R_intro_Monte_Carlo

play with a sequence

```
A_vector = rep(NA, 1001)
A_vector[1] = 1
A_vector[2] = 1
for(n in 2:1000){
  A_vector[n+1] = 2 * A_vector[n] + A_vector[n-1]
}
A_vector[201]
```

```
## [1] 1.795176e+76
```

```
A_vector[901]
```

```
## [1] Inf
```

how to work this out on log scale to avoid "Inf"

```
log_A_vector = rep(NA, 1001)
log_A_vector[1] = 0
log_A_vector[2] = 0
for(n in 2:1000){
  temp <- max(log_A_vector[n-1], log_A_vector[n])
  log_A_vector[n+1] = temp + log(2 * exp(log_A_vector[n] - temp) + exp(log_A_vector[n-1] - temp))
}

log_A_vector[201]
```

```
## [1] 175.5816
```

```
log_A_vector[901]
```

```
## [1] 792.5431
```

Vector and Matrix operations

```r
x <- c(1,2,3,5)
y <- c(3,5,7,6)
x+y
```

```
## [1]  4  7 10 11
```

```r
x*y
```

```
## [1]  3 10 21 30
```

```r
x%*%t(y)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    3    5    7    6
## [2,]    6   10   14   12
## [3,]    9   15   21   18
## [4,]   15   25   35   30
```

```r
x * 4
```

```
## [1]  4  8 12 20
```

```r
x^2
```

```
## [1]  1  4  9 25
```

R list (data frame)

```r
my_list <- list(First_class = c(1, 2, 3, 4, 4, 4),
                Second_class = matrix(1:4, 2, 2),
                last_element = 2.4,
                words = c("i", "love", "statistics"))
my_list$First_class
```

```
## [1] 1 2 3 4 4 4
```

```r
names(my_list)
```

```
## [1] "First_class"  "Second_class" "last_element" "words"
```

sample random variables

```r
# Coin flip

p <- 0.7
my_coin_flips <- rbinom(100, 1, p)

# simulate a Bernoulli random variable

p <- 0.7
y <- rbinom(100000, 1, p)
table(y)
```

```
## y
##     0     1
## 29968 70032
```

```r
# estimate moment
phat <- sum(y)/length(y)
print(c(phat, mean(y)))
```

```
## [1] 0.70032 0.70032
```

```r
print(c(phat*(1-phat), var(y)))
```

```
## [1] 0.2098719 0.2098740
```

uniform distribution

```r
x <- runif(1000, pi, 2*pi)
mean(x)
```

```
## [1] 4.680078
```

```r
# true value of the mean
(3*pi)/2
```

```
## [1] 4.712389
```

```r
# do it for multiple sample sizes

for(size in c(10, 100, 1000, 1000000)) {
  my_vals = runif(size, min = pi, max = 2*pi)
  print(size)
  print(mean(my_vals))
  print(var(my_vals))
}
```

```
## [1] 10
## [1] 5.050056
## [1] 0.7643
## [1] 100
## [1] 4.753062
## [1] 0.8684398
## [1] 1000
## [1] 4.748064
## [1] 0.8228995
## [1] 1e+06
## [1] 4.71075
## [1] 0.8224556
```

```r
# equivalent way of simulating bernoulli experiments
x <- runif(10000, 0, 1)
y <- (x < p)
table(as.integer(x < p))
```

```
##
##    0    1
## 2983 7017
```

what if the coin is biased with head prob. = 0.4?

Gaussian r.v.s N(0, 1)

```r
my_simulation <- list(ten_samples = rnorm(10),
                      twen_samp = rnorm(20),
                      fiftysample = rnorm(50),
                      hundredsamp = rnorm(100),
                      thousampl = rnorm(1000),
                      millionsampl = rnorm(1000000))

sapply(my_simulation, mean)
```

```
##   ten_samples     twen_samp   fiftysample   hundredsamp     thousampl
##  0.0894066124  0.0414342446  0.1229750162 -0.0819672490  0.0605598329
##  millionsampl
## -0.0005179793
```

```r
sapply(my_simulation, var)
```

