 <- rt(10000, 3)
y5 <- rbeta(10000, 0.1, 0.1, 0)
y6 <- rbeta(10000, 1, 1, 0)
y7 <- rbeta(10000, 10, 5, 0)
```

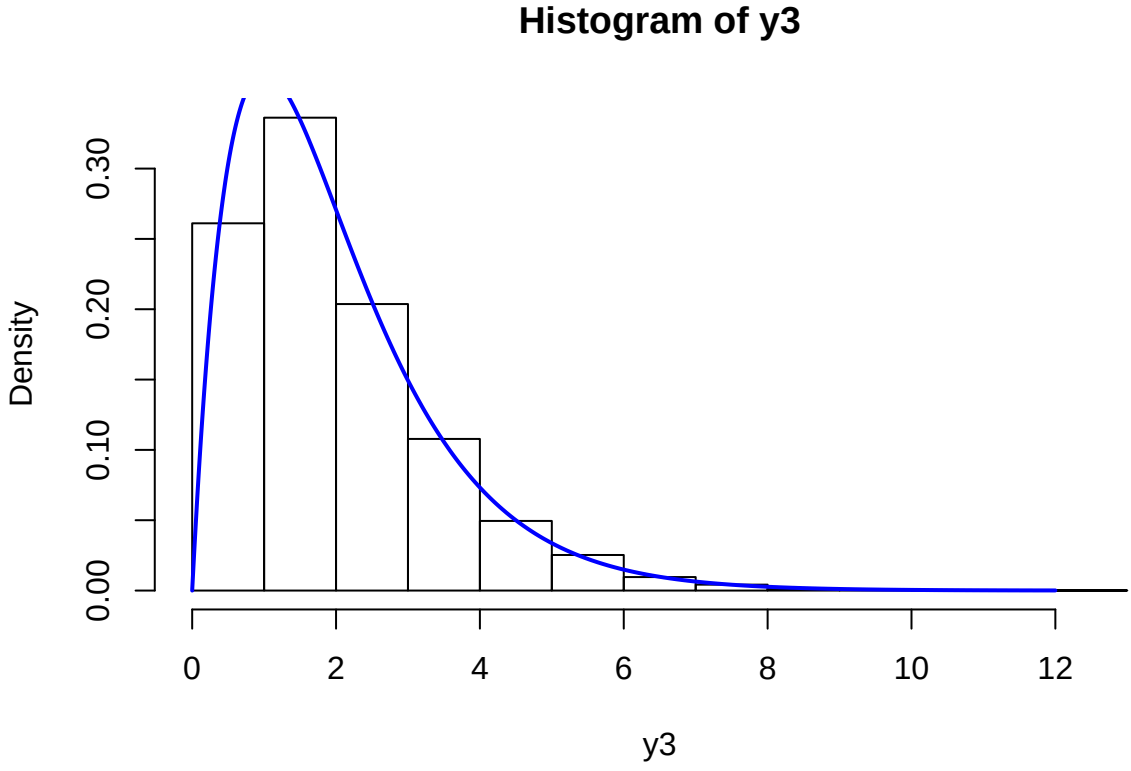
```r
hist(y1, probability = TRUE)
xfit <- seq(-5, 1.1, 0.01)
yfit <- dunif(xfit, min=0, max=1, log=FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```



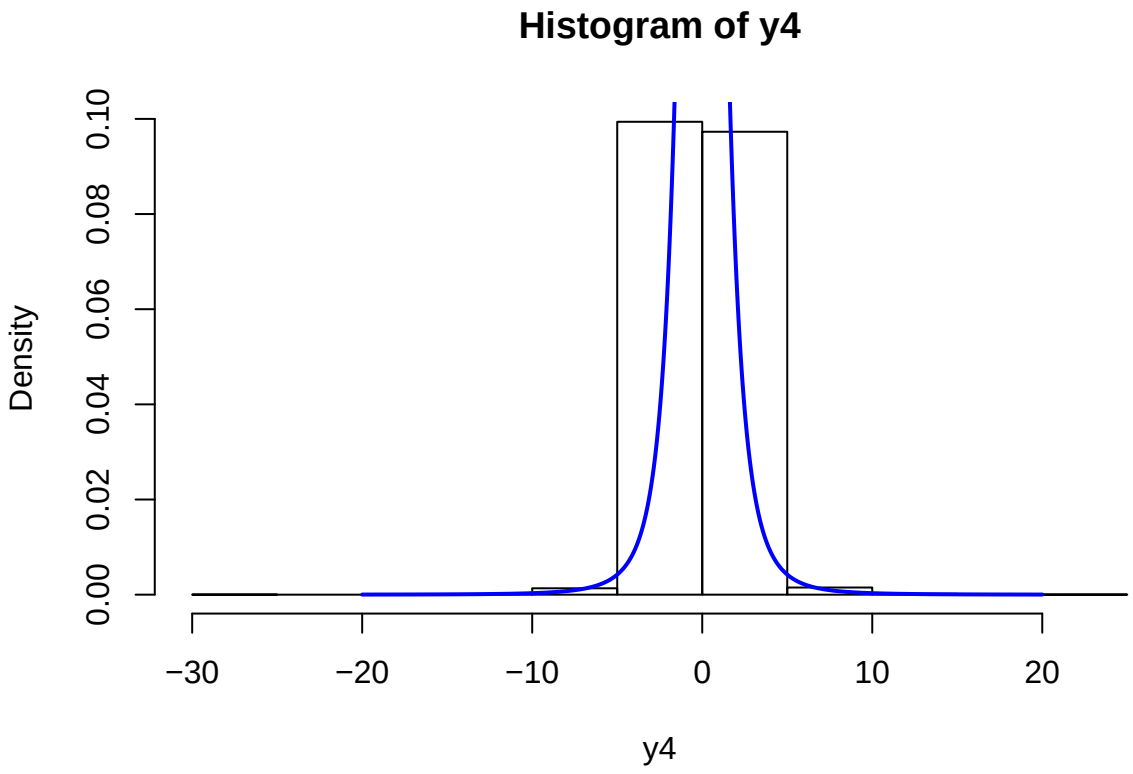

```r
hist(y2, probability = TRUE)
xfit <- seq(0, 3, 0.01)
yfit <- dexp(xfit, rate = 3, beta = 1/3, log = FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```



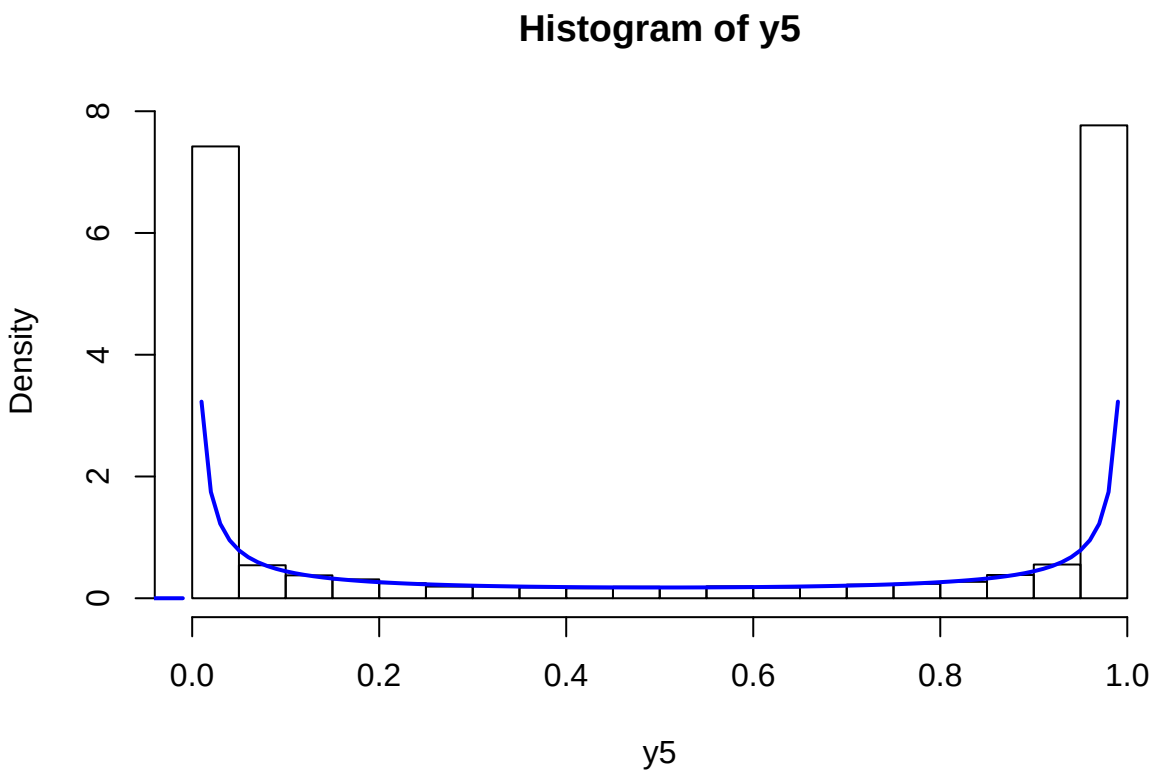

```
hist(y3, probability = TRUE)
xfit <- seq(0, 12, 0.01)
yfit <- dgamma(xfit, 2, 1, 1, 2, 1, log=FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```



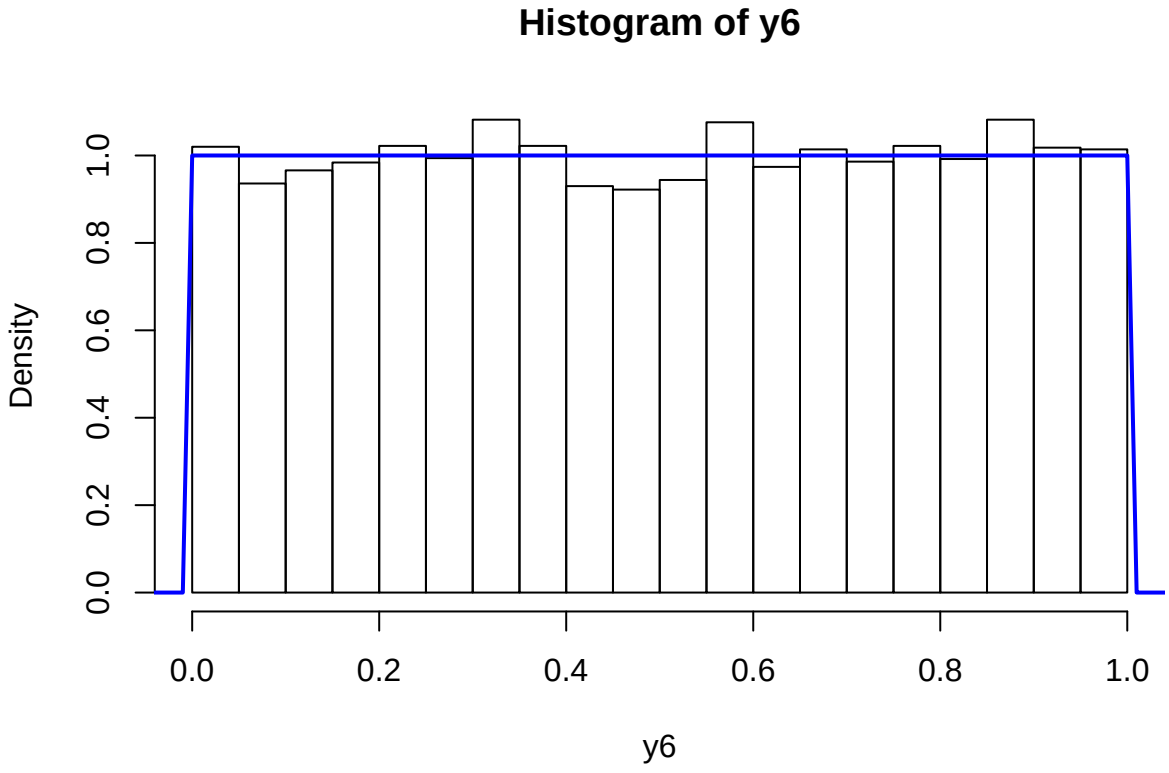
```r
hist(y4, probability = TRUE)
xfit <- seq(-20, 20, 0.01)
yfit <- dt(xfit, 3)
lines(xfit, yfit, col="blue", lwd=2)
```



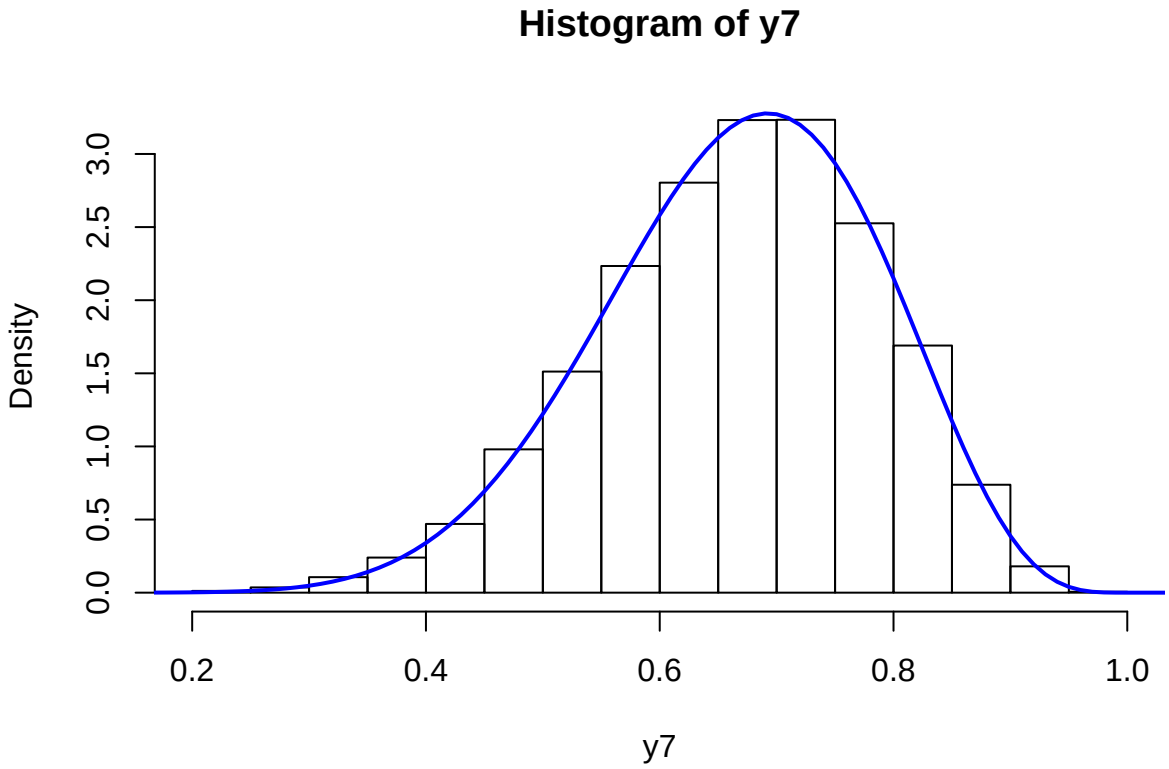

```r
hist(y5, probability = TRUE)
xfit <- seq(-1, 1, 0.01)
yfit <- dbeta(xfit, 0.1, 0.1, 0, log = FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```




