Class-2

Jenna G. Tichon

23/09/2019

Generative Models for Discrete Data

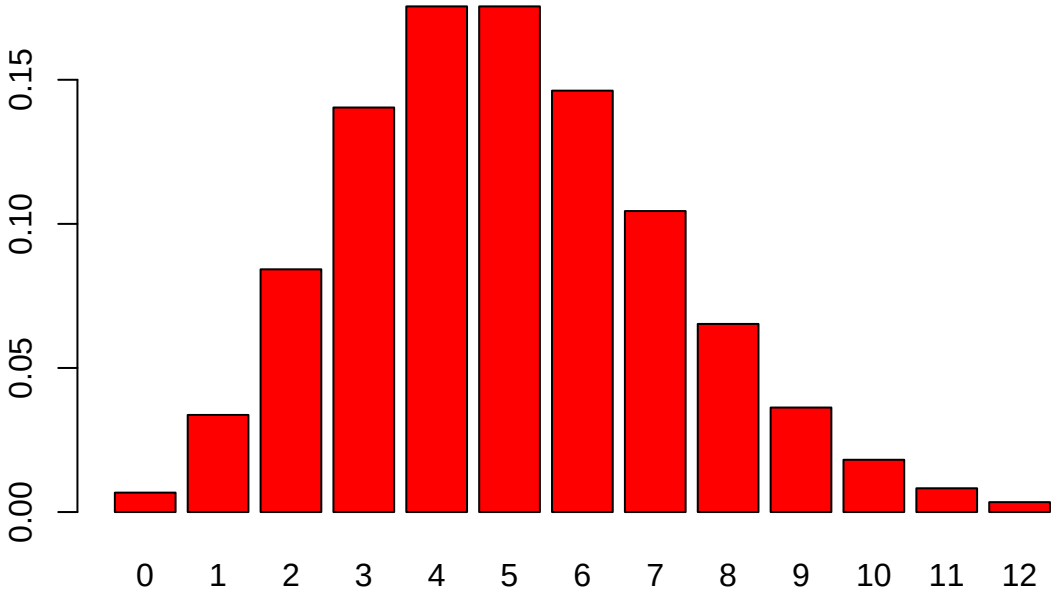
1.1 Goals for this chapter

1.2 A Real Example

```
dpois(x=0:12, lambda=5)
```

```
##  [1] 0.006737947 0.033689735 0.084224337 0.140373896 0.175467370
##  [6] 0.175467370 0.146222808 0.104444863 0.065278039 0.036265577
## [11] 0.018132789 0.008242177 0.003434240
```

```
barplot(dpois(0:12,5), names.arg = 0:12, col = "red")
```



1.3 Using discrete probability models

> *__Bio tip__: __Exchangeable__: The order in which the data observed doesn't matter.

```
genotype = c("AA", "AO", "BB", "AO", "OO", "AO", "AA", "BO", "BO",
             "AO", "BB", "AO", "BO", "AB", "OO", "AB", "BB", "AO", "AO")
table(genotype)
```

```
## genotype
## AA AB AO BB BO OO
##  2  2  7  3  3  2
```

```r
#Access factor levels
genotypeF = factor(genotype)
levels(genotypeF)
```

```
## [1] "AA" "AB" "AO" "BB" "BO" "OO"
```

```r
table(genotypeF)
```

```
## genotypeF
## AA AB AO BB BO OO
##  2  2  7  3  3  2
```

1.3.1 Bernoulli Trials

```r
rbinom(15, prob= 0.5, size = 1)
```

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

```r
rbinom(12, prob = 2/3, size = 1)
```

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

```r
rbinom(1, prob = 2/3, size = 12)
```

```
## [1] 6
```

```r
set.seed(235569515)
rbinom(1, prob = 0.3, size = 15)
```

```
## [1] 5
```

