1 Probability Distributions

A probability distribution describes how the values of a random variable are distributed.

For example, the collection of all possible outcomes of a sequence of coin tossing is known to follow the binomial distribution, whereas the means of sufficiently large samples of a data population are known to resemble the normal distribution (Central limit Theorem). Since the characteristics of these theoretical distributions are well understood, they can be used to make statistical inferences on the entire data population as a whole.

Recall that probability distributions can be categorised as either ***discrete*** or ***continuous***. We will look at

- how to generate random numbers,
- how to use probability mass and density functions,
- how to use cumulative distribution functions,
- how to determine quantiles.

There is a wide variety of probability distributions available, but we shall only look at a few key distribution.

- The Normal Distribution
- The Student $t-$Distribution
- The Uniform Distribution
- The Exponential Distribution
- The Binomial Distribution
- The Poisson Distribution

If you would like to know what distributions are available you can do a search using the command `help.search("distribution")`.

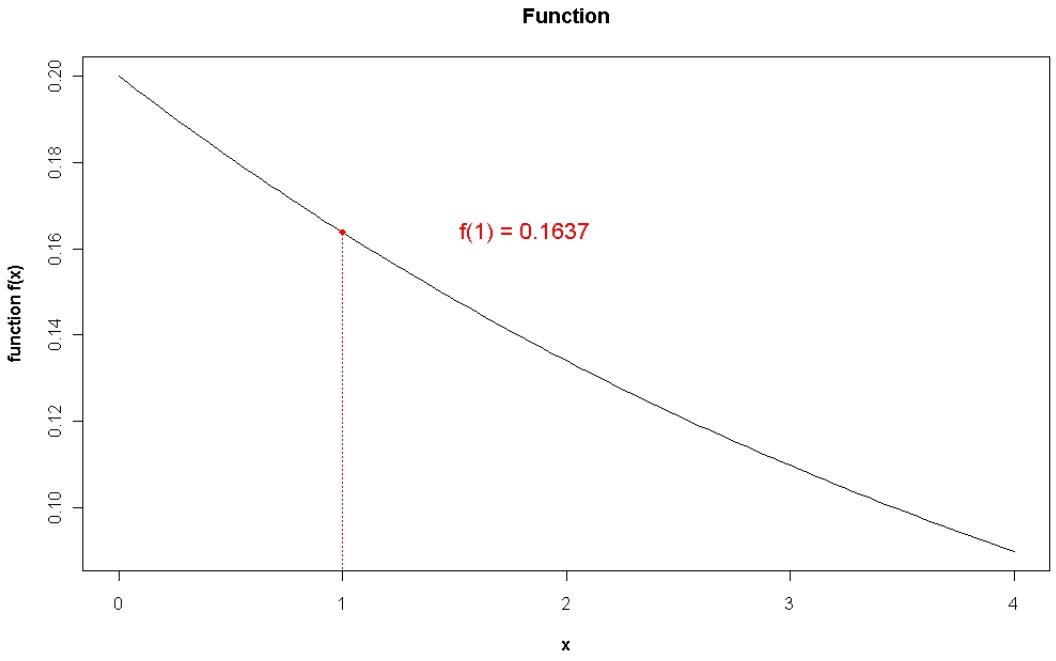
1.1 Probability Mass Functions

For a discrete distribution, the probability mass function (pmf) is the probability that the discrete random variable X has some specified value (k) i.e. $P(X = k)$. Probability mass functions are determined using the `d` family of functions. For example, for the binomial distribution, the command is `dbinom()`, for the Poisson distribution `dpois()`.

1.2 Probability Density Functions

For a discrete distribution, the probability density function (pdf) is the relative likelihood that the continuous random variable X will have some specified value (k). Since for continuous distributions the probability at a single point is zero, it is **not** equivalent to $P(X = k)$.

Rather the density function is the height of a ***density curve*** (such as the well-known Gaussian bell-curve) at point k.



