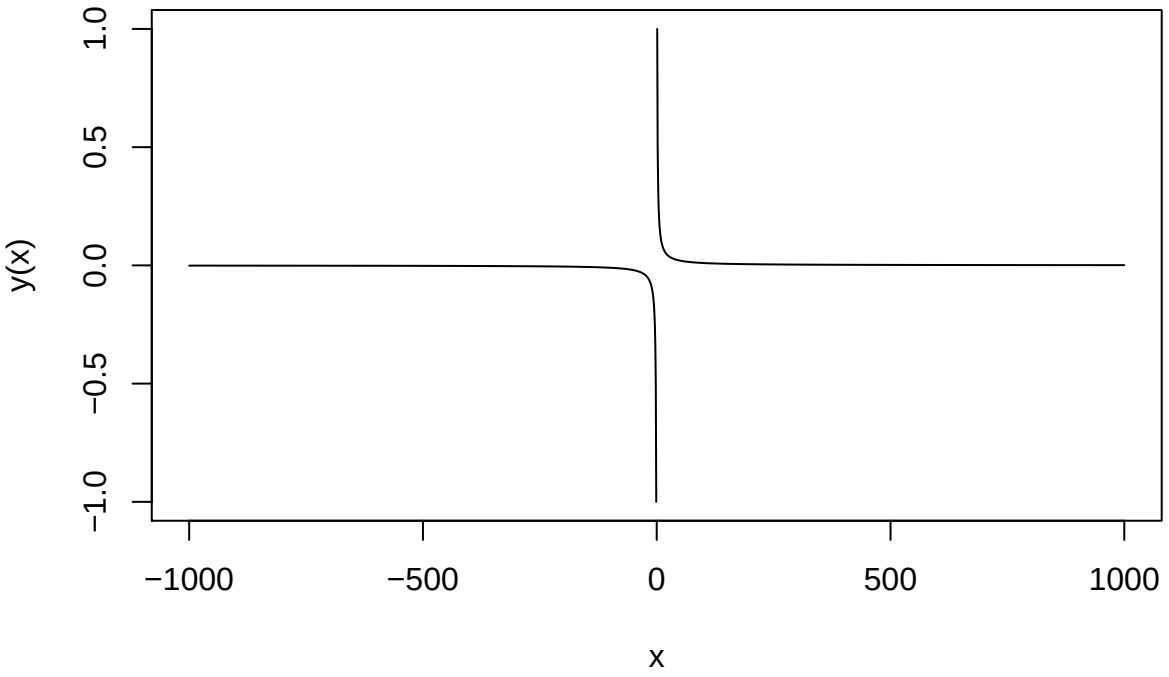
Title

William Murrah

10/06/2014

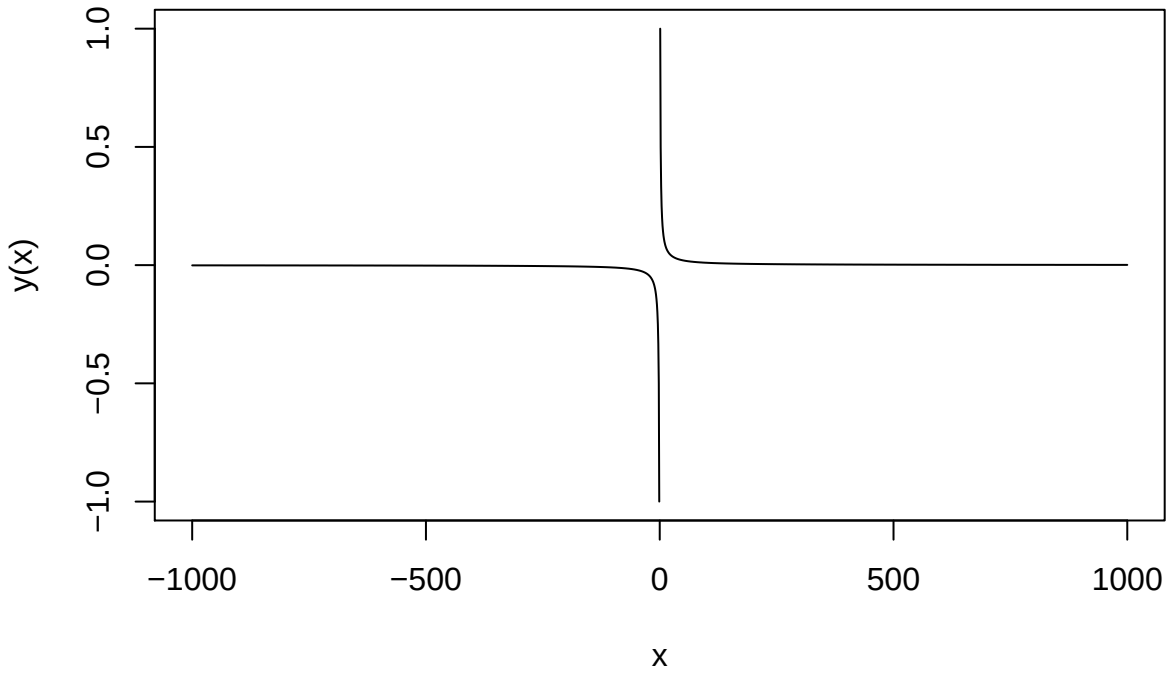
```
library(knitr)
```

```
library(psych)
library(xtable)
options(xtable.comment = FALSE)
x = -1000:1000
y = function(x) 1/x
plot(x,y(x), type='l')
```