1.3.2 Binomial success counts

Q 1.3

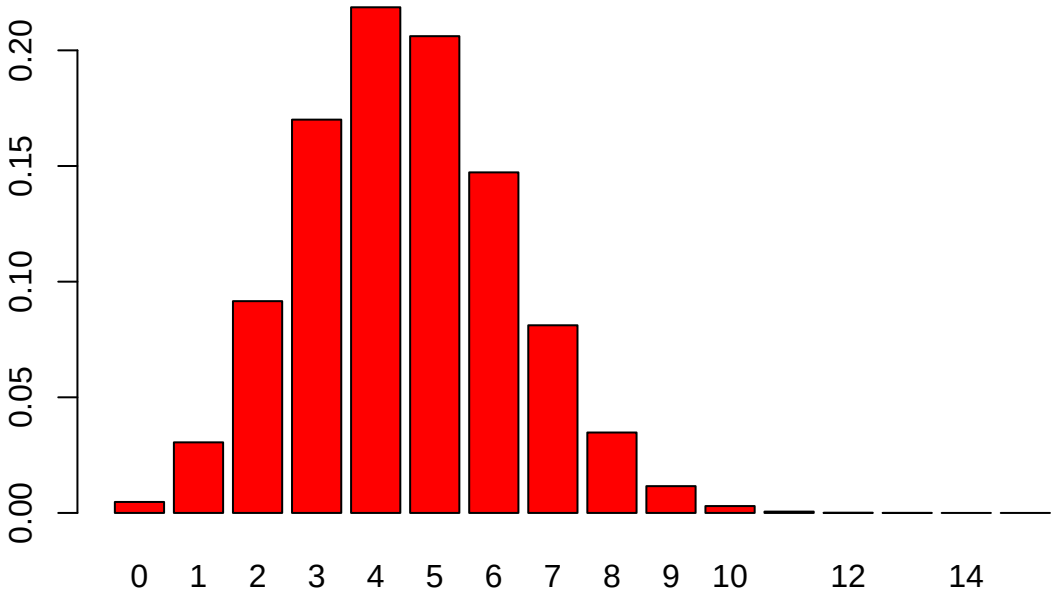
```r
for(j in 1:10)
{
  print(rbinom(1, prob = 0.3, size = 15))
}
```

```
## [1] 5
## [1] 2
## [1] 2
## [1] 4
## [1] 5
## [1] 3
## [1] 5
## [1] 7
## [1] 2
## [1] 5
```

```r
probabilities = dbinom(0:15, prob = 0.3, size = 15)
round(probabilities, 2)
```

```
##  [1] 0.00 0.03 0.09 0.17 0.22 0.21 0.15 0.08 0.03 0.01 0.00 0.00 0.00 0.00
## [15] 0.00 0.00
```

```r
barplot(probabilities, names.arg = 0:15, col ="red")
```



Q 1.4

```r
dbinom(3, size = 4, prob = 2/3)
```

```
## [1] 0.3950617
```

```r
choose(4,3)*(2/3)^3*(1-2/3)^(4-3)
```

```
## [1] 0.3950617
```

1.3.3 Poisson distributions

Q5

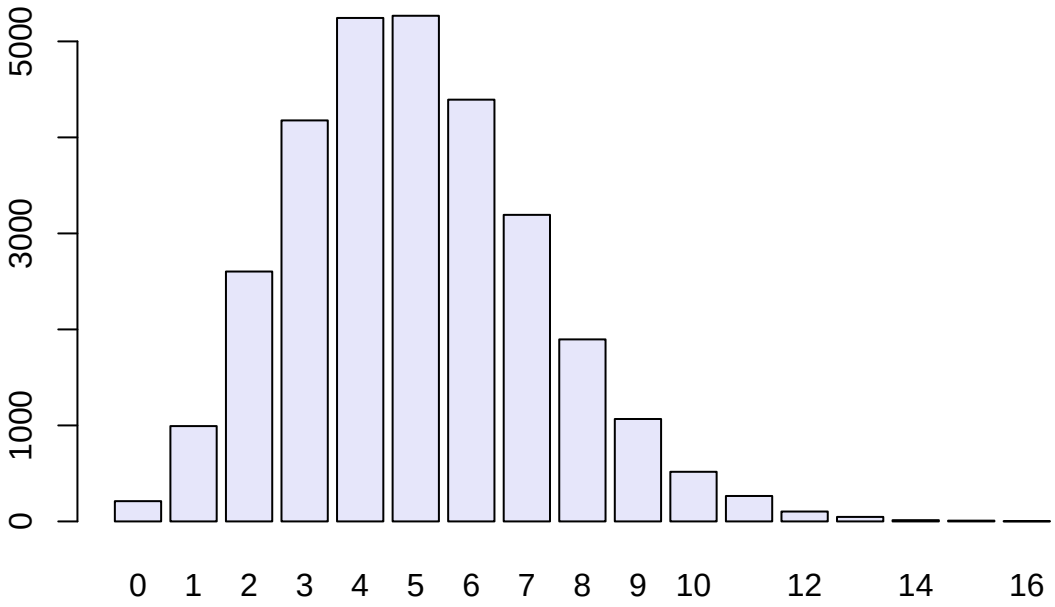
```r
dpois(3,5)
```

```
## [1] 0.1403739
```

```r
5^3*exp(-5)/factorial(3)
```

```
## [1] 0.1403739
```

```r
simulations = rbinom(n = 30000, prob = 5e-4, size = 10000)
barplot(table(simulations), col = "lavender")
```



