Roll a fair dice. Record the result.

X ~ Uniform({1,2,...,6})


```
>> X = randi(6)

X =

     5
```

Again, roll a fair dice. Record the result.

```
>> X = randi(6)

X =

     6
```

Again, roll a fair dice. Record the result.

```
>> X = randi(6)

X =

     1
```

Again, roll a fair dice. Record the result.

```
>> X = randi(6)

X =

     6
```

Again, roll a fair dice. Record the result.

```
>> X = randi(6)

X =

     4
```

Again, roll a fair dice. Record the result.

```
>> X = randi(6)

X =

     1
```

```
>> X = randi(6,20,10)

X =

     2     5     3     4     5     2     1     2     6     3
     4     3     4     1     5     4     3     1     6     4
     6     4     5     1     3     5     2     2     4     5
     6     2     5     2     4     5     5     2     1     1
     1     5     2     6     1     3     3     3     2     6
     6     1     5     2     1     1     6     1     3     5
     6     2     4     5     4     2     2     6     5     3
     3     1     1     2     5     6     2     6     1     3
     5     1     1     6     6     1     1     3     1     3
     1     5     3     3     1     5     1     3     2     2
     3     5     6     2     4     4     6     3     4     4
     6     2     3     2     3     6     4     6     5     4
     5     6     4     4     1     1     4     3     4     5
     6     1     2     3     3     3     1     1     3     5
     4     3     5     3     1     1     6     5     4     4
     1     3     2     5     5     6     4     3     2     3
     6     5     4     4     2     1     3     2     5     5
     6     5     5     4     4     5     4     3     2     4
     5     2     6     6     1     5     3     1     5     3
     5     3     6     2     4     6     1     1     2     6
```

Generate X 200 times. Put the results in a table of size 20×10

`randi` function

- Generate uniformly distributed pseudorandom **integers**
- `randi(imax)` returns a scalar value between 1 and `imax`.
- `randi(imax,m,n)` and `randi(imax,[m,n])` return an *m*-by-*n* matrix containing pseudorandom integer values drawn from the discrete uniform distribution on the interval [1,`imax`].
  - `randi(imax)` is the same as `randi(imax,1)`.
- `randi([imin,imax],...)` returns an array containing integer values drawn from the discrete uniform distribution on the interval [`imin`,`imax`].

`hist` function

- Create histogram plot
- `hist(data)` creates a histogram bar plot of data.
  - Elements in data are sorted into **10 equally spaced bins** along the x-axis **between the minimum and maximum** values of data.
  - Bins are displayed as rectangles such that the height of each rectangle indicates the number of elements in the bin.
  - If data is a vector, then one histogram is created.
  - If data is a matrix, then a histogram is created separately for each column.
    - Each histogram plot is displayed on the same figure with a different color.
- `hist(data,nbins)` sorts data into the number of bins specified by `nbins`.
- `hist(data,xcenters)`
  - The values in `xcenters` **specify the centers** for each bin on the x-axis.

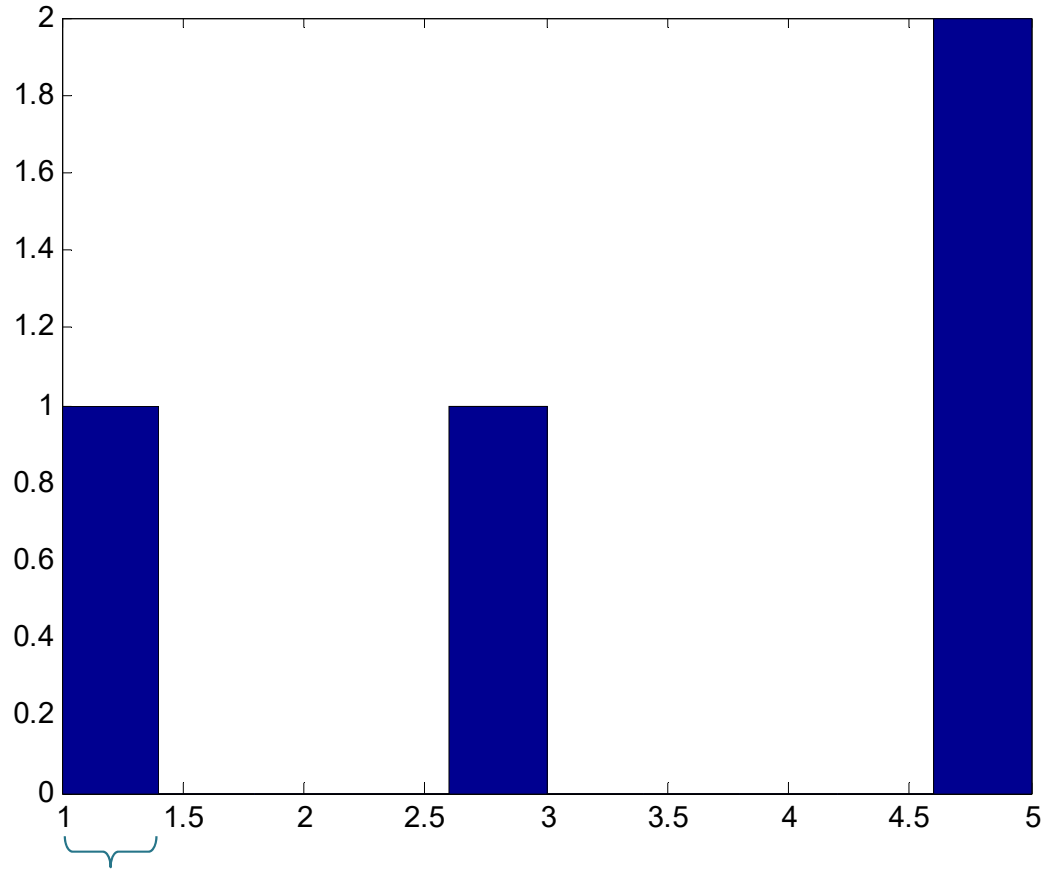
`hist` function: Example

```
>> x = [1 3 5 5]

x =

     1     3     5     5

>> hist(x)
```