Probability density functions are determined using the `d` family of functions. For example, for the normal distribution, the command is `dnorm()`. Necessarily the `d` family of functions are more useful for discrete random variables than for continuous variables.

1.3 Cumulative Distribution Functions

The cumulative distribution function (cdf) is the probability that the variable takes a value less than or equal to x. The cumulative density function is often denoted as $F(x)$.

$$F(x) = P(X \leq x) = \alpha$$

There is little difference in how to treat the cdf for discrete and continuous variables, other than the definition of the complement. Values for the cumulative distribution functioncan be determined using the `p` family of functions.

It is possible to specify the complement probability of the cumulative distribution function (i.e $P(X \geq x)$) directly by additionally specifying the argument `lower=FALSE`.

1.4 Inverse Cumulative Distribution Functions

The inverse cumulative distribution function (otherwise known as the quantile function) is a function that yields the quantile k for some specified probability α such that the variable takes a value less than or equal to k with this probability (Recall that $p(X \leq x) = \alpha$).

$$F^{-1}(\alpha) = x$$

Implementation of the inverse cumulative distribution function requires use of the `q` family of functions. Quantile functions generally tend to be used with continuous random variables, rather than for discrete random variables.

2 The Normal Distribution

There are four functions that can be used to generate the values associated with the normal distribution. The first function we shall look at is `dnorm`. Given an appropriate set of argument values (i.e. for mean and standard deviation), this function returns the height of the desnity curve (i.e. plot of the probability density distribution) for those specifications.

To specify a value for mean, use the argument `mean=`. Similarly to specify a value for mean, use the argument `sd=`.

```
dnorm(3,mean=2,sd=1)
```

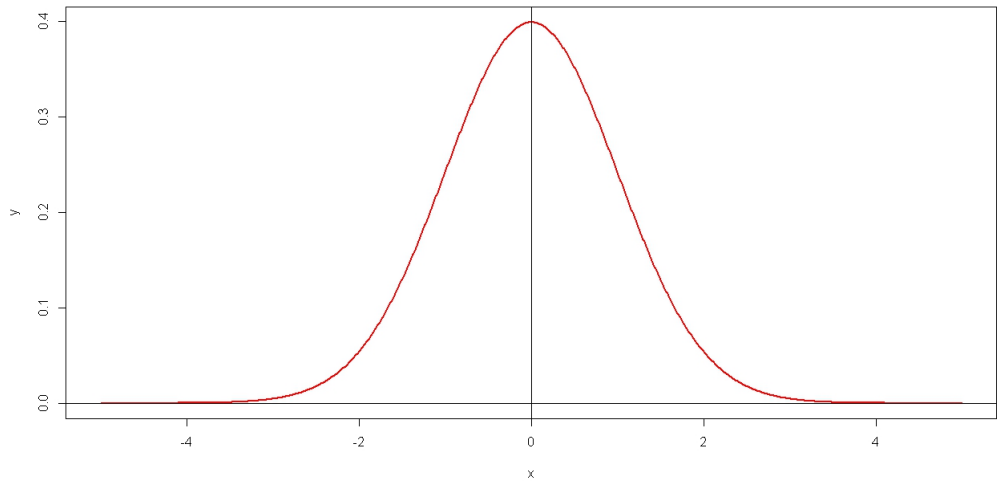
```
> dnorm(3,mean=2,sd=1)
[1] 0.2419707
```

The default values for mean (μ) and standard deviation (σ) are 0 and 1 respectively. This is , by definition, the standard normal distribution, commonly called the Z distribution. Therefore, if the values for mean and standard deviation are not specified, 0 and 1 are used by default.

```
> dnorm(3)
[1] 0.004431848
> dnorm(0)
[1] 0.3989423
>dnorm(0)*sqrt(2*pi)
[1] 1
> vec = c(0,1,2)
> dnorm(vec)
[1] 0.39894228 0.24197072 0.05399097
```