1.3.4 A generative model for epitope detection

*__Bio tip__: An **antibody** is a protein made by certain white blood cells in response to a foreign substance in the body, which is called the **antigen**. An antibody binds to its antigen to destroy the antigen. So destroy antigens directly and others recruit white blood cells to destroy the antigen. An **epitope**, also known as antigenic determinant, is the part of an antigen that is recognized by the immune system, specifically by anitobodies, B celss or T cells.

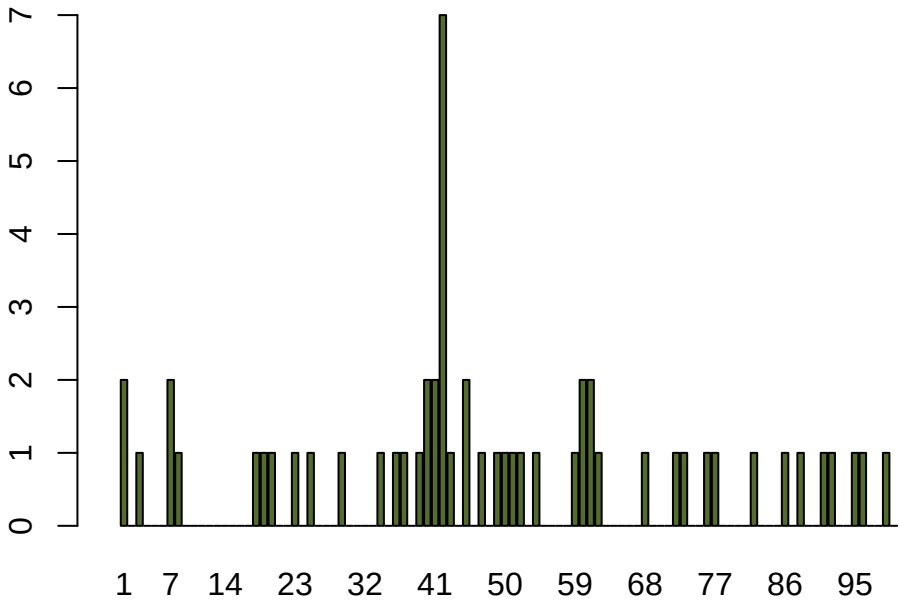
False positive rate: detect an epitope when none is present. Assume 1%. So if no epitopes presents, probability is 1 in 100 of getting a 1.

Test 100 positions on 50 patient samples. Take position i. Sum positions i across all 50 samples. This follows a poisson distribution with mean 0.5 (we expect 1 in 100 so we expect 0.5 in 50).

This is a plot of the 100 positions:

```
load(here("data", "e100.RData"))

barplot(e100, ylim = c(0, 7), width = 0.7, xlim = c(-0.5, 100.5),
        names.arg = seq(along = e100), col = "darkolivegreen")
```



There is a big spike with seeing 7 hits when no epitopes are present (e.g. 7 false positives).