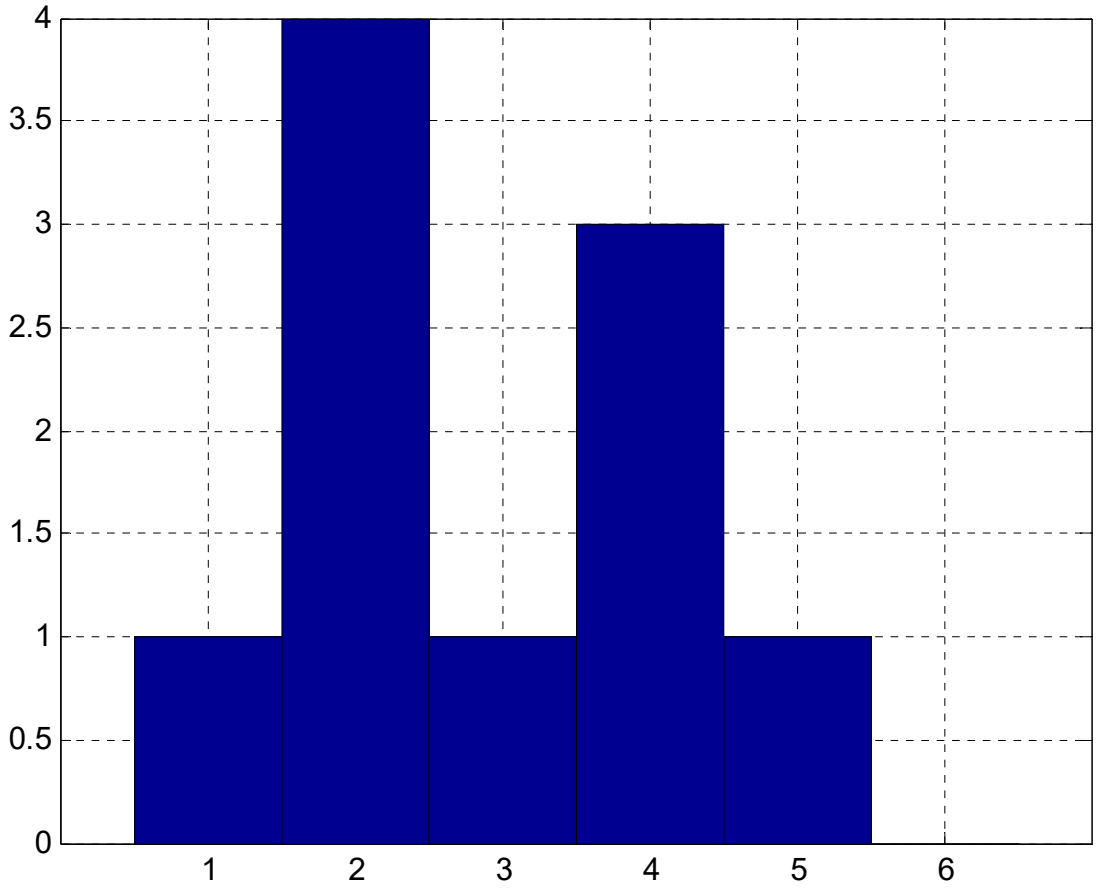
min

max

The width of each bin is $\frac{5-1}{10} = 0.4$

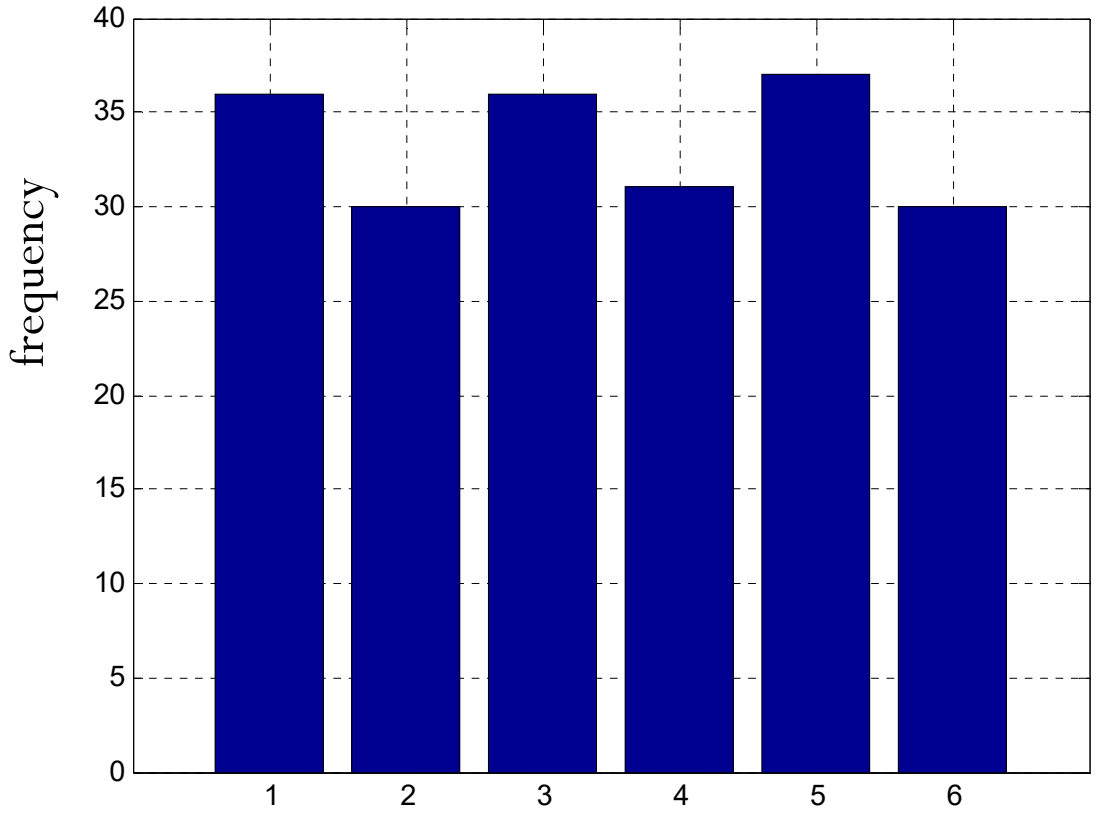
`hist` function: Example

```
>> X = randi(6,1,10)
X =
     4     2     4     5     2     1     2     2     3     4