The command `dnorm()` can be used to plot the bell curve. The bell curve depicted below can be constructed using the following `R` code.

```
x = seq(-5,5,by=.01)
y = dnorm(x)
plot(x,y,type="l",lwd=2,col="red")
abline(h=0)
abline(v=0)
```



The second function we examine is `pnorm()`. Given a number or a list this function computes the cumulative distribution function, i.e. probability that a normally distributed random number will be less than that number. This function accepts the same options as `dnorm()`, i.e. `mean=` and `sd=`.

You can get a full list of them and their options using the help command:

```
>  pnorm(0)
[1] 0.5
>  pnorm(1)
[1] 0.8413447
>  pnorm(0,mean=2)
[1] 0.02275013
>
>  pnorm(0,mean=2,sd=3)
[1] 0.2524925
>
>  vec = c(0,1,2)
>  pnorm(vec)
[1] 0.5000000 0.8413447 0.9772499
>
```

2.1 Applications of the Normal Distribution

Suppose the mass of shire horses is assumed to have a normal distribution with mean 1000kg and standard deviation 50kg.

1. Calculate the probability that the mass of a shire horse is less than 975kgs.
2. Calculate the probability that the mass of a shire horse is more than 975kgs.
3. Calculate the probability that the mass of a shire horse is between 940kg and 1030kg.
4. Calculate the probability that the mass of a shire horse exceeds 1030kg.

Solutions: Let X be the mass of the horses, distributed as

$$X \sim \mathcal{N}(1000, 50^2)$$

.

1. $P(X \leq 975)$: `pnorm(975,mean=1000,sd=50)`.
2. $P(X \geq 975) = 1 - P(X \leq 975)$:`1-pnorm(975,mean=1000,sd=50)`.
3. $P(940 \leq X \leq 1030) = P(X \leq 1030) - P(X \leq 940)$: `pnorm(1030,mean=1000,sd=50) - pnorm(940,mean=1000,sd=50)`.
4. $P(X \geq 1030)$ `pnorm(1030,mean=1000,sd=50,lower=FALSE)`.

```
> pnorm(975,mean=1000,sd=50)
[1] 0.3085375
> 1-pnorm(975,mean=1000,sd=50)
[1] 0.6914625
> pnorm(1030,mean=1000,sd=50)-pnorm(940,mean=1000,sd=50)
[1] 0.6106772
>
> pnorm(1030,mean=1000,sd=50)
[1] 0.7257469
> pnorm(1030,mean=1000,sd=50,lower=FALSE)
[1] 0.2742531
```

The answers are $30.85\%, 69.14\%$ and 61.06% respectively.

The next function we look at is `qnorm()`, which is the inverse of `pnorm()`. The idea behind `qnorm()` is that you specify the probability, and it returns the number whose cumulative distribution matches the probability. For example, if you have a normally distributed random variable with mean zero and standard deviation one, then if you give the function a probability it returns the associated Z-value:

```
> qnorm(0.5)
[1] 0
>#Well Known Z Scores
> qnorm(0.95)
[1] 1.644854
> qnorm(0.975)
[1] 1.959964
> qnorm(0.995)
[1] 2.575829
>
> qnorm(0.5,mean=1)
[1] 1
```

```
>  qnorm(0.5,mean=1,sd=2)
[1] 1
>  qnorm(0.5,mean=2,sd=2)
[1] 2
>  qnorm(0.5,mean=2,sd=4)
[1] 2
>  qnorm(0.25,mean=2,sd=2)
[1] 0.6510205
>
```

