The Gaussian distribution with known mean and variance

the Gaussian distribution

This short handout takes you through the basics of the Gaussian distribution as discussed in lectures, although you should be aware that we assume both the mean and variance are known precisely.

Remember that you can type `?rnorm` at the R prompt to get help on that command.

Random sampling from the Gaussian

We can take random samples from the Gaussian distribution using the `rnorm()` function:

```
rnorm(10)
```

```
##  [1]  0.50064983  0.68804774  0.83051333  0.50969342 -0.21630050 -0.67663638
##  [7]  0.46784358 -1.06966620  0.07603018  0.96173255
```

```
rnorm(10)
```

```
##  [1] -1.1505108 -0.8317241 -0.2085885  1.0272353 -0.6841123 -0.7504832
##  [7] -1.3438092  0.5648671  0.2560330 -1.0852492
```

(see how I do it twice to remind myself that the output changes each time it is called).

We can change the mean and/or the standard deviation:

```
rnorm(10,mean=100)
```

```
##  [1] 102.11386  99.59870 100.59846  99.60177 100.99567 100.06709 101.79206
##  [8]  98.76423  99.52069 100.63590
```

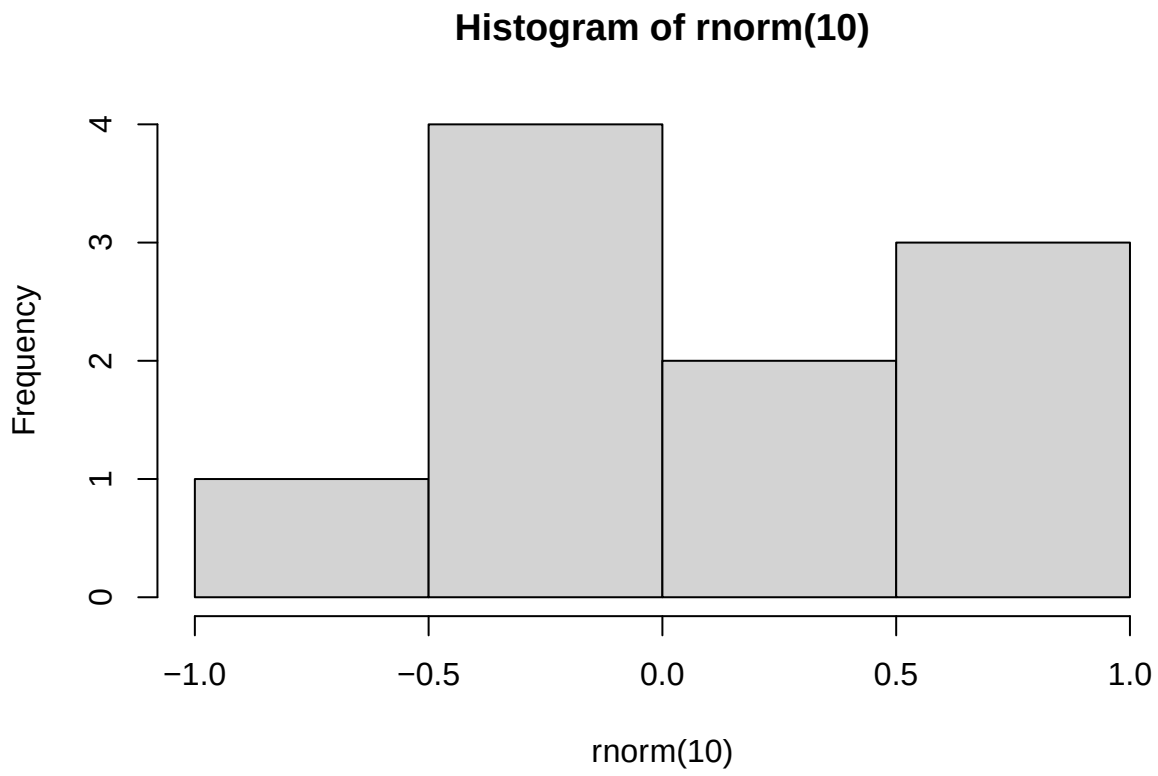
```
rnorm(10,mean=100)
```

```
##  [1] 100.03672  98.83925 101.86798  99.60652  99.93980  98.38865 100.83729
##  [8]  99.50978 100.04302  99.76022
```