>> hist(reshape(X,1,prod(size(X))),1:6)
>> grid on
```



X ~ Uniform({1,2,...,6})

```
X = randi(6,20,10)
```

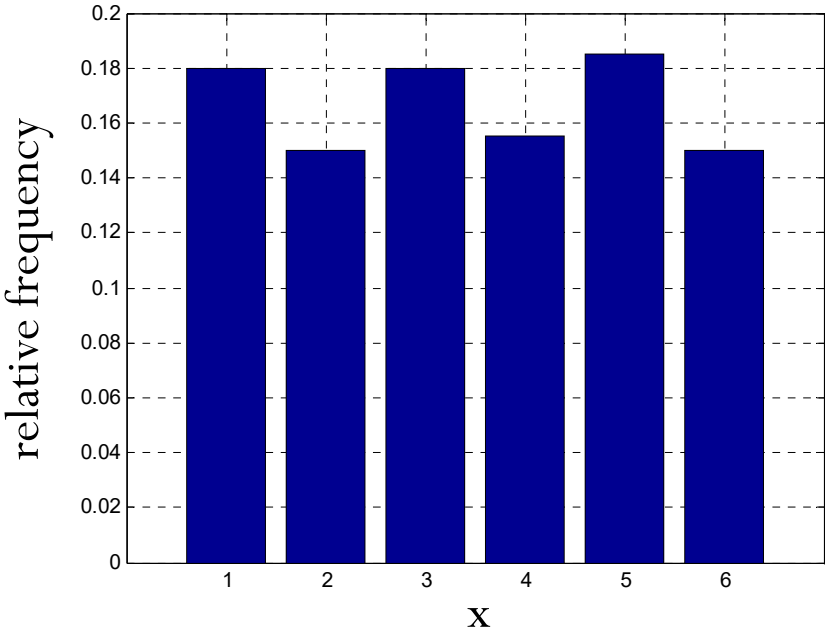



```
[N, x] = hist(reshape(X,1,prod(size(X))),1:6)
bar(x,N)
Grid on
```

