# sampling

## Why Monte-Carlo simulation?

Let's simulate 1000 samples from Binomial(N=10,p=0.4) and compute sample mean and variance using those samples. Compare them with theoretical mean and variance.

```
set.seed(1)
N = 10
p = 0.4
x = rbinom(1000, N, p)
mean(x)
```

```
## [1] 4.02
```

```
N*p  # theoretical mean
```

```
## [1] 4
```

```
var(x)
```

```
## [1] 2.414014
```

```
N*p*(1-p) # theoretical variance
```

```
## [1] 2.4
```

We can easily estimate mean and variance of Y = X^(2.4) + X + log(X+1) and P(Y > 10) when X ~ Binomial(N=10,p=0.4).

```
x = rbinom(1000, N, p)
y = x^(2.4) + x + log(x+1)
mean(y)
```

```
## [1] 39.83337
```

```
var(y)
```

```
## [1] 992.5155
```

```
sum(y > 10)/length(y)
```

```
## [1] 0.803
```

## Simulating other (finite) discrete rv using sample function.

```
x = sample(c(0,1,2), 1000000, replace = TRUE, prob = c(0.5, 0.3, 0.2))
sum(x==0)/length(x)
```

```
## [1] 0.499563
```

```
sum(x==1)/length(x)
```

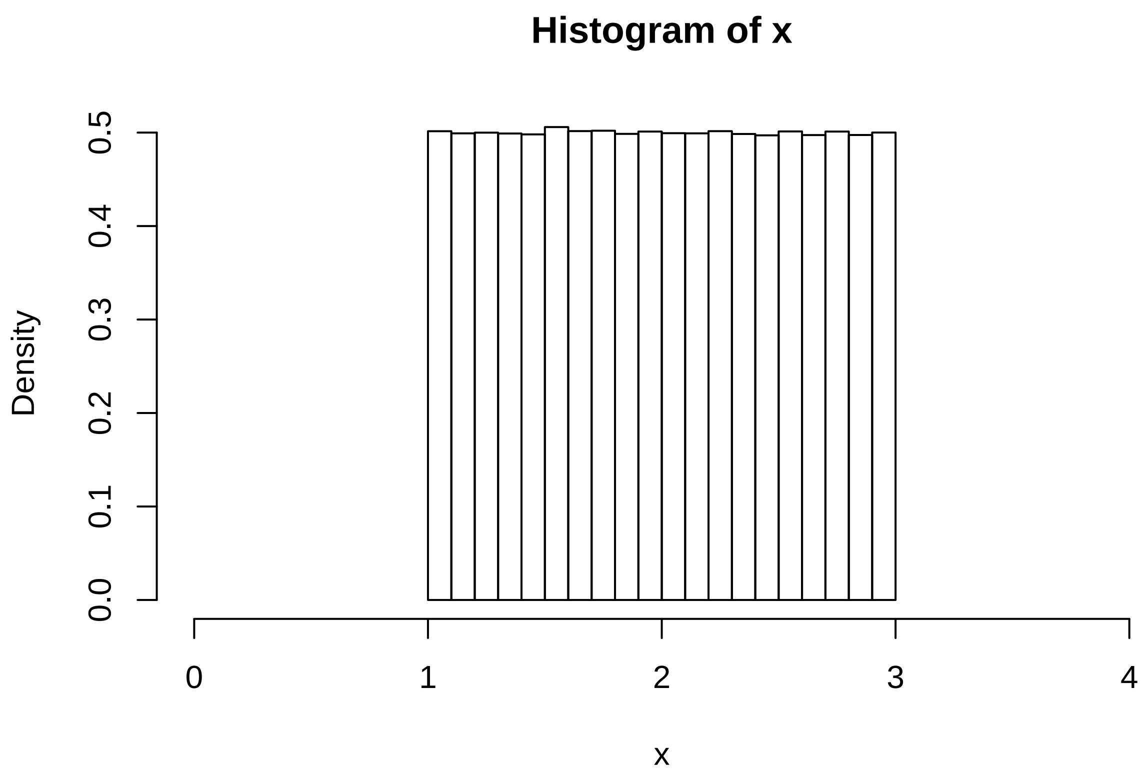
```
## [1] 0.300597
```

```
sum(x==2)/length(x)
```

```
## [1] 0.19984
```