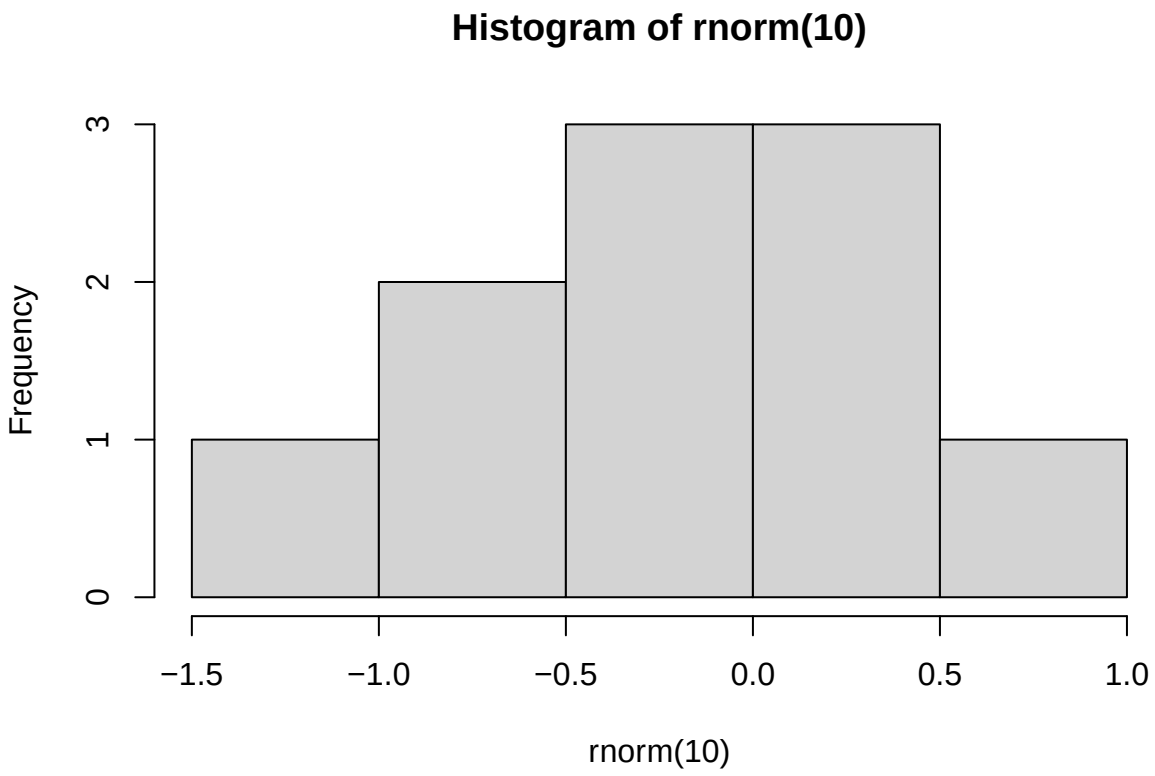
See how the values are clustered around 100.

It is difficult to get a feel for these numbers so we can use R to plot various types of visualization:

```
hist(rnorm(10))
```

