Probability and Simulation

Statistics 427: R Programming

Module 14

2020

Random variables

In the baseball simulation done in Module 7, the proportion of successes out of 30 attempts was recorded. When the process was repeated, it was typical for a new proportion to occur. The proportion of successes is a quantity that will vary during repeated sampling.

Quantities that vary at random are called random variables; they are akin to a mathematical function, and many process generate random variables. Think of capture-and-release methods to study birds and measure their beak length, every different bird measured will vary to some degree.

Random variables

In science random variable are important because they are typically counts or measurement produced by some process by which scientific data are generated. Science is full of quantities that vary: amount of daily rainfall at a weather station, household income among households in a city, growth yields of wheat under certain conditions, distances of galaxies from Earth, the number of craters in 100-km^2 areas on Venus, and the list can go on. Think about a process and most of the time they could be considered random variables.

Probability

What can possibly be learned from processes that have varying levels of outcomes? Look for variability in the patterns of the outcomes.

The study of ways to make reliable conclusions from outcomes of random variables is the branch of **science** called *statistics*. The main driving force of statistics is in building and testing mathematical models of how the outcomes arise, and for use as hypotheses about our understanding of the processes involved. The study of patterns and randomness is a branch of mathematics called *probability*.

Probability

Law of Large Numbers (LLN): if a random process could be repeated many, many, many, *many* times, the *long-run* proportion of successes will stabilize and get closer and closer to the theoretical probability. For a small number of repetitions, considerable variation can be expected to happen.

and the next day... and the next day... and the next day...

LLN

Building a function to generate a sequence of successes and failures. The function will return with a vector of 0s and 1s (0=failure, 1=success) based on input of a sample size and probability of success. The idea is to use the function to simulate samples of size 5, then size 6, size 7, and so on until maybe 500. For each block, the mean is calculated and look at the variability in the means decreasing.

Simulation of LLN

```
outcomes=function(n,p){
  u=runif(n)
  x=1*(u<=p) # easier way to convert logical (T/F) to numeric (0,1)
  return(x)
}
# test it out
outcomes(10,.8)
```

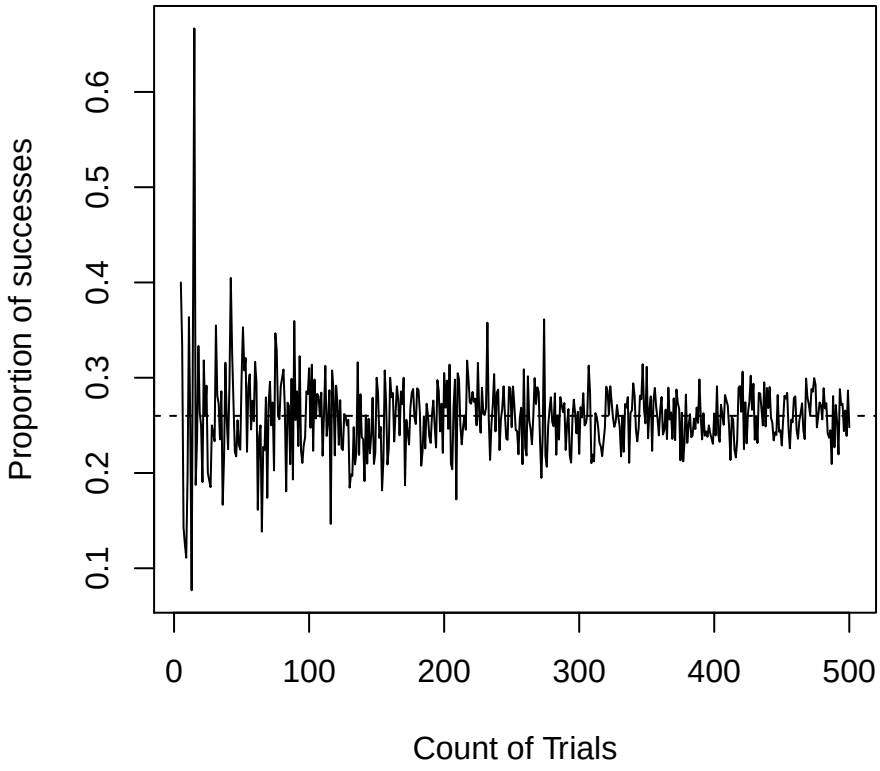
```
 [1] 1 1 0 1 1 0 1 1 1 1
```

Simulation of LLN

```
p=0.26; max.n=500 # p(1) and max sample size
n=5:max.n # n increasing from sample size 5 to 500
phat=numeric(length(n))
for(i in 1:length(n)){
  phat[i]=mean(outcomes(n[i],p))
}
```