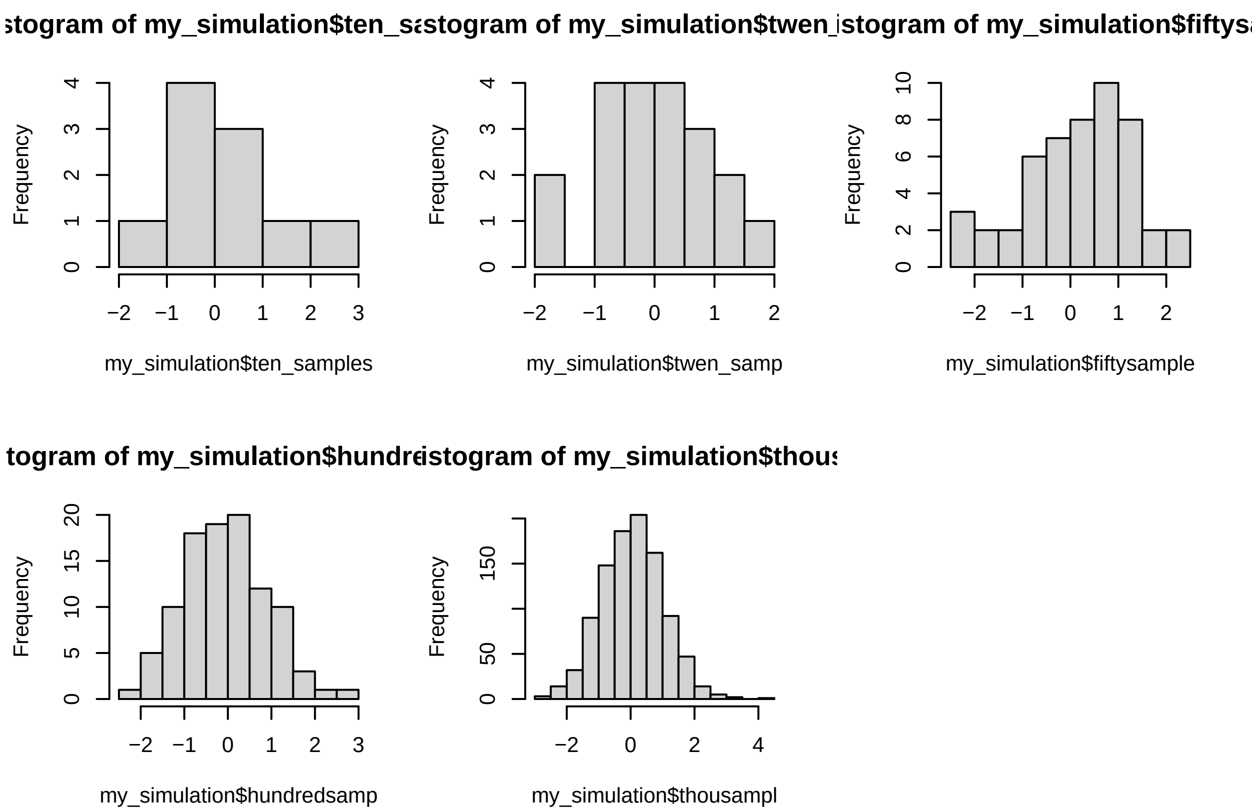
```
##  ten_samples    twen_samp  fiftysample  hundredsamp    thousampl millionsampl
##    1.3246898    0.9158714    1.1730845    0.8796913    0.9472722    0.9987203
```

```r
# recover density using Monte Carlo Samples

par(mfrow = c(2, 3))
```

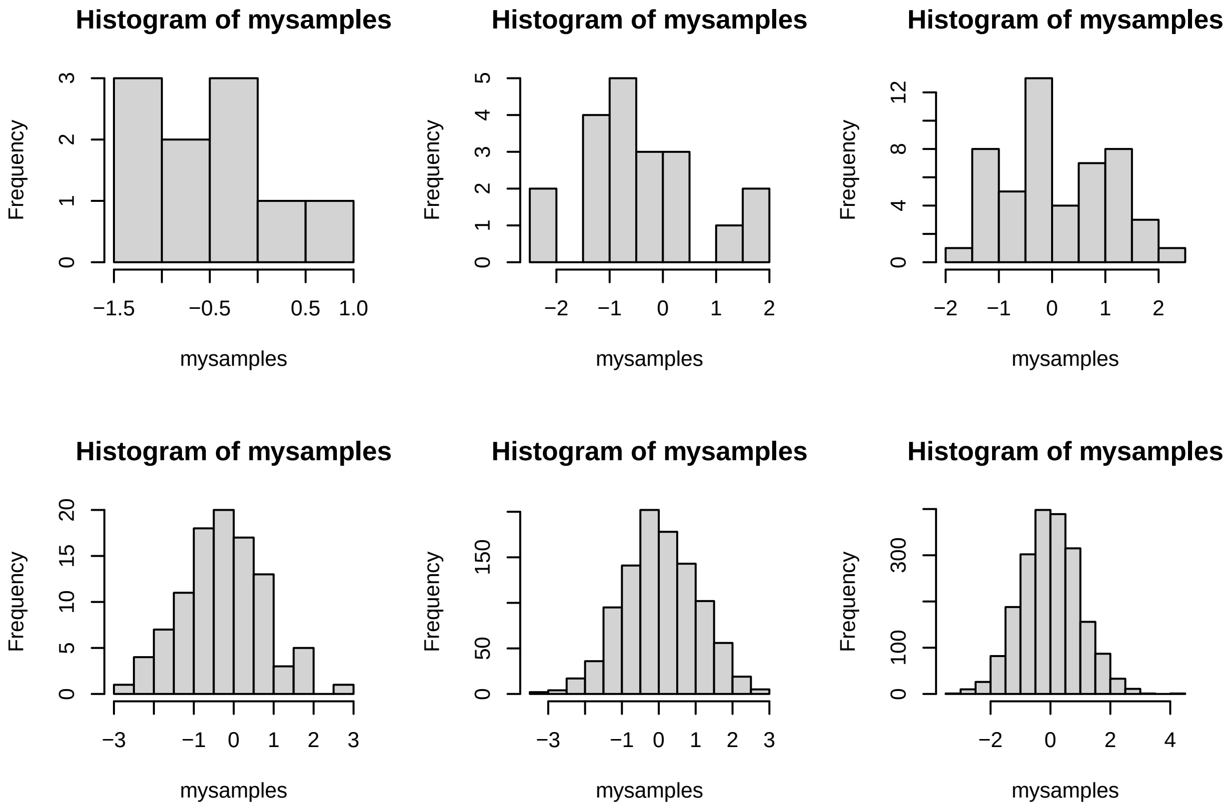
```r
hist(my_simulation$ten_samples)
hist(my_simulation$twen_samp)
hist(my_simulation$fiftysample)
hist(my_simulation$hundredsamp)
hist(my_simulation$thousampl)

# Compute Expectation and Variance of r.v.s using Monte Carlo

print(mean(my_simulation$ten_samples))
```

```
## [1] 0.08940661
```