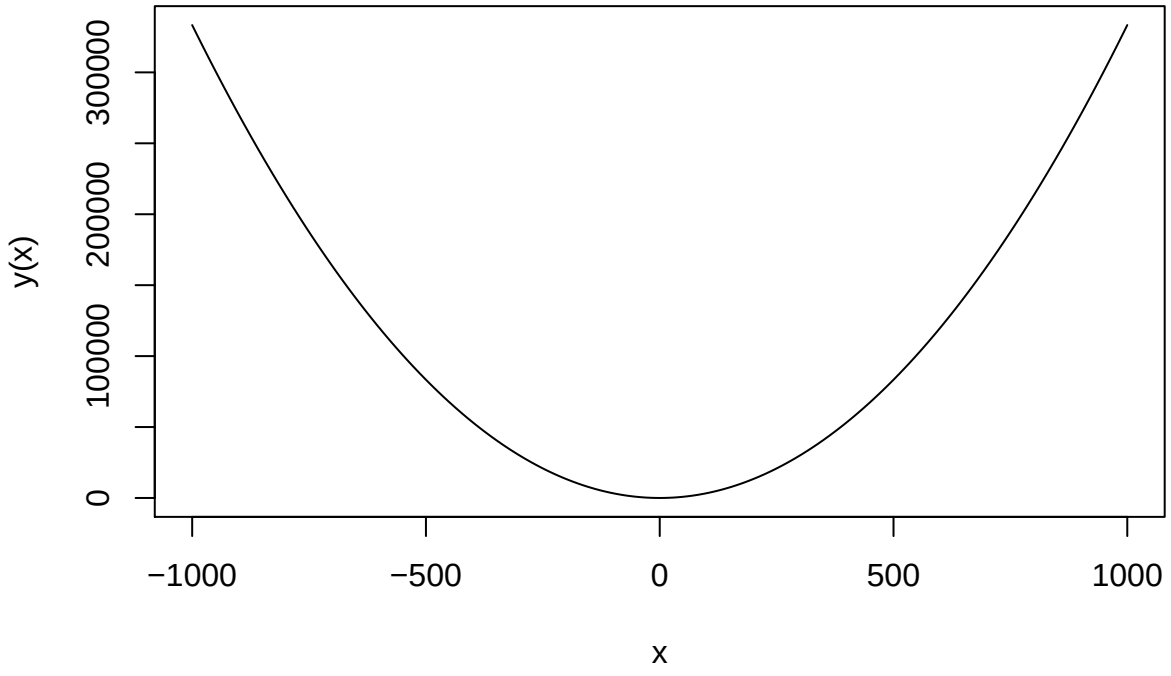


$$\lim_{x \to \infty}, \frac{1}{x}$$

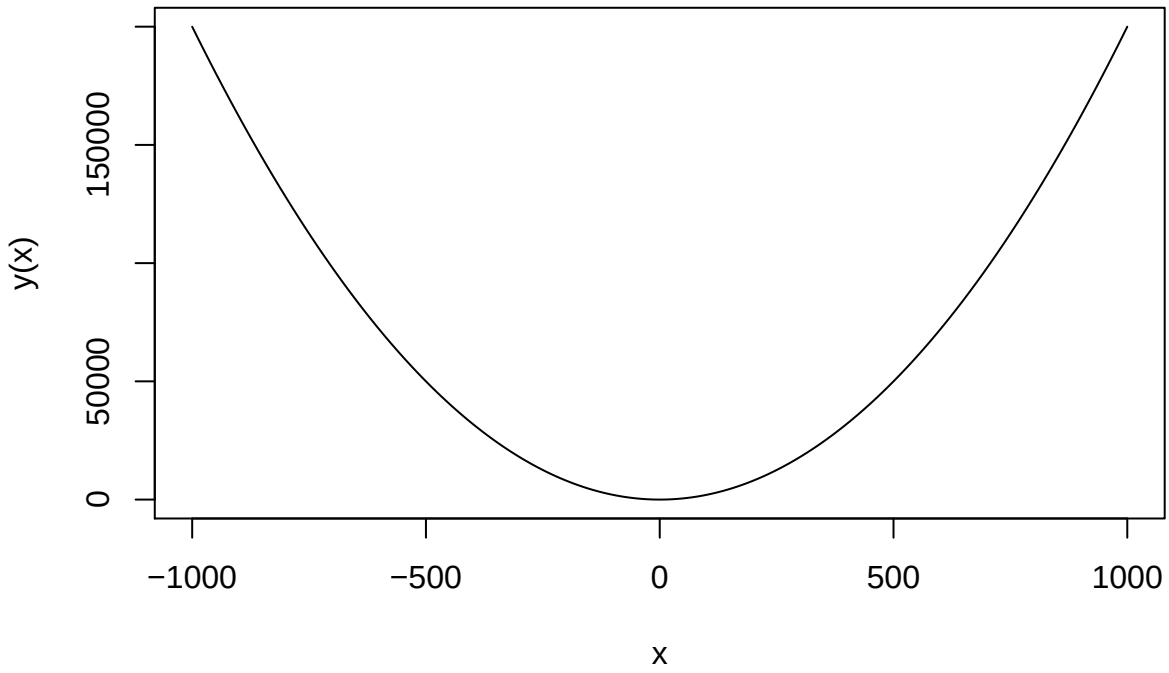
```
x = 1000:-1000
y = function(x) 1/x
plot(x,y(x), type='l')
```




```
x = -1000:1000
y = function (x) (x^2 + 5)/3
plot(x, y(x), type='l')
```