Simulation of LLN plot

```
plot(n,phat,type='l',xlab='Count of Trials',ylab='Proportion of successes')
abline(h=p,lty=2)
```



LLN

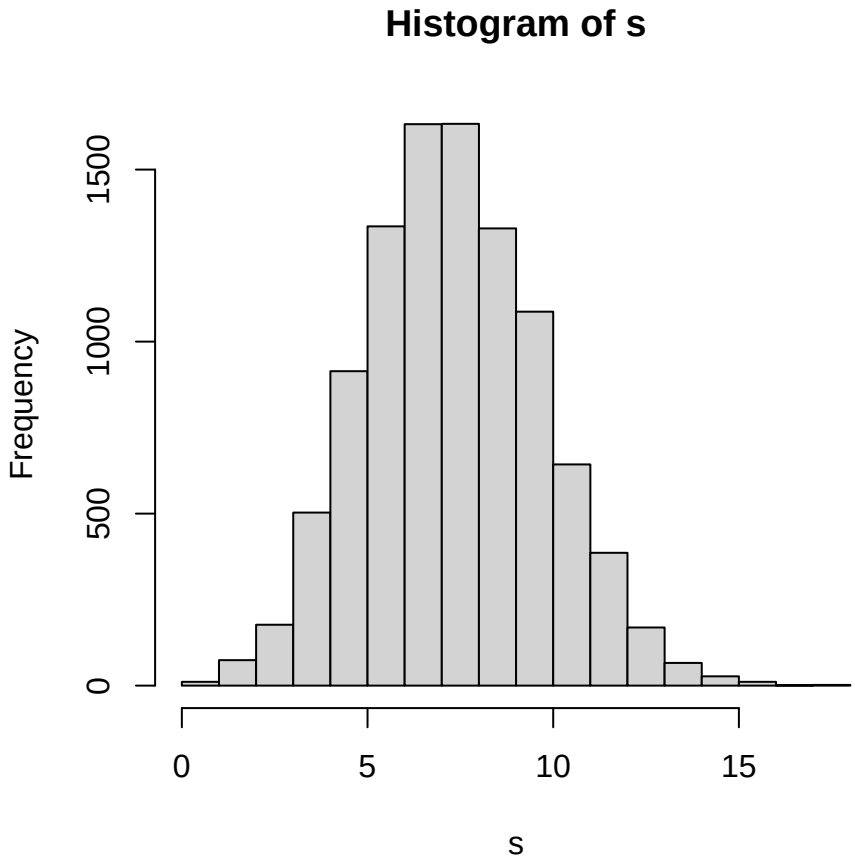
You can see in the graph, as we take more and more samples, the variation is reducing and the probability will converge to a single number (the theoretical probability).

In practice, an event probability might never actually be known but rather estimated from many trials.

Probability distribution of counts

Rather than an average now, keep track of the number of successes. Simulate samples of size 30 for 10,000 iterations.

```
n=30; p=0.26; max.n=10000 # p(1) and max sample size
s=numeric(max.n)
for(i in 1:max.n){
  s[i]=sum(outcomes(n,p))
}
hist(s)
```



Probability distribution of counts

The collection of the possible outcomes $(0, 1, \ldots, 30)$ of the random variable (successes out of n trials) along with the probabilities of the outcomes is called the probability distribution of a sample.

Binomial distribution

Basically what was just simulated (the sum of the outcomes) is a type of probability distribution called the *binomial distribution*. A random process that produces a binomial distribution has the following properties (1) There are a fixed number of trials n, with only two possible outcomes (success (S) or failure (F)) (2) For

each trial, the probability of success p is the same (3) Trials are independent of each other (the outcome on any one trial has no effect on the outcome of any other trial) (4) The random variable is the count of y successes from n trials.