```
hist(y6, probability = TRUE)
xfit <- seq(-0.1, 1.1, 0.01)
yfit <- dbeta(xfit, 1, 1, 0, log = FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```



```r
hist(y7, probability = TRUE)
xfit <- seq(-0.1, 1.1, 0.01)
yfit <- dbeta(xfit, 10, 5, 0, log = FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```

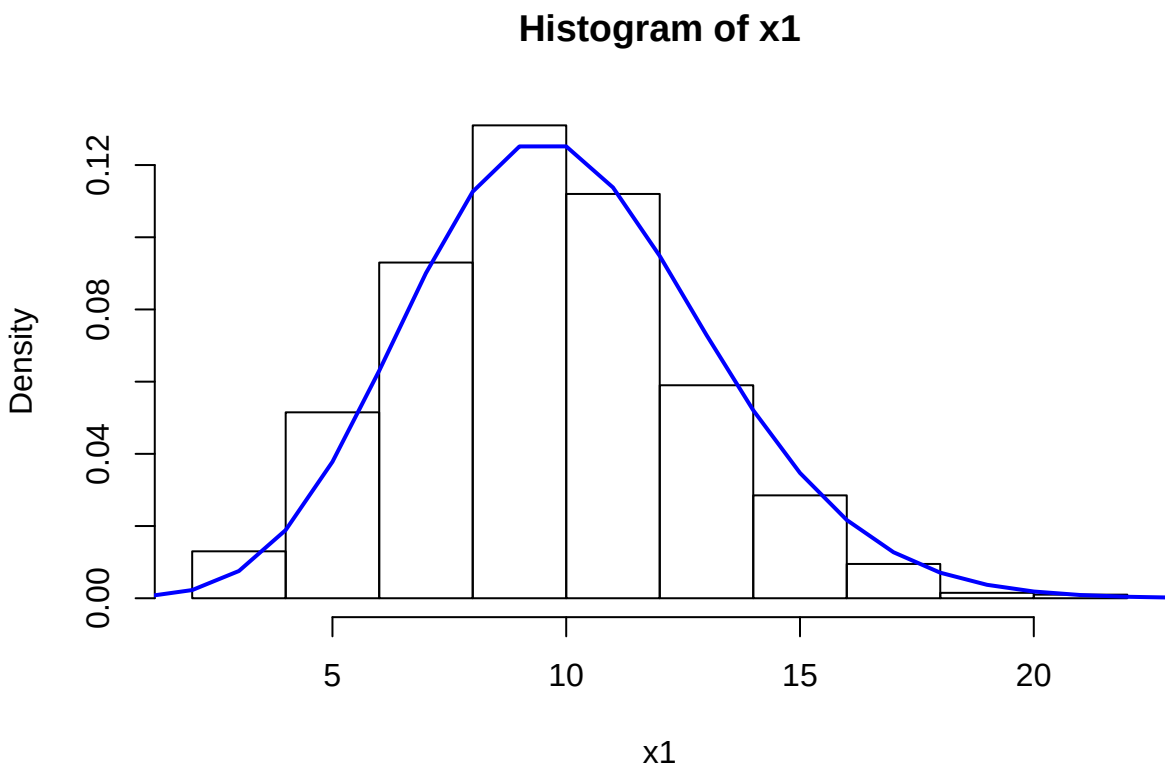


Uppgift 3 Relation mellan fordelningar

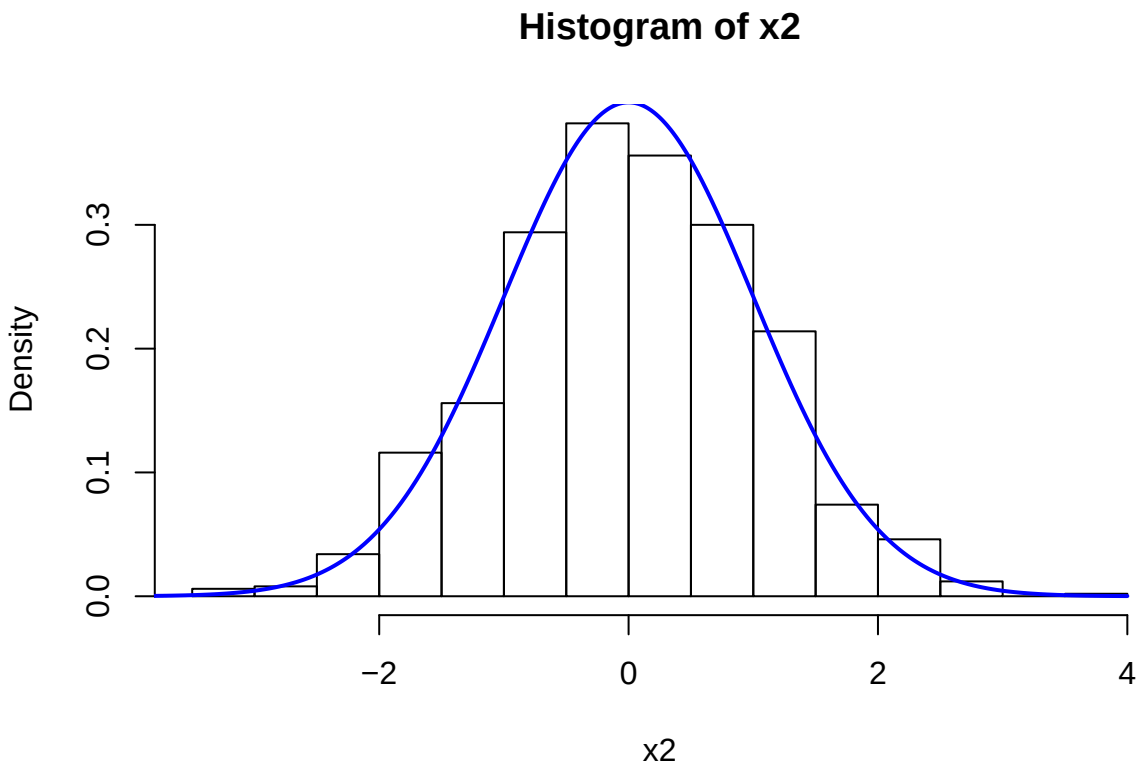
a) Simulera 1000 varden fran respektive fordelning och visualisera fordelningen i ett histogram tillsammans med fordelningens tathetsfunktionen.

```r
x1 <- rbinom(n = 1000, size = 10000, prob = 0.001)
x2 <- rt(1000, 10000)
```

```r
hist(x1, probability = TRUE)
xfit <- seq(0, 25, 1)
yfit <- dbinom(xfit, size = 10000, prob = 0.001)
lines(xfit, yfit, col="blue", lwd=2)
```




```
hist(x2, probability = TRUE)
xfit <- seq(-4, 4, 0.01)
yfit <- dt(xfit, 10000)
lines(xfit, yfit, col="blue", lwd=2)
```



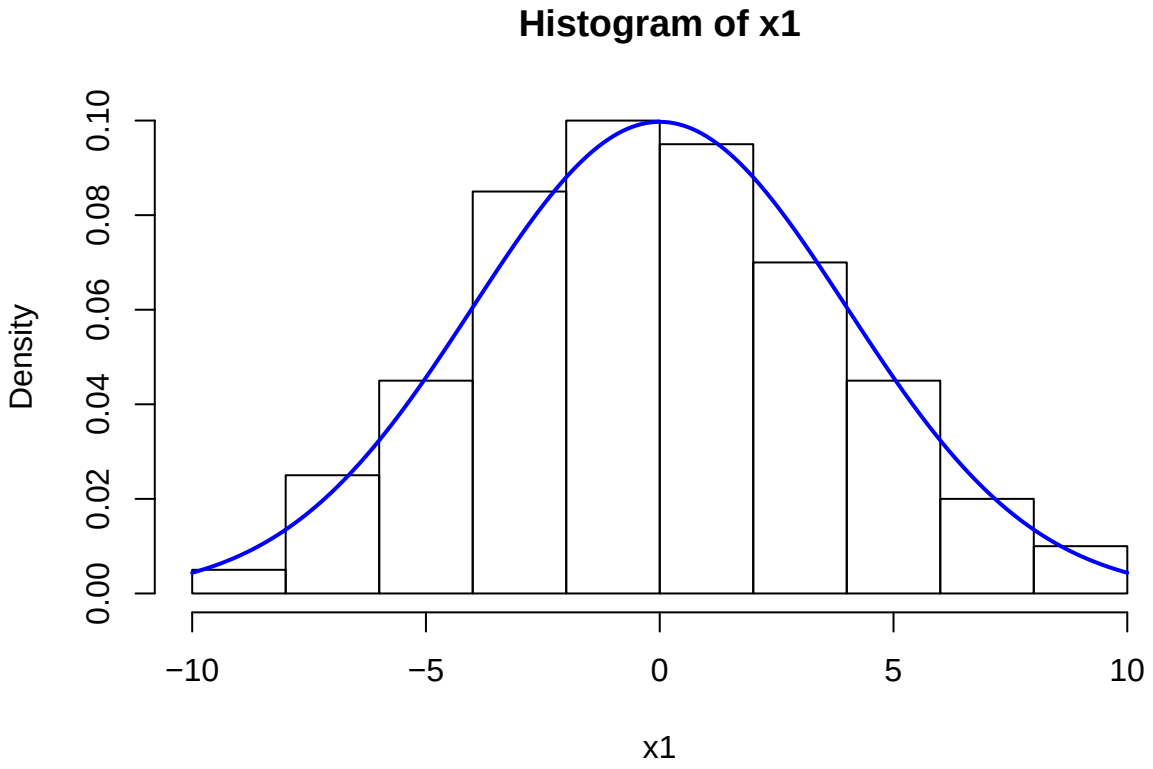
b) Ta reda pa (ex. via Wikipedia) vilken annan fordelning som respektive fordelning borjar konvergera mot.

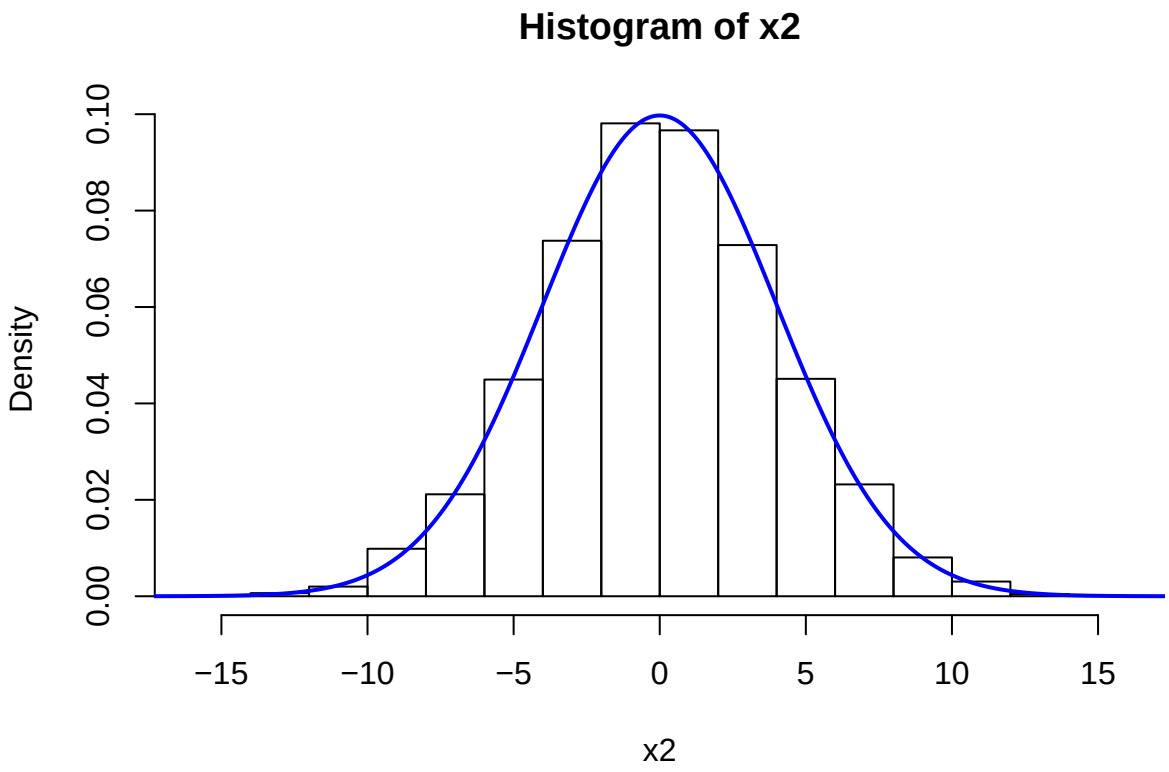
Binomialfördelningen börjar konvergera mot Poissonfördelning och Student-t-fördelningen börjar konvergera mot en normalfördelning.

c) Simulera dragningar fran dessa fordelningar och jamfor resultatet med de resultat du fick i a).