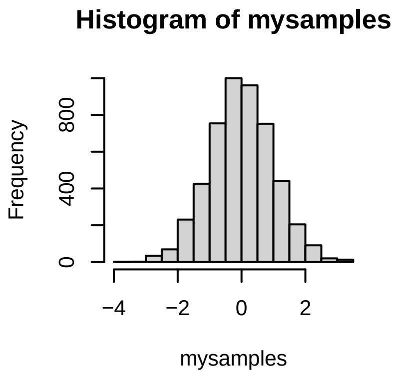
```r
par(mfrow = c(2, 3))
```



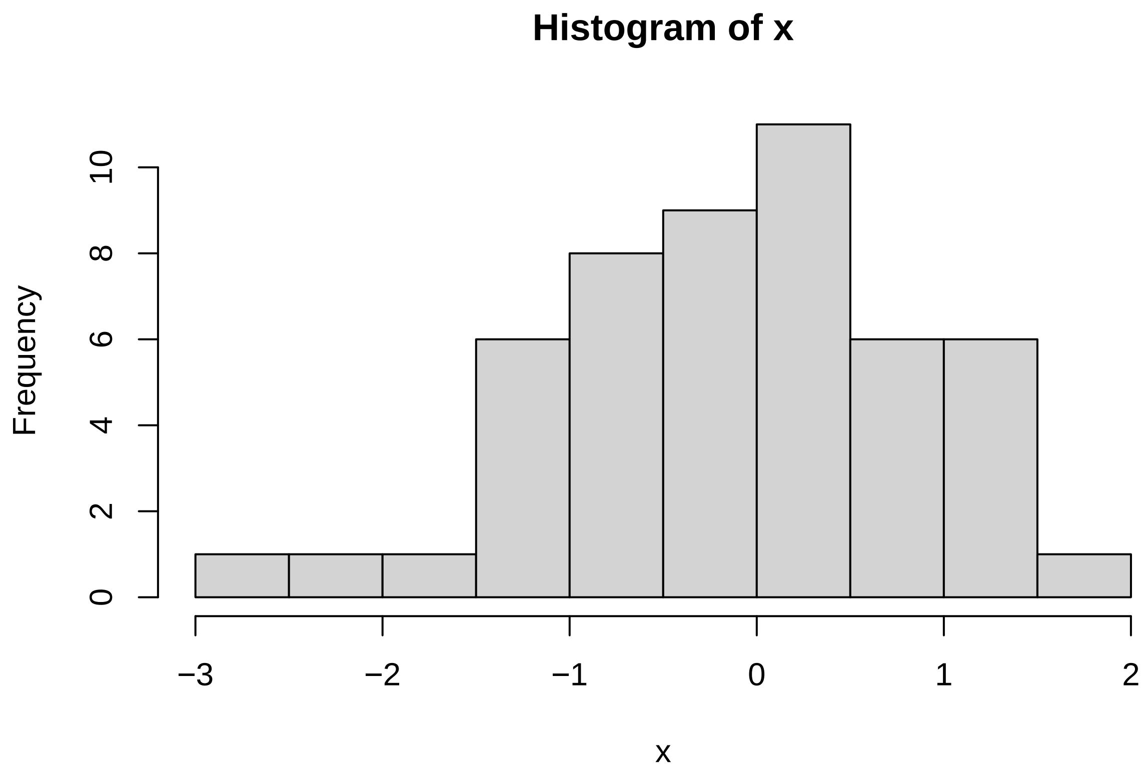

```r
Nsamples_vec <- c(10, 20, 50, 100, 1000, 2000, 5000)
for(k in Nsamples_vec){
  mysamples <- rnorm(k)
  hist(mysamples)
}
```