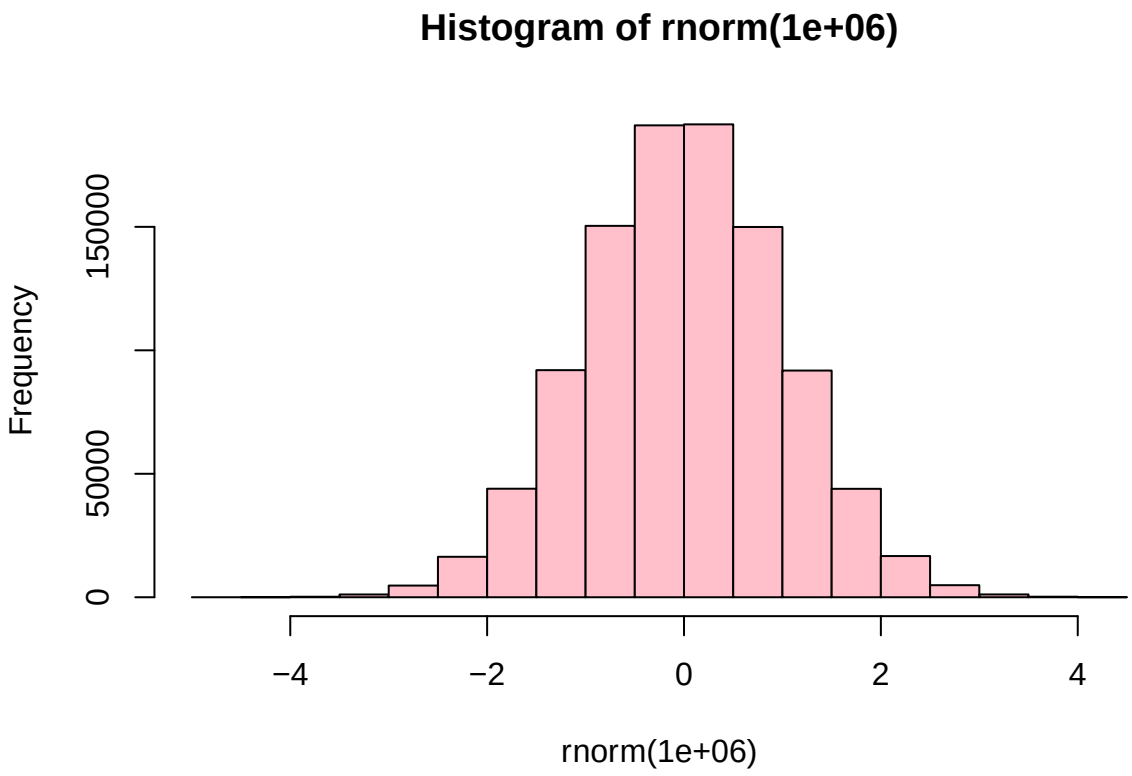



```
hist(rnorm(10))
```



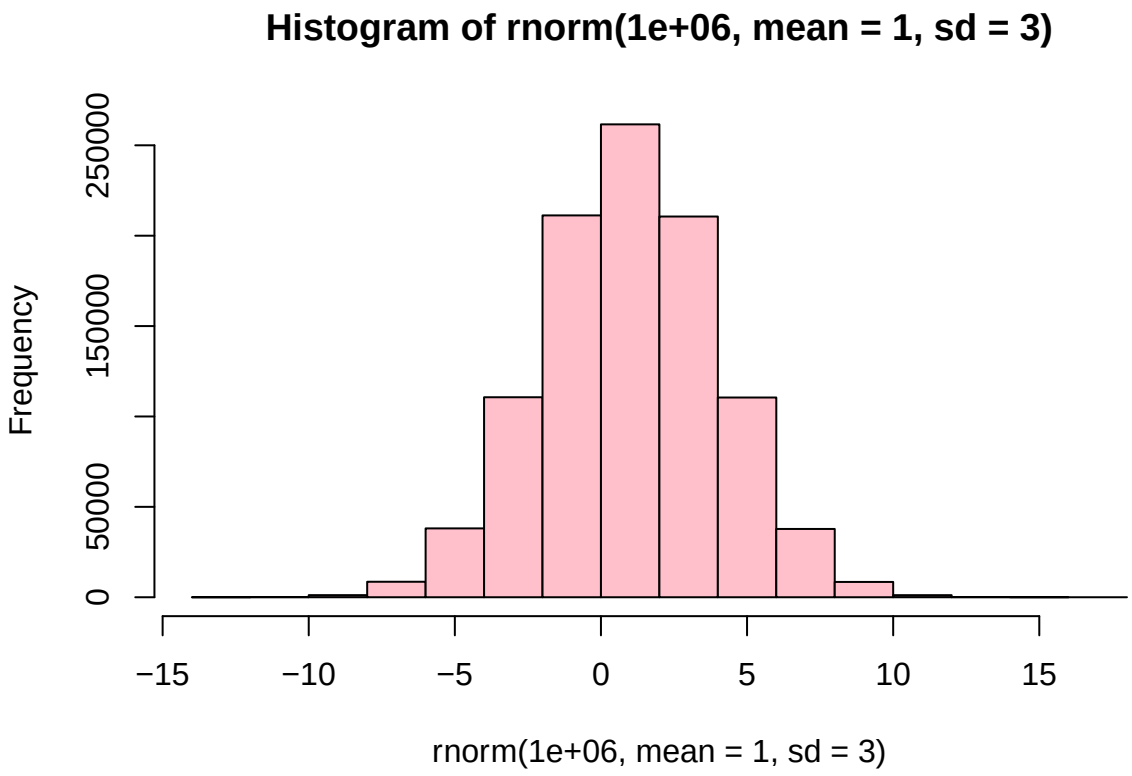
(these are bad visualizations because of the small number of observations). But this is easy to rectify; we can use a higher number of observations:

```
hist(rnorm(1e6),col='pink')
```