The last function we examine is the `help(rnorm)` function which can generate normally distributed random numbers. The argument that must be specified are the number of random numbers required, as well as optional arguments to specify the mean and standard deviation. Randomly generated numbers will be specified to several decimal places, so it is worth recalling functions such as `help(round())` that can be used to manage precision.

```
>  rnorm(4)
[1] -0.79949714 -1.14181543  0.09373155 -0.18318997
>
>  rnorm(4,mean=3)
[1] 3.948279 4.214592 1.488769 1.354725
>
>  rnorm(4,mean=3,sd=3)
[1] 1.259427 1.554388 1.654530 1.076075
>
>  rnorm(4,mean=3,sd=3)
[1] 2.802254 2.681757 1.528246 6.496170
>
```

The help file for thes commands can be obtained by specifying any one of the fuctions as an argument to the `help()` function: e.g.`help(dnorm)`.

3 Student t-Distribution

The Students t-distribution is a probability distribution that is used to estimate population parameters when the sample size is small and/or when the population variance is unknown. The parameters are the ***degrees of freedom*** (df = n-1 where n is sample size).

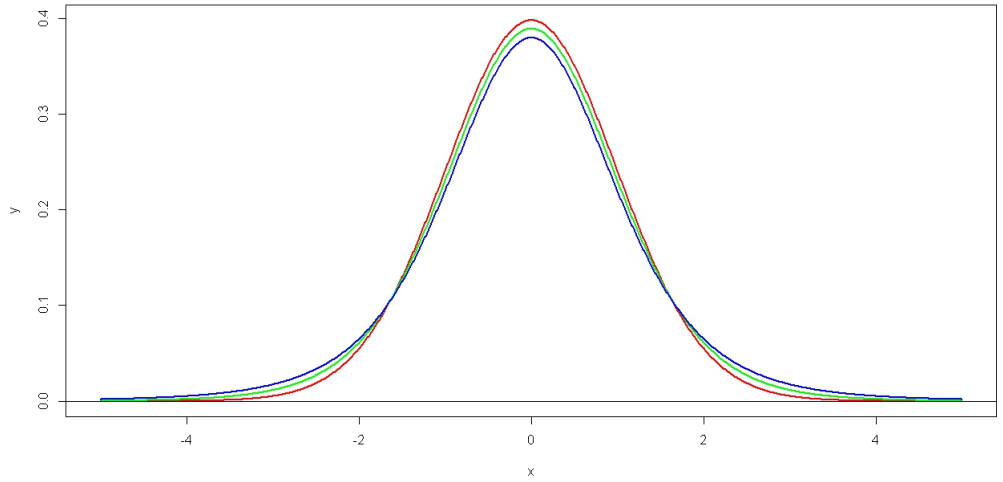
As with the normal distribution, there are four functions that can be used to generate the values associated with the $t-$distribution.The commands follow the same kind of naming convention, and the names of the commands are `dt(), pt(), qt()` and `rt()`.

One difference between the $t-$distribution and the normal distribution is that, for the $t-$distribution, it is assumed that the values are ***standardized*** to mean zero and standard deviation one. Also the number of degrees of freedom must be specified.

```
> pt(-3,df=10)
[1] 0.006671828
>  pt(3,df=10)
[1] 0.9933282
>  1-pt(3,df=10)
[1] 0.006671828
>  pt(3,df=20)
[1] 0.9964621
```

The density curves for various degrees of freedom (red with df=5, blue with df=15 and green with df=100) are depicted on the following graph.

```
x = seq(-5,5,by=.01)
y = dt(x,df=100)
plot(x,y,type="l",lwd=2,col="red")
abline(h=0)
w = dt(x,df=10)
points(x,w,type="l",lwd=2,col="green",add=TRUE)
z = dt(x,df=5)
points(x,z,type="l",lwd=2,col="blue",add=TRUE)
```



Next we have the inverse cumulative probability distribution function. Recall computing critical values for small sample hypothesis tests. The values computed here are equivalent of the value found in statistical tables.

```
> qt(0.05,df=10)
[1] -1.812461
>  qt(0.95,df=10)
[1] 1.812461
>  qt(0.05,df=20)
[1] -1.724718
>  qt(0.95,df=20)
[1] 1.724718
```

