

# Introduction to Probability Distributions

## 4. Basic Probability Distributions

### Contents

- [The Normal Distribution](#)
- [The t Distribution](#)
- [The Binomial Distribution](#)
- [The Chi-Squared Distribution](#)

We look at some of the basic operations associated with probability distributions. There are a large number of probability distributions available, but we only look at a few. If you would like to know what distributions are available you can do a search using the command `help.search("distribution")`.

Here we give details about the commands associated with the normal distribution and briefly mention the commands for other distributions. The functions for different distributions are very similar where the differences are noted below.

For this chapter it is assumed that you know how to enter data which is covered in the previous chapters.

To get a full list of the distributions available in R you can use the following command:

```
help(Distributions)
```

For every distribution there are four commands. The commands for each distribution are prepended with a letter to indicate the functionality:

"d"	returns the height of the probability density function
"p"	returns the cumulative density function
"q"	returns the inverse cumulative density function (quantiles)
"r"	returns randomly generated numbers

### 4.1. The Normal Distribution

There are four functions that can be used to generate the values associated with the normal distribution. You can get a full list of them and their options using the help command:

```
> help(Normal)
```

The first function we look at it is *dnorm*. Given a set of values it returns the height of the probability distribution at each point. If you only give the points it assumes you want to use a mean of zero and standard deviation of one. There are options to use different values for the mean and standard deviation, though:

```
dnorm(0)
```

```
[1] 0.3989423
```

```
dnorm(0) * sqrt(2*pi)
```

```
[1] 1
```

```
dnorm(0, mean=4)
```

```
[1] 0.0001338302
```

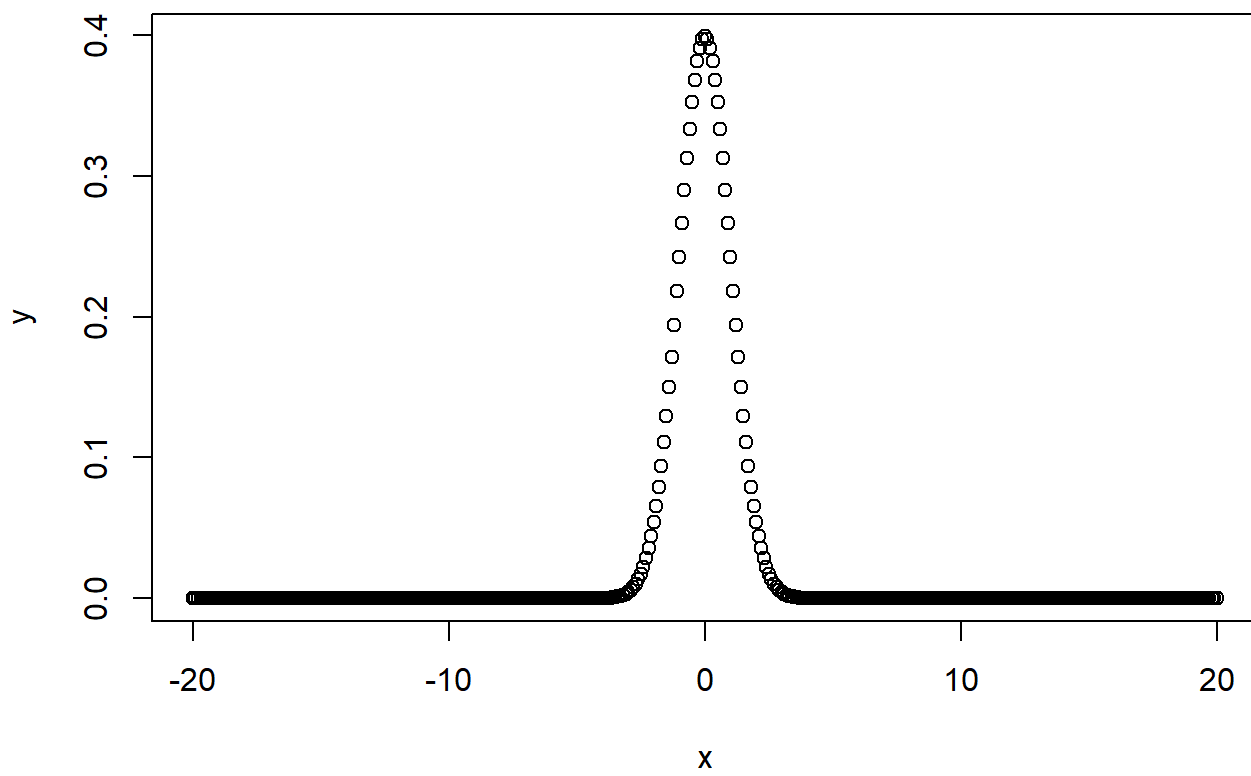
```
dnorm(0, mean=4, sd=10)
```

```
[1] 0.03682701
```

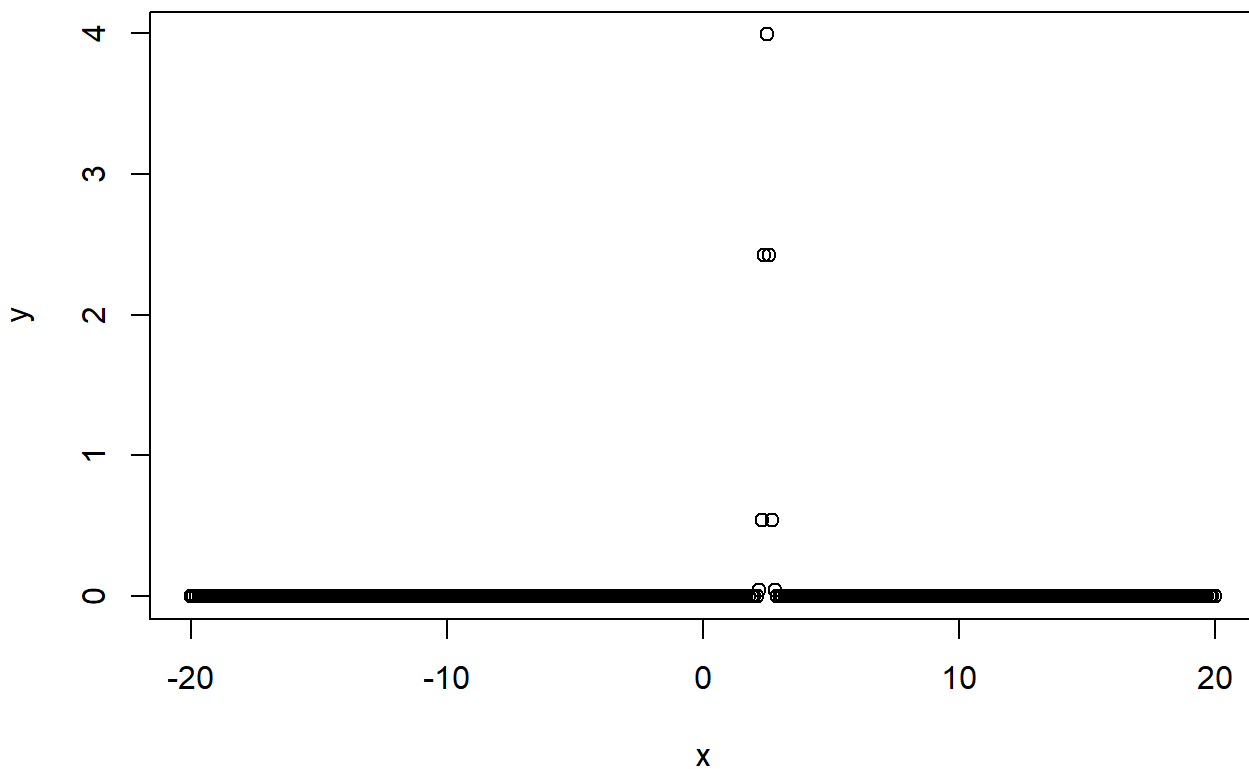
```
v <- c(0,1,2)  
dnorm(v)
```

```
[1] 0.39894228 0.24197072 0.05399097
```

```
x <- seq(-20,20,by=.1)  
y <- dnorm(x)  
plot(x,y)
```



```
y <- dnorm(x,mean=2.5,sd=0.1)
plot(x,y)
```