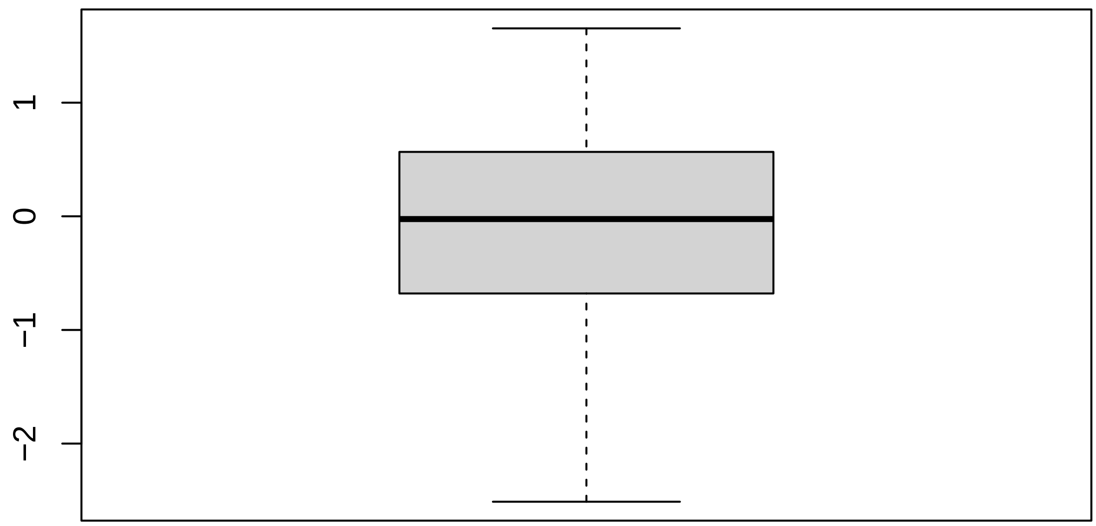
Histogram of mysamples
Frequency
mysamples
−1.5
−0.5
0.5
1.0
0
1
2
3
Histogram of mysamples
Frequency
mysamples
−2
−1
0
1
2
0
1
2
3
4
5
Histogram of mysamples
Frequency
mysamples
−2
−1
0
1
2
0
4
8
12
Histogram of mysamples
Frequency
mysamples
−3
−1
0
1
2
3
0
5
10
15
20
Histogram of mysamples
Frequency
mysamples
−3
−1
0
1
2
3
0
50
150
Histogram of mysamples
Frequency
mysamples
−2
0
2
4
0
100
300



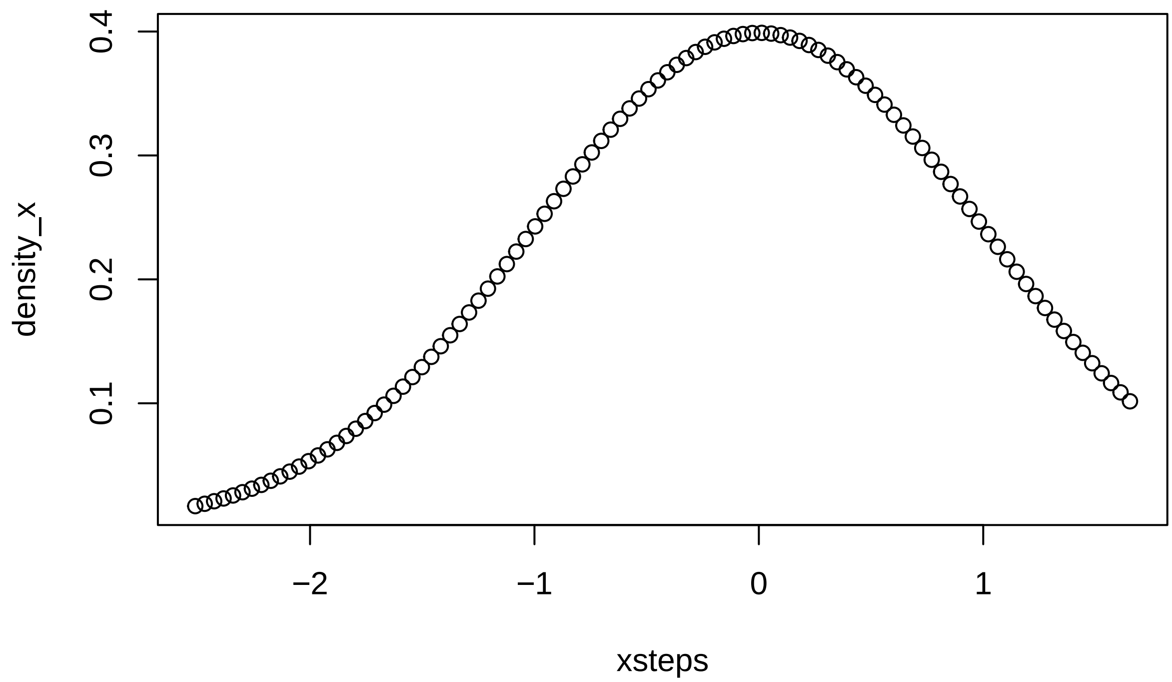

```
Nsamples <- 50
x <- rnorm(Nsamples)
hist(x)
```