```
x1 <- rnorm(100, mean = 0, sd = 4)
x2 <- rnorm(10000, mean = 0, sd = 4)
```

I figurerna nedan visas resultatet av dragningarna som ett histogram tillsammans med täthetsfunktionen.



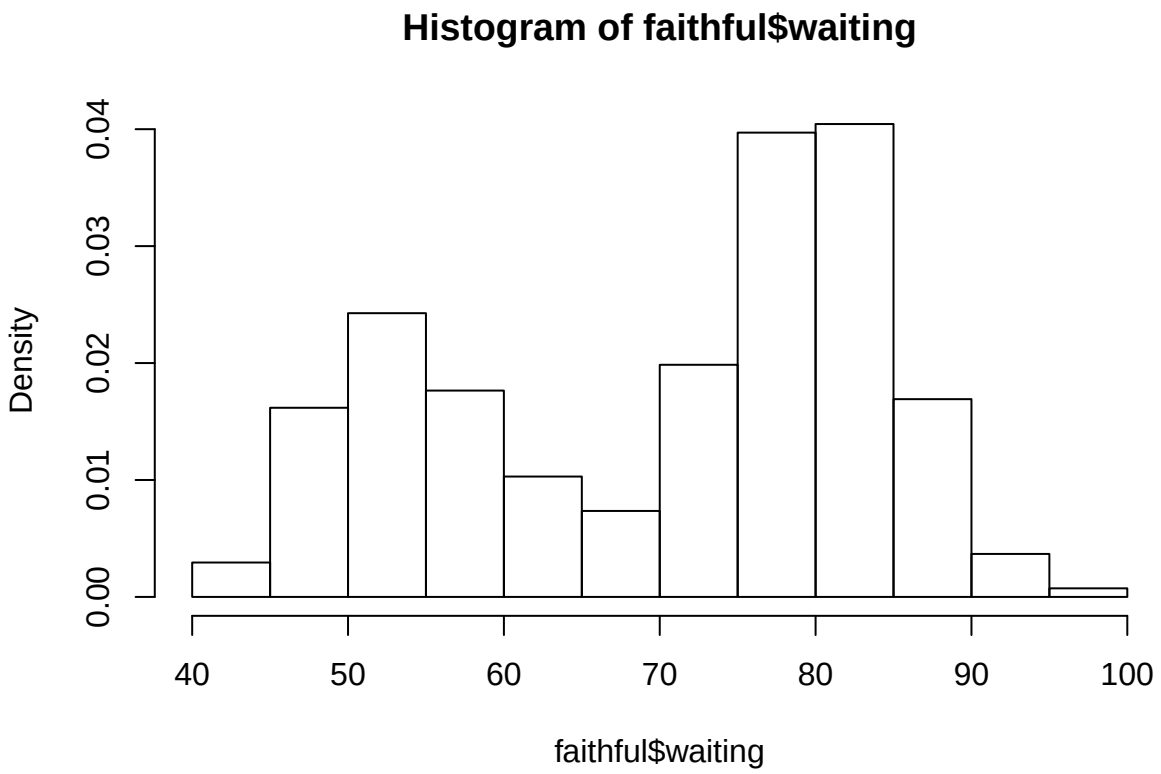


Uppgift 4 Hiearkiska sannolikhetsfördelningar

Vi ska nu simulera en blandad sannolikhetsfördelning (mixture distribution). Börja att läsa in data-materialet faithful i R med data(faithful).

a) Visualisera variabeln waiting i ett histogram.

```
data(faithful)
hist(faithful$waiting, probability = TRUE)
```

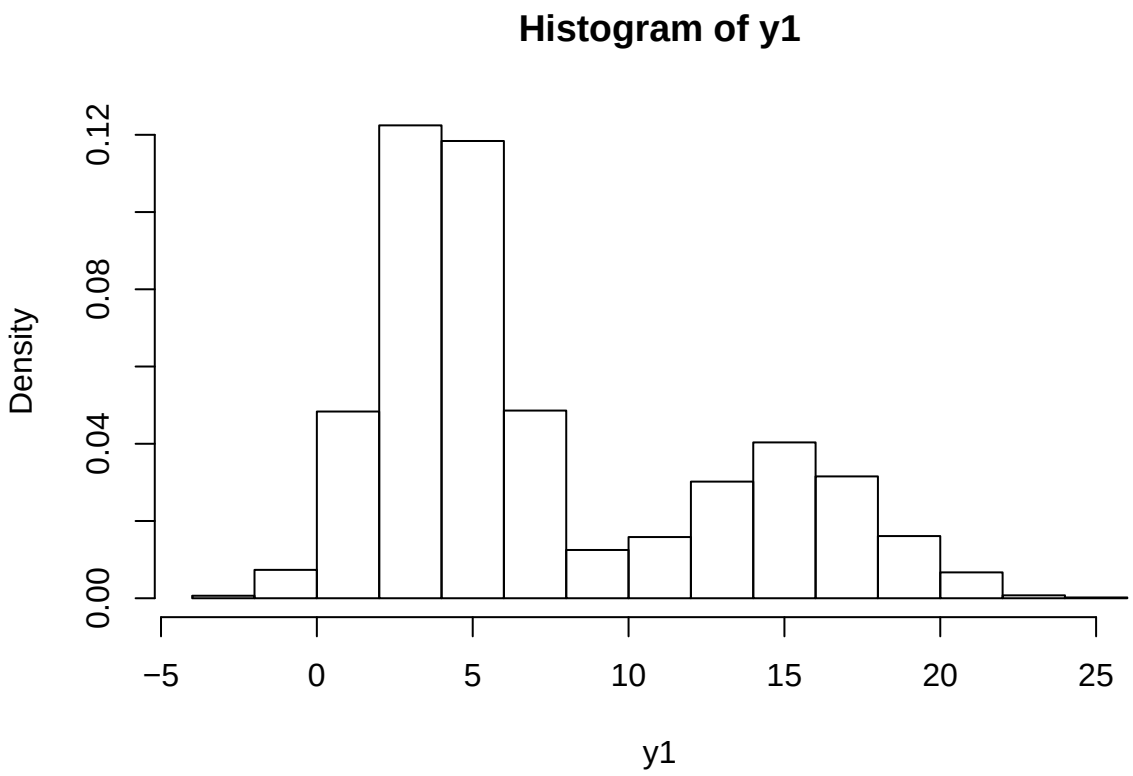


b) Simulera 10000 från följande modell och visualisera fördelningen i ett histogram. Simulera först från X och sedan från Y . Sätt p = 0.3, 1 = 15, 1 = 3, 2 = 4 och 2 = 2.

Xi <- Bernoulli(p)

```
x1 <- rbern(10000, 0.3)
y1 <- numeric(10000)
for(i in 1:10000) {
  y1[i] <- x1[i] * rnorm(n = 1, mean = 15, sd = 3) + (1-x1[i]) * rnorm(n = 1, mean = 4, sd = 2)
}
```

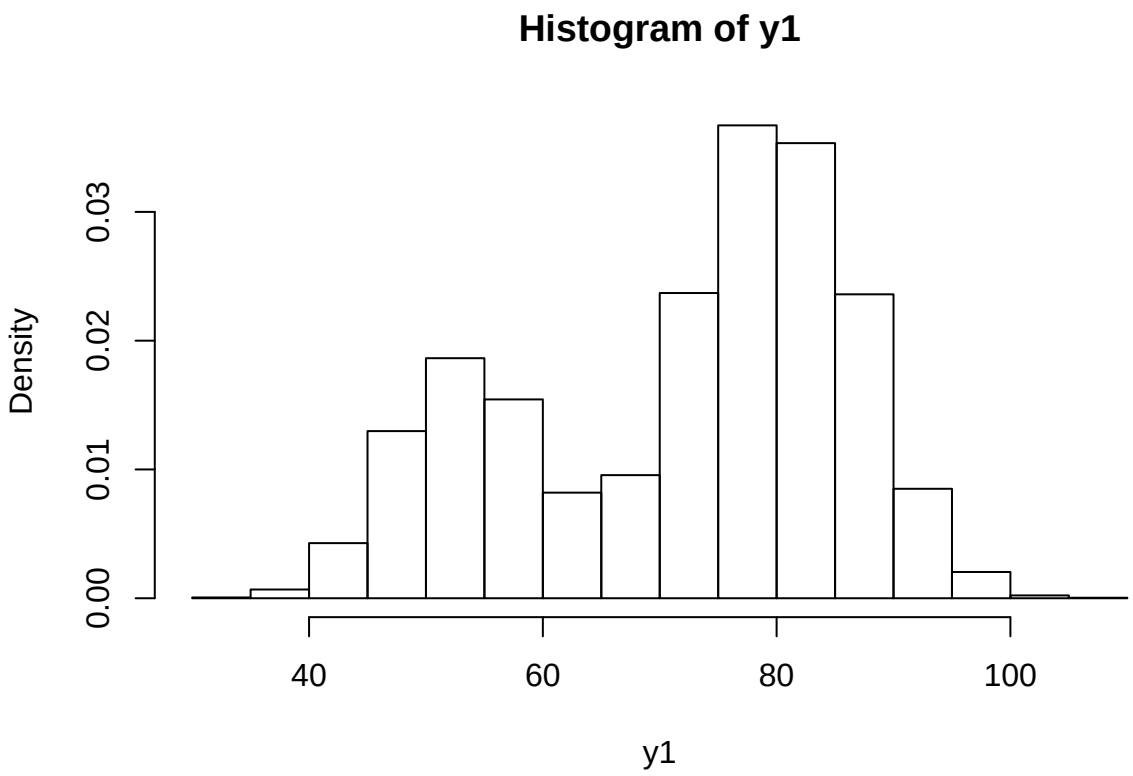
```
hist(y1, probability = TRUE)
xfit <- seq(40, 100, 1)
yfit <- dbern(xfit, 0.3, log = FALSE)
lines(xfit, yfit, col="blue", lwd=2)
```



c)

```
x1 <- rbern(10000, 0.3)
y1 <- numeric(10000)
for(i in 1:10000) {
  y1[i] = x1[i] * rnorm(n = 1, mean = 53, sd = 6) + (1-x1[i]) * rnorm(n = 1, mean = 80, sd = 7)
}
```

```
hist(y1, probability = TRUE)
```



Uppgift 5 Analytisk sannolikhet och approximation med ”Monte Carlo” - metoder

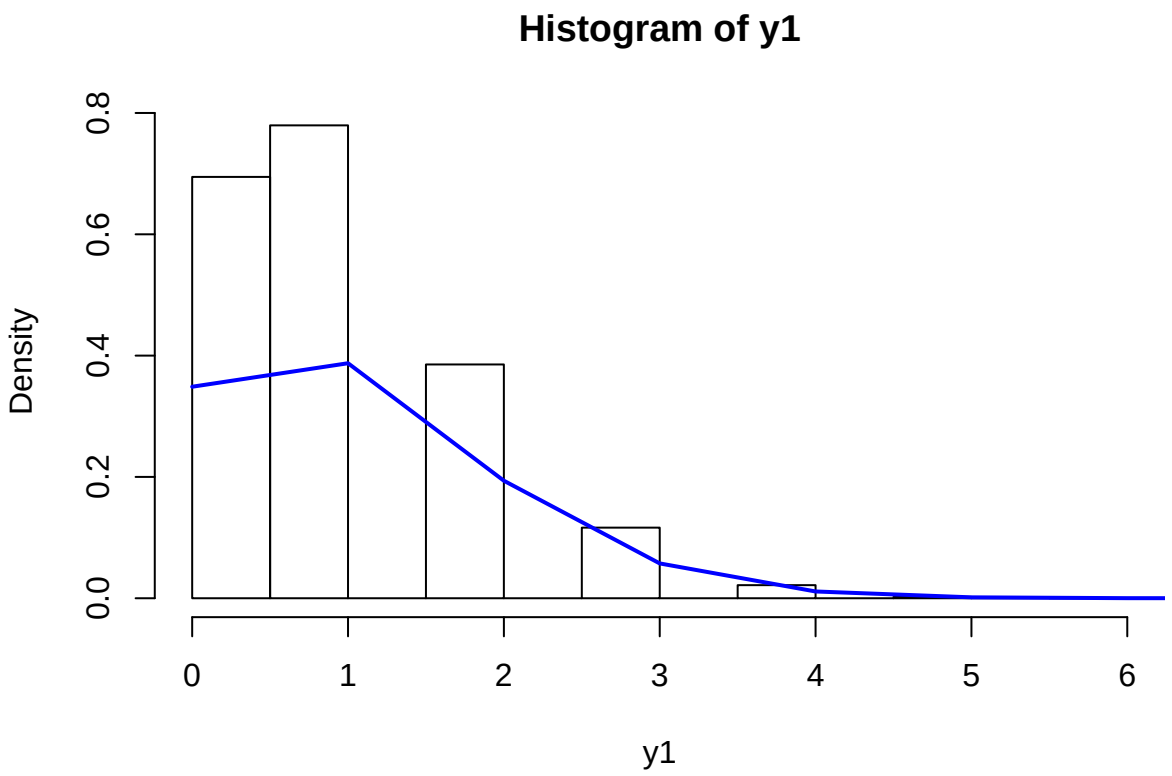
```
x1 <- rnorm(10000, 0, 1)
y1 <- rbinom(10000, 10, 0.1)
```

a)