The second function we examine is `pnorm`. Given a number or a list it computes the probability that a normally distributed random number will be less than that number. This function also goes by the rather ominous title of the “Cumulative Distribution Function.” It accepts the same options as `dnorm`:

```
pnorm(0)
```

```
[1] 0.5
```

```
pnorm(1)
```

```
[1] 0.8413447
```

```
pnorm(0, mean=2)
```

```
[1] 0.02275013
```

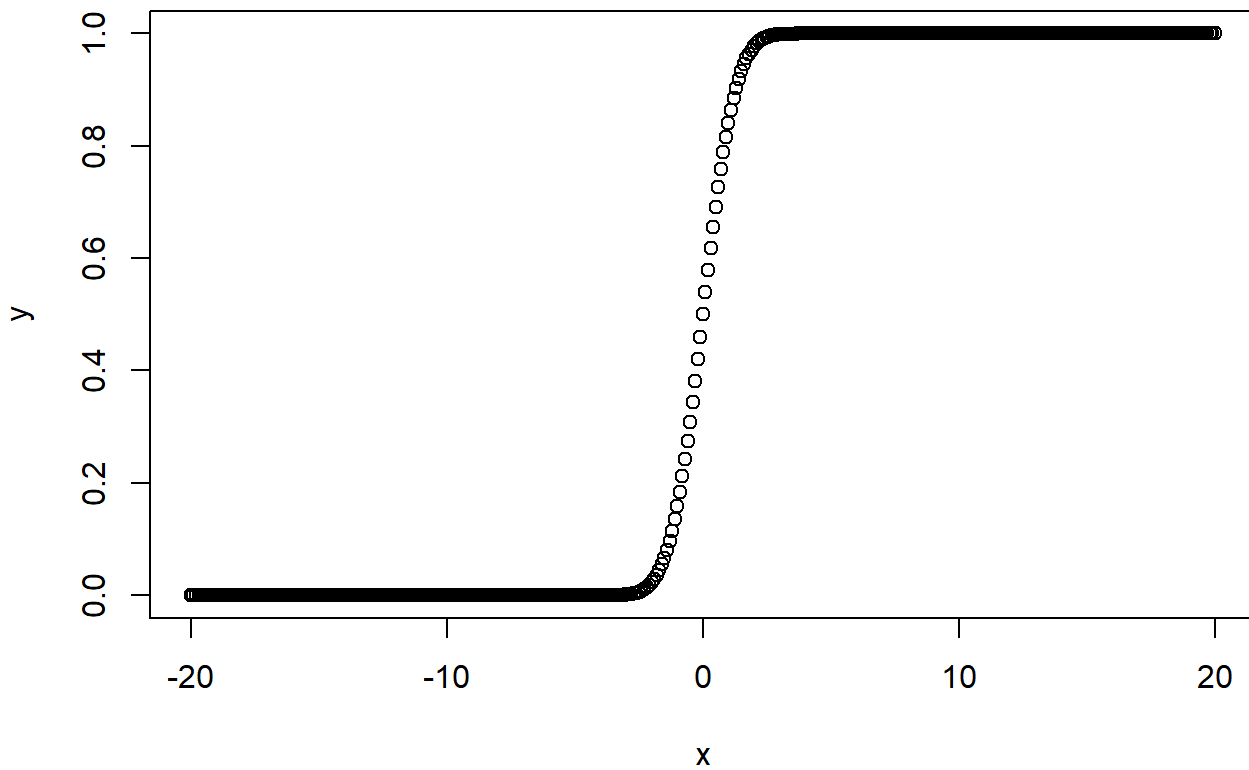
```
pnorm(0, mean=2, sd=3)
```

```
[1] 0.2524925
```

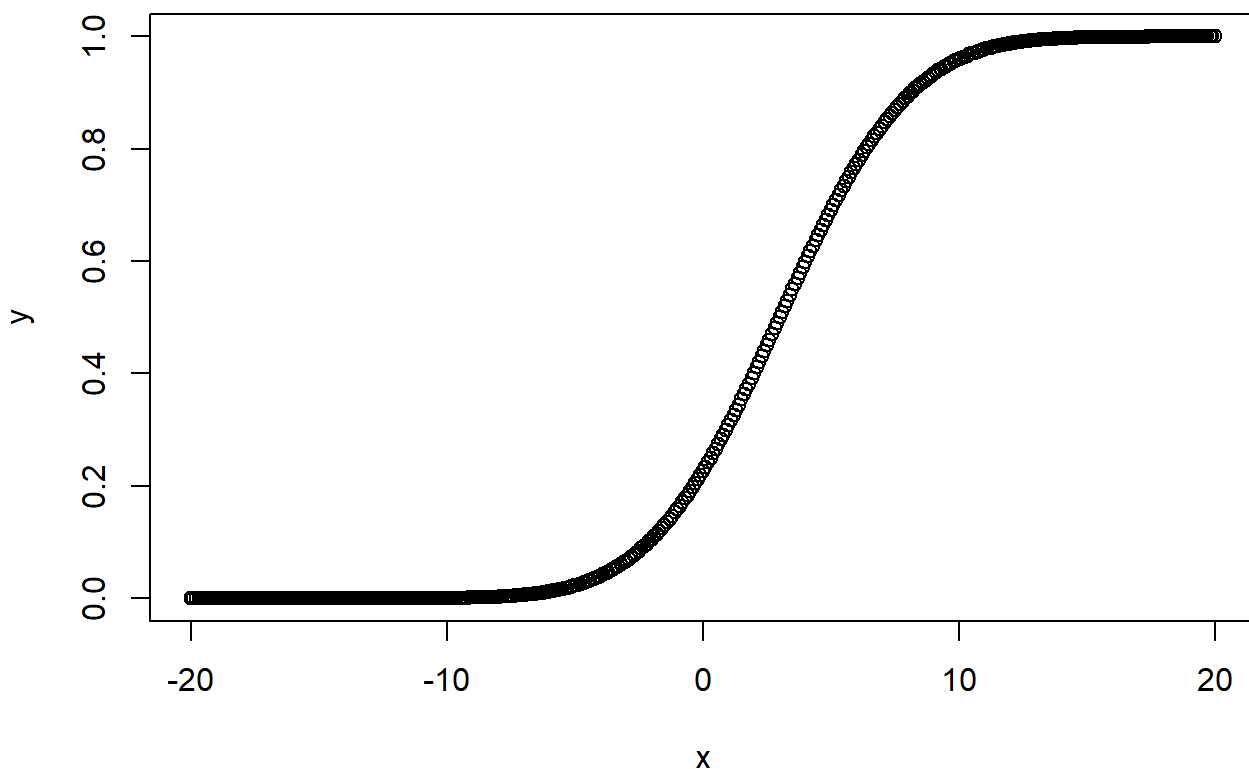
```
v <- c(0,1,2)
pnorm(v)
```

```
[1] 0.5000000 0.8413447 0.9772499
```

```
x <- seq(-20,20,by=.1)  
y <- pnorm(x)  
plot(x,y)
```



```
y <- pnorm(x,mean=3,sd=4)  
plot(x,y)
```



If you wish to find the probability that a number is larger than the given number you can use the *lower.tail* option:

```
pnorm(0, lower.tail=FALSE)
```

```
[1] 0.5
```

```
pnorm(1, lower.tail=FALSE)
```

```
[1] 0.1586553
```

```
pnorm(0, mean=2, lower.tail=FALSE)
```

```
[1] 0.9772499
```

The next function we look at is *qnorm* which is the inverse of *pnorm*. The idea behind *qnorm* is that you give it a 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-score:

```
qnorm(0.5)
```

```
[1] 0
```

```
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
```

```
qnorm(0.333)
```

```
[1] -0.4316442
```

```
qnorm(0.333,sd=3)
```

```
[1] -1.294933
```