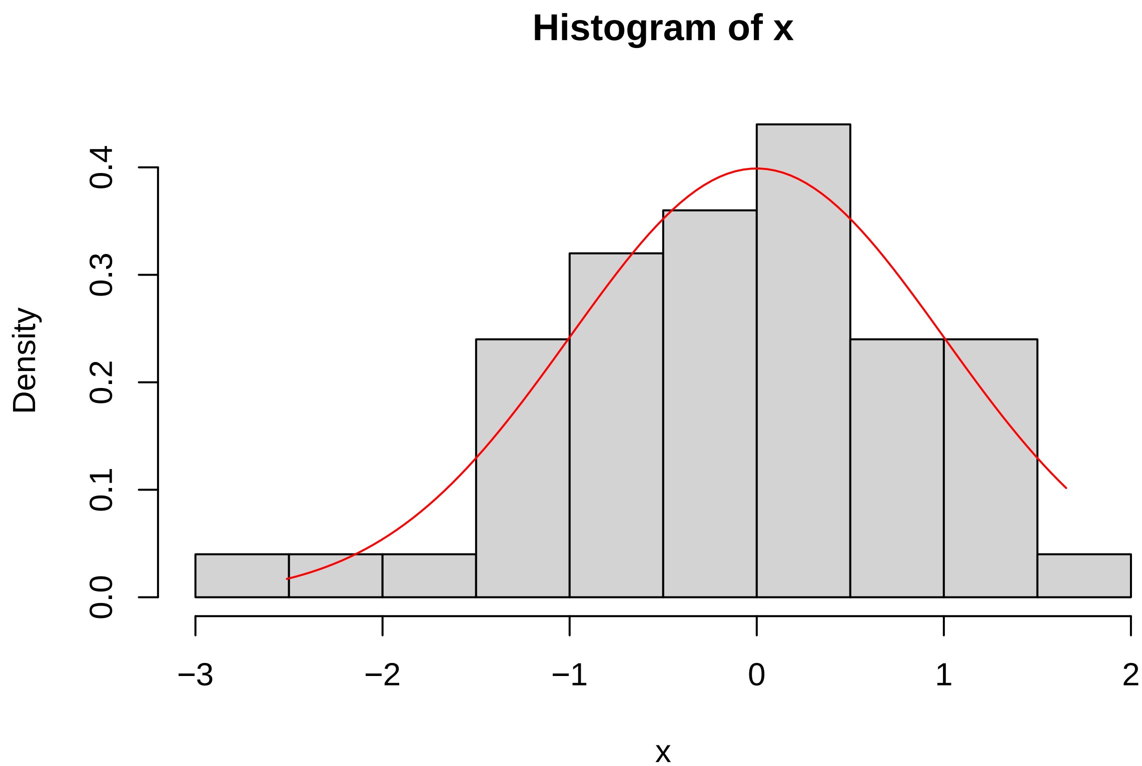

```
boxplot(x)
```




```
xrange <- range(x)
xsteps <- seq(xrange[1], xrange[2], length.out = 100)
density_x <- dnorm(xsteps)
plot(xsteps, density_x)
```



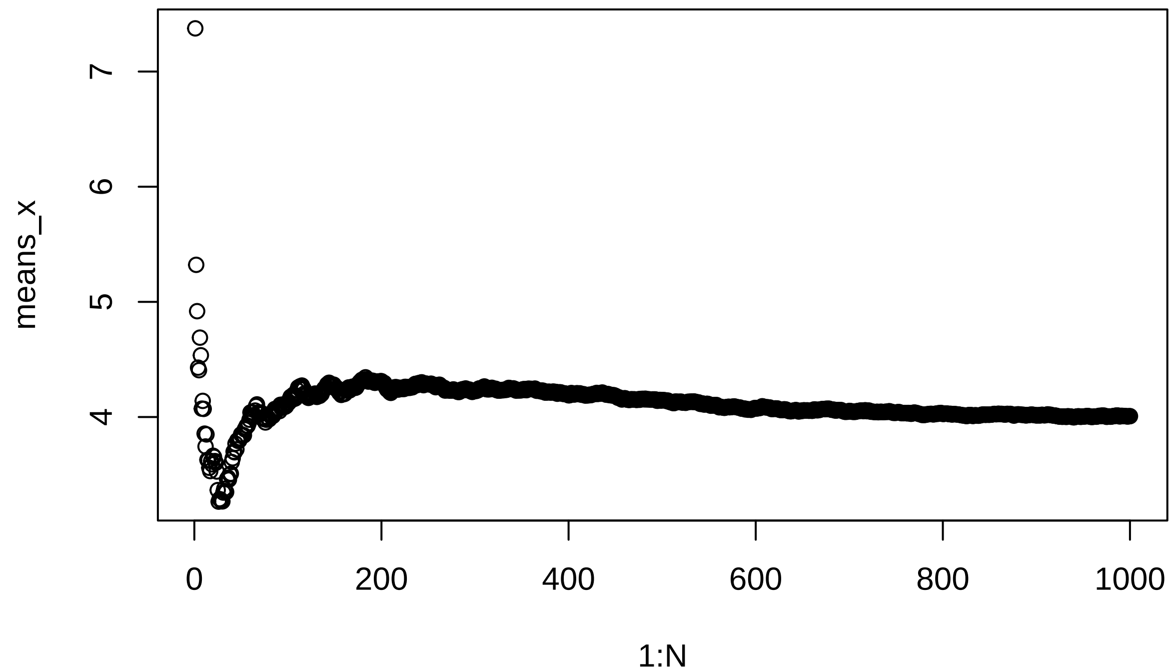

```r
hist(x, freq = FALSE)
lines(xsteps, density_x, col = "red")
```