If independent, the probability of 7 or more for this particular position has probability:

```
1-ppois(6,0.5)
```

```
## [1] 1.00238e-06
```

```
ppois(6, 0.5, lower.tail=FALSE)
```

```
## [1] 1.00238e-06
```

But! We were looking at 100 positions. If we repeat this sample 100 times, what's the probability the maximum is 7 or larger?

$$P(max >= 7) = 1 - P(\text{all less than } 7)) = 1 - P(x \leq 6)^{100}$$

```
1-ppois(6,0.5)^100
```

```
## [1] 0.000100233
```

It goes from about 1 in 1,000,000 to about 1 in 10,000. (i.e. a factor of 100)

> **ALERT!!**: This doesn't replicate the text but don't forget, this is simulated without a set seed, that's why!

```
#Code for simulating this probability
maxes = replicate(100000, {
  max(rpois(100,0.5))
})
table(maxes)
```

```
## maxes
##     1     2     3     4     5     6     7     9
##     8 23102 60779 14362  1595   138    15     1
```

```
mean( maxes >= 7 )
```

```
## [1] 0.00016
```

*__R tip__: The `mean` function works this way because we created a logical vector of TRUE/FALSEs.

1.4 Multinomial distributions: the case of DNA

One sample with four categories from a multinomial distribution with probabilies 1/8, 3/8, 3/8, 1/8.

```
rmultinom(1, size = 4, prob=c((1/8),(3/8),(3/8),(1/8)))
```

```
##      [,1]
## [1,]    2
## [2,]    1
## [3,]    1
## [4,]    0
```

q1.7

The answer to the actual Q1.7 for finding the probability

```
pvec = rep(1/4, 4)
dmultinom(c(4,2,0,0), size=6, prob = pvec)
```

```
## [1] 0.003662109
```

Step one for building this through simulation: generate one sample

```
t(rmultinom(1, prob = pvec, size = 8))
```

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

q1.8

t() stands for transpose. Removing it and the vector displays vertically.

```
rmultinom(1, prob = pvec, size = 8)
```

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

q1.9

This generates 8 samples of size 1.

```
rmultinom(n = 8, prob = pvec, size = 1)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    1    1    0    1    0    1    0    0
## [2,]    0    0    0    0    0    0    1    1
## [3,]    0    0    0    0    1    0    0    0
## [4,]    0    0    1    0    0    0    0    0
```

This generates one sample of size 8.

```
rmultinom(n = 1, prob = pvec, size = 8)
```

```
##      [,1]
## [1,]    4
## [2,]    0
## [3,]    2
## [4,]    2
```

1.4.1 Simulating for power

Power (True Positive Rate): Probability of detecting something that is there. (Aim for 80%+ power). $P(\text{reject } H_0 | H_A)$

If we find the extreme 5% tail for the distribution under H_0 (uniform distribution) using simulation, we can estimate the rejection region. It's upper 5% because this is a chi-squared distribution we are estimating for a test statistic.

```
#Randomly generated under null hypothesis, 1000 observations where all 4 categories are equally likely.
obsunder0 = rmultinom(1000, prob = pvec, size = 20)
dim(obsunder0)
```

```
## [1]    4 1000
```

```
#Print first 11 observations to show. Remember, displayed vertically.
obsunder0[, 1:11]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## [1,]    2    5    5    6    4    3    3    3    6     6     3
## [2,]    5    4    4    2    9    4    6    6    4     4     6
## [3,]    7    6    6    9    4    4    6    7    7     5     3
## [4,]    6    5    5    3    3    9    5    4    3     5     8
```

Now we find the expected counts under uniform to calculate our test statistic.

```
#Expected counts if all equally likely
expected0 = pvec * 20
```

Now we will create a function to calculate the deviations under the chi-squared test statistic:

$$\sum_i \frac{(E_i - x_i)^2}{E_i}$$

```
#Function to create test stat - chi-squared
stat = function(obsvd, exptd = 20 * pvec) {
  sum((obsvd - exptd)^2 / exptd)
}
```

We can use this function to what the test statistic would have been for the first sample:

```
#Give sum contribution of first observation
stat(obsunder0[, 1])
```

```
## [1] 2.8
```

Using this, we will find what the test statistic would have been for each of our 1000 samples. This will allow us to decide what is considered an extreme deviation under H_0 empirically.

```
#Apply the test stat to each column element of obsunder0
#apply's second option is 1 or 2 for row or column
S0 = apply(obsunder0, 2, stat)
summary(S0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.200   2.400   2.982   4.000  22.000
```

*__R tip__: The apply(x,1 or 2, f) function says apply function f to each row (1) or each column (2) of x.