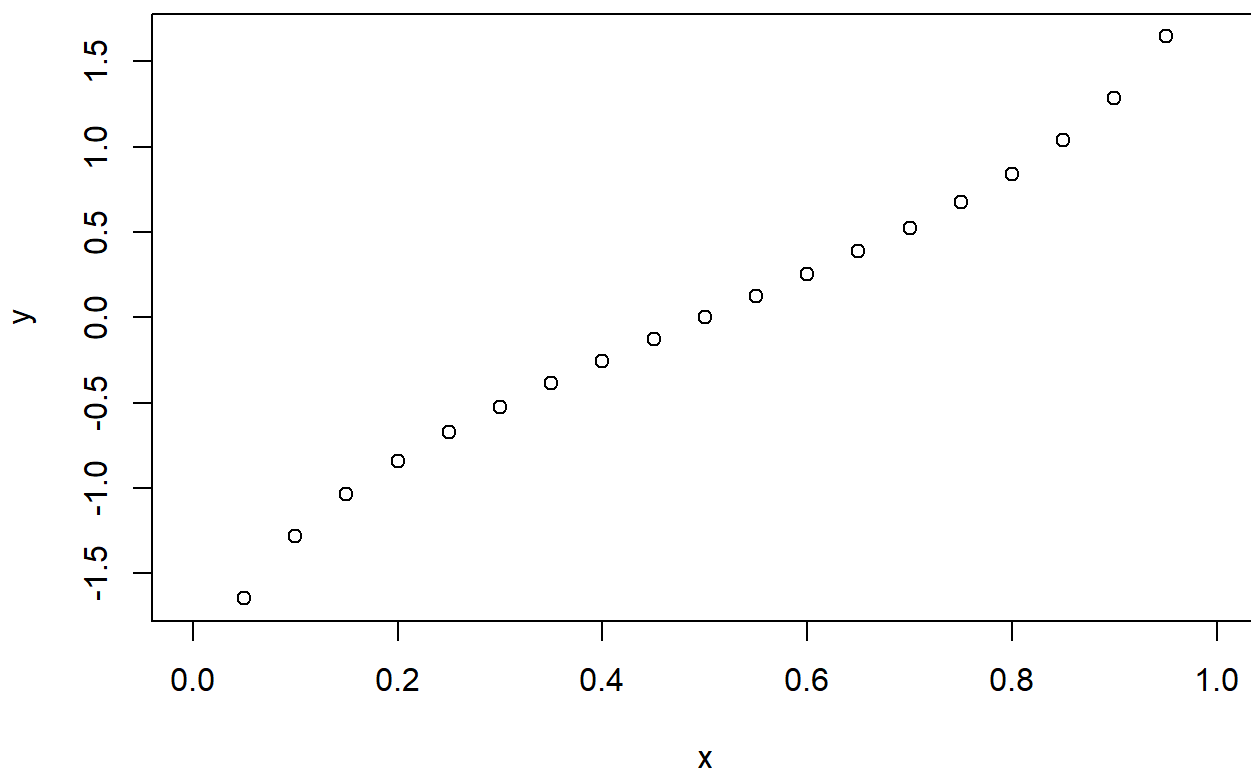
```
qnorm(0.75,mean=5,sd=2)
```

```
[1] 6.34898
```

```
v = c(0.1,0.3,0.75)  
qnorm(v)
```

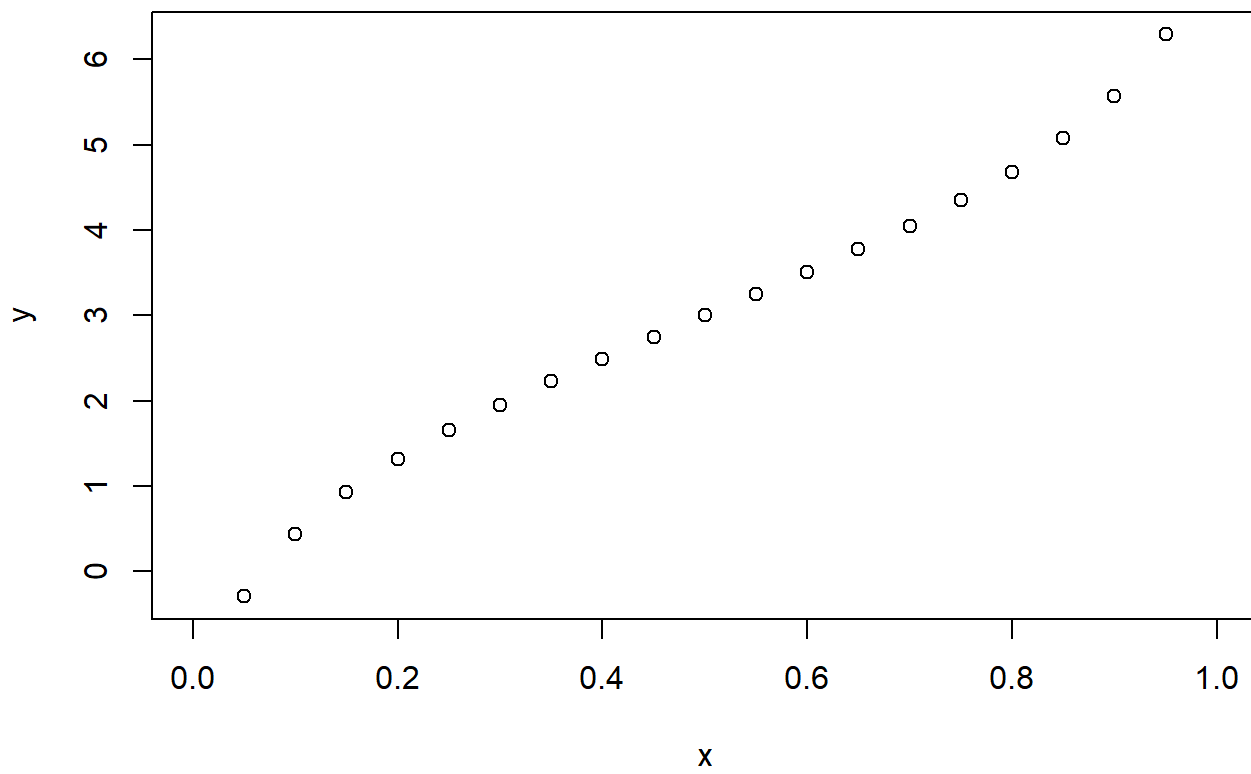
```
[1] -1.2815516 -0.5244005 0.6744898
```

```
x <- seq(0,1,by=.05)  
y <- qnorm(x)  
plot(x,y)
```

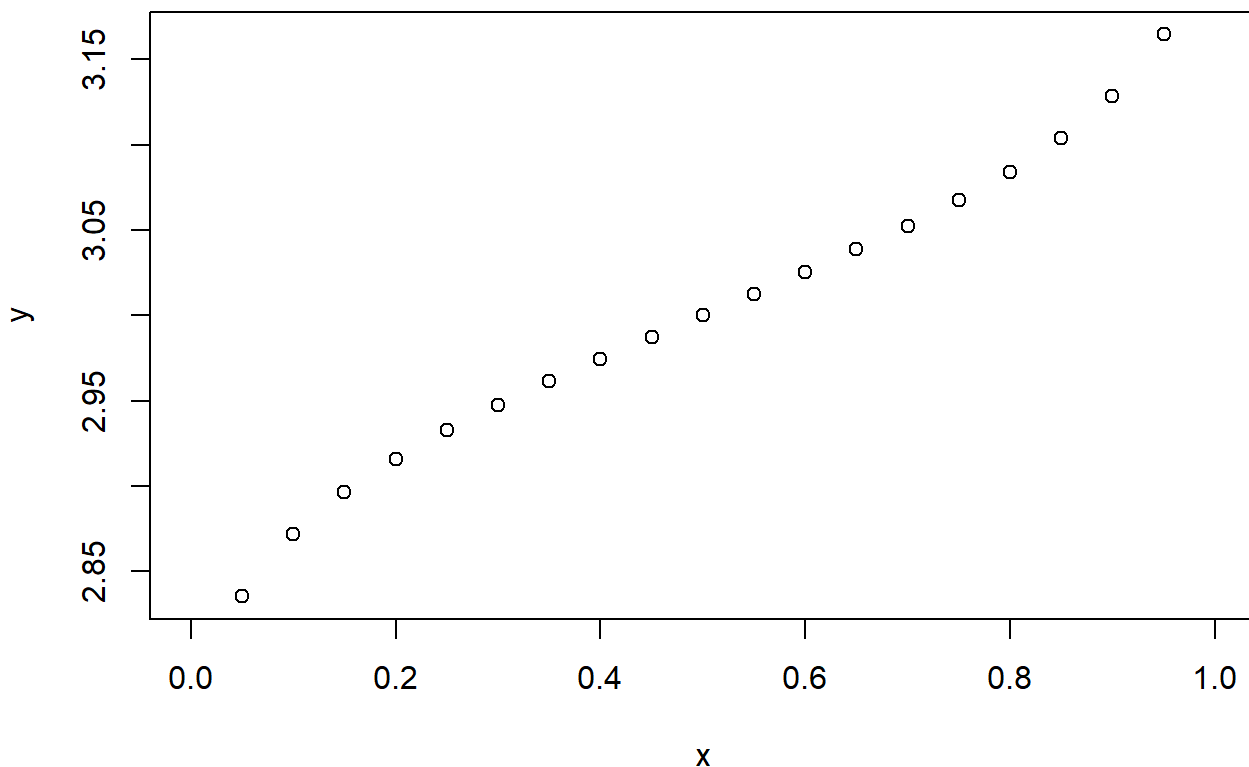


```
y <- qnorm(x,mean=3,sd=2)
plot(x,y)
```





```
y <- qnorm(x,mean=3,sd=0.1)
plot(x,y)
```