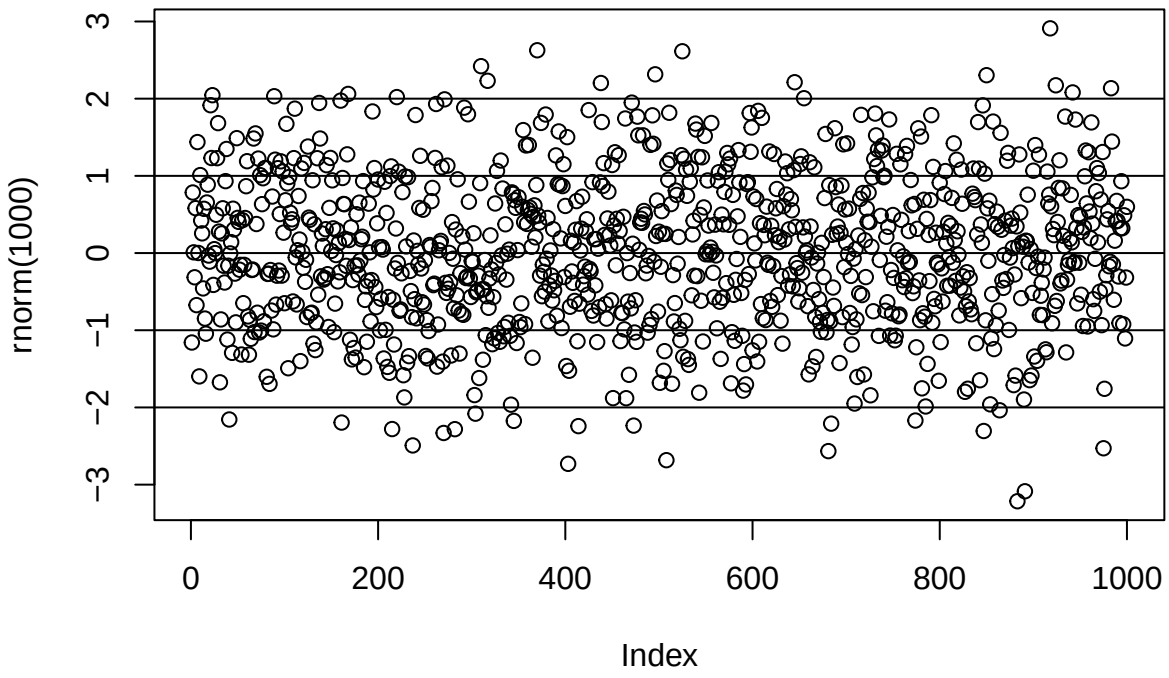
We can change the mean and standard deviation using the appropriate argument:

```
hist(rnorm(1e6,mean=1,sd=3),col='pink')
```



But histograms are not the only type of visualization:

```
plot(rnorm(1000))
abline(h=c(-2:2))
```



This shows the actual values of the observations. If the R idiom uses unfamiliar commands, remember that you can type `?abline` (or whatever) to get online help.