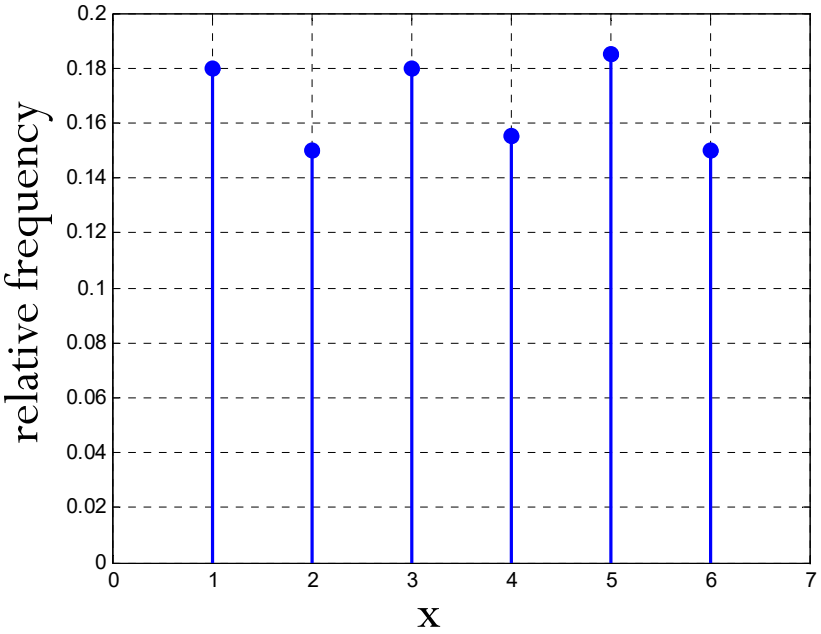
The `hist` command is used here for counting.

The "histogram" itself is drawn using the `bar` command.

Relative Frequency



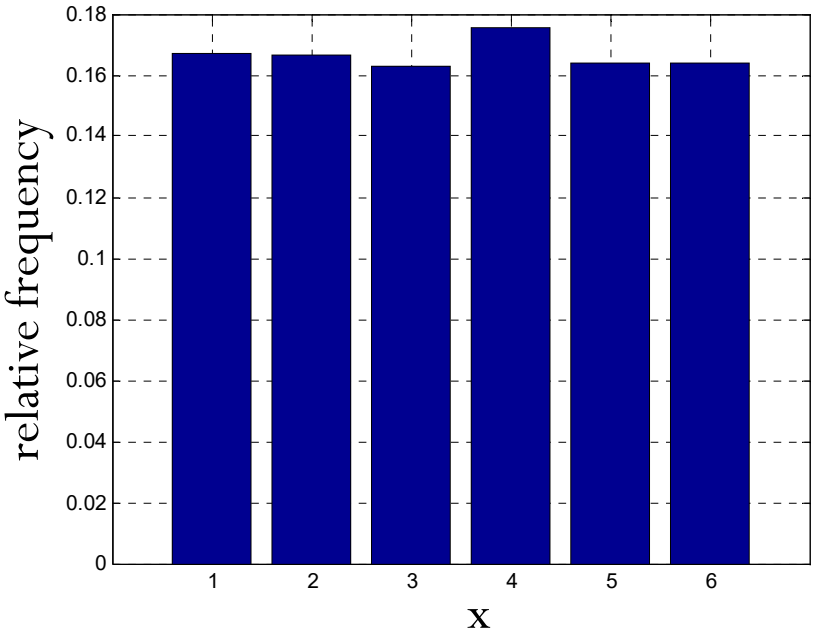

```
rf = N/prod(size(X))
bar(x,rf)
grid on
```