The last function we examine is the `rnorm` function which can generate random numbers whose distribution is normal. The argument that you give it is the number of random numbers that you want, and it has optional arguments to specify the mean and standard deviation:

```
rnorm(4)
```

```
[1] 0.5486528 2.1348439 0.4554851 0.6609455
```

```
rnorm(4,mean=3)
```

```
[1] 1.647380 2.178437 1.493164 2.852338
```

```
rnorm(4,mean=3,sd=3)
```

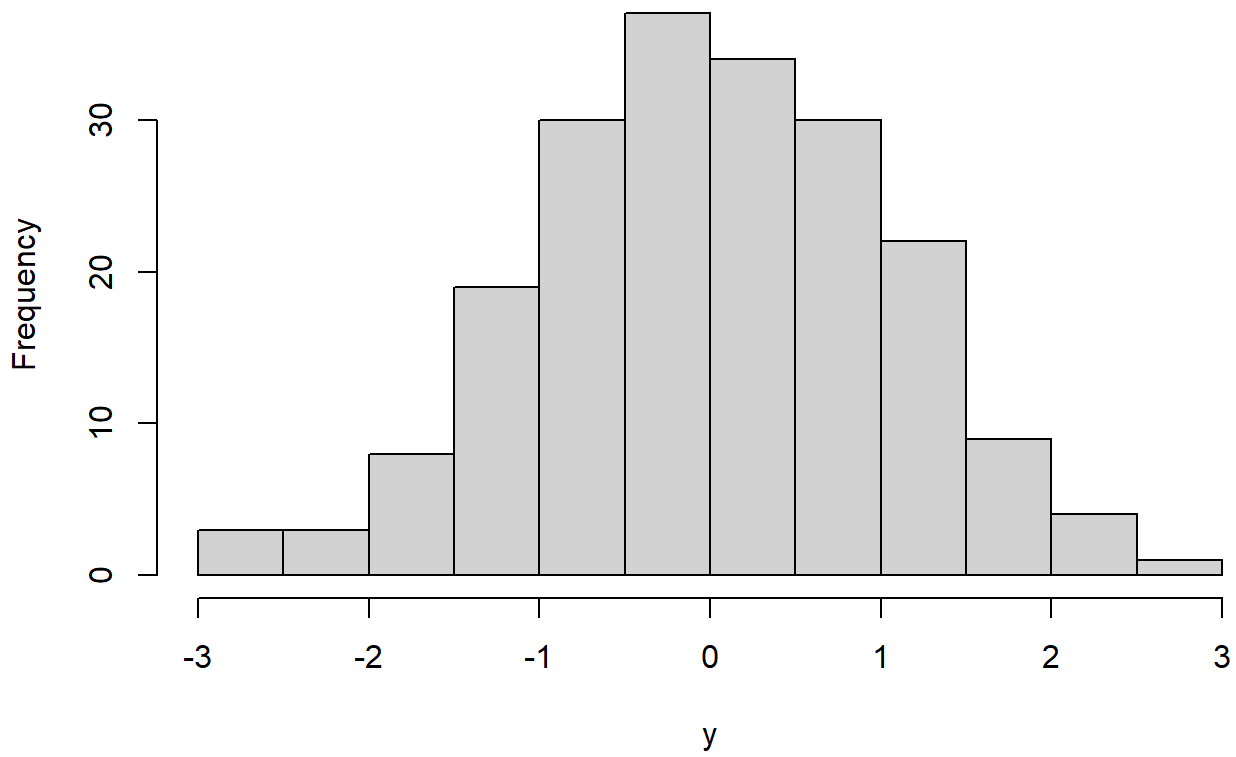
```
[1] 11.287850 1.823686 9.085679 2.879095
```

```
rnorm(4,mean=3,sd=3)
```

```
[1] -1.714814 1.277867 1.203975 -1.539575
```