Now we use the quantile function to find the 95th percentile of these deviations under H_0

```
#Quantiles for distribution of deviations (contributions to test stat)
q95 = quantile(S0, probs = 0.95)
q95
```

```
## 95%
## 7.6
```

This means that we would reject H_0 if our test stat is greater than 7.6 (i.e. if the weighted mean of all deviations is larger than what would be an extreme deviation under H_0).

To determine the power of a test with a 3/8, 1/4, 3/12, 1/8 distribution, we find the probability that we would reject H_0 when we have an observed dataset that truly comes from this different distribution when we take 1000 samples of size 20.

```
pvecA = c(3/8, 1/4, 3/12, 1/8)
observed = rmultinom(1000, prob = pvecA, size = 20)
dim(observed)
```

```
## [1]    4 1000
```

We can find the average across each row to show that it is more or less in line with what we expect.

```
#display first seven, calculate the average counts across each row.
observed[, 1:7]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]   10    6    6    6    8    7    7
## [2,]    3    6    7    7    5    3    6
## [3,]    4    6    6    6    4    7    5
## [4,]    3    2    1    1    3    3    2
```

```
apply(observed, 1, mean)
```

```
## [1] 7.573 4.974 4.991 2.462
```

```
#calculate expected counts under new distribution
expectedA = pvecA * 20
expectedA
```

```
## [1] 7.5 5.0 5.0 2.5
```

Test calculate the deviation of the first sample, then find the test statistic for each sample.

```
stat(observed[, 1])
```

```
## [1] 6.8
```

```
S1 = apply(observed, 2, stat)
```

Knowing that the rejection region is all observed chi-squares greater than 7.6, count how samples would have led to a rejection of our test.

```
q95
```

```
## 95%
## 7.6
```

```
sum(S1 > q95)
```

```
## [1] 216
```

The power is the percentage of observations that fall in the rejection region.

```
power = mean(S1 > q95)
power
```

```
## [1] 0.216
```

> *__R tip__: Remember, when set up as S1>q95, we are taking the mean of a logical vector so it is averaging 0's and 1's which is the same as taking a percentage.

1.5 Summary of this chapter

1.6 Further Reading

1.7 Exercises

Exercise 1.1

Geometric `rgeom(n, p)`

Hypergeometric `rhyper(N,A,B,k)`

Exercise 1.2

```
dbinom(0,10,.3)+dbinom(1,10,.3)+dbinom(2,10,.3)
```

```
## [1] 0.3827828
```

```
pbinom(2,10,.3)
```

```
## [1] 0.3827828
```

Exercise 1.3

Here is my first overly complicated attempt at the problem but it does give us a function for calculating the probability the max is exactly equal to a number!

```
#Function to find probability the max of n poisson samples with mean lambda is
#equal to m
max.pois<-function(m,n,lambda)
{
  #Probability max is less than m (i.e. all n values less than m)
  prob.lessm<- ppois(m-1,lambda)^n
```

```
  #Probability max is less than m+1 (i.e. all n values less than m+1)
  prob.lessm1<- ppois(m,lambda)^n

  #Probability max is equal to m (P(max<m+1)-P(max<m))
  probm<-prob.lessm1-prob.lessm
}

asbigmax.pois<-function(m,n,lambda){
  #Find probability of all maxes less than m
  max.prob<-c(0:m-1)
  for(j in 0:m-1)
  {
    max.prob[j]<-max.pois(j,n,lambda)
  }

  #Find probability of max at least as large as m (i.e. 1 - prob less than m)
  prob<-1-sum(max.prob)
  prob
}
```

Test case for $m = 9$, $n = 20$, $\lambda = 4$

```
asbigmax.pois(9,20,4)
```

```
## [1] 0.3507249
```

And then I realized that was way too complicated and found a better function that matched what we had done earlier:

```
maxormore<-function(m,n,lambda)
{
  1-ppois(m-1,lambda)^n
}
```

Confirm the test cases agree:

```
maxormore(9,20,4)
```

```
## [1] 0.3507249
```