```
#P(Y=0 enligt täthetsfunktionen)
dbinom(0, size = 10, prob = 0.1)
```

```
## [1] 0.3486784
```

```
#Simulera 10000 dragningar från Y och beräkna andelen dragningar där y = 0
hist(y1, probability = TRUE)
xfit <- seq(0, 10, 1)
yfit <- dbinom(xfit, 10, 0.1)
lines(xfit, yfit, col="blue", lwd=2)
```




```
length(y1[y1 == 0])/10000
```

```
## [1] 0.3473
```

b)

1 P(X < 0)

```
pnorm(0, mean = 0, sd = 1)
```

```
## [1] 0.5
```

2 P(-1 < X < 1)

```
pnorm(1, mean = 0, sd = 1) - pnorm(-1, mean = 0, sd = 1)
```

```
## [1] 0.6826895
```

3 P(1.96 < X)

```
1 - pnorm(1.96, mean = 0, sd = 1)
```

```
## [1] 0.0249979
```

4 P(0 < Y < 10) <=> P(1 <= Y <= 9) (p g a diskret fördelning)

```
pbinom(9, size = 10, prob = 0.1)  - pbinom(1, size = 10, prob = 0.1)
```

```
## [1] 0.2639011
```

5 P(Y = 0)

```
pbinom(0.0, size = 10, prob = 0.1)
```

```
## [1] 0.3486784
```

6 P (0 <= Y <= 10)

```
pbinom(10.0, size = 10, prob = 0.1) - pbinom(0.0, size = 10, prob = 0.1)
```

```
## [1] 0.6513216
```

c)

1 P(X < 0)

```
x1 <- rnorm(10000, 0, 1)
p = ecdf(x1)
p(0.0)
```

```
## [1] 0.5018
```

2 P(1 < X < 1)

```
x2 <- rnorm(10000, 0, 1)
p = ecdf(x2)
p(1.0) - p(-1.0)
```

```
## [1] 0.6793
```

3 P(1.96 < X)

```
x3 <- rnorm(10000, 0, 1)
p = ecdf(x3)
1 - p(1.96)
```

```
## [1] 0.0269
```

4 P(0 < Y < 10)

```
y1 <- rbinom(10000, 10, 0.1)
p = ecdf(y1)
p(10.0) - p(0.0)
```

```
## [1] 0.6573
```

5 P(Y = 0)

```
y2 <- rbinom(10000, 10, 0.1)
p = ecdf(y2)
p(0.0)
```

```
## [1] 0.346
```

6 P (0 <= Y <= 10)

```
y3 <- rbinom(10000, 10, 0.1)
p = ecdf(y3)
p(10.0) - p(0.0)
```

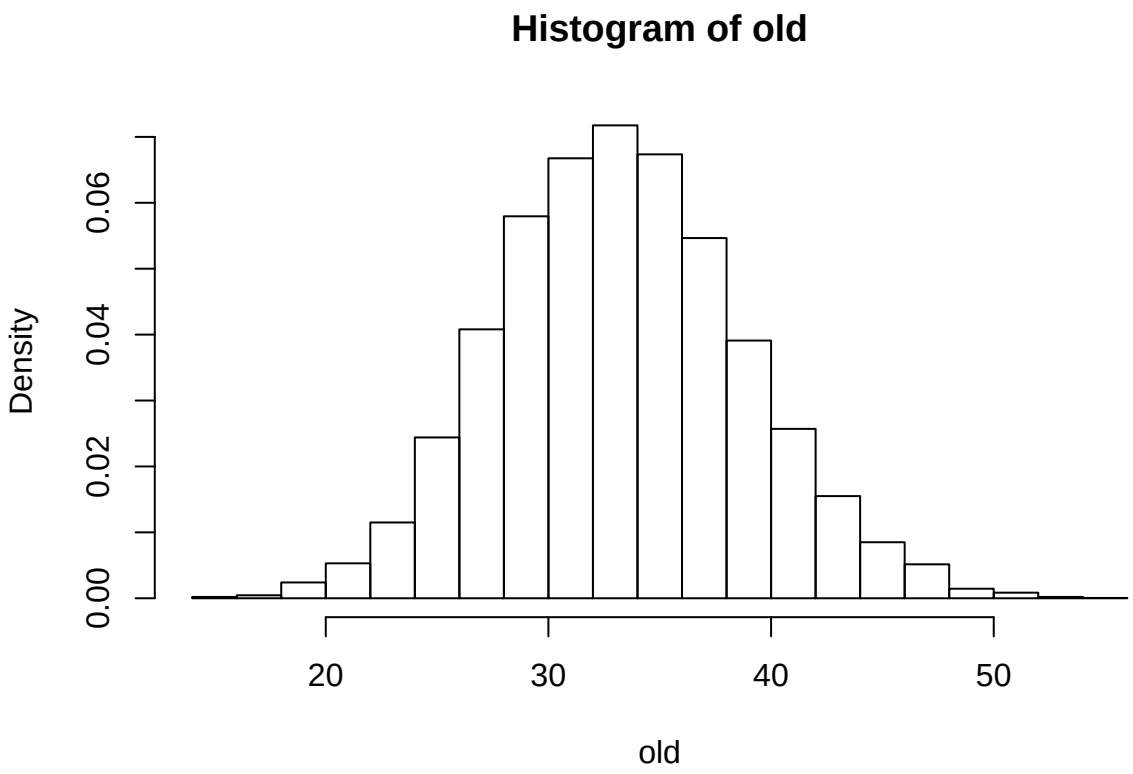
```
## [1] 0.6451
```

Uppgift 6 Beräkna (icke-triviala) sannolikheter

a)

Förväntade antalet fel: Gamla systemet

```
old <- rbinom(10000, size = 337, prob = 0.1)
hist(old, probability = TRUE)
```

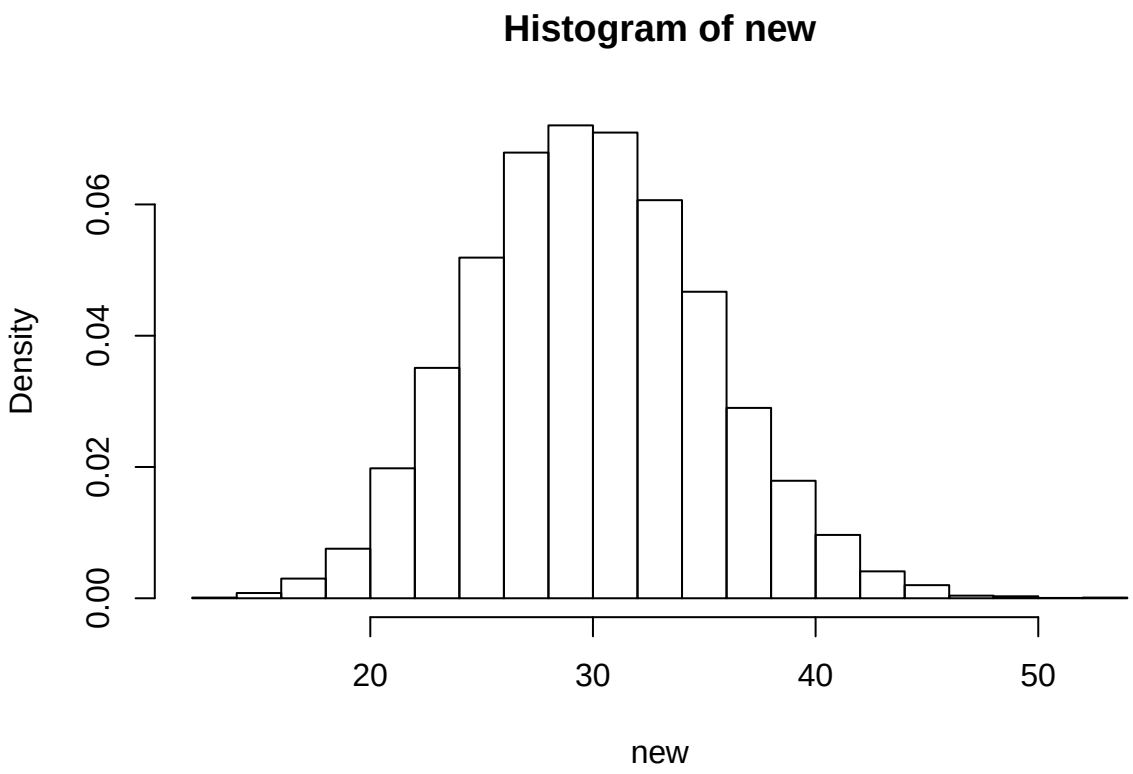



```
evold <- mean(old)
print(evold)
```

```
## [1] 33.7818
```

Förväntade antalet fel: Nya systemet

```
newprob <- runif(10000, min = 0.02, max = 0.16)
evnewprob <- mean(newprob)
new <- rbinom(10000, size = 337, prob = evnewprob)
hist(new, probability = TRUE)
```




```
evnew <- mean(new)
print(evnew)
```

```
## [1] 30.4229
```

b)

P(färre fel i gamla systemet än nya) = intervallet [0.1, 0.16] / intervallet [0.02, 0.16] = 0.06/0.14 = 0.429

c)

sannolikheten att du kommer få fler än 50 fel i gamla systemet

```
p <- ecdf(old)
print(1-p(50))
```

```
## [1] 0.0022
```

sannolikheten att du kommer få fler än 50 fel i nya systemet

```
p <- ecdf(new)
print(1-p(50))
```

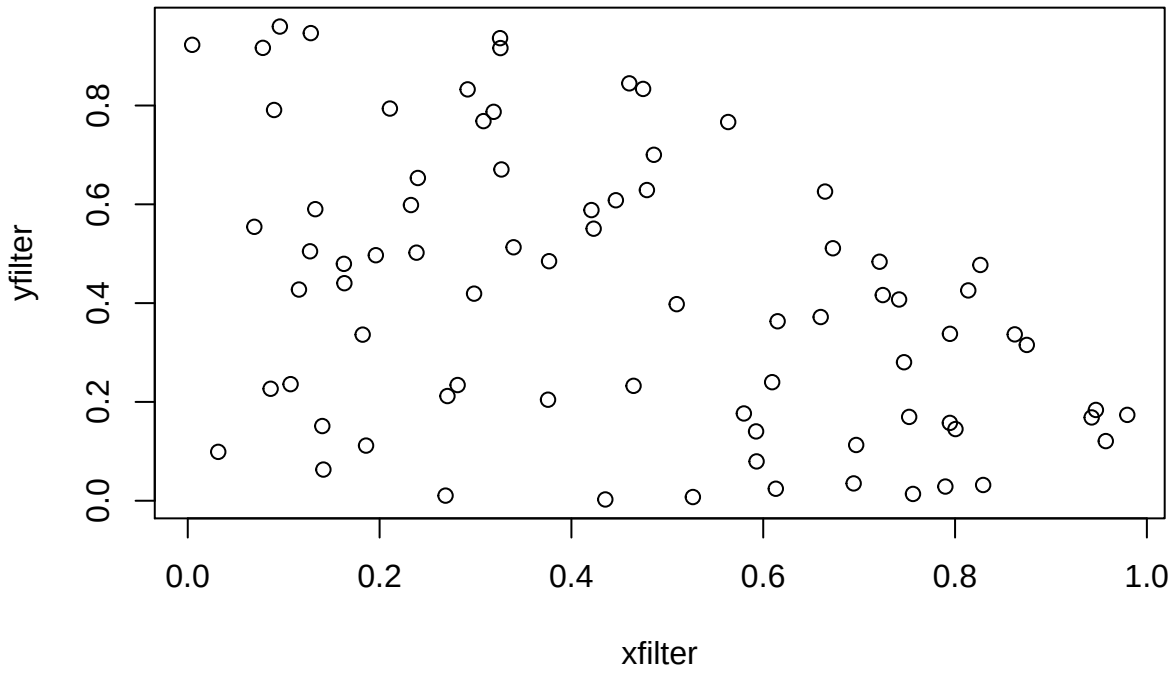
```
## [1] 3e-04
```

Uppgift 7 Monte Carlo metoder för integrering

a)

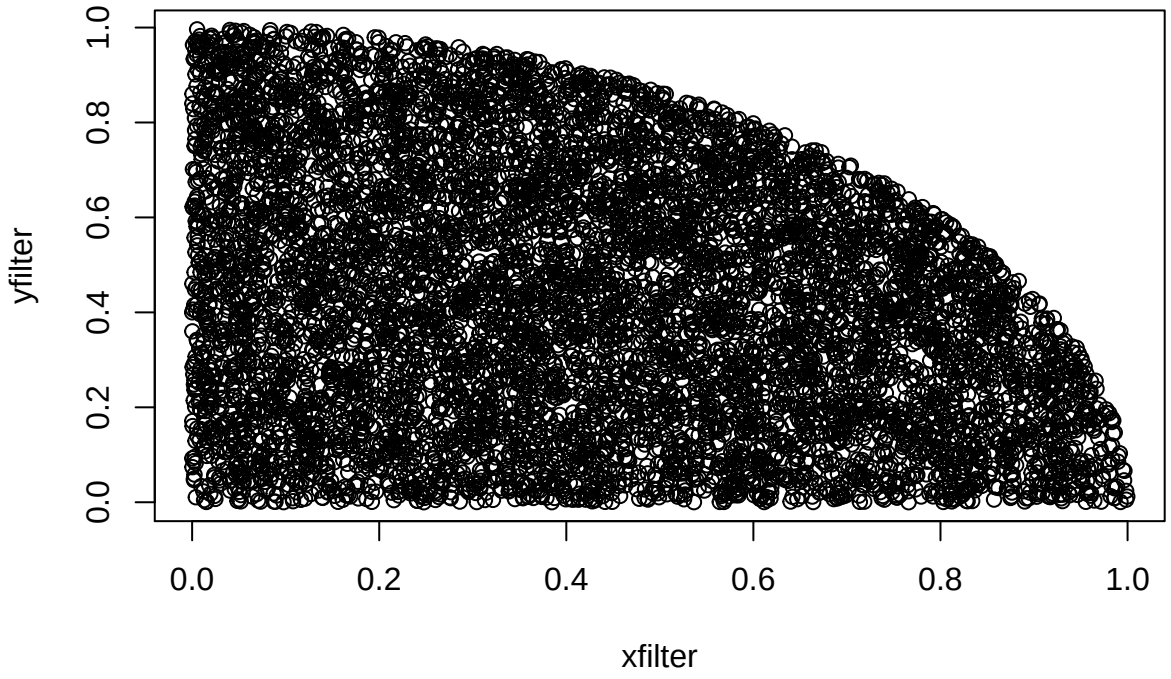
```
montecarlocalcef <- function(n) {
  x <- runif(n, min = 0.0, max = 1.0)
  y <- runif(n, min = 0.0, max = 1.0)

  xfilter <- x[((x**2 + y**2)**(1/2))<=1]
  yfilter <- y[((x**2 + y**2)**(1/2))<=1]

  plot(xfilter, yfilter)

  z <- length(xfilter)

  const <- 4 * z/n
  return(const)