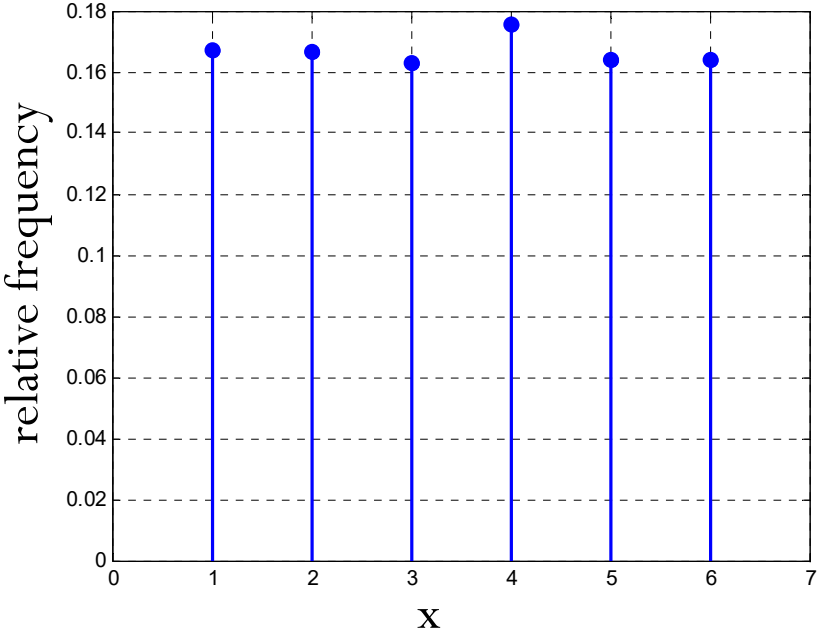

```
stem(x,rf,'filled','LineWidth',1.5)
grid on
```

```
X = randi(6,20,10)
```

With larger number of samples




```
rf = N/prod(size(X))
bar(x,rf)
grid on
```




```
stem(x,rf,'filled','LineWidth',1.5)
grid on
```

```
X = randi(6,100,100);
```

20-Sided Dice

Dice in Dungeons & Dragons

- A fantasy tabletop role-playing game (RPG)
- First published in 1974
- Widely regarded as the beginning of modern role-playing games and the role-playing game industry

D&D uses polyhedral dice to resolve random events. These are abbreviated by a 'd' followed by the number of sides. Shown counter-clockwise from the bottom are: d4, d6, d8, d10, d12 and d20 dice.

DUNGEONS & DRAGONS

http://en.wikipedia.org/wiki/File:Dice_(typical_role_playing_game_dice).jpg
http://en.wikipedia.org/wiki/Dungeons_%26_Dragons

D20 Bowl Set

[http://geekologie.com/2014/09/dungeons-dragons-approved-d20-serving-bo.php]

Flip an unfair coin 10 times. (The probability of getting heads for each time is 0.3.)
Count the number of heads.

n p

X ~ binomial(10,0.3)

```
>> X = binornd(10,0.3)

X =

     3
```

Again, flip an unfair coin 10 times. Count #H.

```
>> X = binornd(10,0.3)

X =

     2
```

Again, flip an unfair coin 10 times. Count #H.

```
>> X = binornd(10,0.3)

X =

     2
```

Again, flip an unfair coin 10 times. Count #H.

```
>> X = binornd(10,0.3)

X =

     5
```

Again, flip an unfair coin 10 times. Count #H.

```
>> X = binornd(10,0.3)

X =

     1
```

Again, flip an unfair coin 10 times. Count #H.

```
>> X = binornd(10,0.3)

X =

     4
```