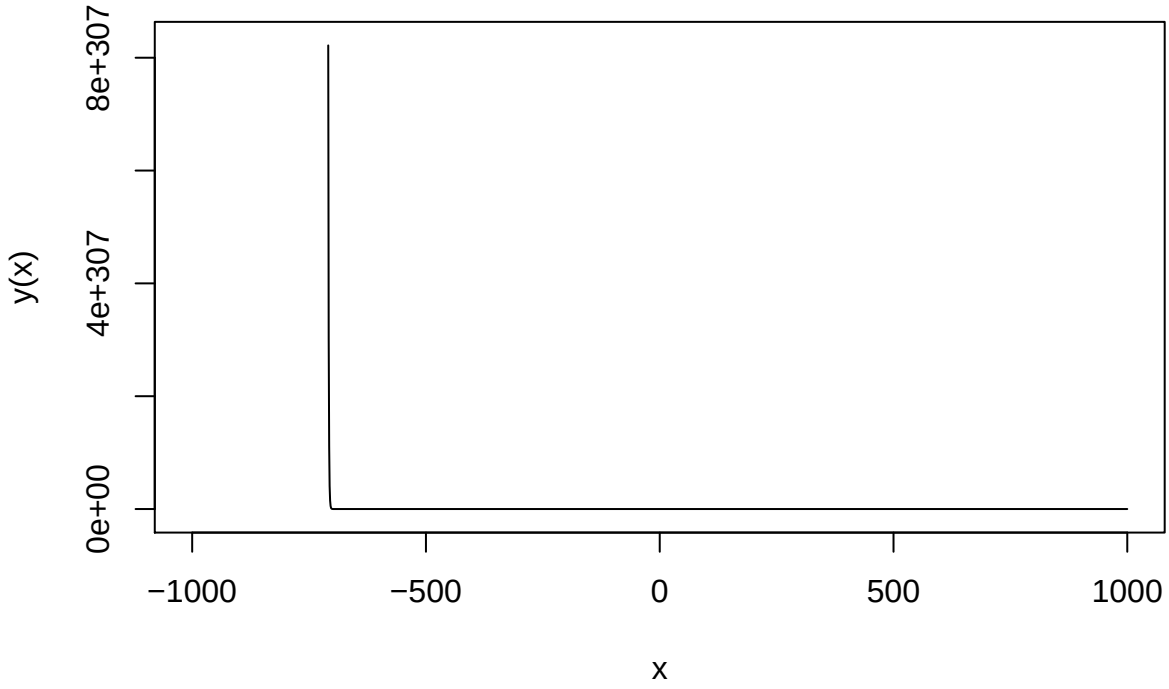



```
x = 1000:-1000
y = function (x) (x^2 - 1)/5
plot(x, y(x), type='l')
```




```
x = -1000:1000
y = function (x) exp(-x)