Binomial distribution function

The probability of y successes from n trials is

$$P(Y = y) = \binom{n}{y} p^y (1-p)^{n-y}$$

$$\binom{n}{y} = \frac{n!}{y!(n-y)!}$$

The mean ($E(Y)$), variance ($V(Y)$), and standard deviation ($SD(Y)$):

$$E(Y) = np \qquad V(Y) = np(1-p) \qquad SD(Y) = \sqrt{V(Y)}$$

R functions for binomial distribution

`dbinom(x,n,p,...)`$= P(Y = y)$ *or* $P(X = x)$
`pbinom(x,n,p,...)`$= P(Y \leq y)$ (less than or equal to argument; a range)

`x`: vector of quantiles (argument)
`n`: sample size
`p`: probability of success
`...`: other options

$P(Y > y) = 1 - binom(y)$ and $P(Y \geq y) = 1 - P(Y < y)$

Binomial example I

```
p=.26; n=30
y=0:30
py=dbinom(y,n,p); py
```

```
 [1] 1.193855e-04 1.258388e-03 6.410975e-03 2.102338e-02 4.985950e-02
 [6] 9.109465e-02 1.333593e-01 1.606490e-01 1.622772e-01 1.393732e-01
[11] 1.028348e-01 6.569302e-02 3.654544e-02 1.777886e-02 7.585191e-03
[16] 2.842738e-03 9.363749e-04 2.709384e-04 6.875163e-05 1.525641e-05
[21] 2.948198e-06 4.932634e-07 7.089904e-08 8.664513e-09 8.879174e-10
[26] 7.487304e-11 5.058989e-12 2.633308e-13 9.913030e-15 2.402039e-16
[31] 2.813199e-18
```

```
data.frame(y,py)
```

```
    y           py
1   0 1.193855e-04
2   1 1.258388e-03
3   2 6.410975e-03
4   3 2.102338e-02
5   4 4.985950e-02
6   5 9.109465e-02
7   6 1.333593e-01
8   7 1.606490e-01
```

```
9   8 1.622772e-01
10  9 1.393732e-01
11 10 1.028348e-01
12 11 6.569302e-02
13 12 3.654544e-02
14 13 1.777886e-02
15 14 7.585191e-03
16 15 2.842738e-03
17 16 9.363749e-04
18 17 2.709384e-04
19 18 6.875163e-05
20 19 1.525641e-05
21 20 2.948198e-06
22 21 4.932634e-07
23 22 7.089904e-08
24 23 8.664513e-09
25 24 8.879174e-10
26 25 7.487304e-11
27 26 5.058989e-12
28 27 2.633308e-13
29 28 9.913030e-15
30 29 2.402039e-16
31 30 2.813199e-18
```

Binomial example II

```
pbinom(y,n,p)
```

```
 [1] 0.0001193855 0.0013777732 0.0077887486 0.0288121271 0.0786716262
 [6] 0.1697662787 0.3031255671 0.4637745942 0.6260518208 0.7654250545
[11] 0.8682598999 0.9339529216 0.9704983638 0.9882772275 0.9958624185
[16] 0.9987051567 0.9996415316 0.9999124700 0.9999812216 0.9999964780
[21] 0.9999994262 0.9999999195 0.9999999904 0.9999999990 0.9999999999
[26] 1.0000000000 1.0000000000 1.0000000000 1.0000000000 1.0000000000
[31] 1.0000000000
```