Exercise 1.4

Same deal with complicated version then simpler version. I used $m = 10$, $n = 20$, $\lambda = 5$ for my default values.

```
#Function to find probability the max of n poisson samples with mean lambda is
#equal to m
max.pois<-function(m,n,lambda)
{
  #Probability max is less than m (i.e. all n values less than m)
  prob.lessm<- ppois(m-1,lambda)^n

  #Probability max is less than m+1 (i.e. all n values less than m+1)
  prob.lessm1<- ppois(m,lambda)^n

  #Probability max is equal to m (P(max<m+1)-P(max<m))
  probm<-prob.lessm1-prob.lessm
}
```

```r
asbigmaxd.pois<-function(m=10,n=20,lambda=5){
  #Find probability of all maxes less than m
  max.prob<-c(0:m-1)
  for(j in 0:m-1)
  {
    max.prob[j]<-max.pois(j,n,lambda)
  }

  #Find probability of max at least as large as m (i.e. 1 - prob less than m)
  prob<-1-sum(max.prob)
  prob
}
```

Test the function:

```r
asbigmaxd.pois()
```

```
## [1] 0.4763395
```

Way simpler version with same defaults:

```r
maxormored<-function(m=10,n=20,lambda=5)
{
  1-ppois(m-1,lambda)^n
}
```

Show the test agrees:

```r
maxormored()
```

```
## [1] 0.4763395
```

Exercise 1.5

ALERT!!: Not sure what this question means. 100 trials mean 100 samples or 100 positions to test? What do they mean by "number of simulations to prove"? Leaving this for group discussion.

ALERT!!: Okay, what it meant was, that if you need to prove with a p-value of, say .01, then you need to show that your value is more extreme than 100 values. So you would need at least 100 observations.

We would need $\frac{1}{0.000001} = 1,000,000$ simulations.

ALERT!!: Take a peek at the replicate function in the "leader" text.

Exercise 1.6

```r
?Distributions
```

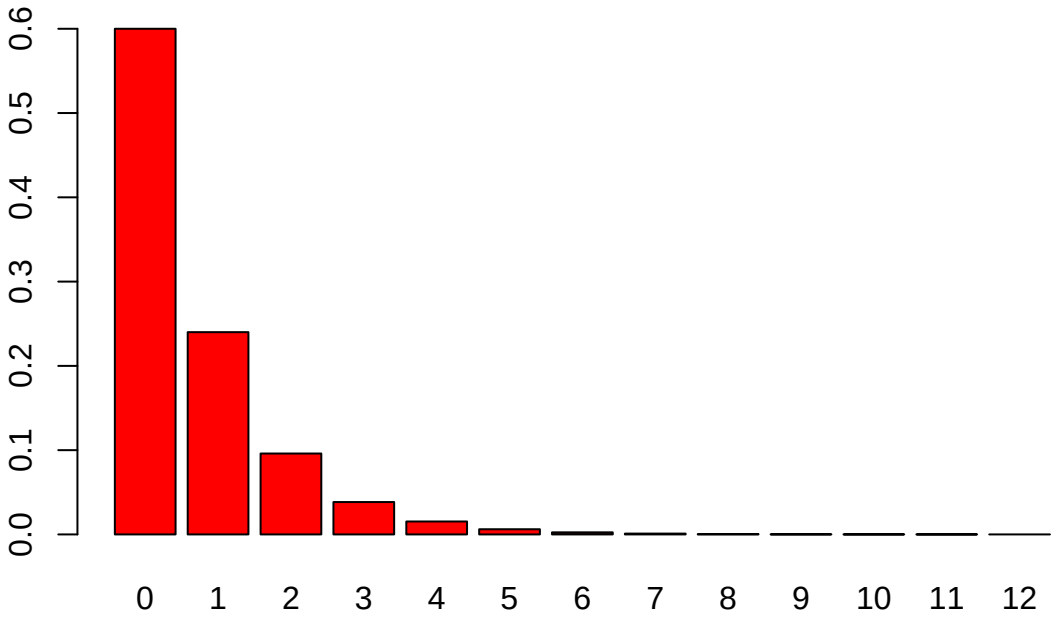
Some not discrete distributions: chi-squared, exponential, F, gamma, normal.