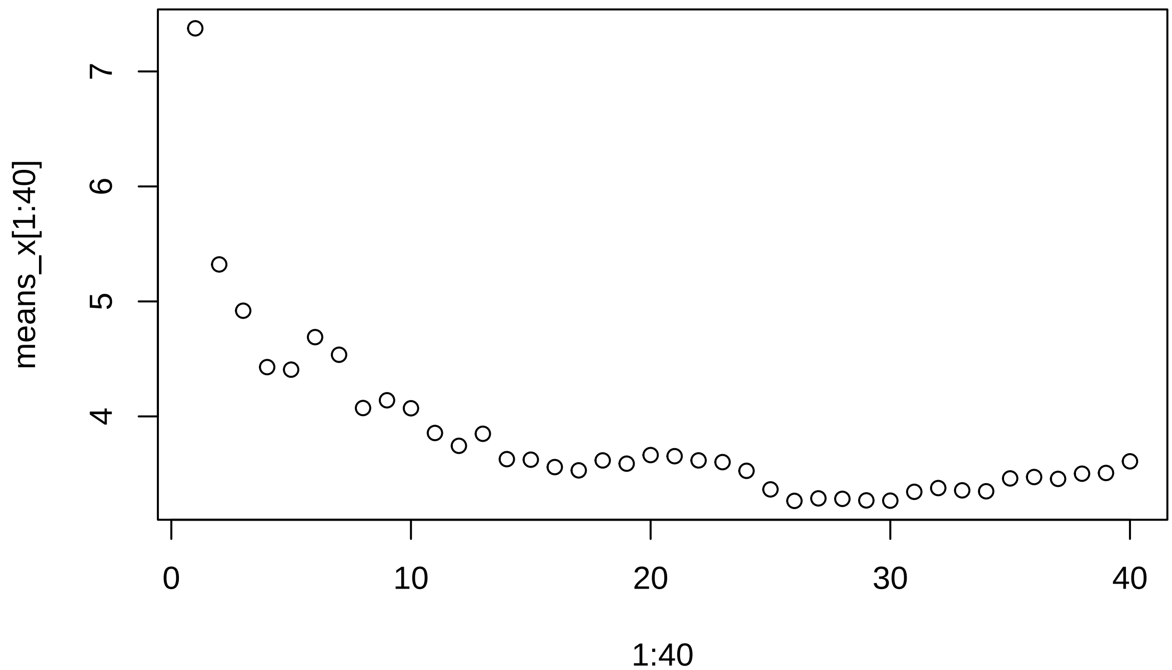
test out the same things with chi-squared or gamma distribution

law of large numbers

```
N <- 1000
x <- rnorm(N, 4, 2)
means_x <- cumsum(x) / (1:N)
plot(1:N, means_x)
```