plot(x, y(x), type='l')
```

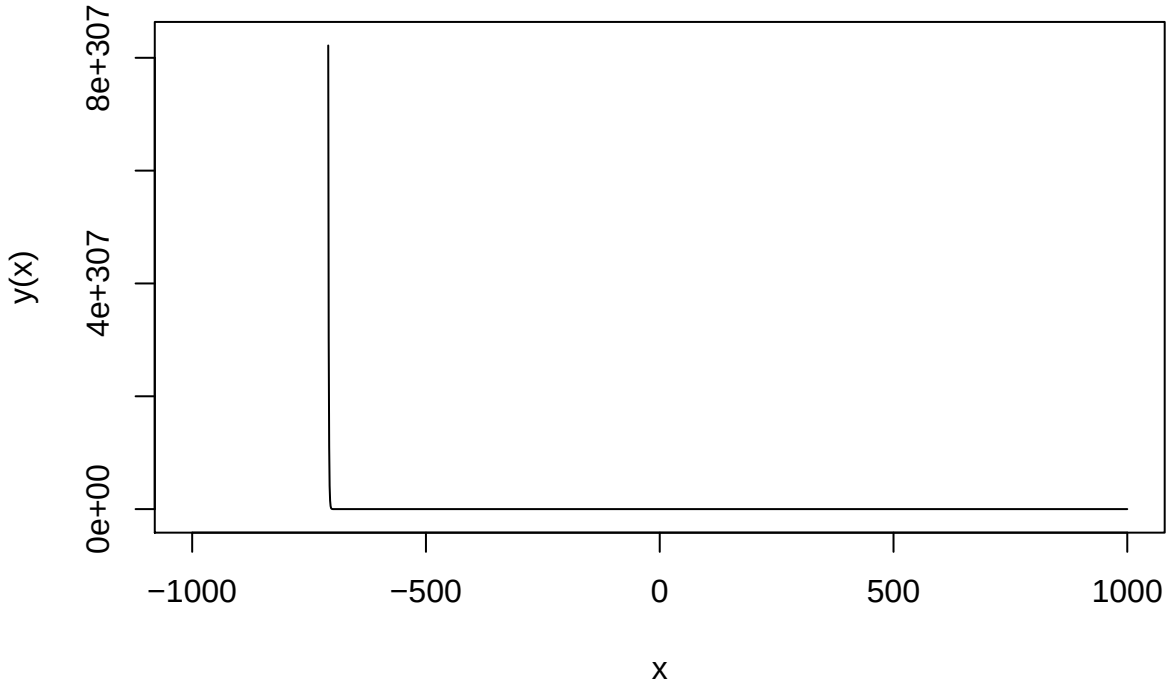
```
## Warning in plot.window(...): Internal(pretty()): very large range..
## corrected
```