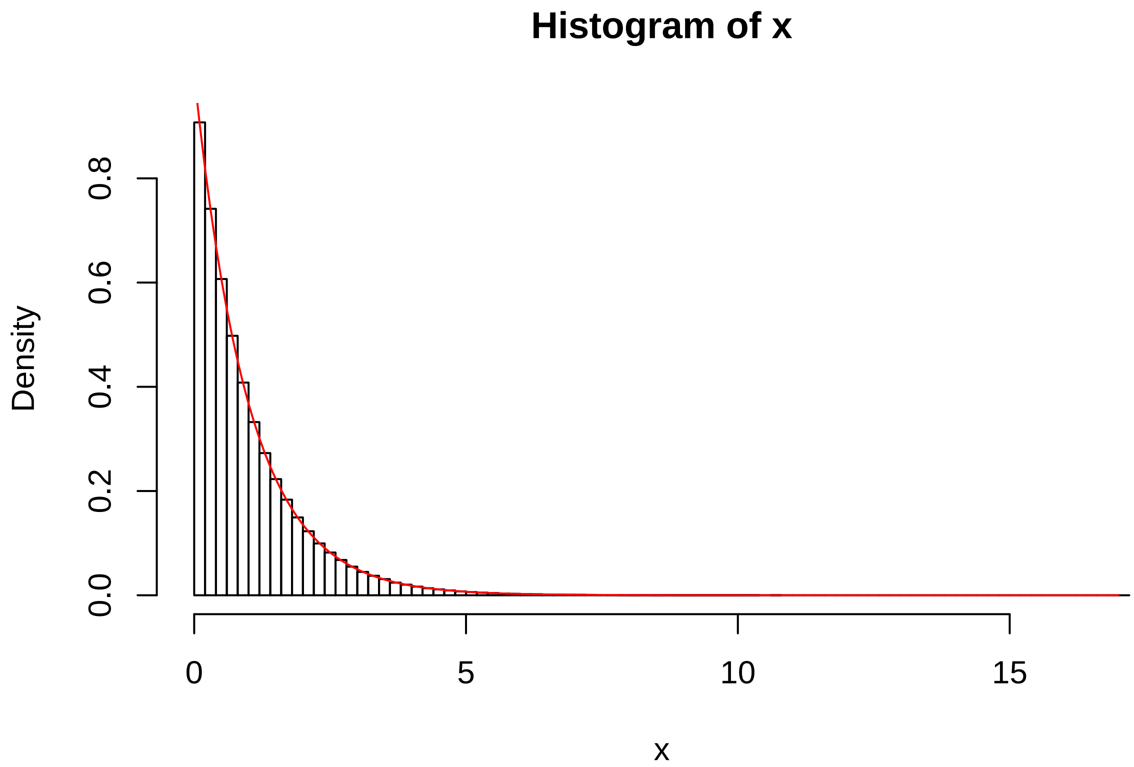
## Simulating X ~ U(1,3)

```
u = runif(1000000, 0, 1)
x = 2*u + 1
hist(x,breaks=20,xlim=c(0,4), freq = FALSE)
```




## Simulating X ~ exp(1)

```
lambda = 1
u = runif(1000000, 0, 1)
x = -1/lambda*log(1-u)
hist(x,breaks=100,freq = FALSE)
x <- seq(0, max(x), .1)
lines(x, dexp(x, lambda), col="red")
```


Histogram of x
Density
0.0
0.2
0.4
0.6
0.8
0
5
10
15
x