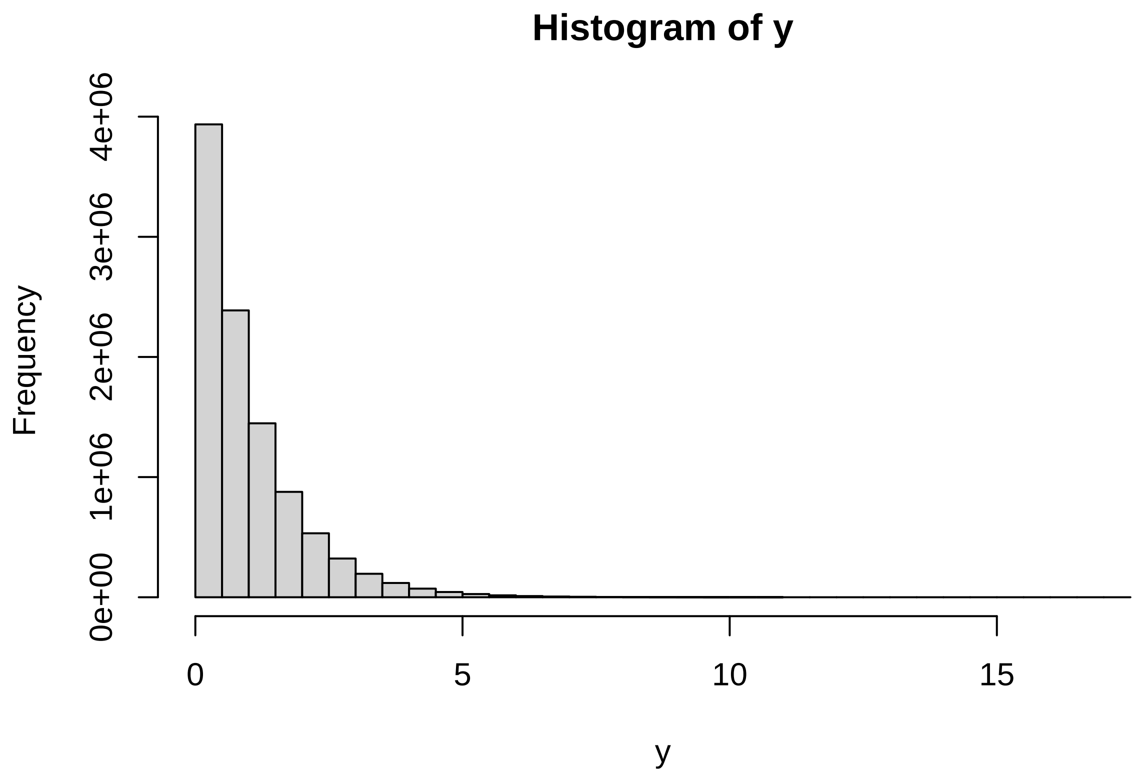



```
plot(1:40, means_x[1:40])
```

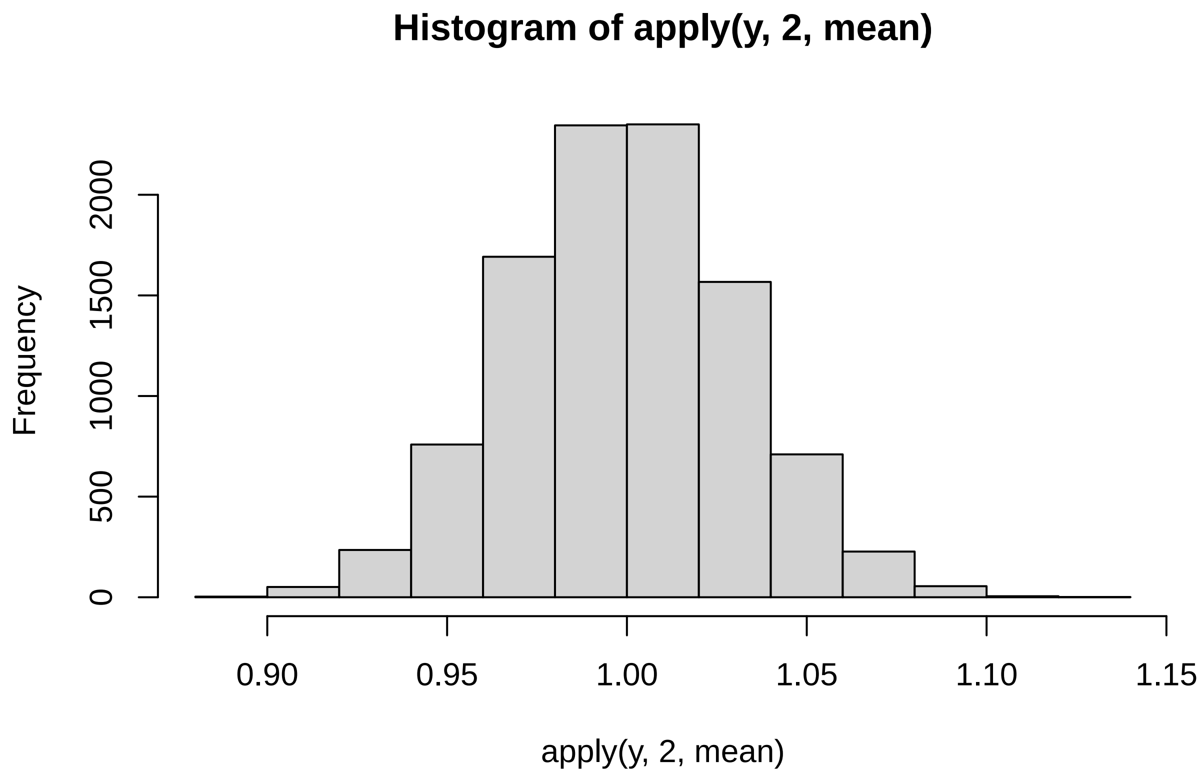


central limit theorem

```
y <- array(rgamma(1000 * 10000, 1), c(1000, 10000))
hist(y)
```




```
hist(apply(y, 2, mean))
```



Monte Carlo Integration

```
x <- rgamma(100, 1, 1)
# Computing expectation
mean(x)
```

```
## [1] 1.189894
```

```
# Computing variance
var(x)
```

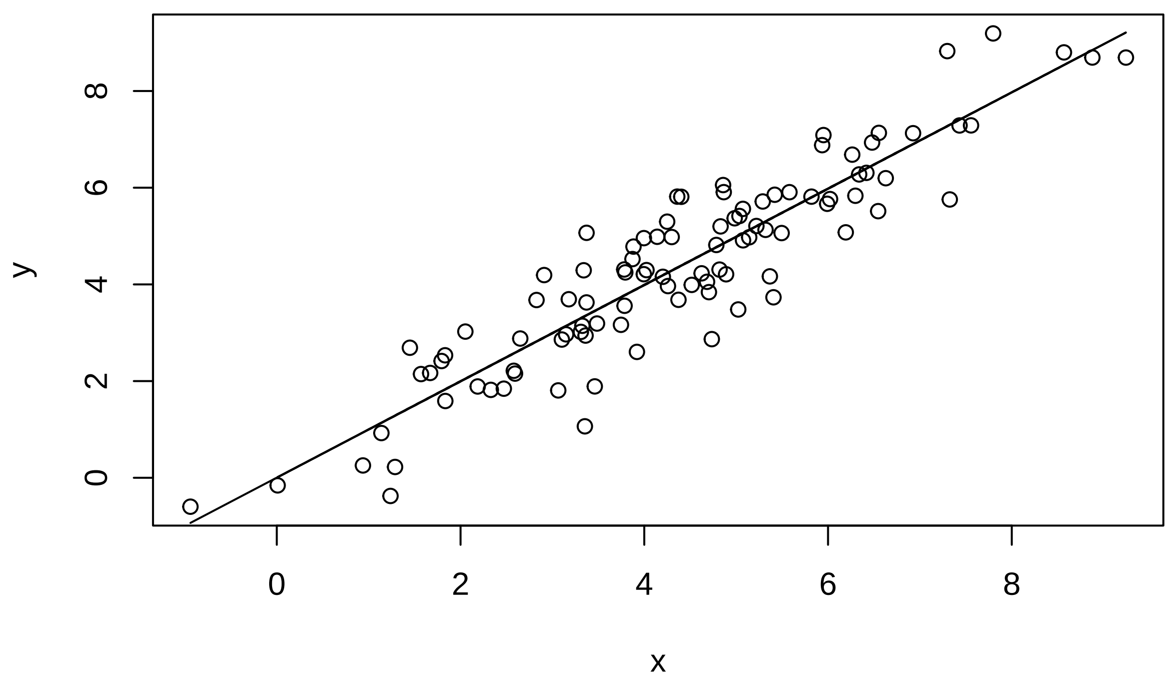
```
## [1] 1.672165
```

```
# Computing tail probability
sum(x > 3) / length(x)
```

```
## [1] 0.11
```