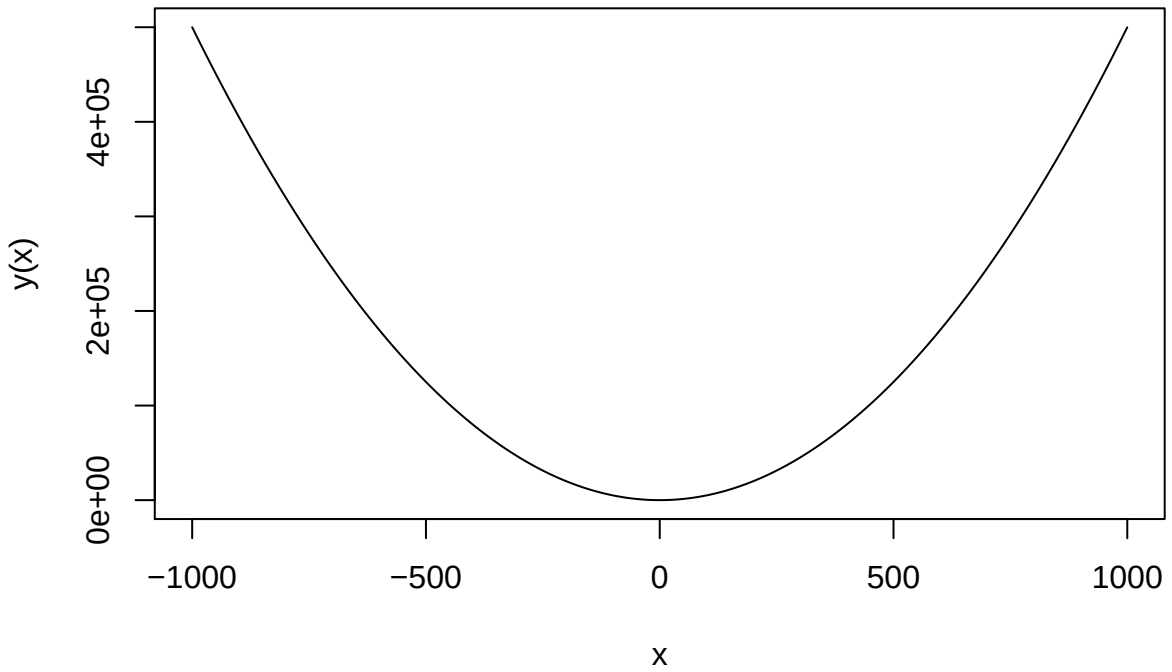

```
x = 1000:-1000
y = function (x) exp(-x)
plot(x, y(x), type='l')
```

```
## Warning in plot.window(...): Internal(pretty()): very large range..
```