Verify that the two visualizations are consistent with each other.

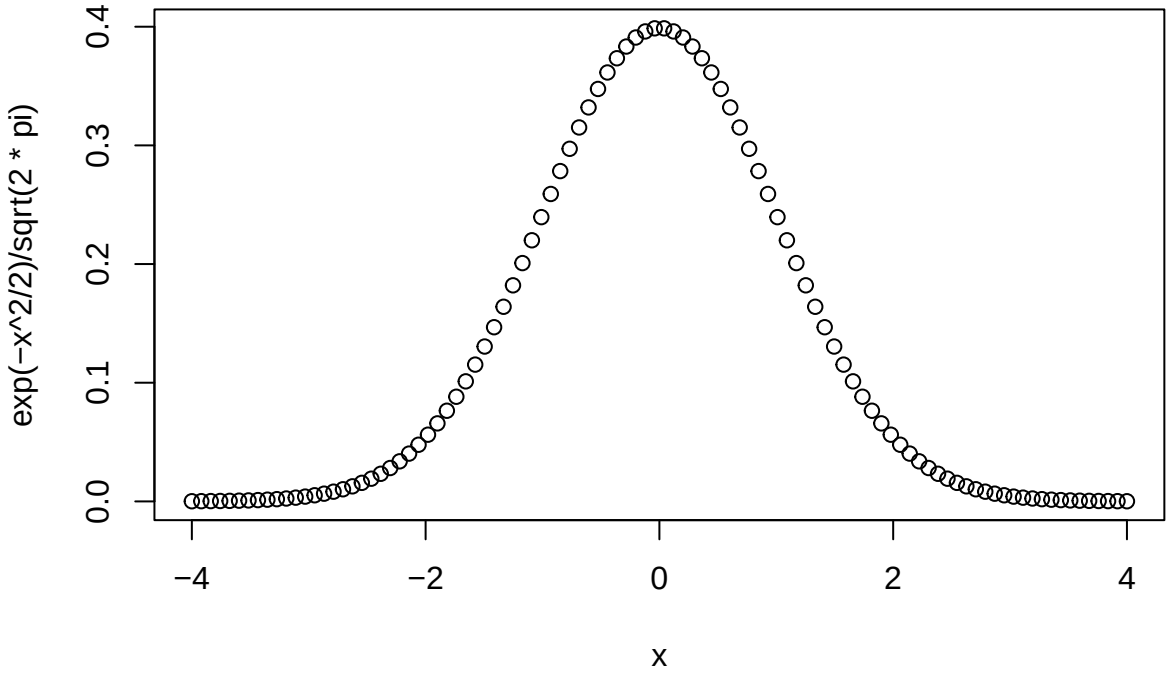
The density function of the Gaussian

The probability density function of the standard Gaussian is given by

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{x^2}{2}\right]$$

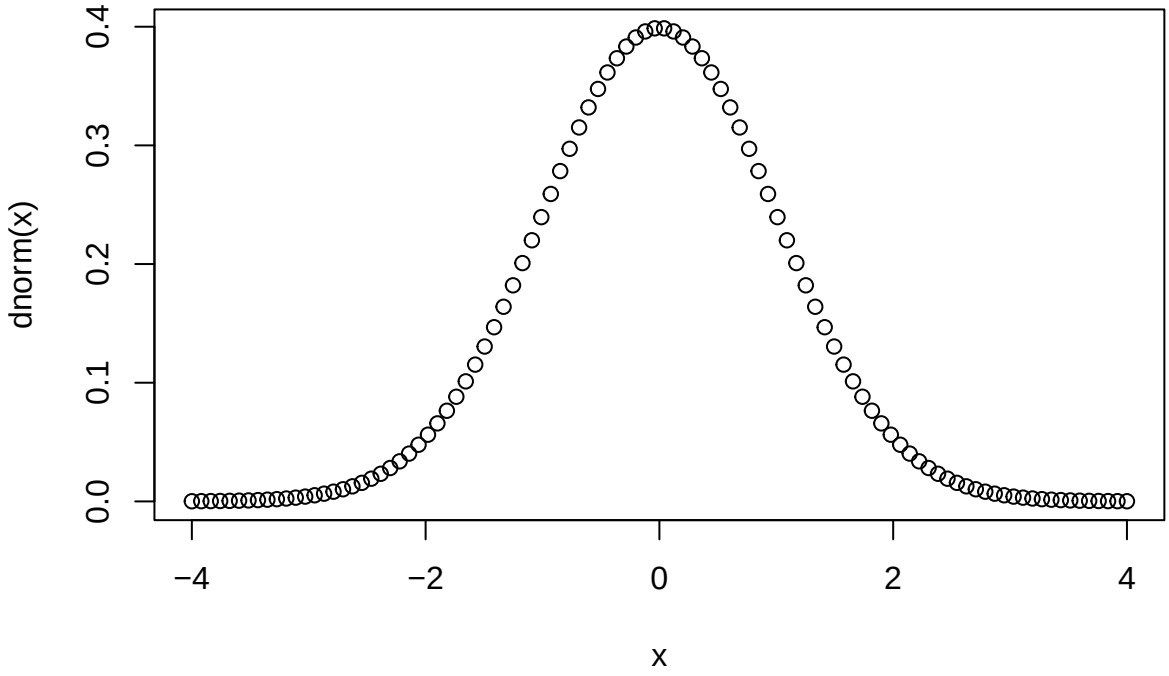