Finally random numbers can be generated according to the $t-$distribution using the `rt()`.

```
> rt(3,df=10)
[1] 0.9440930 2.1734365 0.6785262
> rt(3,df=20)
[1] 0.1043300 -1.4682198  0.0715013
```

4 Continuous Probability Distributions

In addition to the normal distribution and the $t-$distribution , we will look at two more continuous distribution functions:

- The Uniform Distribution
- The Exponential Distributions

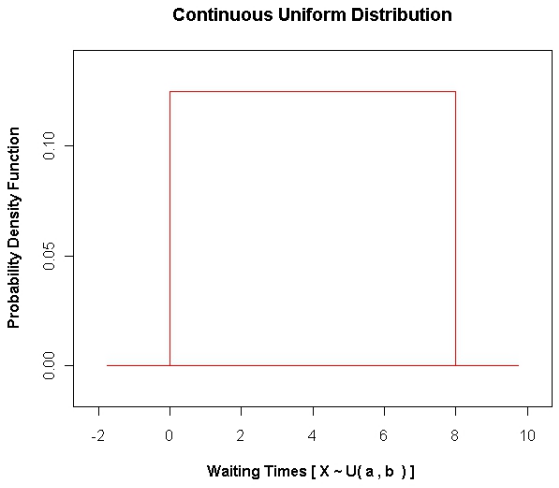
4.1 The Uniform Distribution

A random variable X is called a continuous uniform random variable over the interval (a, b) if it's probability density function is given by

$$f_X(x) = \frac{1}{b-a} \qquad \text{when } a \leq x \leq b \text{ (otherwise } f_X(x) = 0)$$

The corresponding cumulative density function is

$$F_x(x) = \frac{x-a}{b-a} \qquad \text{when } a \leq x \leq b$$



The continuous distribution is very simple to understand and implement, and is commonly used in computer applications (e.g. computer simulation). It is known as the 'Rectangle Distribution' for obvious reasons. We specify the word "continuous" so as to distinguish it from it's discrete equivalent: the discrete uniform distribution. The continuous uniform distribution is characterized by the following parameters

- The lower limit a (or with `R`: `min`)
- The upper limit b (or with `R`: `max`)
- We denote a uniform random variable X as $X \sim U(a, b)$

It is not possible to have an outcome that is lower than a or larger than b ,i.e. $P(X < a) = P(X > b) = 0$.

We wish to compute the probability of an outcome being within a range of values.We shall call this lower bound of this range L and the upper bound U. Necessarily L and U must be possible outcomes.The probability of X being between L and U is denoted $P(L \leq X \leq U)$.

In the absence of the specified upper and lower limits, the default values of 0 and 1 would be used.

4.1.1 Generating Random Values

The most common use of the uniform distribution is the generation of uniformly distributed random variables. The relevant command is `runif()`. Functions that can be used to manage precision are useful when generating random values.

```
> runif(10)
 [1] 0.2558100 0.8738507 0.4521578 0.7868320 0.2310644 0.5265236
 [7] 0.9041761 0.3948904 0.1928505 0.5793142
> runif(10,min=1,max=7)
 [1] 2.822485 4.603547 5.794749 3.766398 2.016349 2.116504 5.863682
 [8] 3.911420 3.434373 6.986899
>
> Another Way of Simulating Dice Rolls
> floor(runif(10,min=1,max=7))
 [1] 1 5 1 6 6 2 1 2 6 3
```

4.2 The Exponential Distribution

The exponential distribution describes the arrival time of a randomly recurring independent event sequence. If μ is the mean "duration" or "waiting time" for the next event recurrence, its probability density function is given as

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

The parameter λ is called ***rate*** parameter. It is the inverse of the expected duration (μ).

$$\mu = \frac{1}{\lambda}$$

.The cumulative distribution function of an exponential distribution is

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

If the expected duration is 5 (e.g. five minutes) then the rate parameter value is 0.2. When implemented with `R`, the exponential distribution is characterized by the single parameter `rate`.The commands follow the same kind of naming convention, and the names of the commands are `dexp()`, `pexp()`, `qexp()`, and `rexp()`. A few examples are given below to show how to use the various commands.