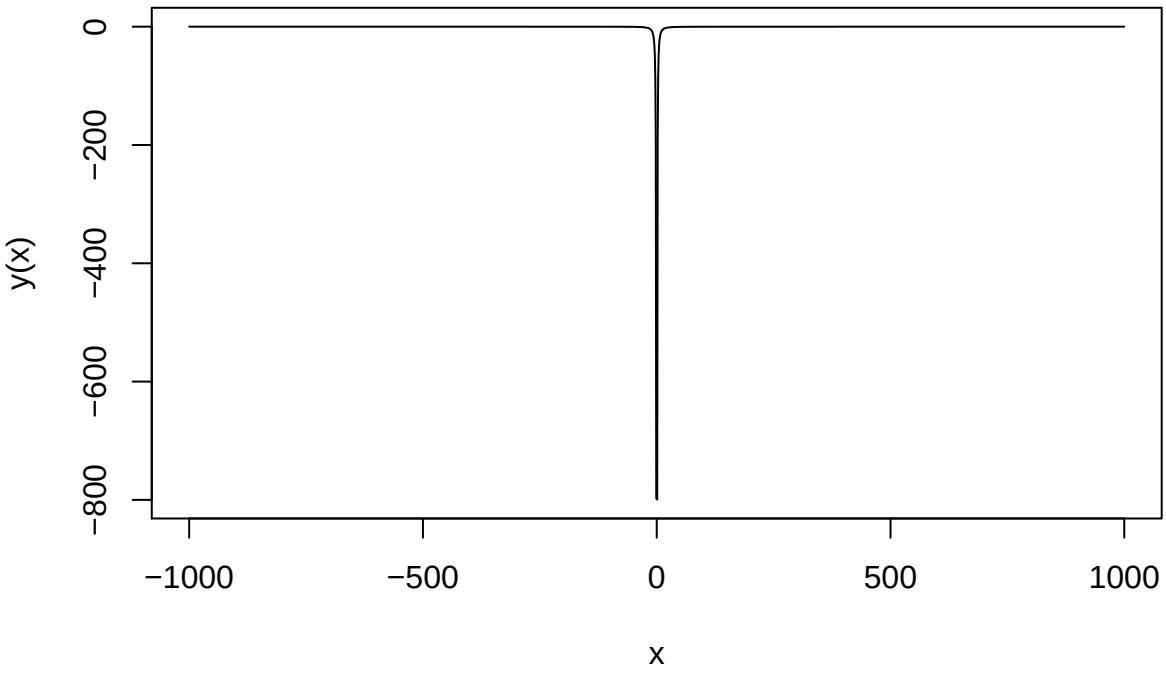
```
## corrected
```




```
x = -1000:1000
y = function (x) x/2*x
plot(x, y(x), type='l')
```




```
x = -1000:1000
mu = 0
sigma = 1:2001
y = function (x) 1/(sigma*sqrt(2*pi)*(-(x - 0)^2)/(2*sigma^2))
plot(x, y(x), type='l')
```



Matix Algebra

```
A = matrix(c(3,-1,2,3), nrow=2)
A
```

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

```
B = matrix(c(3,4,10,1,3,4,-1,5,1), nrow=3)
B
```

```
##      [,1] [,2] [,3]
## [1,]    3    1   -1
## [2,]    4    3    5
## [3,]   10    4    1
```

```
C = matrix(c(1,7,4),nrow=1)
C
```

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

```
D = matrix(c(2,9,3), nrow=3)
D
```

```
##      [,1]
## [1,]    2
## [2,]    9
## [3,]    3
```

```
E = matrix(c(4,7,6, 1,1,8, 3,8,7), nrow=3)
E
```

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

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

```
BE <- B + E
print(xtable(BE, digits=0, caption="$B + E$"))
```

	1	2	3
1	7	2	2
2	11	4	13
3	16	12	8

Table 1: $B + E$

```
E - B
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    4
## [2,]    3   -2    3
## [3,]   -4    4    6
```

```
# A%*%B non-conformable
```

```
C %*% D
```

```
##      [,1]
## [1,]   77
```

```
D %*% C
```

```
##      [,1] [,2] [,3]
## [1,]    2   14    8
## [2,]    9   63   36
## [3,]    3   21   12
```

```
# D %*% E non-conformable
```

```
t(E)
```

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

```
det(A)
```

```
## [1] 11
```

```
det(B)
```

```
## [1] 9
```

```
solve(E)
```

```
##             [,1]       [,2]        [,3]
## [1,]  0.72151899 -0.2151899 -0.06329114
## [2,]  0.01265823 -0.1265823  0.13924051
## [3,] -0.63291139  0.3291139  0.03797468
```

```
tr(B)
```

```
## [1] 7
```

Chapter 2

1.