}
```

```
montecarlocalcef(100)
```



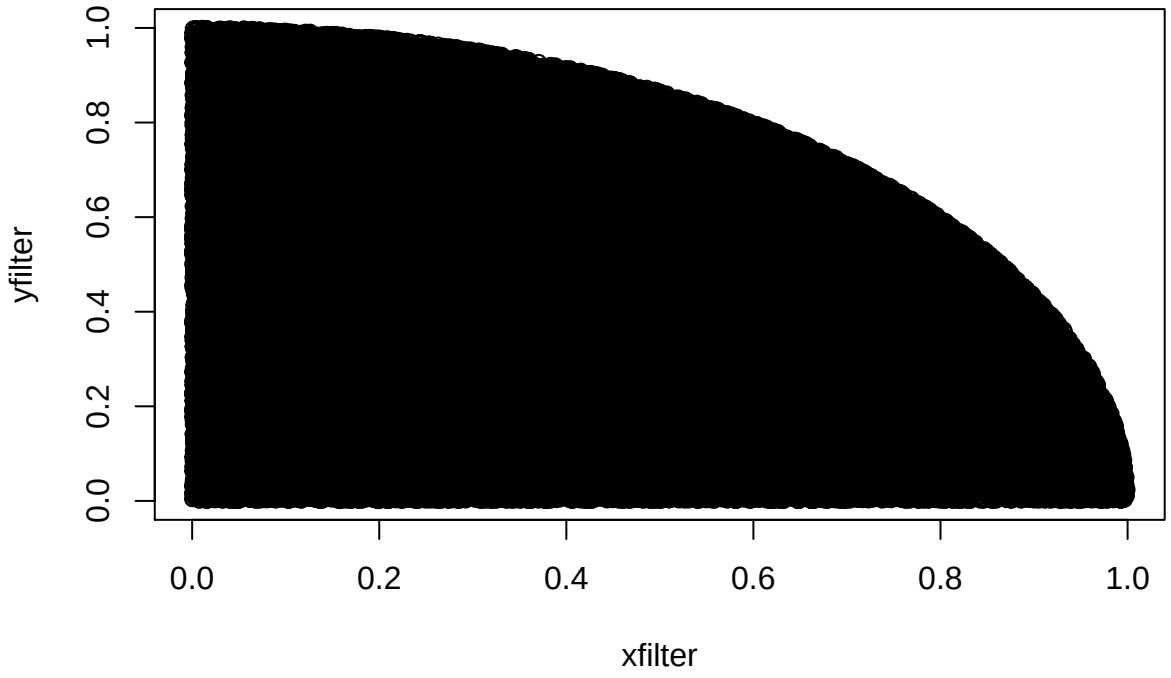

```
## [1] 3.12
```

```
montecarlocalcef(10000)
```




```
## [1] 3.1456
```

```
montecarlocalcef(100000)
```




```
## [1] 3.14156
```

```
montecarlocalcef(1000000)
```


```
## [1] 3.142204
```

b)

Vi försöker approximera pi, vilket verkar gå ganska väl både sett till talet man får fram samt den grafiska kvartscirkeln.

c)

Approximationen blir exaktare, men beräkningen tar längre tid.

d)

```
x2 <- function(n) {
  return(n*n)
}
integrate(x2, lower = 0, upper = 2)
```

```
## 2.666667 with absolute error < 3e-14
```

e)

```
montecarlocalc <- function(n) {

x <- runif(n, min = 0.0, max = 2.0)
```

```
y <- runif(n, min = 0.0, max = 4.0)


return(mean(y < x**2)*2*4)
}

montecarlocalc(100000)
```

```
## [1] 2.65248
```

Uppgift 8 Stora talens lag

```
binomfunc <- function(n) {
  x <- rbinom(n, size = 10, prob = 0.2)
  return(x)
}

gammafunc <- function(n) {
  y <- rgamma(n, shape = 2,  scale = 2)
  return(y)
}
```

a)

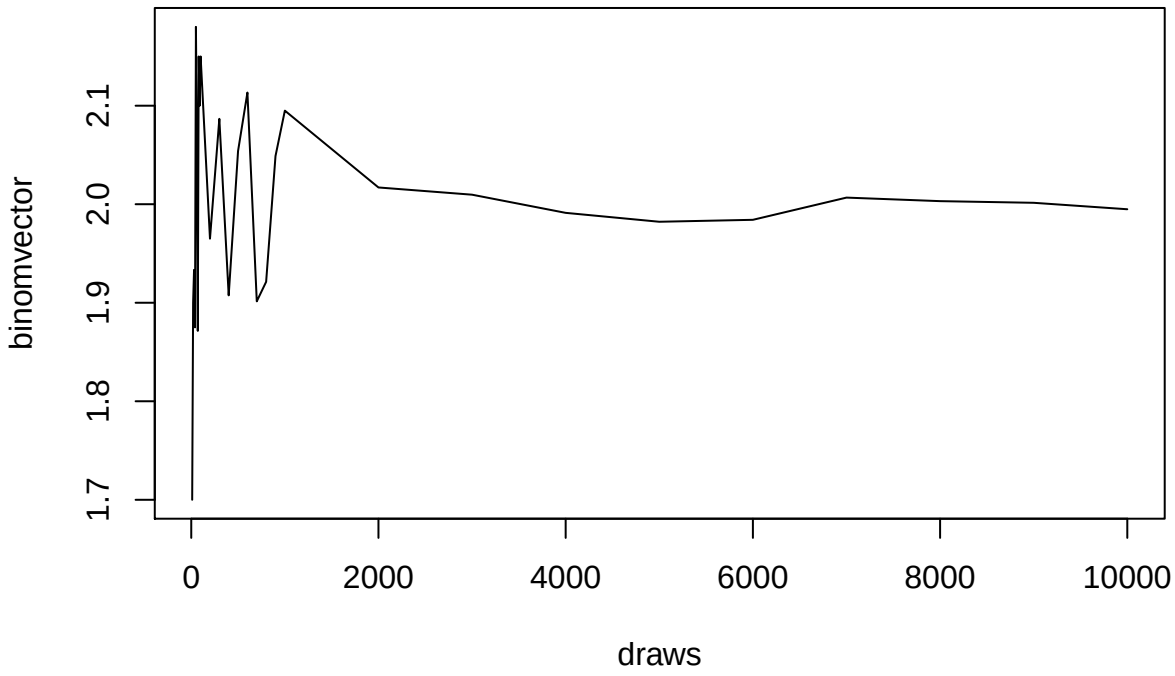
E[X] = size * prob = 10 * 0.2 = 2

E[Y] = shape / rate = shape / (1/scale) = shape * scale = 2 * 2 = 4

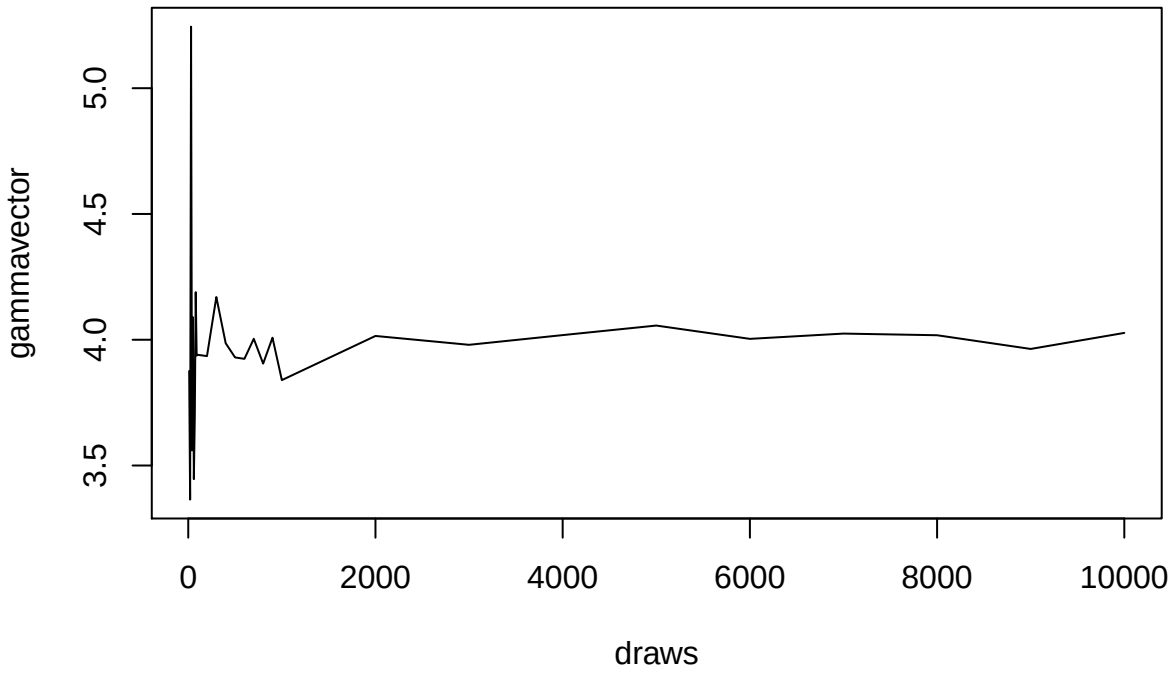
b)

```
binomvector <- c()
draws <- c()

n <- 0
while (n < 10000) {
  if(n < 100) {
    n <- n + 10
  }
  else if (n < 1000) {
    n <- n + 100
  }
  else if (n < 10000) {
    n <- n + 1000
  }
  x <- mean(binomfunc(n))
  binomvector <- c(binomvector, x)
  draws <- c(draws, n)
}
```

```
plot(draws, binomvector, type = "l")
```




```
gammavector <- c()
draws <- c()

n <- 0
while (n < 10000) {
  if(n < 100) {
    n <- n + 10
  }
  else if (n < 1000) {
    n <- n + 100
  }
  else if (n < 10000) {
    n <- n + 1000
  }
  x <- mean(gammafunc(n))
  gammavector <- c(gammavector, x)
  draws <- c(draws, n)
}

plot(draws, gammavector, type = "l")
```



Uppgift 9 Väntevärde och varians

```
exponentialfunc <- function(n) {
 x <- rexp(n, rate = 10)
 return(x)
}

poissonfunc <- function(n) {
  y <- rpois(n, lambda = 3)
  return(y)
}
```

a)

exponential distribution: E[X] = 1/rate = 1/10 = 0.1 Var[X] = 1/rate^2 = 1/100 = 0.01

poisson distribution: E[Y] = Var[Y] = lambda = 3

b)

```
# Beräknar E(X)
mean(exponentialfunc(10000))
```

```
## [1] 0.09972648
```

```r
# Beräknar Var(X)
var(exponentialfunc(10000))
```

```
## [1] 0.009887765
```

```r
# Beräknar E(Y)
mean(poissonfunc(10000))
```

```
## [1] 2.9813
```

```r
# Beräknar Var(Y)
var(poissonfunc(10000))
```

```
## [1] 3.008813
```

c)

1. $E(3) = 3$
2. $E(3 * X + 2) = 3 * E(X) + 2 = 3 * 0.1 + 2 = 2.3$
3. $E(X + Y) = E(X) + E(Y) = 0.1 + 3 = 3.1$
4. $E(X * Y) = E(X) * E(Y) = 0.1 * 3 = 0.3$
5. $E(3 * X + 2 * Y - 3) = 3 * E(X) + 2 * E(Y) - 3 = 3 * 0.1 + 2 * 3 - 3 = 3.3$
6. $Var(2 * X - 5) = 2^2 * Var(X) = 4 * 0.01 = 0.04$
7. $Var(X + Y) = Var(X) + Var(Y) = 0.01 + 3 = 3.01$

d)

```r
x <- exponentialfunc(10000)
y <- poissonfunc(10000)