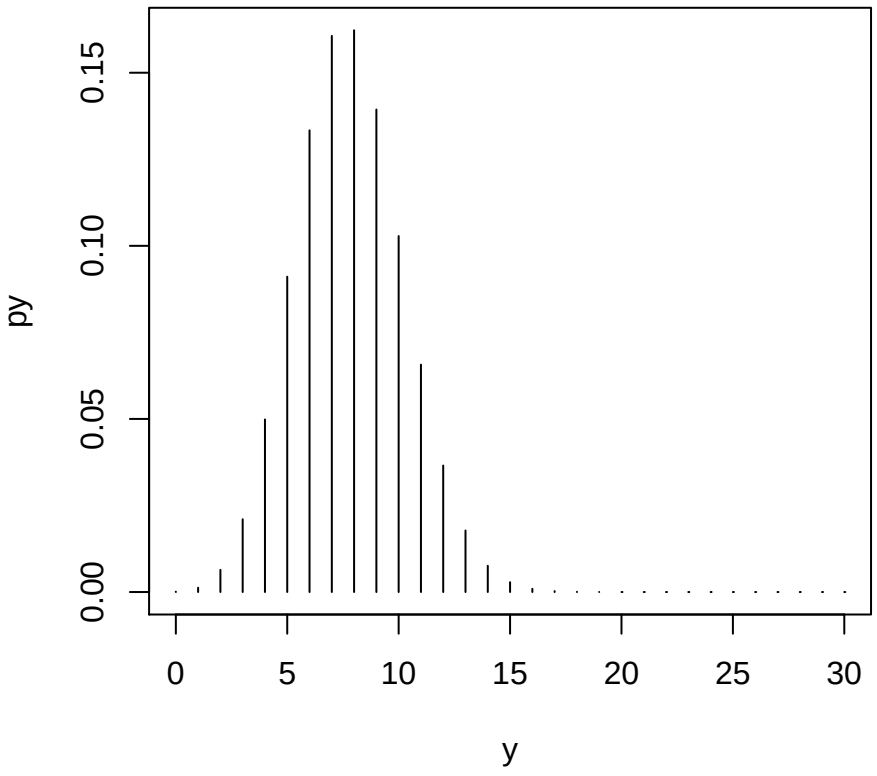
Binomial example III

```
rbinom(20,n,p) # generates 20 random values from binom with n=30,p=.26
```

```
 [1]  5  6  6  9  6  6  4 12  7  6  9 10  8  7  6 11 11 10  9 12
```

Binomial example IV

```
py=dbinom(y,n,p)
plot(y,py,type='h')
```



Probability distributions of measurements

Life is discrete; every measurement device we have is limited to a certain degree of accuracy, like to the second decimal place. The set of measurements that can be output by the device is a discrete set, meaning we can count the members of the set. Even the set of numbers that can be stored in the floating point system on a computer is a discrete set (granted, it is a *large* set).

Discrete vs. continuous

Members of the set of real numbers in a interval, say between 0 and 1, cannot be "counted" like the members of a discrete set. Real numbers are a mathematical abstraction beyond our everyday experience; there are numerous circumstances in science in which modeling the numerical output of some phenomenon as a set of real numbers offers great calculation convenience at little expense of realism.

In probability, if the range of output of a random process is "fine-grained" with many possible values, we can often achieve great simplification by using a *continuous probability distribution* to model the process. A random variable with a continuous probability distribution takes a real values within a range (interval) of possible values.

Uniform distribution

When a random variable has a continuous probability distribution, we assign probabilities to intervals instead of individual numbers. In earlier modules, we have used the *uniform distribution*. The random variable U takes a real value between 0 and 1 (in computers U takes on a value from any of the possible 16-decimal place numbers between 0 and 1). The probability that U takes on a value between a and b is $b - a$ where $0 \leq a \leq b \leq 1$, that is

$$P(a \leq U \leq b) = b - a$$

`runif()`

`runif(n)` where `n` is the desired number of randomly sampled values. This function will allow a convenient way to do many kinds of probability simulations and randomizations, as used in previous modules with random generators.

Normal distribution

For probabilities for area to the left, use `pnorm()`; for area to the right use `1-pnorm()`; for area between two values (a<b) use `pnorm(b)-pnorm(a)`.

$P(Y = y) = dne$ because Y is continuous

`pnorm(x,mean=0,sd=1,...)`
`x`: (y) argument; quantity or vector of quantities
`mean`: (mu) default is 0
`sd`: (sigma) default is 1 (with mean=0 gives standard normal => Z, as in $Z \sim N(\mu = 0, \sigma = 1)$)

$$Y \sim N(\mu, \sigma)$$

With $z = \frac{y-\mu}{\sigma}$

Normal example

```
# Y~N(500,100)
mu=500; sd=100
pnorm(600,mu,sd) # P(Y<600)
```

```
[1] 0.8413447
```

```
1-pnorm(600,mu,sd) # P(Y>600)
```

```
[1] 0.1586553
```

```
pnorm(700,mu,sd)-pnorm(400,mu,sd) # P(400<Y<700)
```