This has mean 0 and standard deviation 1. In R we can visualise this as follows:

```
x <- seq(from=-4,to=4,len=100)
plot(x,exp(-x^2/2)/sqrt(2*pi))
```



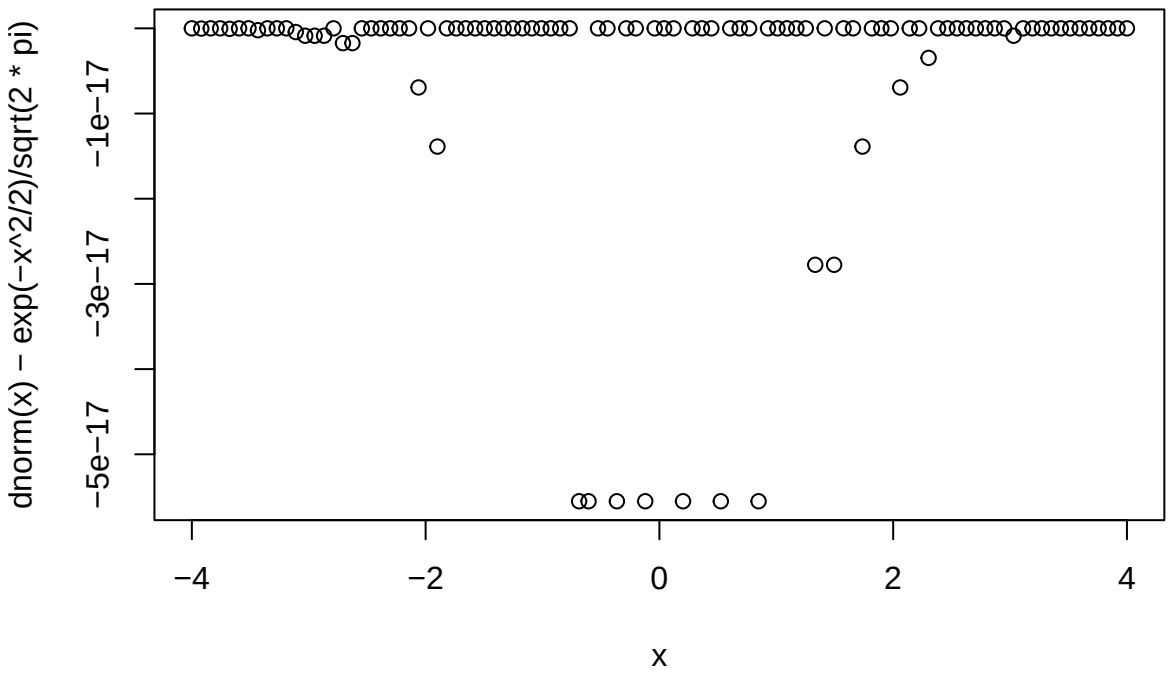
Make sure you understand this before moving on. R provides more convenient ways of evaluating this function, specifically the `dnorm()` function:

```
plot(x,dnorm(x))
```