#1
mean(3)
```

```
## [1] 3
```

```r
#2
mean(3*x+2)
```

```
## [1] 2.297308
```

```r
#3
mean(x+y)
```

```
## [1] 3.087003
```

```r
#4
mean(x*y)
```

```
## [1] 0.295966
```

```r
#5
mean(3*x+2*y-3)
```

```
## [1] 3.273108
```

```r
#6
var(2*x-5)
```

```
## [1] 0.04104537
```

```r
#7
var(x+y)
```

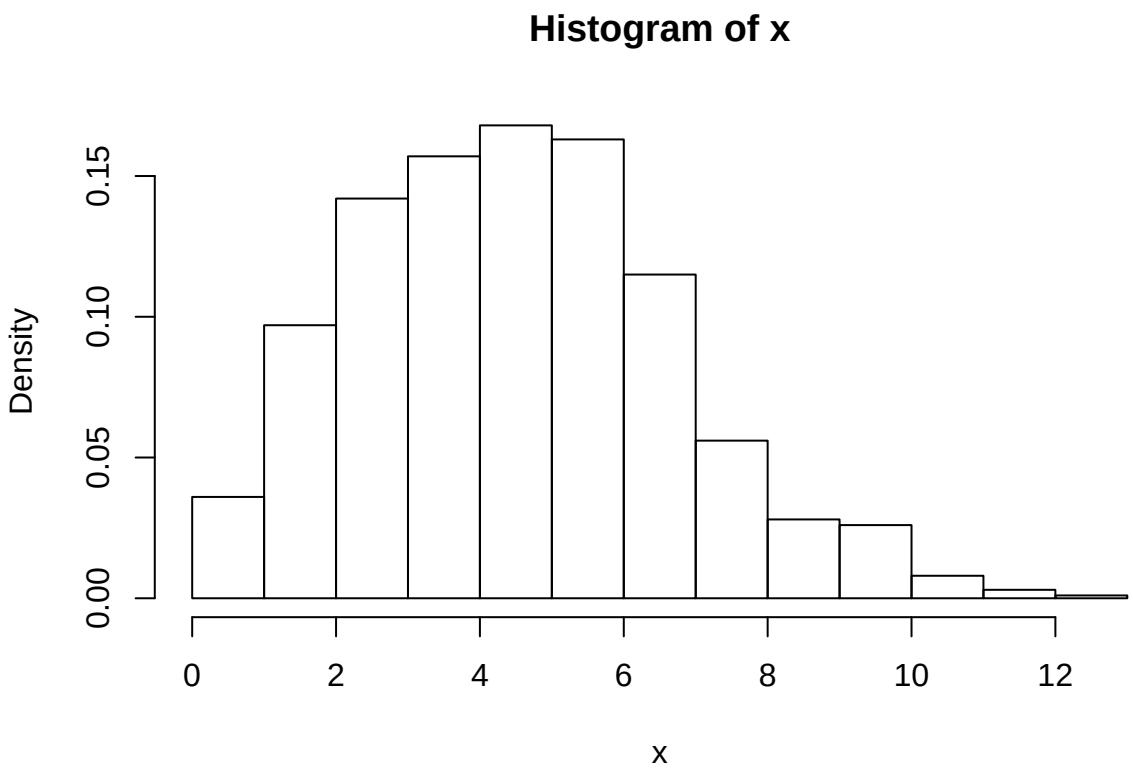
```
## [1] 2.987027
```

Uppgift 10 Centrala gränsvärdessatsen

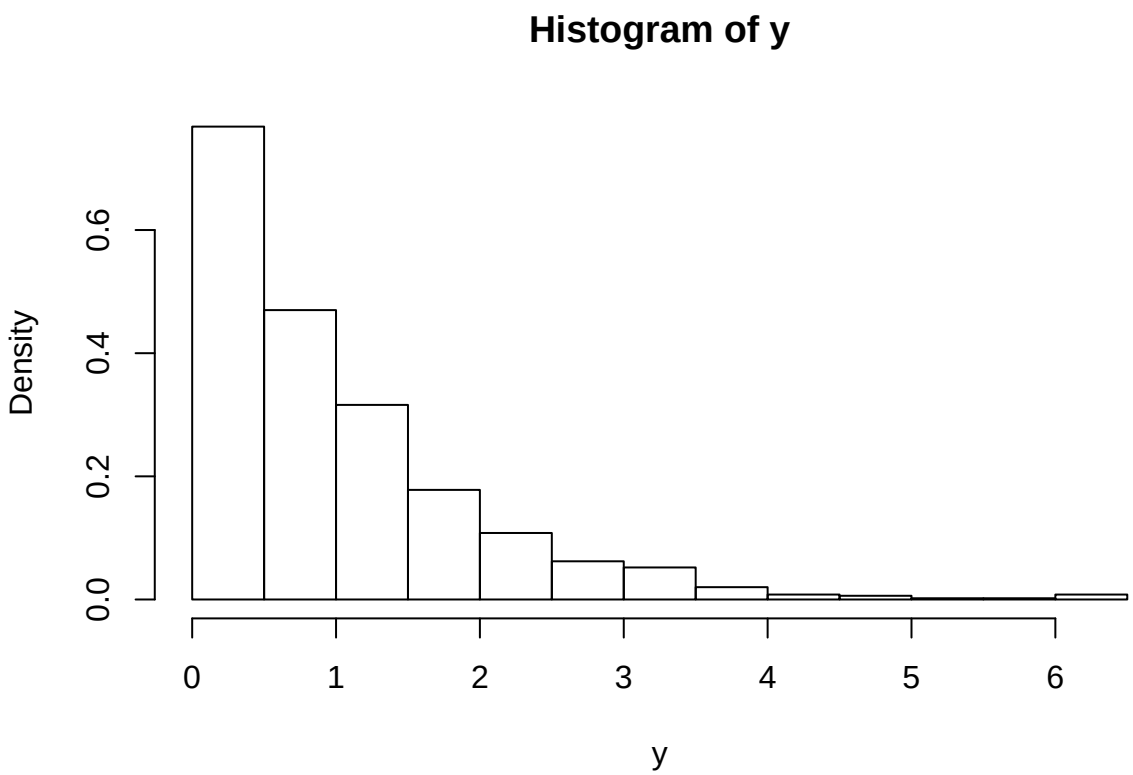
```r
poissonfunc <- function(n) {
  x <- rpois(n, lambda = 5)
  return(x)
}

exponentialfunc <- function(n) {
 y <- rexp(n, rate = 1)
 return(y)
}

binomfunc <- function(n) {
  z <- rbinom(n, size = 10, prob = 0.01)
  return(z)
}
```

a)

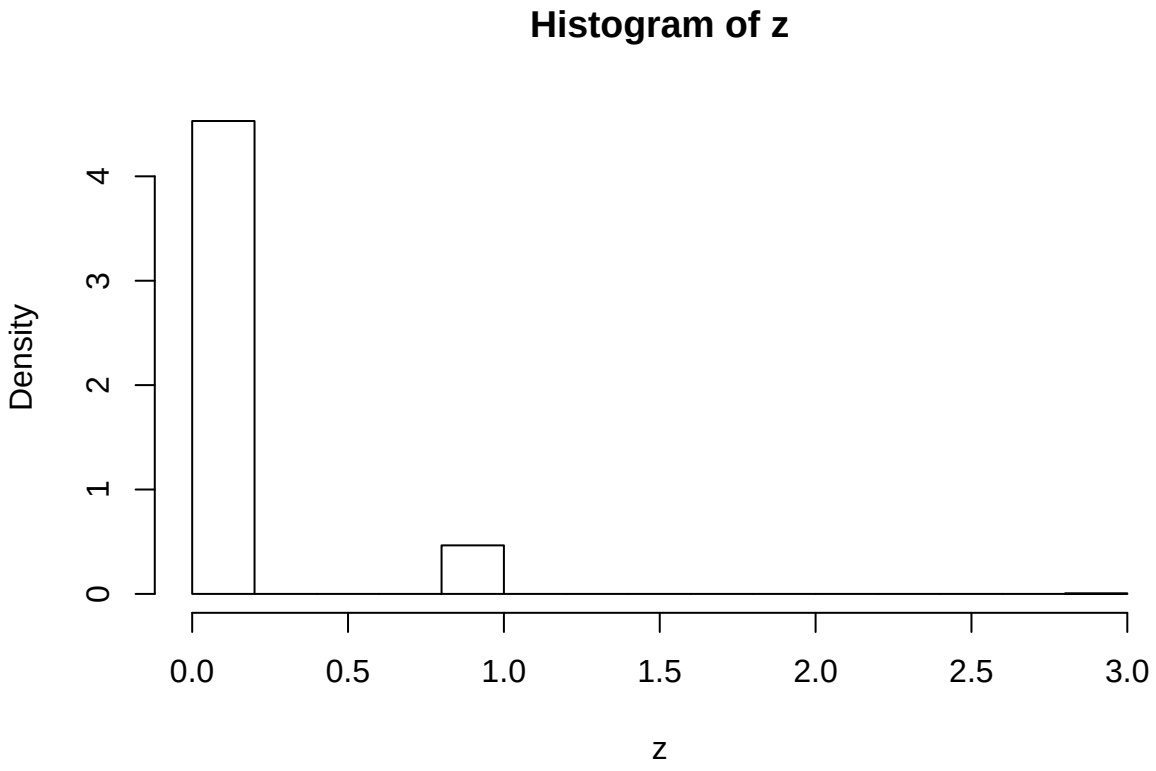
```r
x <- poissonfunc(1000)
hist(x, probability = TRUE)
```




```r
y <- exponentialfunc(1000)
hist(y, probability = TRUE)
```



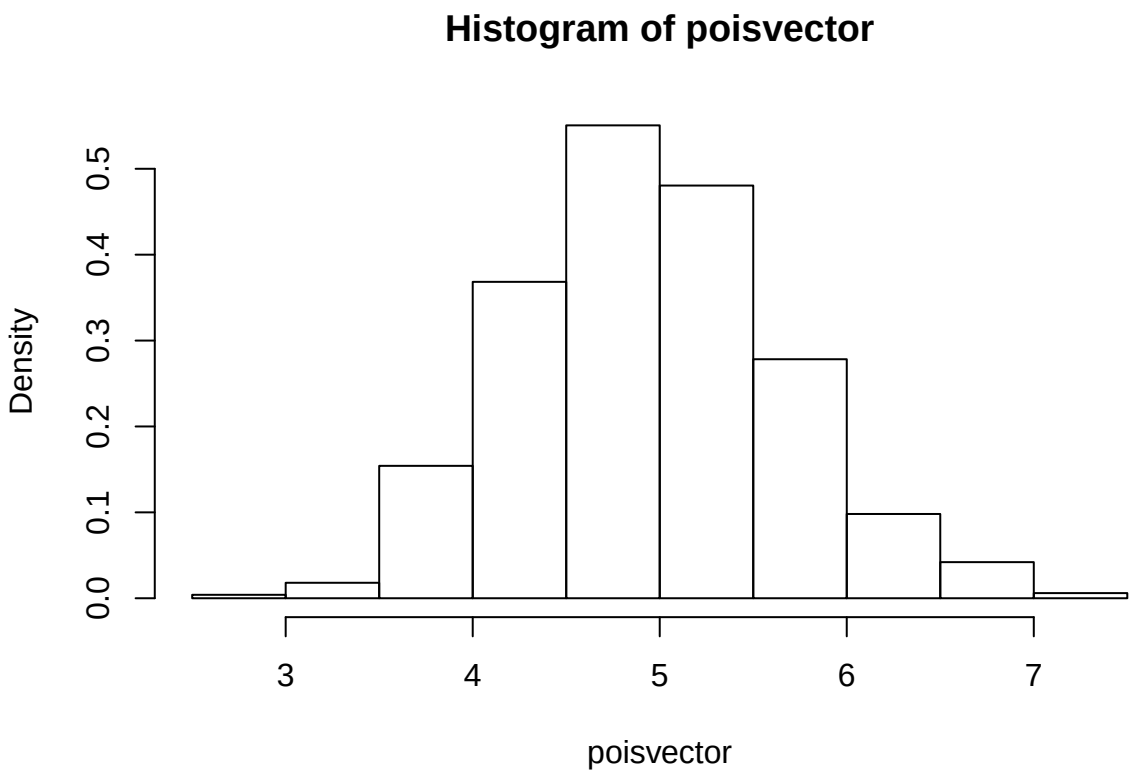
```r
z <- binomfunc(1000)
hist(z, probability = TRUE)
```



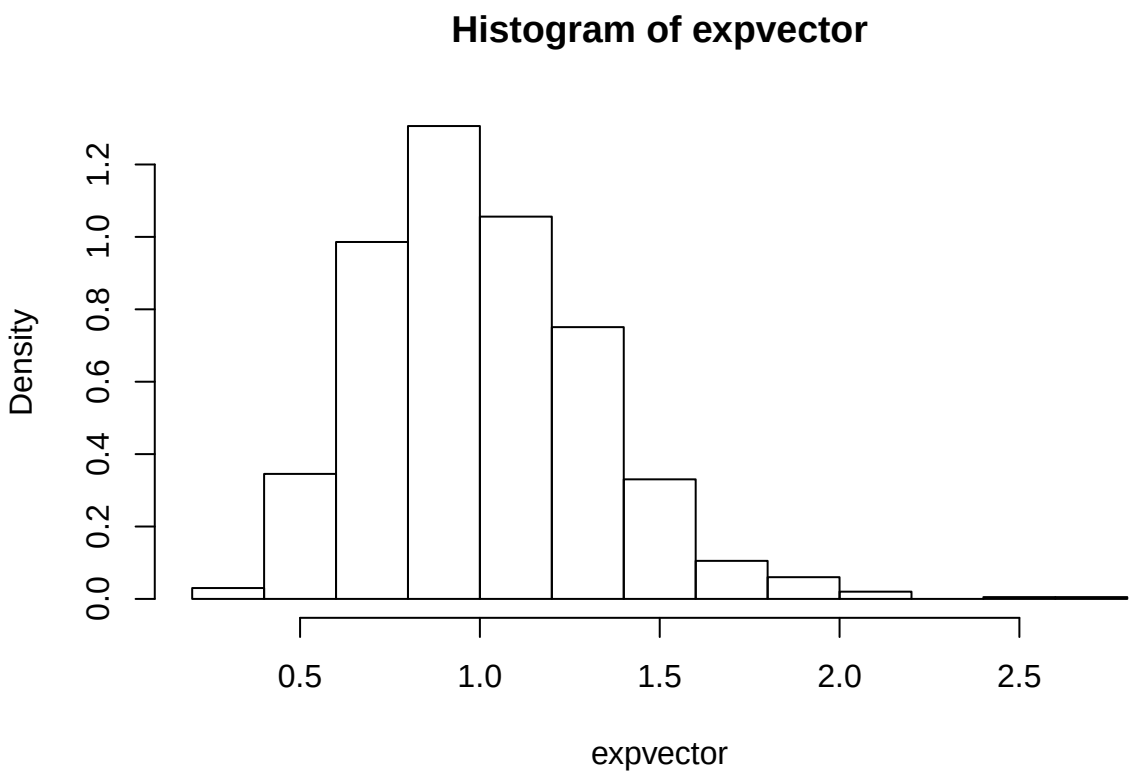
b)