2. Probability of finding 3 heads in a row:

```
.5 * .5 * .5
```

```
## [1] 0.125
```

3. Probability of finding 3 heads in a row with bend coin (p = .7)

```
.7 * .7 * .7
```

```
## [1] 0.343
```

```
# or

.7^3
```

```
## [1] 0.343
```

4. 3 heads or 3 tails in a row

```
.5^3 + .5^3
```

```
## [1] 0.25
```

5. 3 heads and one tail (order irrelevant):

$$\binom{n}{k} p^k (1 - p^{n-k})$$

```
choose(4,3)*(.5^3 * .5^1)
```

```
## [1] 0.25
```

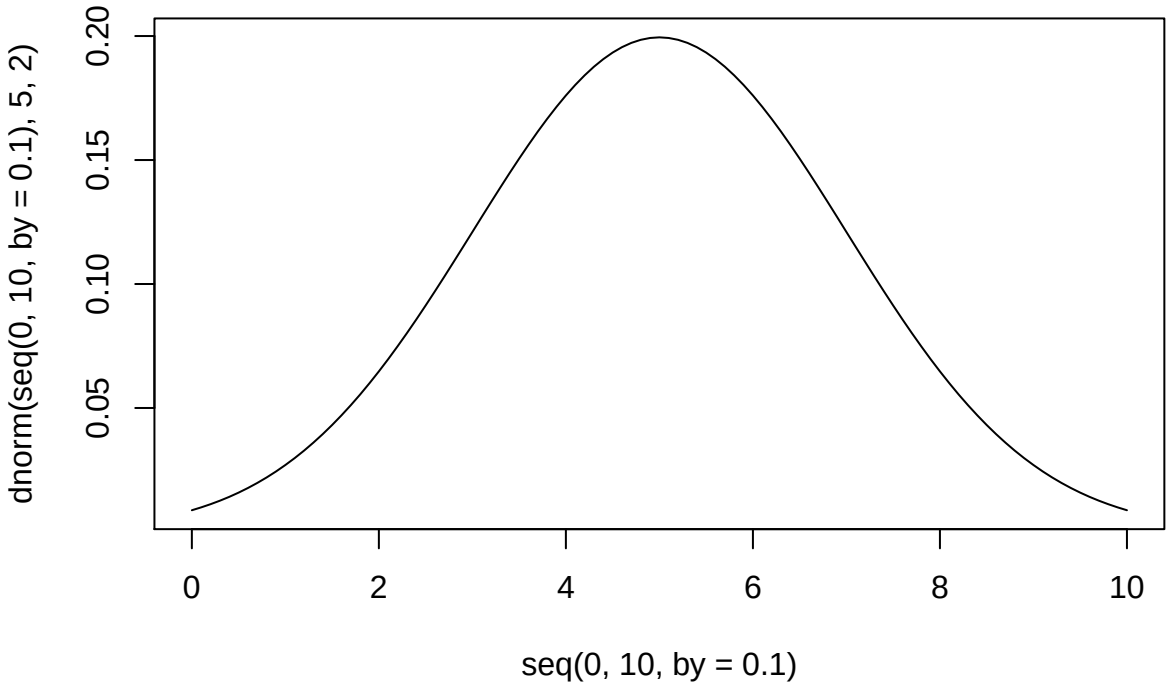
6. normal approximation of binomial:

$$\mu_x = np, \quad \sigma_x = \sqrt{np(1-p)}$$

```
mu = 200*.5
sigma.x = sqrt(200*.5*(1 - .5))
z = (130 - mu)/ sigma.x
```