4.2.1 Exponential Distribution : Problem

Suppose the mean checkout time of a supermarket cashier is three minutes. Find the probability of a customer checkout being completed by the cashier in less than two minutes, three minutes and four minutes. (i.e. what percentage of "waiting times" are less than two, three and four minutes?)

Solution The checkout processing rate is equals to one divided by the mean checkout completion time. Hence the processing rate is 1/3 checkouts per minute. We then apply the function `pexp()` of the exponential distribution with rate=1/3. Also compute the probability that the waiting time exceeds 5 minutes.

```
> pexp(2,rate=1/3)
[1] 0.4865829
> pexp(3,rate=1/3)
[1] 0.6321206
> pexp(4,rate=1/3)
[1] 0.7364029
> pexp(5,rate=1/3,lower=FALSE)
[1] 0.1888756
```

The probabilities of finishing a checkout in under two, three and four minutes by the cashier are approximately 48.6 %, 63.21% and 73.6% respectively.

What is the median waiting time? To answer this question we would use the `qexp()` function. Recall that the median is value of x such that $P(X \leq x) = 0.50$. Also determine the first and third quartiles Q_1 and Q_3.

```
> qexp(0.5,rate=1/3)
[1] 2.079442
> qexp(0.25,rate=1/3)
[1] 0.8630462
> qexp(0.75,rate=1/3)
[1] 4.158883
```

5 Discrete Probability Distributions

The discrete distributions we will look at in this course are

- The binomial distribution
- The Poisson distribution

5.1 Binomial Distribution

A binomial experiment (known as a Bernoulli trial) is a statistical experiment that has the following properties:

- The experiment consists of n repeated trials.
- Each trial can result in just two possible outcomes. We call one of these outcomes a success and the other, a failure.
- The probability of success, conventionally denoted mathematically as p, is the same on every trial.
- The trials are independent; that is, the outcome on one trial does not affect the outcome on other trials.
- There are n independent trials.

As with other functions, there are four functions that can be used to generate the values associated with the binomial distribution; `dbinom()`,`pbinom()`, `qbinom()` and `rbinom()`. The arguments to these commands are `size=` and `prob=`.

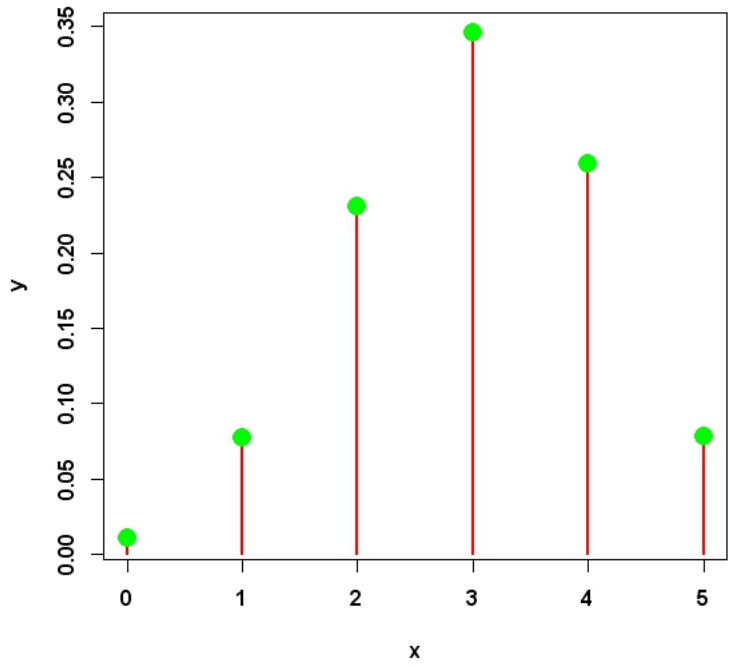
Suppose that there are ten independent trials, and that the probability of success on each trial is 0.6, the probability of 5 successes over ten trials can be computed using the following code.

```
dbinom(5,size=10,prob=0.60)
```