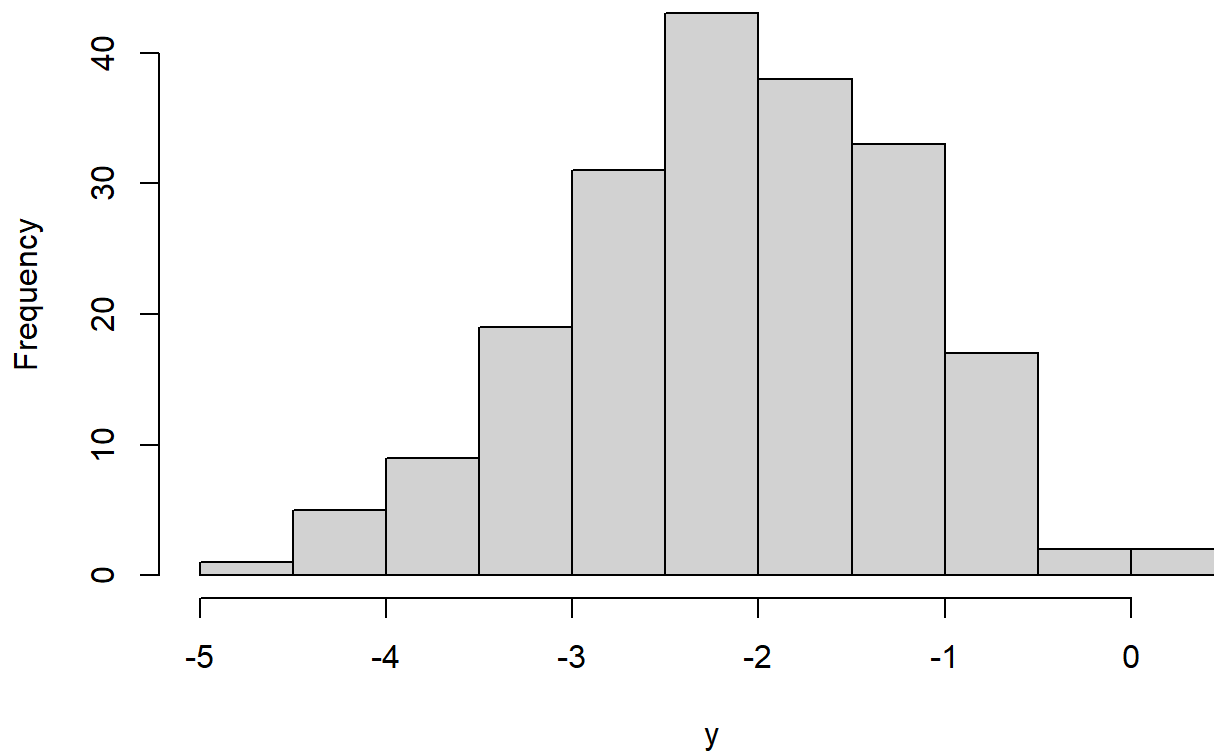
```
y <- rnorm(200)  
hist(y)
```

## Histogram of y



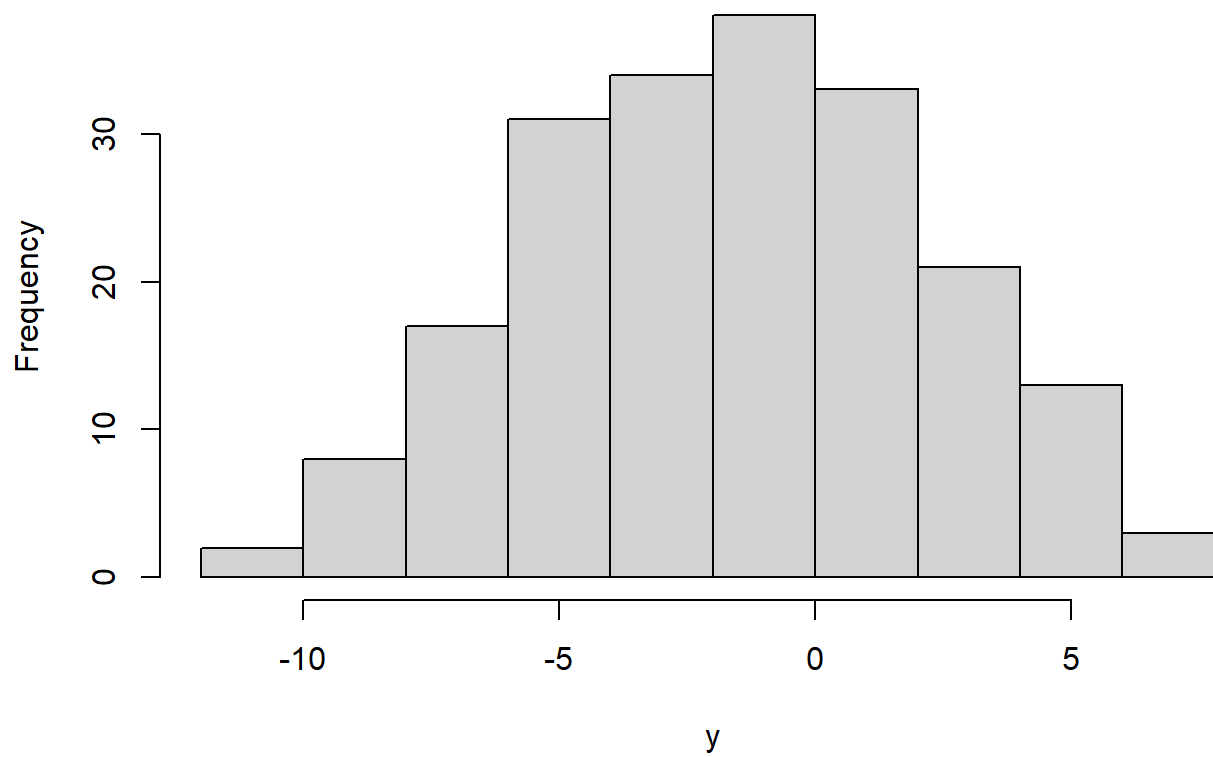
```
y <- rnorm(200, mean=-2)
hist(y)
```

## Histogram of y



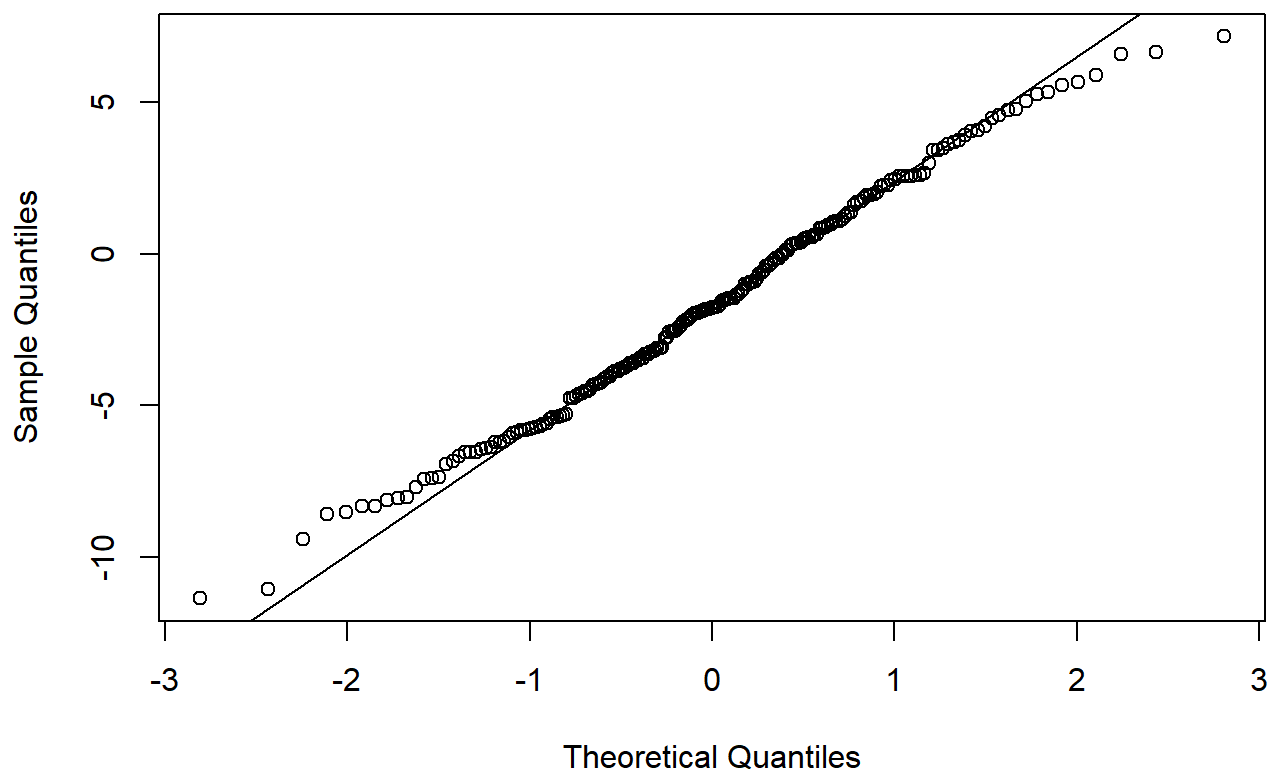
```
y <- rnorm(200, mean=-2, sd=4)
hist(y)
```

## Histogram of y



```
qqnorm(y)  
qqline(y)
```

## Normal Q-Q Plot



## 4.2. The t Distribution

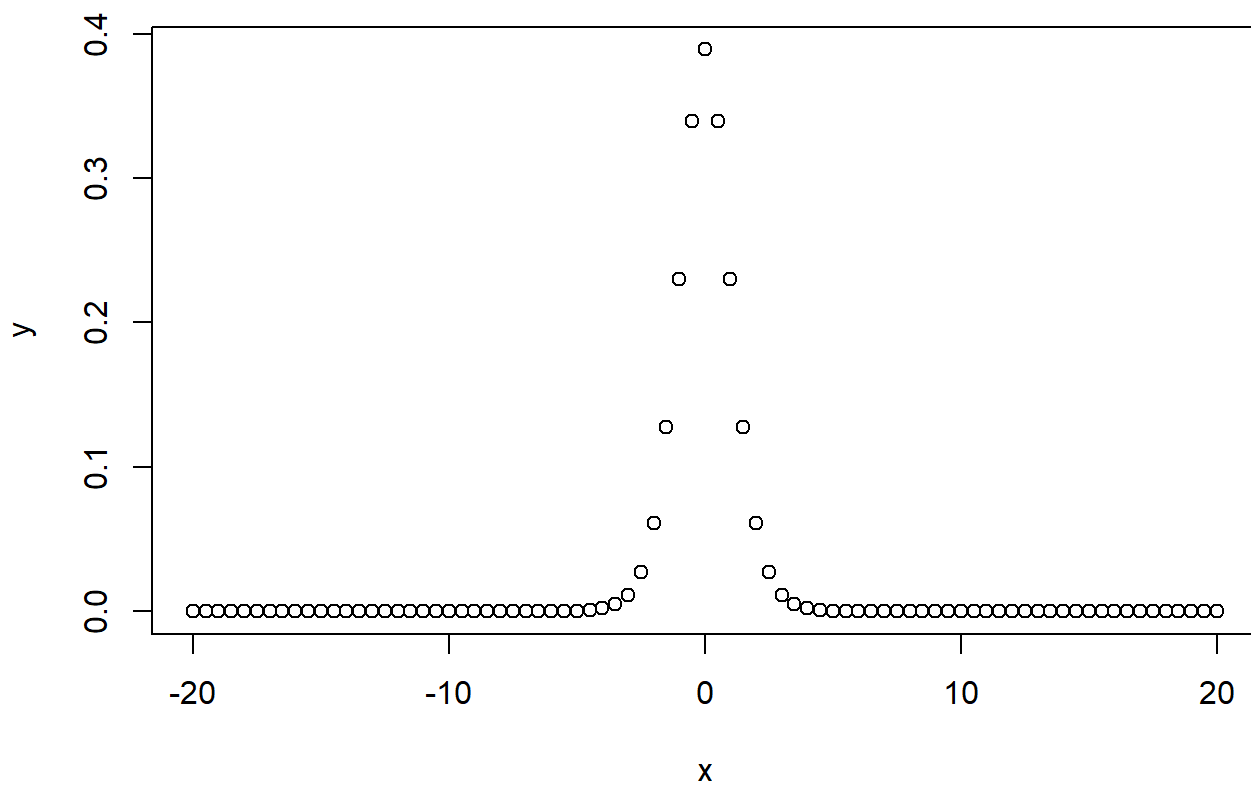
There are four functions that can be used to generate the values associated with the t distribution. You can get a full list of them and their options using the help command:

```
> help(TDist)
```