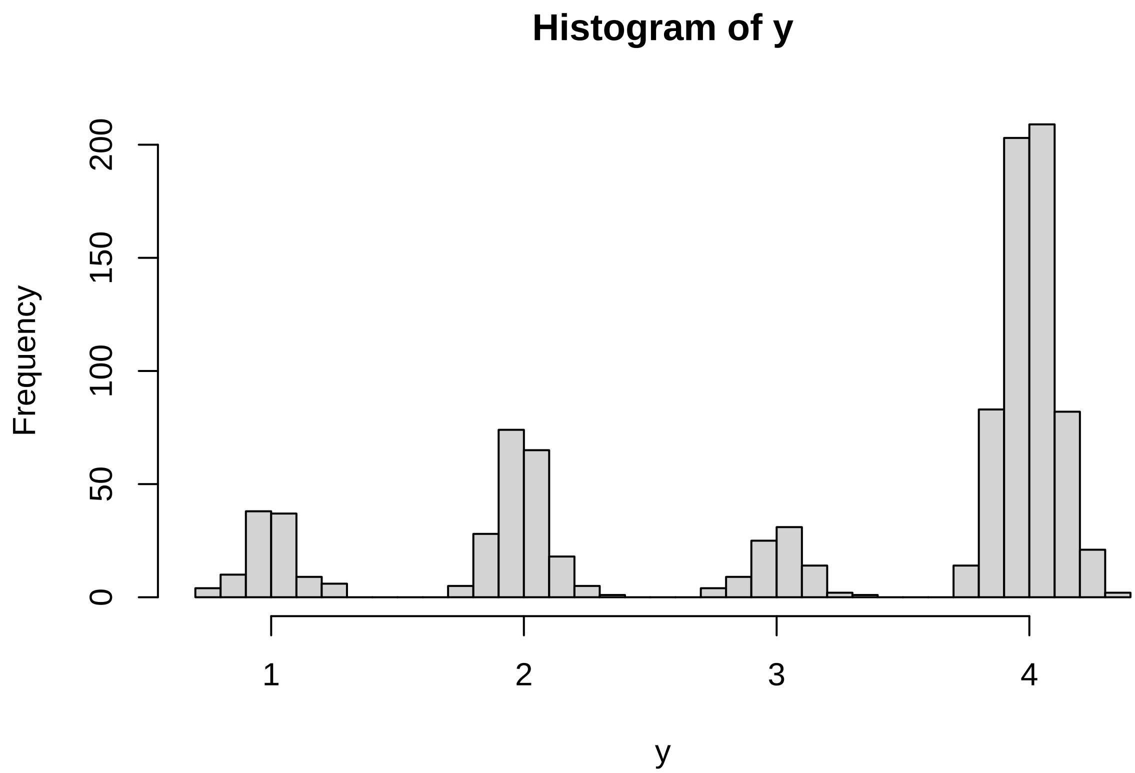
Regression Models

```
N <- 100
beta <- 1
sigma = 0.8

x <- rnorm(N, 4, 2)
y <- beta * x + rnorm(N) * sigma

lm_result <- lm(y~x)
plot(x, y)
lines(x, lm_result$fitted.values)
```



Clustering Models

```
N <- 1000
cluster <- sample(c(1, 2, 3, 4), N, replace = TRUE, prob = c(0.1, 0.2, 0.1, 0.6))
mu <- c(1, 2, 3, 4)
sigma <- rep(0.1, 4)
y <- rnorm(N) * sigma[cluster] + mu[cluster]
hist(y, 50)
```



QUIZ game

```
probquiz <- runif(1)
#probquiz <- sample(0.1*(1:6), 1, replace = TRUE, prob = rep(1, 6))
B <- rbinom(n = 1, size = 1, prob = probquiz)
print(list(as.integer(B), probquiz))
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 0.8565726
```

Exercise: if we have 26 lectures during the semester, what is the expected number of quizzes? What is the corresponding standard deviation?

Exercise: can you verify the results above theoretically?