The probability that the numbers not excessing 5 successes is computed using the `pbinom()` function.

```
pbinom(5,size=10,prob=0.60)
```

```
> dbinom(5,size=10,prob=0.60)
[1] 0.2006581
> pbinom(5,size=10,prob=0.60)
[1] 0.3668967
```



This plot was implemented with the following R code.

```
x = seq(0,5,by=1)
y = dbinom(x,5,0.6)
plot(x,y,type="h",col="red",lwd=2,font.lab=2,font.axis=2)
points(x,y,type="p",col="green",cex=2,pch=16)
```

The heights of each line are as follows:

```
> dbinom(0:5,size=5,prob=0.60)
[1] 0.01024 0.07680 0.23040 0.34560 0.25920 0.07776
```

5.1.1 Exercise

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers, and only one of them is correct. Find the probability of having four or less correct answers if a student attempts to answer every question at random.

Solution: Since only one out of five possible answers is correct, the probability of answering a question correctly by random is 1/5=0.2. To find the probability of having four or less correct answers by random attempt, we apply the function pbinom with x = 4, size = 12, prob = 0.2.

```
> pbinom(4, size=12,prob=0.2)
[1] 0.92744
```

The probability of four or less questions answered correctly by random in a twelve question multiple choice quiz is 92.7%.

5.1.2 Generating Random Values

Suppose we wish to simulate the number of defective components in a sequence of twenty batches of 100 items, where the probability of defect is 0.02. We would implement this simulation as follows:

```
rbinom(20,size=100,prob=0.02)
```

```
> rbinom(20,size=100,prob=0.02)
 [1] 0 3 1 1 3 2 0 2 2 0 5 0 1 1 3 1 2 4 0 3
```

5.1.3 Quantiles

The inverse cumulative distribution function can be implemented using the `qbinom()` function. However the output is generally not as useful as the `pbinom()` function.

```
> qbinom(0.95,100,0.02)
[1] 5
> qbinom(0.90,100,0.02)
[1] 4
> qbinom(0.80,100,0.02)
[1] 3
> qbinom(0.85,100,0.02)
[1] 3
```

5.2 The Poisson Distribution

A Poisson distribution is the probability distribution that results from a Poisson experiment. A Poisson experiment, ,with X as the number of successes, is a statistical experiment that has the following properties:

- The experiment results in outcomes that can be classified as successes or failures.
- The average number of successes (λ or lambda) that occurs in a specified region is known.
- The probability that a success will occur is proportional to the size of the region.
- The probability that a success will occur in an extremely small region is virtually zero.

Note that the specified region could take many forms. For instance, it could be a length, an area, a volume, a period of time, etc. The Poisson distribution will feature significantly in the ***Stochastic Processes*** module that many of you will study in the future.

There are four functions that can be used to generate the values associated with the Poisson distribution. The commands follow the same kind of naming convention, and the names of the commands are `dpois()`, `ppois()`, `qpois()`, and `rpois()`.

These commands work just like the commands for the other distribution. The Poisson distribution requires the lambda. parameters. A few examples are given below to show how to use the different commands. First we have the density function, `dpois()`.