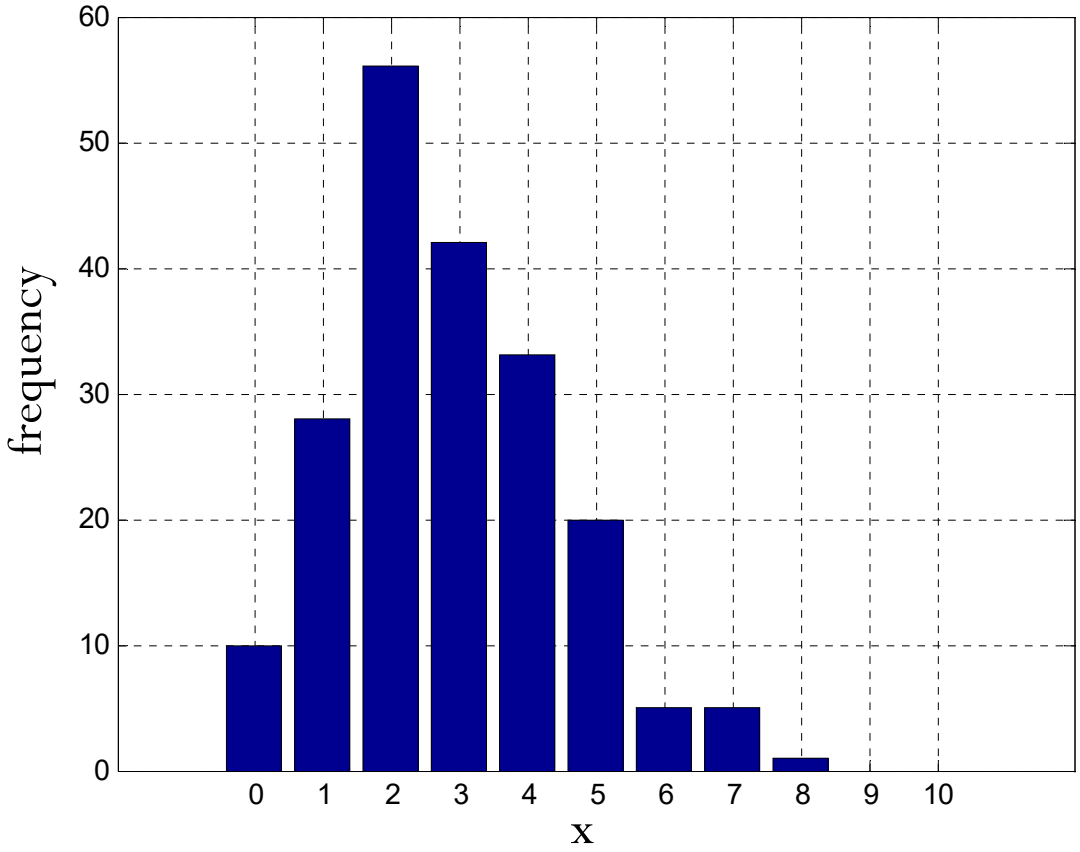
```
>> X = binornd(10,0.3,20,10)

X =

     3     4     4     5     7     2     2     2     2     1
     3     5     3     1     0     4     2     1     2     3
     5     2     2     6     4     2     2     4     3     1
     1     2     2     4     2     4     3     3     3     5
     4     1     4     3     3     4     2     2     2     2
     2     1     3     1     5     2     5     2     1     2
     4     0     3     3     2     1     2     1     3     1
     4     4     0     2     3     6     2     3     1     1
     5     0     3     3     7     1     3     1     3     8
     1     2     4     4     1     5     2     4     5     1
     5     2     4     6     3     2     3     3     5     0
     2     4     0     0     2     2     3     2     0     2
     4     3     3     2     2     2     1     2     7     4
     2     4     2     1     3     3     4     3     5     2
     5     3     2     3     4     2     3     3     1     2
     2     6     2     3     4     4     4     5     6     7
     5     1     2     4     3     3     0     5     0     2
     1     4     1     3     1     4     2     4     2     4
     5     2     2     3     3     5     3     5     2     1
     4     2     4     3     2     5     7     2     3     1
```

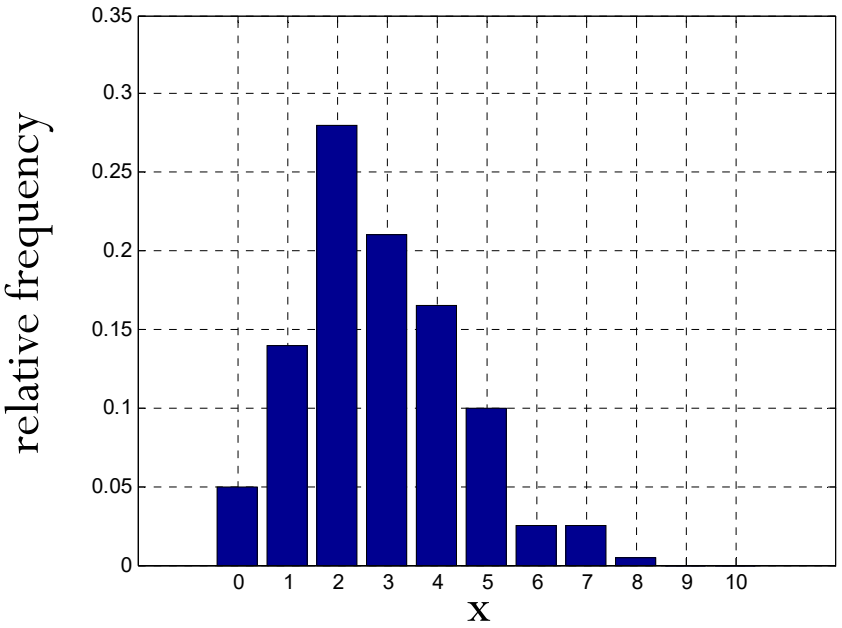
Generate X 200 times. Put the results in a table of size 20×10