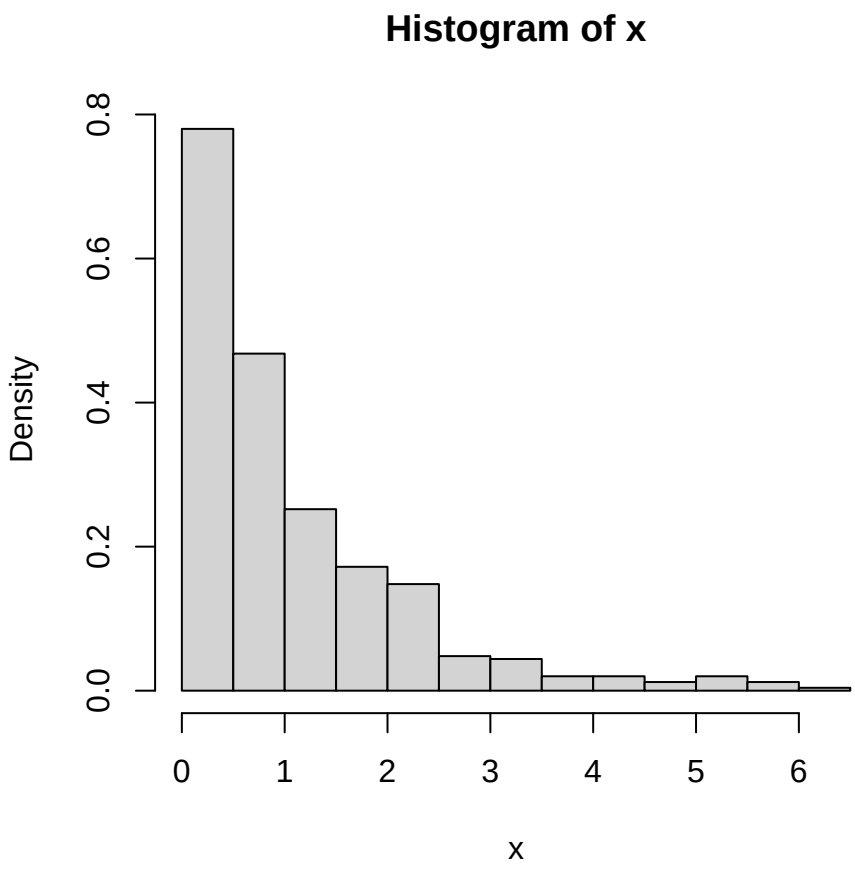
```
[1] 0.8185946
```

Central Limit Theorem (CLT) redux

A simulation for the CLT for the sampling distribution of the sample mean using the normal distribution was shown in Module 6. Using other distributions was hinted at for the example, and here are examples of CLT with the exponential and binomial distributions.

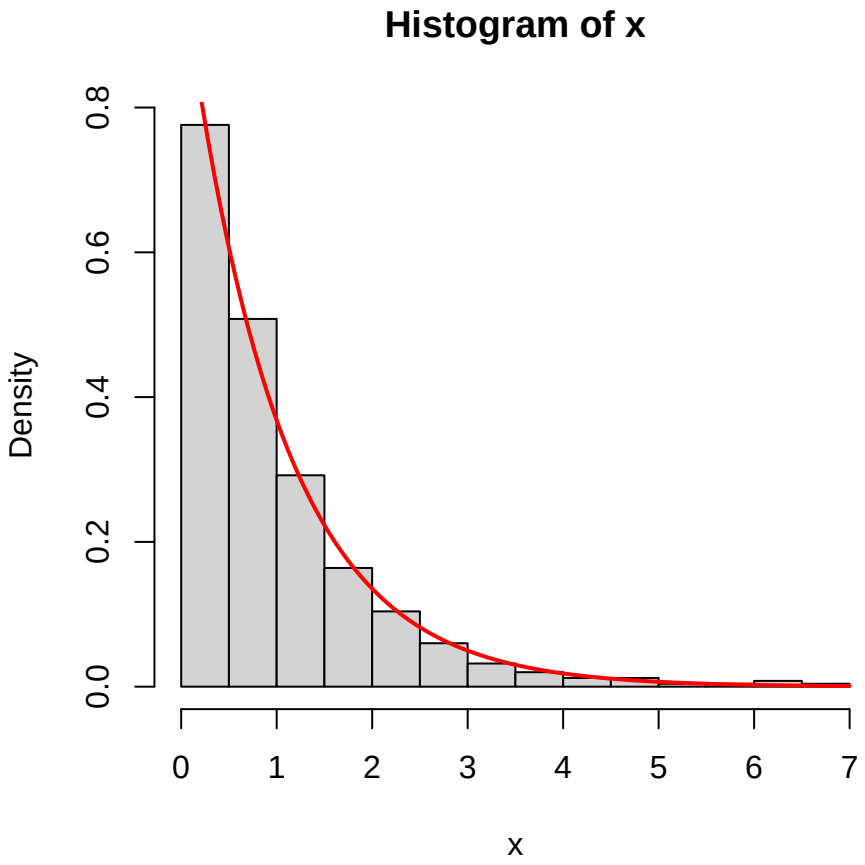
CLT exponential

```
x=rexp(500)
hist(x,prob=TRUE)
```