which _looks_the same. But, it is important to test the similarity as rigorously as is possible:

```
plot(x,dnorm(x) - exp(-x^2/2)/sqrt(2*pi))
```



(see how we are plotting the *difference* between the two methods on the vertical axis). If the two methods give different answers, which one is more likely to be accurate? Which one is faster?

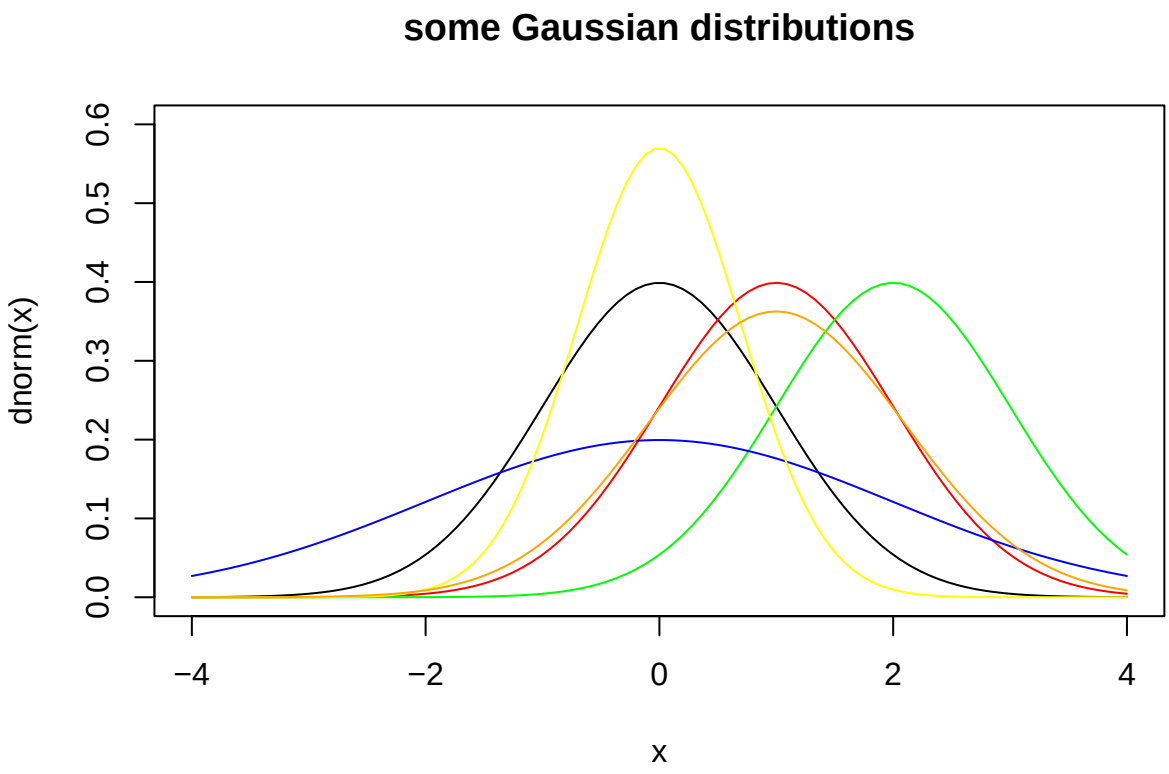
Generalized value of the mean and standard deviation