CDF of Geometric($p = 0.6$)

```r
dgeom(x=0:12, p=.6)
```

```
##  [1] 6.000000e-01 2.400000e-01 9.600000e-02 3.840000e-02 1.536000e-02
##  [6] 6.144000e-03 2.457600e-03 9.830400e-04 3.932160e-04 1.572864e-04
## [11] 6.291456e-05 2.516582e-05 1.006633e-05
```

```r
barplot(dgeom(0:12,p=0.6), names.arg = 0:12, col = "red")
```

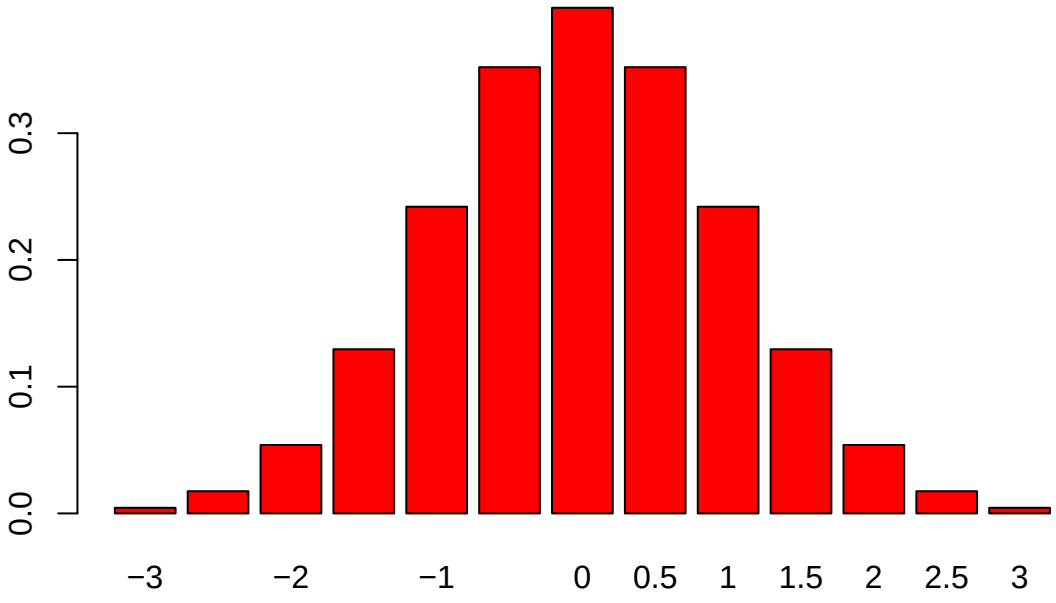


CDF of Standard Normal

```r
dnorm(x=seq(-3,3, by=0.1))
```

```
##  [1] 0.004431848 0.005952532 0.007915452 0.010420935 0.013582969
##  [6] 0.017528300 0.022394530 0.028327038 0.035474593 0.043983596
## [11] 0.053990967 0.065615815 0.078950158 0.094049077 0.110920835
## [16] 0.129517596 0.149727466 0.171368592 0.194186055 0.217852177
## [21] 0.241970725 0.266085250 0.289691553 0.312253933 0.333224603
## [26] 0.352065327 0.368270140 0.381387815 0.391042694 0.396952547
## [31] 0.398942280 0.396952547 0.391042694 0.381387815 0.368270140
## [36] 0.352065327 0.333224603 0.312253933 0.289691553 0.266085250
## [41] 0.241970725 0.217852177 0.194186055 0.171368592 0.149727466
## [46] 0.129517596 0.110920835 0.094049077 0.078950158 0.065615815
## [51] 0.053990967 0.043983596 0.035474593 0.028327038 0.022394530
## [56] 0.017528300 0.013582969 0.010420935 0.007915452 0.005952532
## [61] 0.004431848
```

```r
barplot(dnorm(x=seq(-3,3, by=0.5)), names.arg = seq(-3,3, by=0.5), col = "red")
```



Exercise 1.7

```
x.pois<-rpois(100,3)
mean(x.pois)
```

```
## [1] 3.08
```

```
var(x.pois)
```

```
## [1] 3.286465
```

Exercise 1.8

ALERT!!: No idea. Let's see what the biologists know about this package.