```r
poisvector <- c()
expvector <- c()
i = 1
while (i < 1000) {
  poisvector <- c(poisvector, mean(poissonfunc(10)))
  expvector <- c(expvector, mean(exponentialfunc(10)))
  i <- i + 1
}
```

```r
hist(poisvector, probability = TRUE)
```




```
hist(expvector, probability = TRUE)
```

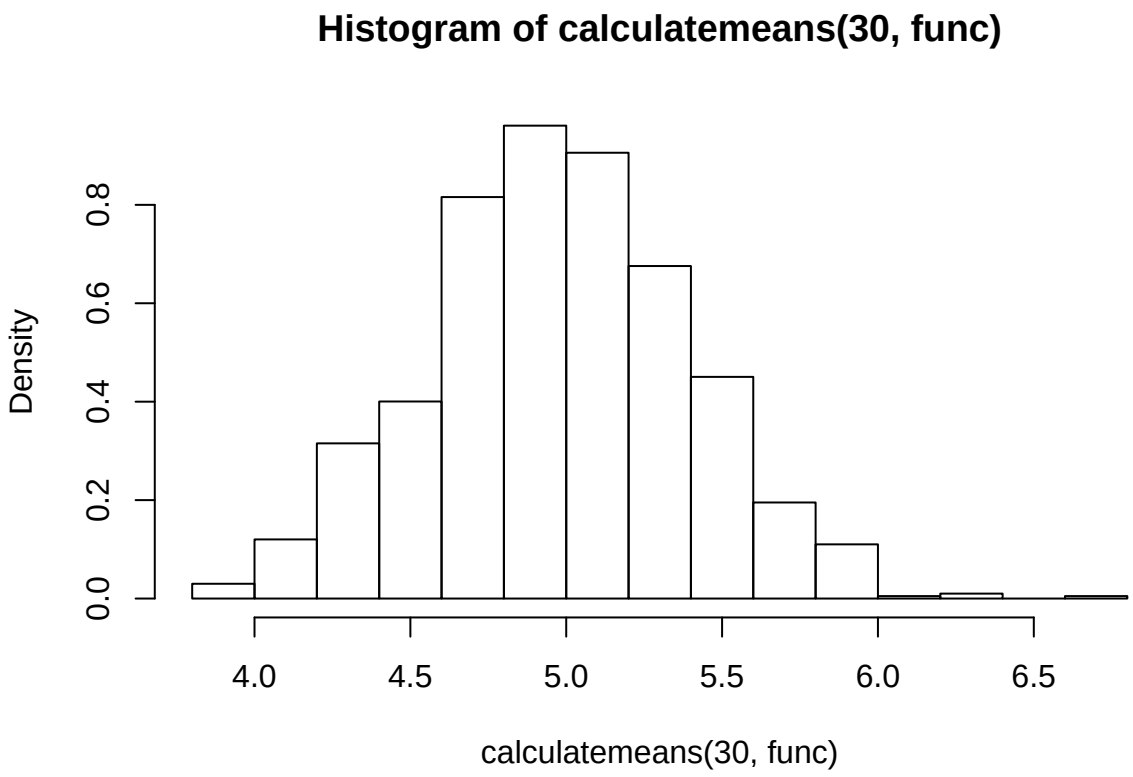


c)

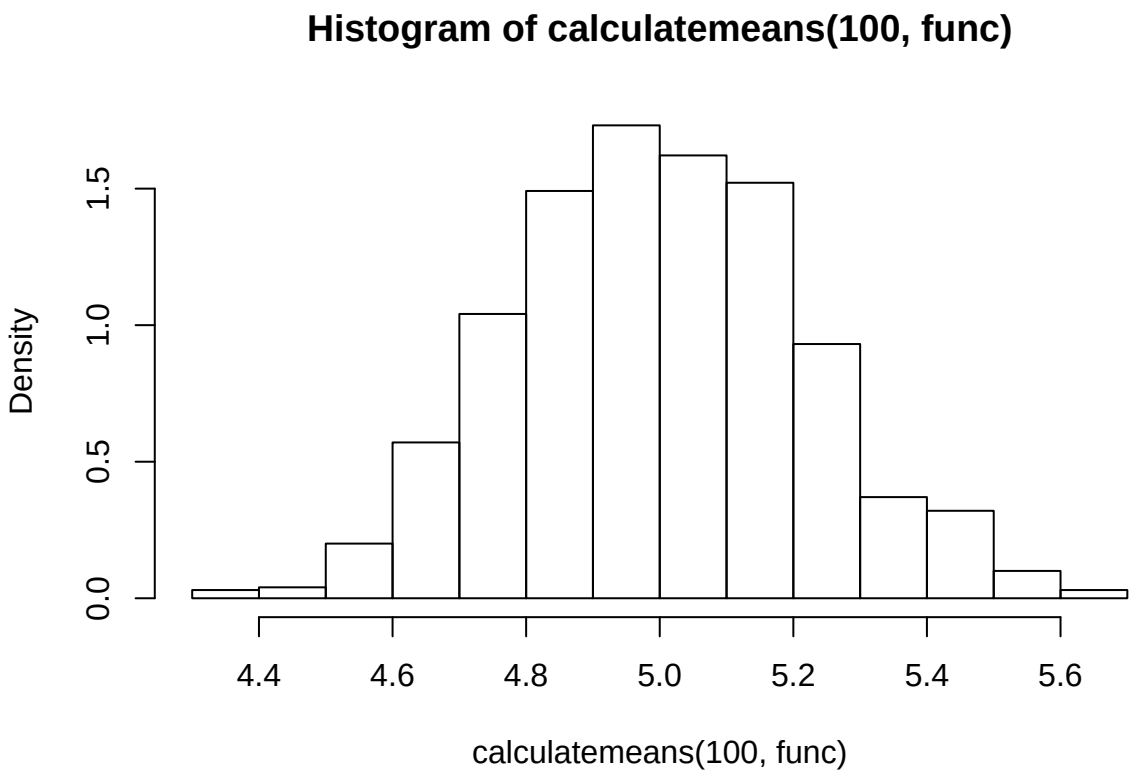
```
calculatemeans <- function(n, func) {
  vector <- c()
  i = 1
  while (i < 1000) {
    vector <- c(vector, mean(func(n)))
    i <- i + 1
  }
  return(vector)
}
```

Poisson

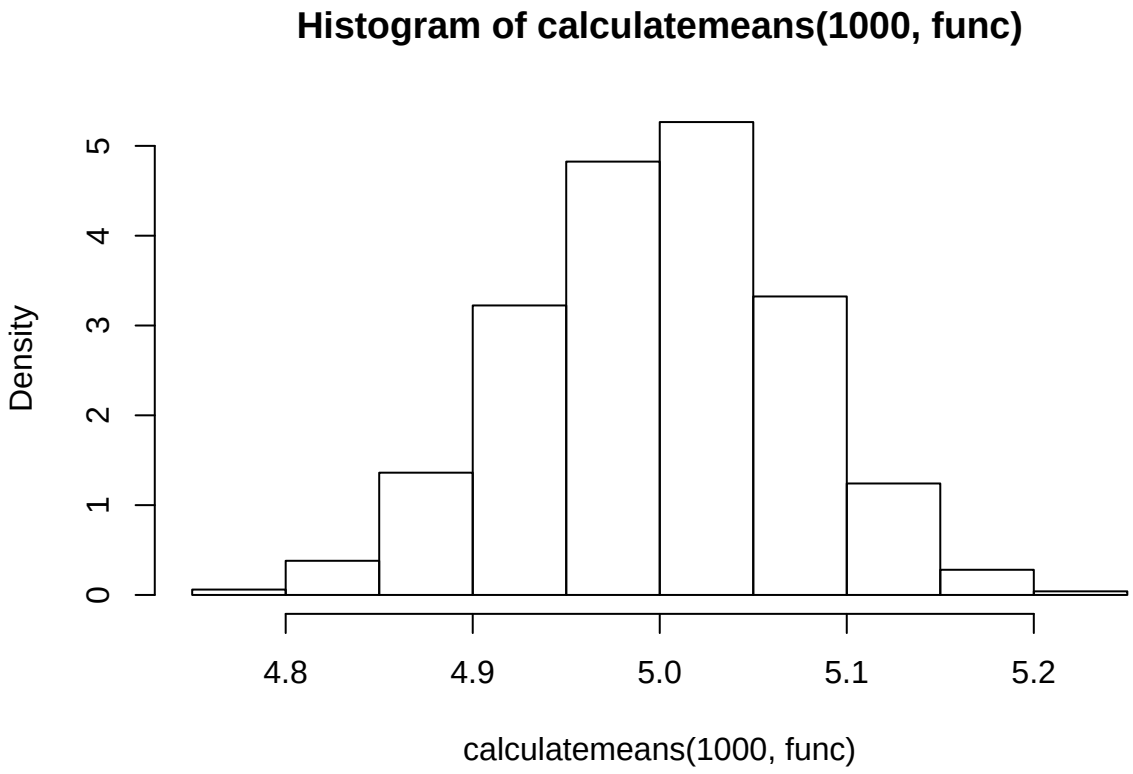
```
func <- poissonfunc
hist(calculatemeans(30, func), probability = TRUE)
```




```
func <- poissonfunc
hist(calculatemeans(100, func), probability = TRUE)
```

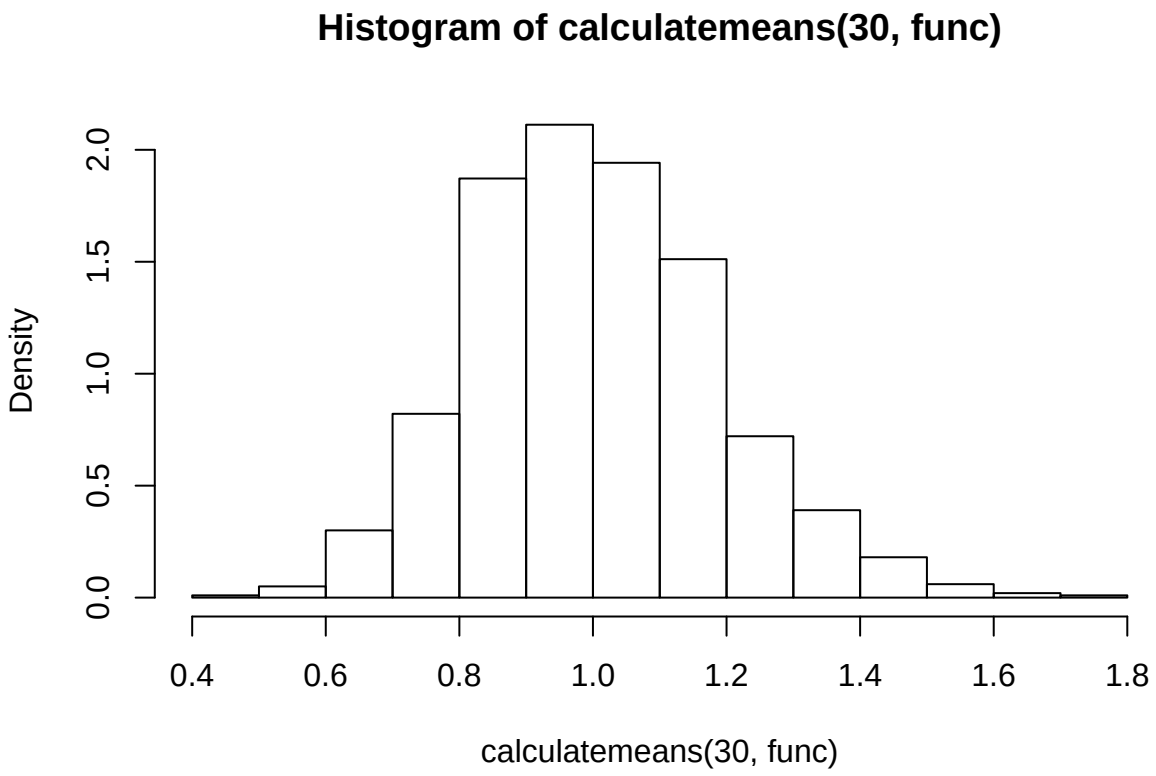