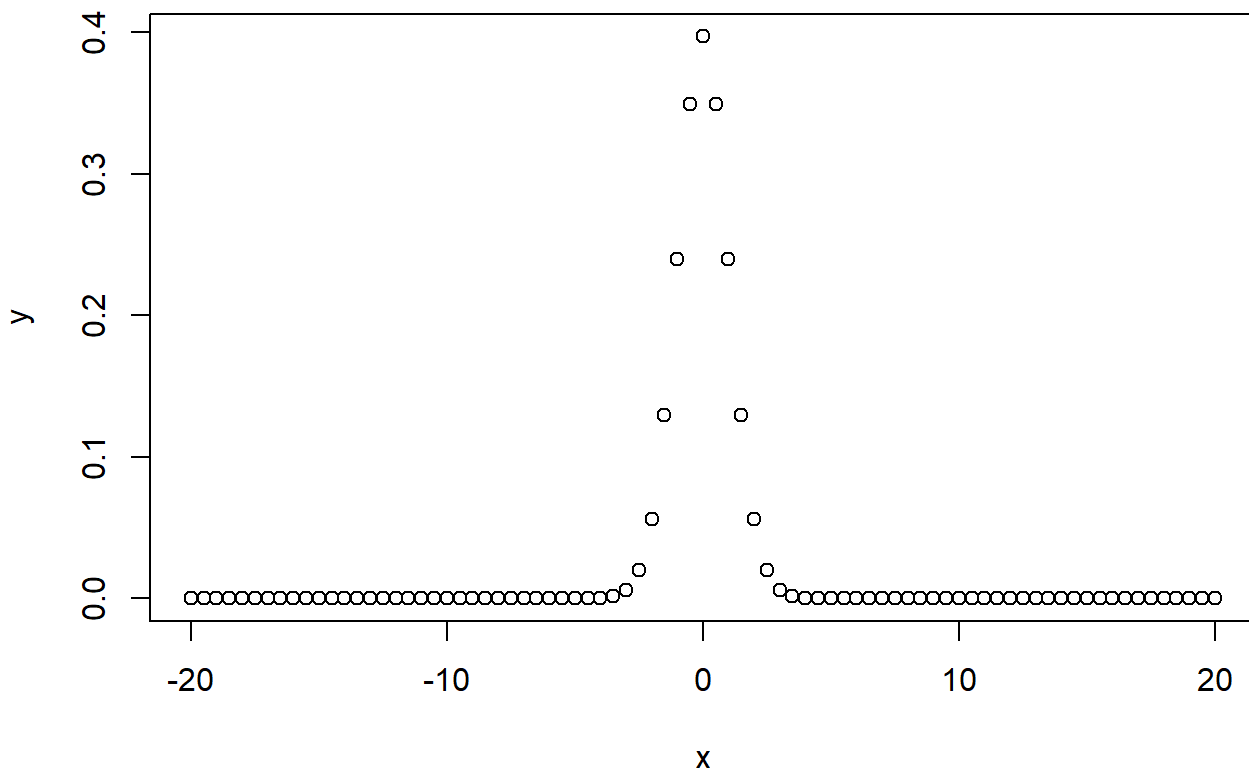
These commands work just like the commands for the normal distribution. One difference is that the commands assume that the values are normalized to mean zero and standard deviation one, so you have to use a little algebra to use these functions in practice. The other difference is that you have to specify the number of degrees of freedom. The commands follow the same kind of naming convention, and the names of the commands are *dt*, *pt*, *qt*, and *rt*.

A few examples are given below to show how to use the different commands. First we have the distribution function, *dt*:

```
x <- seq(-20,20,by=.5)
y <- dt(x,df=10)
plot(x,y)
```



```
y <- dt(x,df=50)
plot(x,y)
```



Next we have the cumulative probability distribution function:

```
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
```

```
x = c(-3,-4,-2,-1)
pt((mean(x)-2)/sd(x),df=20)
```

```
[1] 0.001165548
```



```
pt((mean(x)-2)/sd(x),df=40)
```

```
[1] 0.000603064
```

Next we have the inverse cumulative probability distribution function:

```
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
```

```
v <- c(0.005,.025,.05)  
qt(v,df=253)
```

```
[1] -2.595401 -1.969385 -1.650899
```

```
qt(v,df=25)
```

```
[1] -2.787436 -2.059539 -1.708141
```

Finally random numbers can be generated according to the t distribution:

```
rt(3,df=10)
```

```
[1] 0.05929613 -1.88332462 -1.21287707
```

```
rt(3,df=20)
```

```
[1] 1.7205644 -1.9506461 0.4829968
```

```
rt(3,df=20)
```

```
[1] -0.4146348 -0.6216872 -0.8946080
```

## 4.3. The Binomial Distribution

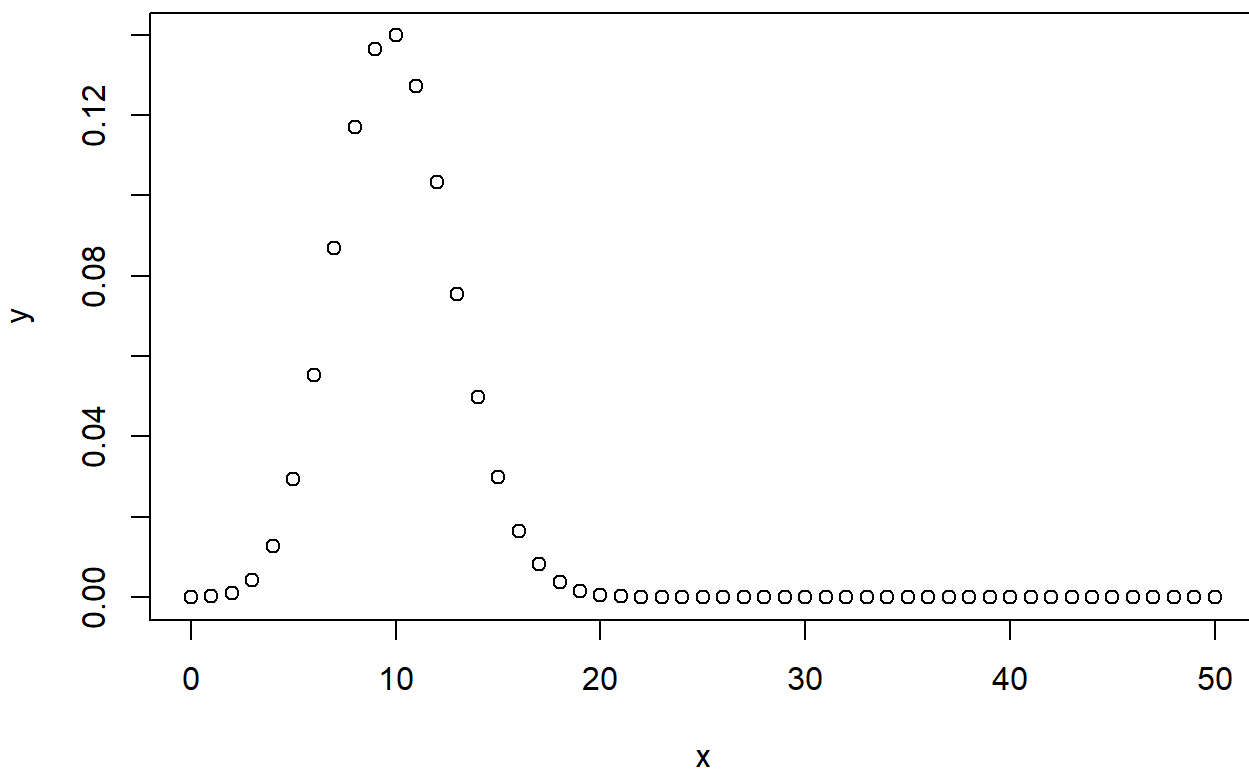
There are four functions that can be used to generate the values associated with the binomial distribution. You can get a full list of them and their options using the help command:

```
> help(Binomial)
```