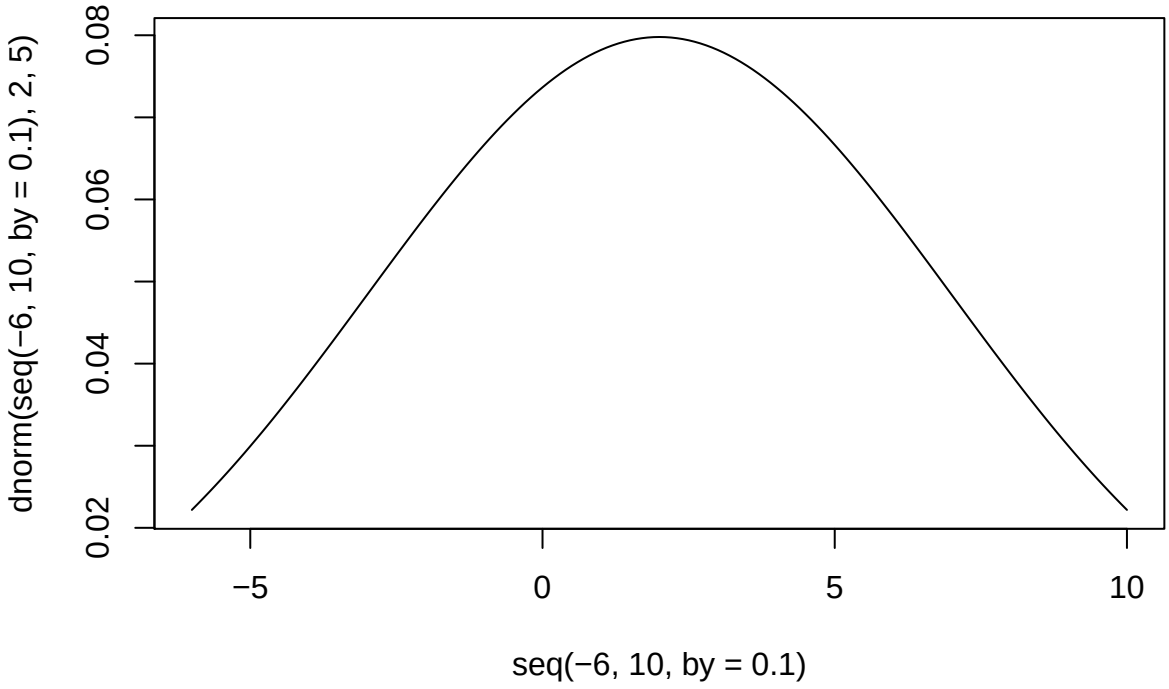
7. Plot a normal distribution with parameters $\mu = 5$ and $\sigma = 2$.

```
plot(seq(0, 10, by=.1), dnorm(seq(0, 10, by=.1), 5,2), type='l')
```



8. Plot a normal distribution with parameters $\mu = 2$ and $\sigma = 5$.

```
plot(seq(-6,10, by=.1), dnorm(seq(-6,10,by=.1), 2,5), type='l')
```



8. Plot a normal distribution with parameters $\mu = 2$ and $\sigma = 5$.
9. Plot the t(0, 1, df = 1) and t(0, 1, df = 10) distributions. Note: Γ is a function. The function is: $\Gamma(n) = e^{-u}u^{n-1}du$. For integers, $\Gamma(n) = 0(n-1)$ Thus, $\Gamma(4) = (4-1)! = 6$. However, when the argument to the function is not an integer, this formula will not work. Instead, it is easier to use a software package to compute the function value for you.

```
# This program adds two numbers
num1 = 1.5
num2 = 6.3
# Add two numbers
sum = float(num1) + float(num2)
# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

```
## The sum of 1.5 and 6.3 is 7.8
```

Does `engine = 'c'` work?

```
void square(double *x) {
  *x = *x * *x;
}
```

Test the `square()` function:

```
.C('square', 9)
```

```
## [[1]]
## [1] 81
```

```
.C('square', 123)
```

```
## [[1]]
## [1] 15129
```