```
func <- poissonfunc
hist(calculatemeans(1000, func), probability = TRUE)
```

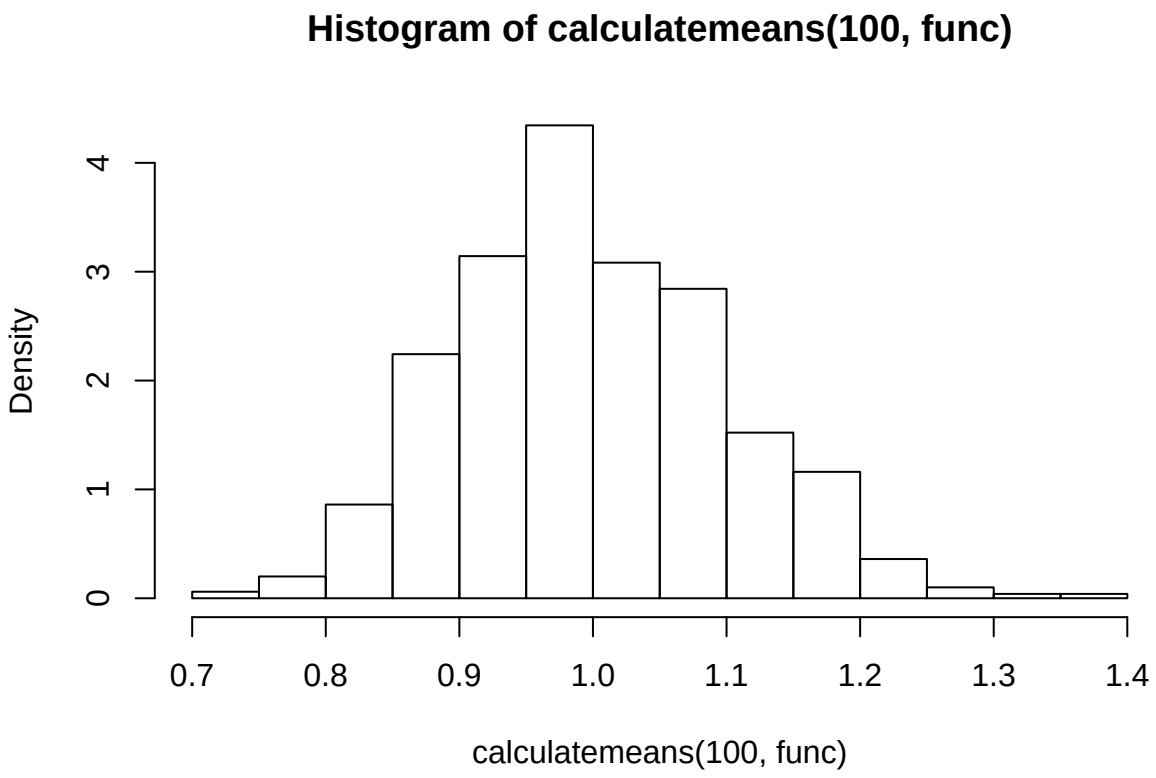


Exponential

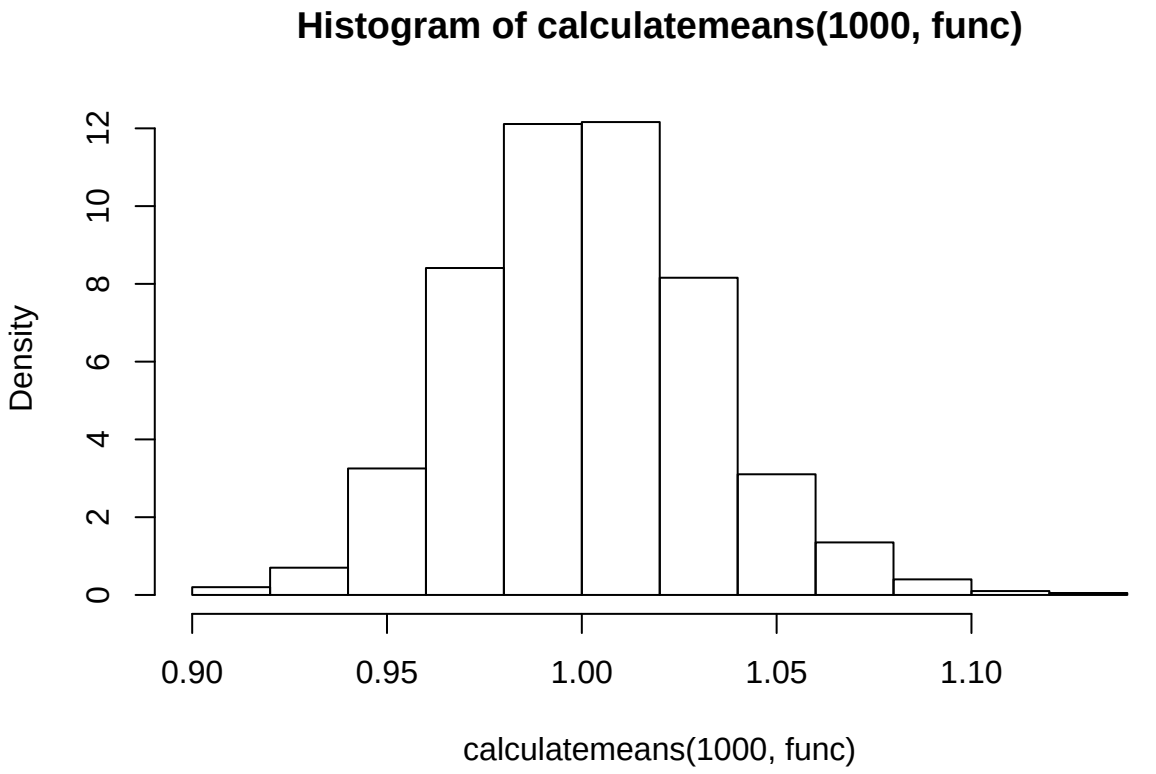
```
func <- exponentialfunc
hist(calculatemeans(30, func), probability = TRUE)
```




```
func <- exponentialfunc
hist(calculatemeans(100, func), probability = TRUE)
```

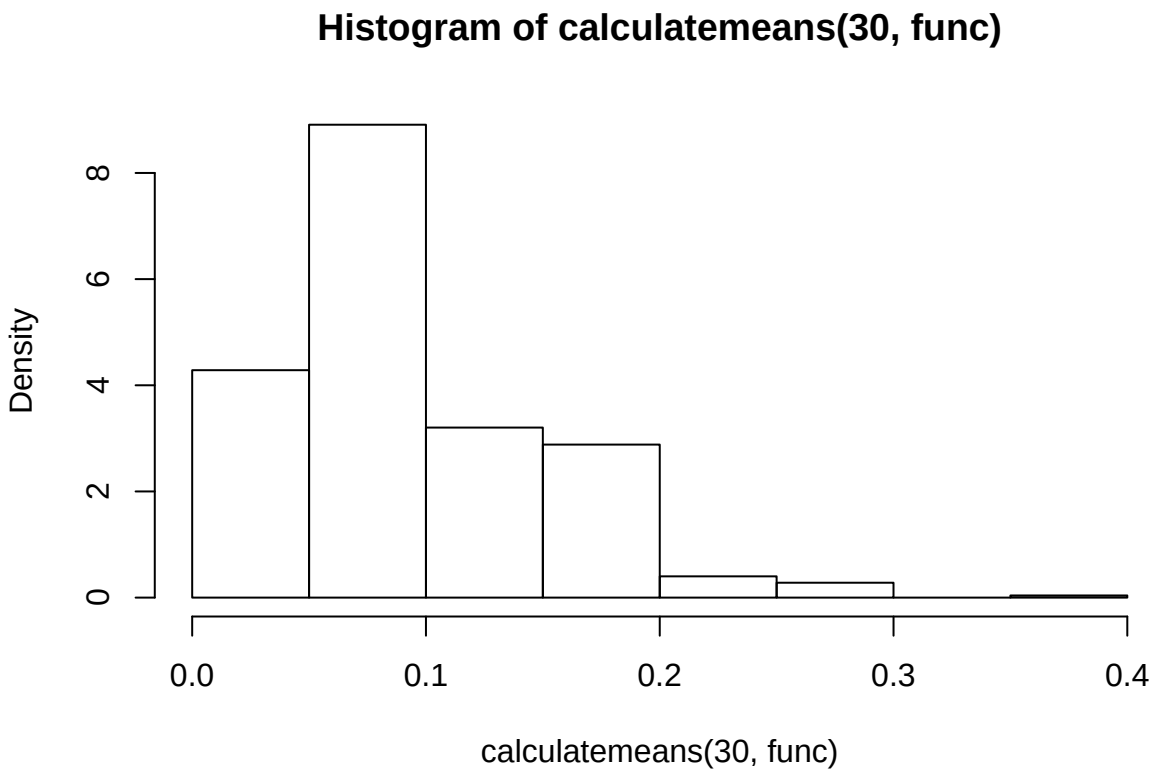



```r
func <- exponentialfunc
hist(calculatemeans(1000, func), probability = TRUE)
```

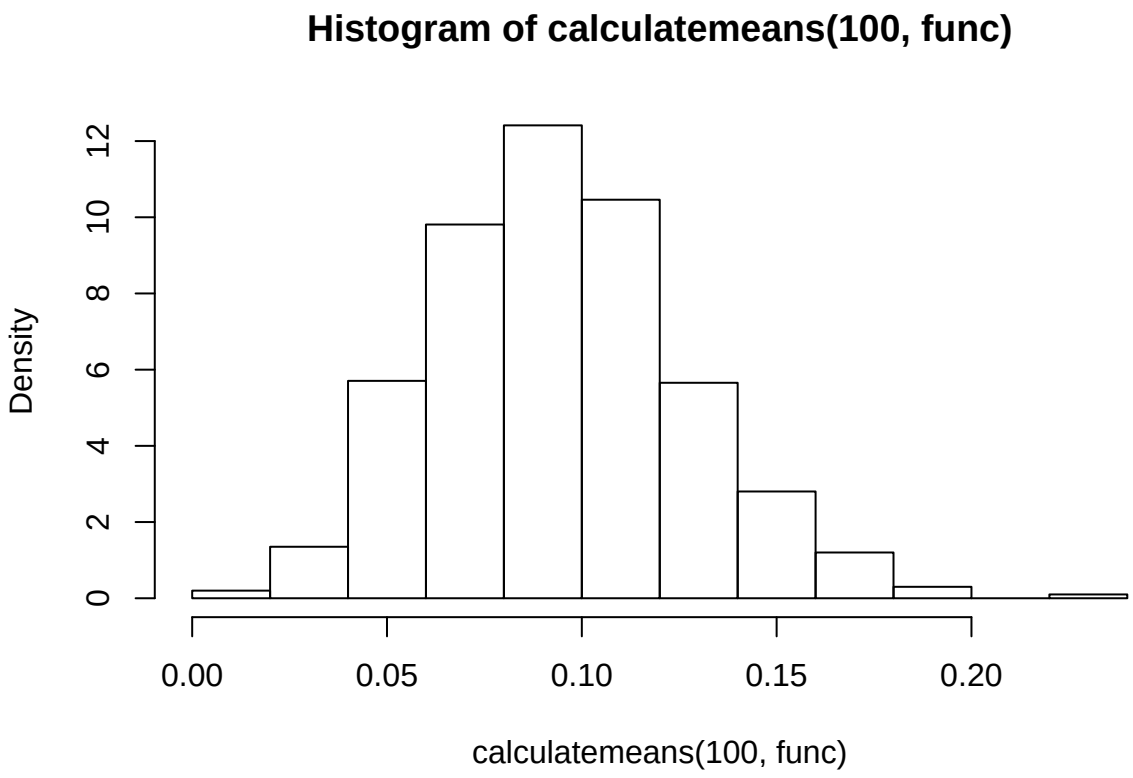


Binomial

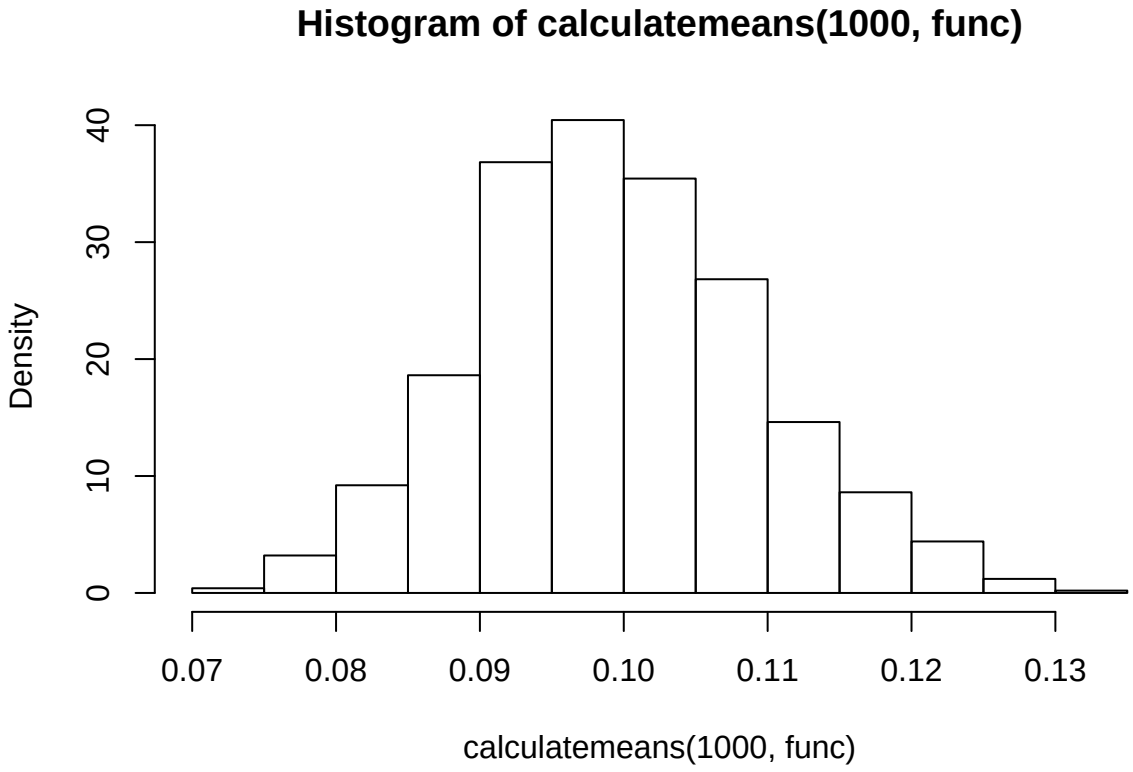
```
func <- binomfunc
hist(calculatemeans(30, func), probability = TRUE)
```




```
func <- binomfunc
hist(calculatemeans(100, func), probability = TRUE)
```




```
func <- binomfunc
hist(calculatemeans(1000, func), probability = TRUE)
```



Medelvärdena konvergerar mot en normalfördelning. Då väntevärdena för Y och Z är nära 0 ser det i början

ut som att histogrammen för dem bara visar ena halvan av en normalfördelning, se uppgift 10 a). Därmed ser det ut som att X konvergerar snabbast mot normalfördelningen.