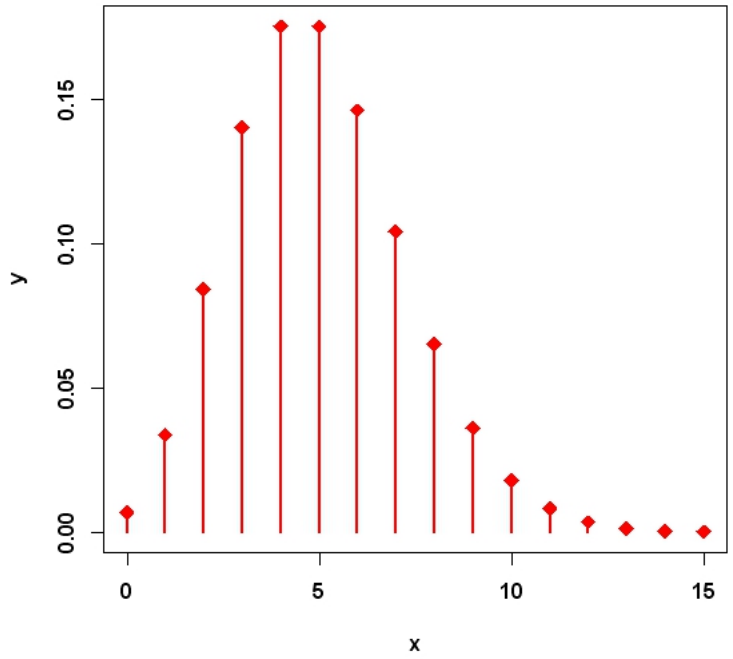
5.2.1 Example

Suppose that the expected number of occurences per unit space is 5, compute the probability of exactly 4 and exactly 5 occurences, i.e. $P(X = 4)$ and $P(X = 5)$ respectively.

```
dpois(4,lambda=5)
dpois(5,lambda=5)
```

The probability of both 4 and 5 successes is 17.5%.

```
> dpois(4,lambda=5)
[1] 0.1754674
> dpois(5,lambda=5)
[1] 0.1754674
```



This plot was implemented with the following R code. The upper limit of 15 was arbitrarily chosen. There is no upper bound in the sample space of potential successes.

```
x = seq(0,15,by=1)
y = dpois(x,lambda=5)
plot(x,y,type="h",col="red",lwd=2,font.lab=2,font.axis=2)
points(x,y,type="p",col="red",cex=1.60,pch=18)
```

Compute the probability that the number of successes is less than or equal to 15, i.e. $P(X \leq 15)$. To compute this, we would use the `ppois()` command. The answer is 99.99%.

```
ppois(3,lambda=5)
```

```
> ppois(15,lambda=5)
[1] 0.999931
```

As mentioned previously, the Poisson distribution is particularly useful in simulating stochastic processes, by generating suitable random values. Suppose the expected number of occurrences is 5 per month. To simulate the number of occurences for two year (i.e. 24 months), we use the `rpois()` function.

```
> rpois(24,lambda=5)
 [1] 4 6 5 6 5 4 7 7 1 2 3 6 7 1 2 6 2 2 7 9 1 4 5 1
```

5.3 Exercise

If there are twelve cars crossing a bridge per minute on average, find the probability of having seventeen or more cars crossing the bridge in a particular minute (i.e. $P(X \geq 17)$).

The probability of having sixteen or less cars crossing the bridge in a particular minute is given by the `ppois()` function.

```
> ppois(16, lambda=12)
[1] 0.89871
```

Hence the probability of having seventeen or more cars crossing the bridge in a minute is in the upper tail of the probability density function.

```
> ppois(16, lambda=12, lower=FALSE)   # upper tail
[1] 0.10129
```

If there are twelve cars crossing a bridge per minute on average, the probability of having seventeen or more cars crossing the bridge in a particular minute is 10.1%.

6 Summary of Distributions

Distribution	R name	Arguments
beta	beta	shape1, shape 2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1,df2,ncp
normal	norm	mean, sd
Poisson	pois	lambda
Student's t	t	df, ncp
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n