CLT exponential

```
x=rexp(500);
hist(x,prob=TRUE)
curve(dexp(x),add=TRUE,lwd=2,col="red")
```



CLT exponential

```
mean(x)
```

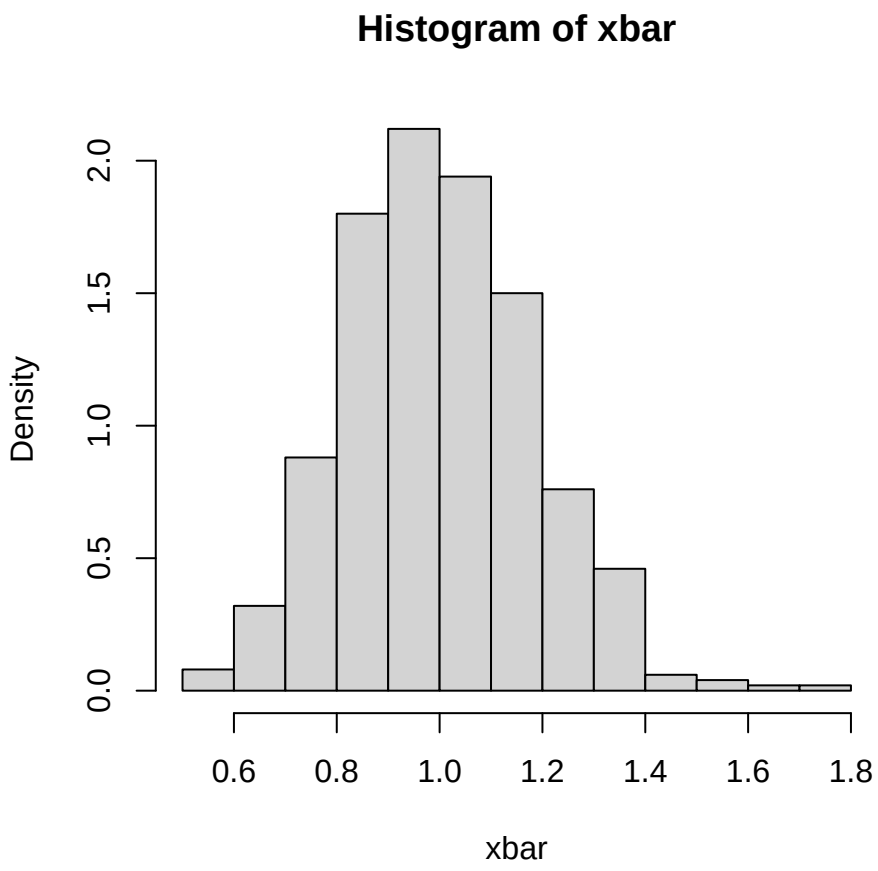
```
[1] 1.007934
```

```
x=rexp(500)
mean(x)
```

```
[1] 1.009407
```