Histogram: X ~ binomial(10,0.3)

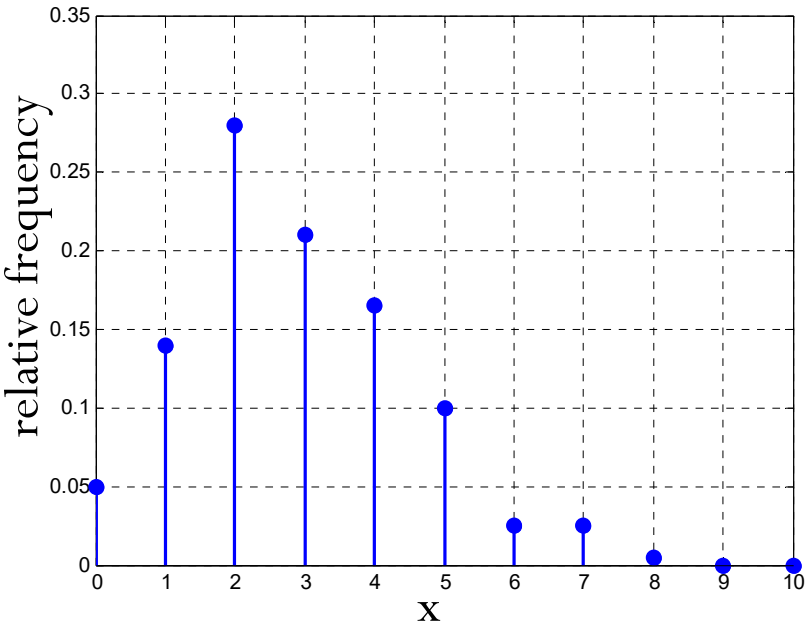



```
[N, x] = hist(reshape(X,1,prod(size(X))),0:10)
bar(x,N)
Grid on
```

Relative Freq.: X ~ binomial(10,0.3)



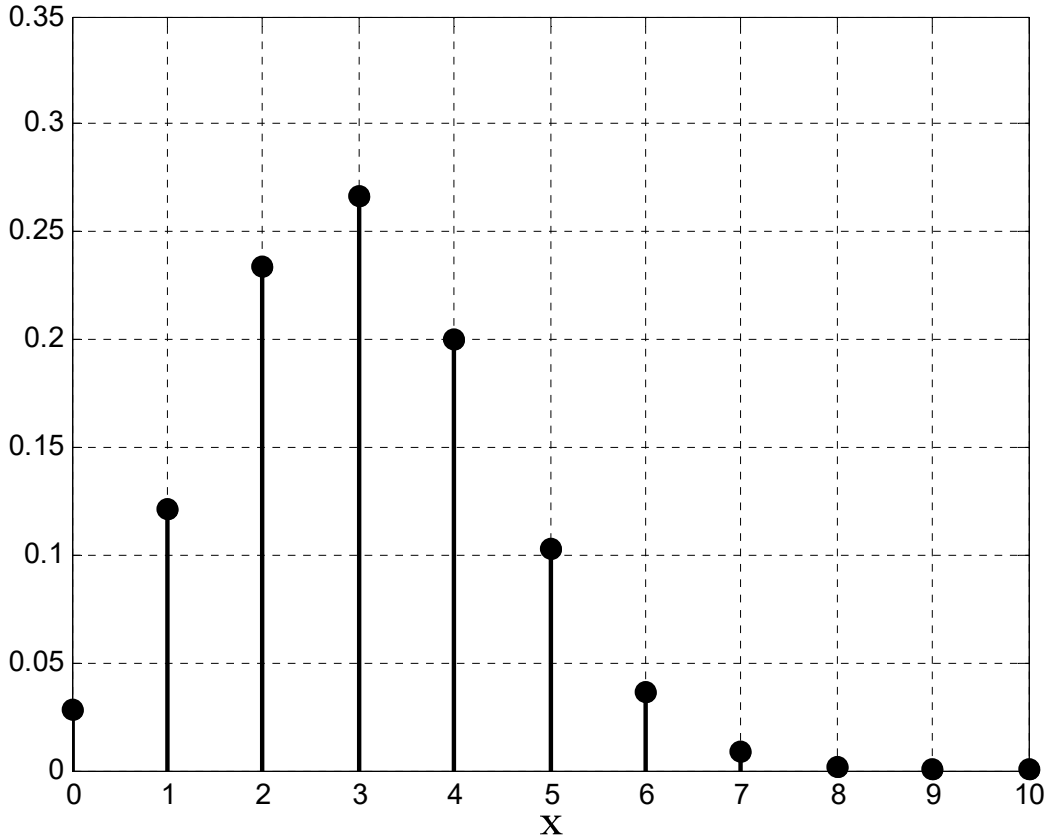

```
rf = N/prod(size(X))
bar(x,rf)
grid on
```




```
stem(x,rf,'filled','LineWidth',1.5)
grid on
```

pmf for X ~ binomial(10,0.3)