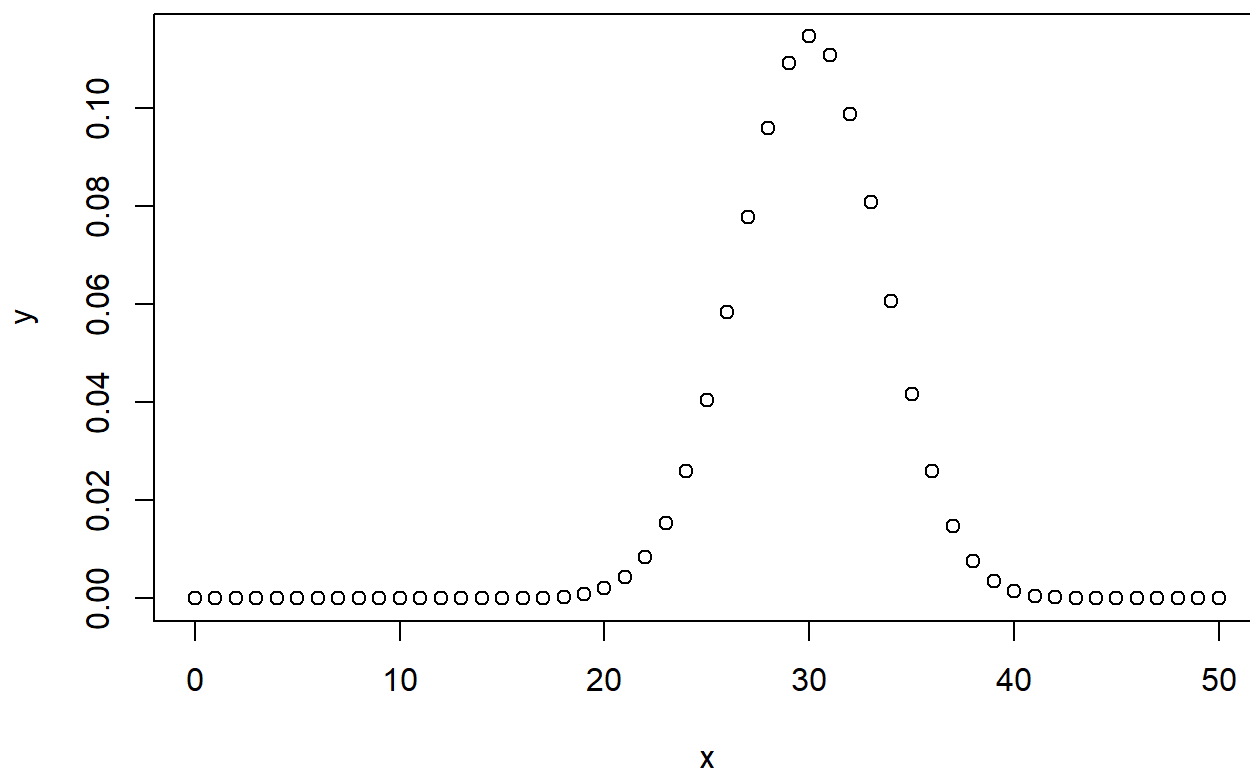
These commands work just like the commands for the normal distribution. The binomial distribution requires two extra parameters, the number of trials and the probability of success for a single trial. The commands follow the same kind of naming convention, and the names of the commands are `dbinom`, `pbinom`, `qbinom`, and `rbinom`.

A few examples are given below to show how to use the different commands. First we have the distribution function, *dbinom*:

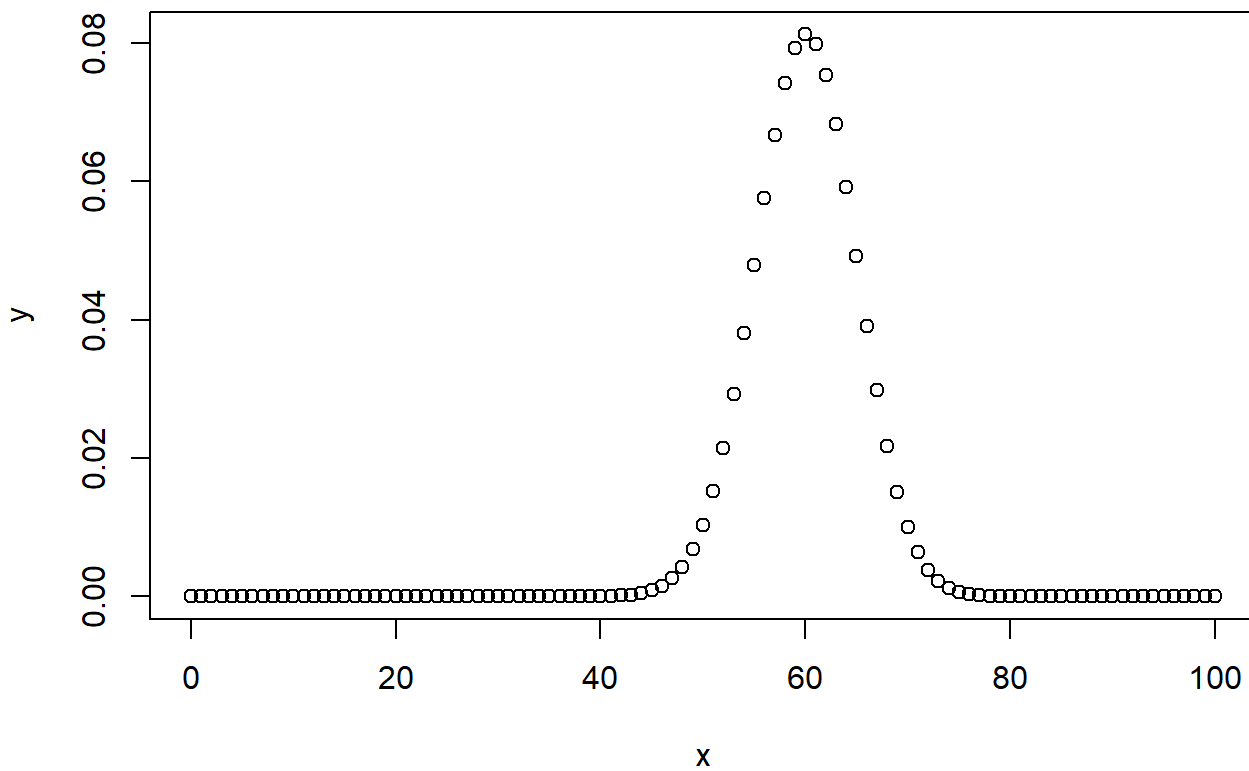
```
x <- seq(0,50,by=1)
y <- dbinom(x,50,0.2)
plot(x,y)
```



```
y <- dbinom(x,50,0.6)
plot(x,y)
```



```
x <- seq(0,100,by=1)
y <- dbinom(x,100,0.6)
plot(x,y)
```



Next we have the cumulative probability distribution function:

```
pbinom(24,50,0.5)
```

```
[1] 0.4438624
```

```
pbinom(25,50,0.5)
```

```
[1] 0.5561376
```

```
pbinom(25,51,0.5)
```

```
[1] 0.5
```

```
pbinom(26,51,0.5)
```

```
[1] 0.610116
```

```
pbinom(25,50,0.5)
```

```
[1] 0.5561376
```

```
pbinom(25,50,0.25)
```

```
[1] 0.999962
```

```
pbinom(25,500,0.25)
```

```
[1] 4.955658e-33
```

Next we have the inverse cumulative probability distribution function:

```
qbinom(0.5,51,1/2)
```

```
[1] 26
```

```
qbinom(0.25,51,1/2)
```

```
[1] 23
```

```
pbinom(23,51,1/2)
```

```
[1] 0.2879247
```

```
pbinom(22,51,1/2)
```

```
[1] 0.200531
```

Finally random numbers can be generated according to the binomial distribution:

```
rbinom(5,100,.2)
```

```
[1] 18 15 16 21 22
```

```
rbinom(5,100,.7)
```

```
[1] 75 62 71 67 78
```

## 4.4. The Chi-Squared Distribution

There are four functions that can be used to generate the values associated with the Chi-Squared distribution. You can get a full list of them and their options using the help command:

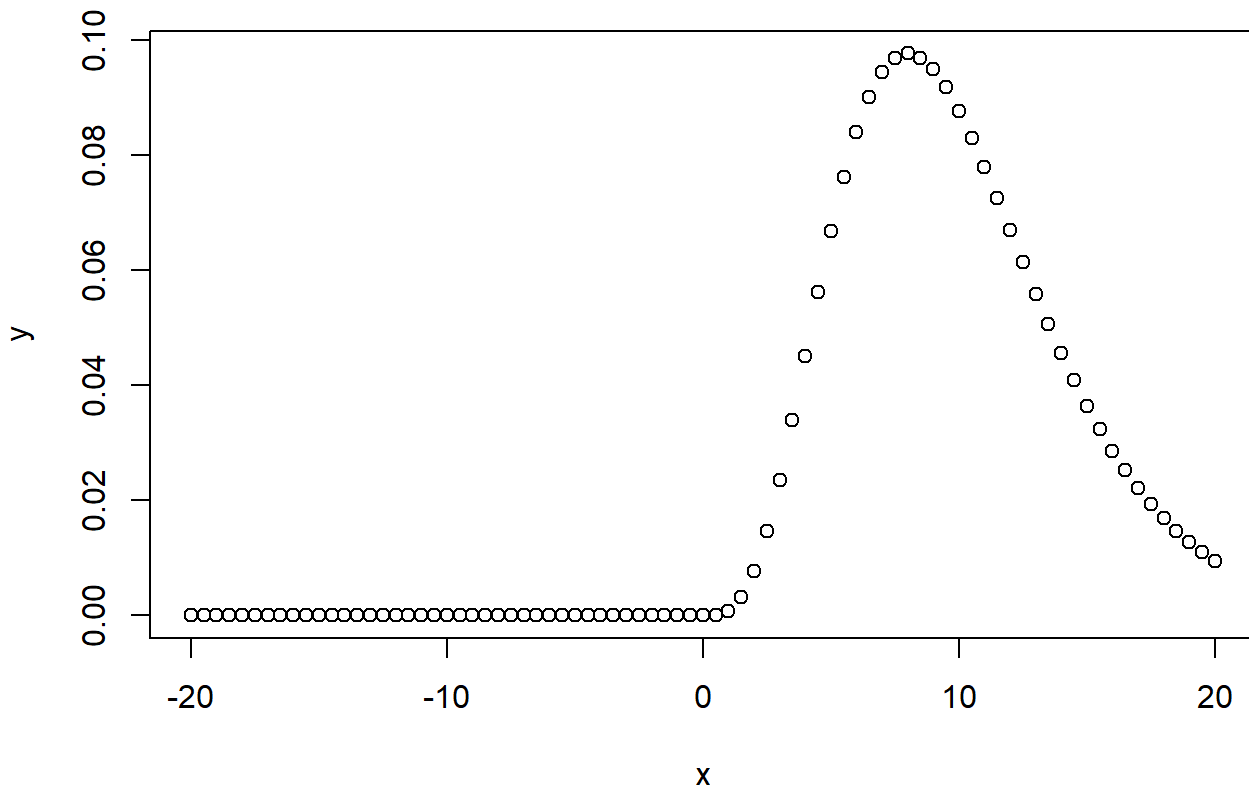
```
> help(Chisquare)
```

These commands work just like the commands for the normal distribution. The first difference is that it is assumed that you have normalized the value so no mean can be specified. The other difference is that you

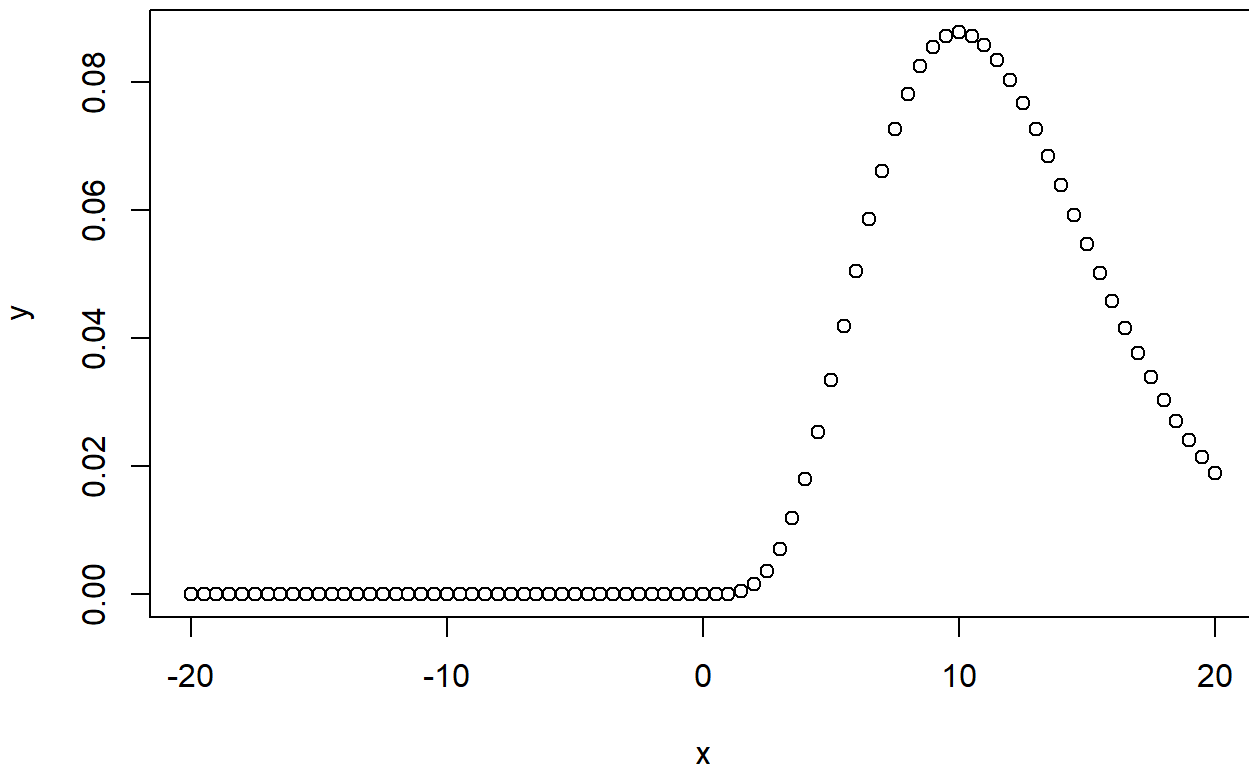
have to specify the number of degrees of freedom. The commands follow the same kind of naming convention, and the names of the commands are *dchisq*, *pchisq*, *qchisq*, and *rchisq*.

A few examples are given below to show how to use the different commands. First we have the distribution function, *dchisq*:

```
x <- seq(-20, 20, by = 0.5)
y <- dchisq(x, df = 10)
plot(x, y)
```



```
y <- dchisq(x, df = 12)
plot(x, y)
```



Next we have the cumulative probability distribution function:

```
pchisq(2, df = 10)
```

```
[1] 0.003659847
```

```
pchisq(3, df = 10)
```

```
[1] 0.01857594
```

```
1 - pchisq(3, df = 10)
```

```
[1] 0.9814241
```

```
pchisq(3, df = 20)
```

```
[1] 4.097501e-06
```

```
x <- c(2, 4, 5, 6)
pchisq(x, df = 20)
```

```
[1] 1.114255e-07 4.649808e-05 2.773521e-04 1.102488e-03
```

Next we have the inverse cumulative probability distribution function:

```
qchisq(0.05, df = 10)
```

```
[1] 3.940299
```

```
qchisq(0.95, df = 10)
```

```
[1] 18.30704
```

```
qchisq(0.05, df = 20)
```

```
[1] 10.85081
```

```
qchisq(0.95, df = 20)
```

```
[1] 31.41043
```

```
v <- c(0.005, 0.025, 0.05)  
qchisq(v, df = 253)
```

```
[1] 198.8161 210.8355 217.1713
```

```
qchisq(v, df = 25)
```

```
[1] 10.51965 13.11972 14.61141
```

Finally random numbers can be generated according to the Chi-Squared distribution:

```
rchisq(3, df = 10)
```

```
[1] 20.984843 5.793830 9.350879
```

```
rchisq(3, df = 20)
```

```
[1] 24.17747 26.82920 13.33649
```

```
rchisq(3, df = 20)
```

```
[1] 20.02197 22.49461 16.08661
```