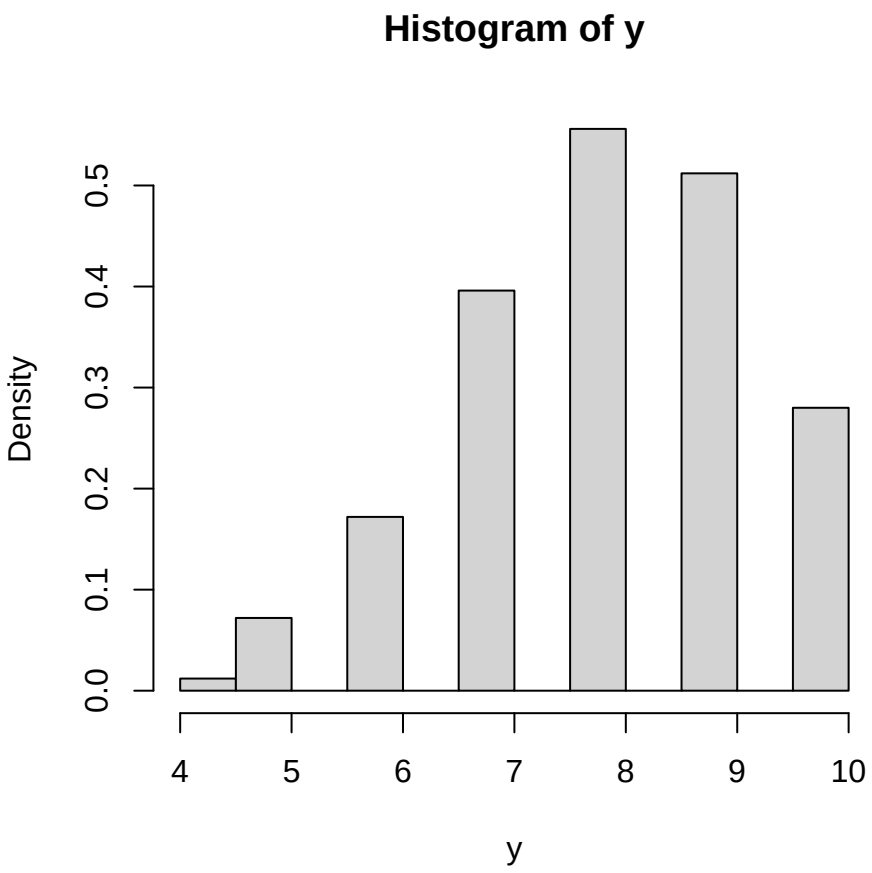
CLT exponential

```
mu=1; sigma=1; n=30
xbar=rep(0,500)
for (i in 1:500)
{ xbar[i]=mean(rexp(n)) }
hist(xbar,prob=TRUE,breaks=12)
```



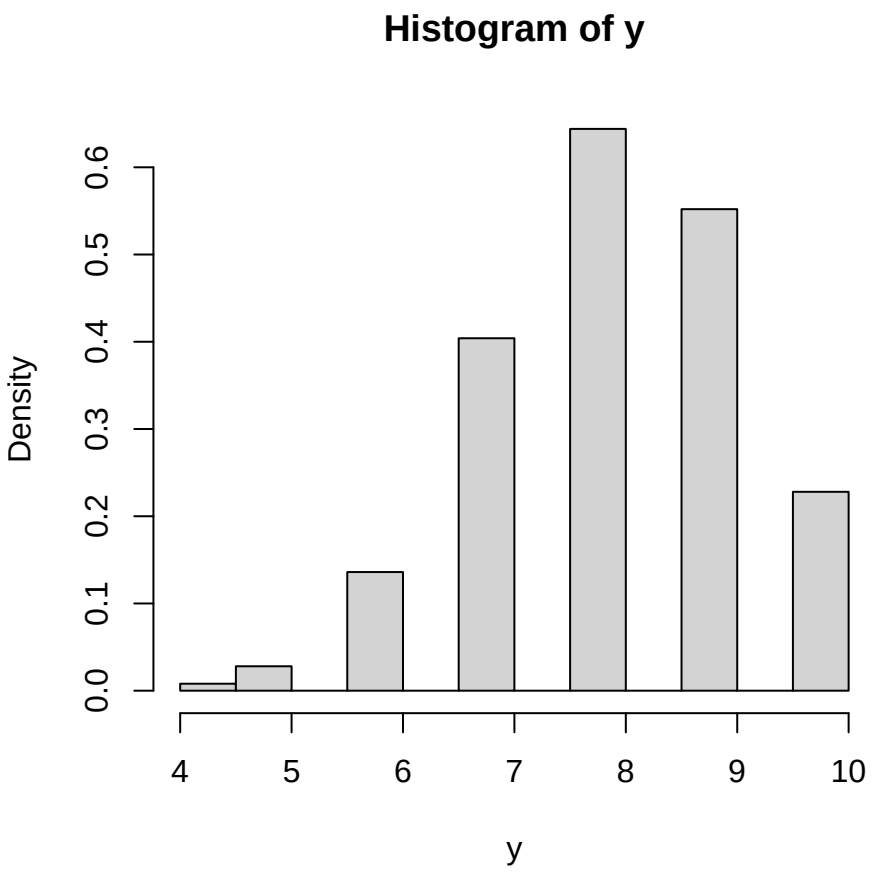
CLT binomial

```
# start with n=10, p=.8, 500 iterations
y=rbinom(500,10,.8)
hist(y,prob=T)
```