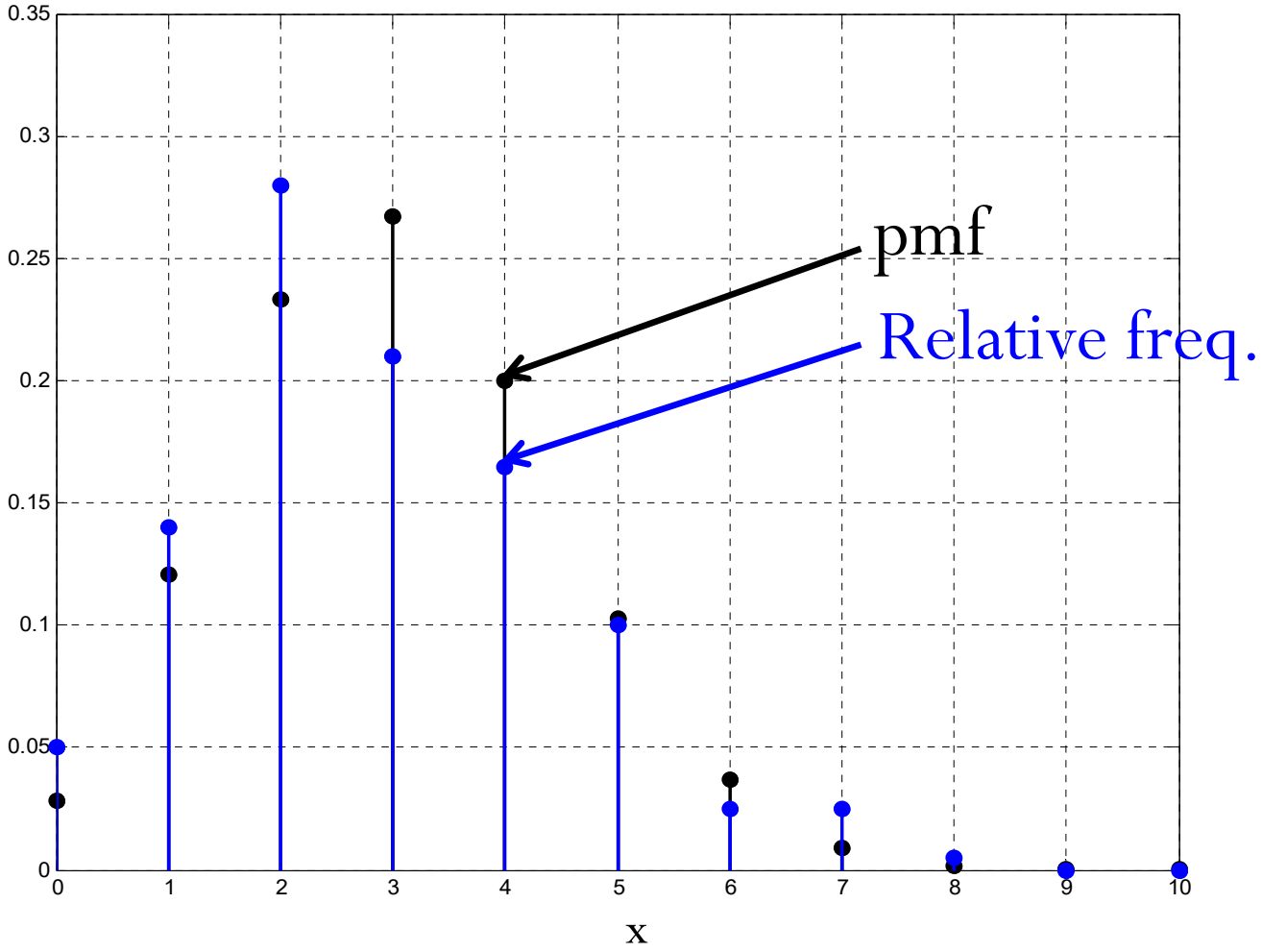
$$p_X(x) = \binom{10}{x} 0.3^x (1-0.3)^{10-x}$$




```
p = binopdf(x,10,0.3)
stem(x,p,'k','filled','LineWidth',1.5); grid on
```

X ~ binomial(10,0.3)

```
X = binornd(10,0.3,20,10);
```



X ~ binomial(10,0.3)

```
X = binornd(10,0.3,100,100);
```