CLT binomial

```
y=rbinom(500,10,.8)
hist(y,prob=T)
```



CLT binomial

```
mean(y)
```

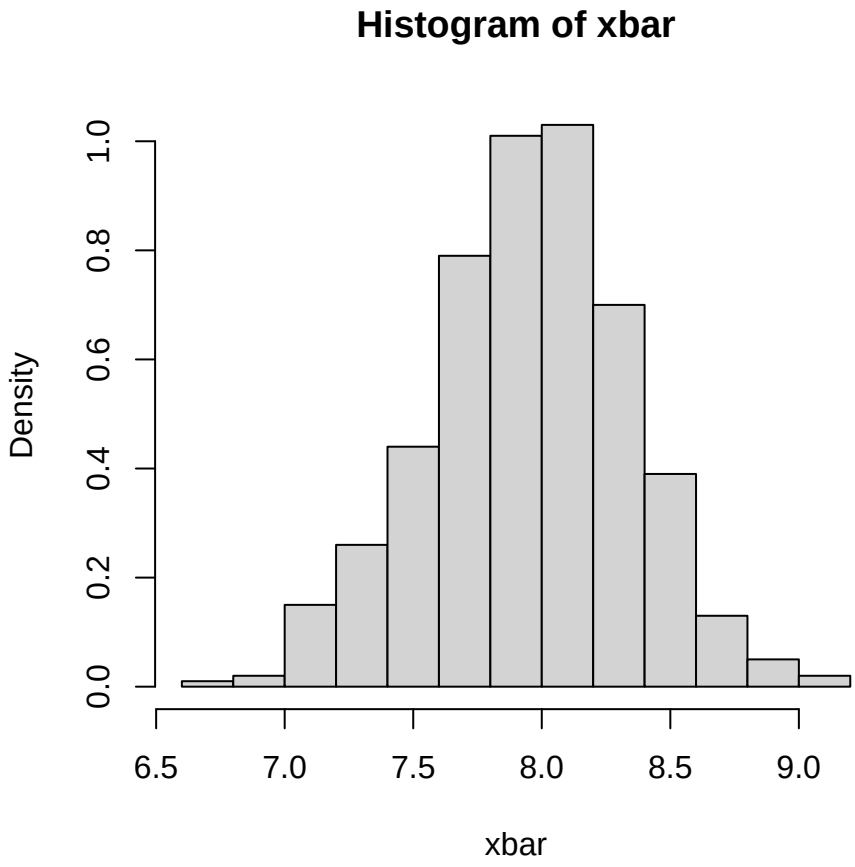
```
[1] 8.108
```

```
y=rbinom(500,10,.8)
mean(y)
```

```
[1] 7.978
```

CLT binomial

```
mu=8; sigma=1.26; n=10
xbar=rep(0,500)
for (i in 1:500)
{ xbar[i]=mean(rbinom(n,10,.8)) }
hist(xbar,prob=TRUE,breaks=15)
```


Histogram of xbar
Density
0.0
0.2
0.4
0.6
0.8
1.0
6.5
7.0
7.5
8.0
8.5
9.0
xbar