We can generalize the Gaussian mean μ and standard deviation σ easily:

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

But it is easier to use `dnorm()` and specify the mean and standard deviation directly:

```
x <- seq(from=-4,to=4,len=100)
plot(x,dnorm(x),pch=16,ylim=c(0,0.6),type='l',main='some Gaussian distributions')
points(x,dnorm(x,mean=1),col="red",type='l')
points(x,dnorm(x,mean=2),col="green",type='l')
points(x,dnorm(x,sd=2),col="blue",type='l')
points(x,dnorm(x,sd=0.7),col="yellow",type='l')
points(x,dnorm(x,mean=1,sd=1.1),col="orange",type='l')
```



Make some other plots and get a feel for how the system works.

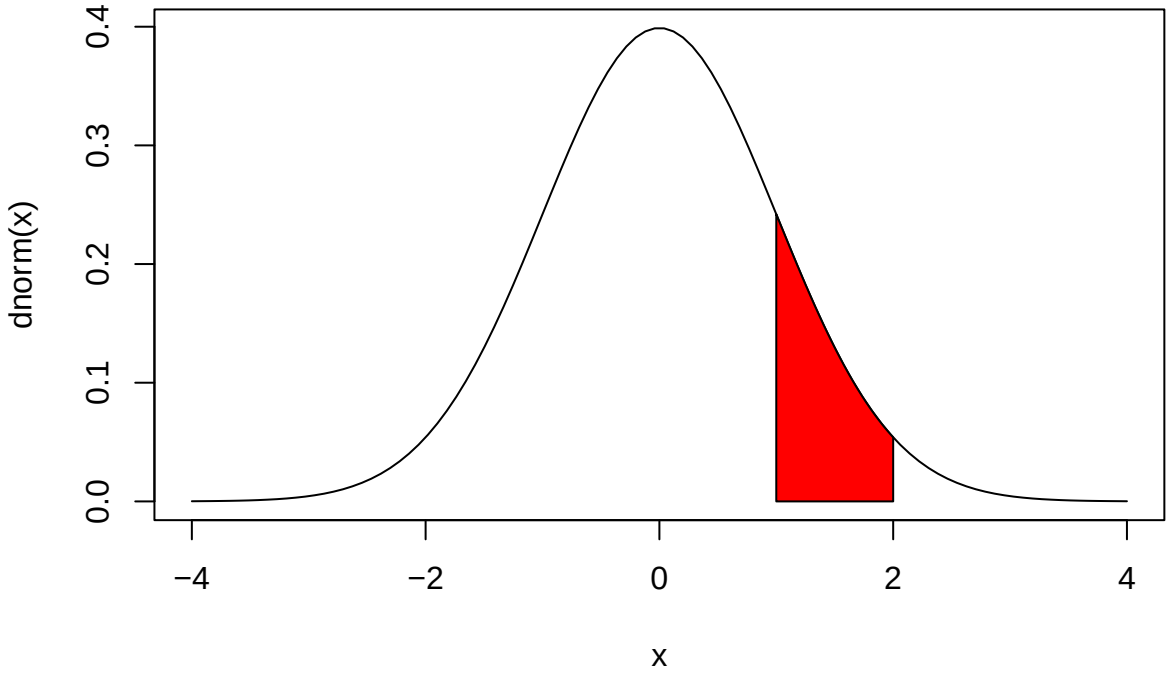
Areas under the curve

Quite often we are interested in areas under a Gaussian curve. To plot this we need a helper function:

```
areaplotter <- function(x1,x2){
  x <- seq(from=-4,to=4,len=100)
  plot(x,dnorm(x),type='l')
  jj <- seq(from=x1,to=x2,len=100)
  polygon(x=c(jj,x2,x1),y=c(dnorm(jj),0,0),col='red')
}
```

Now we can use this:

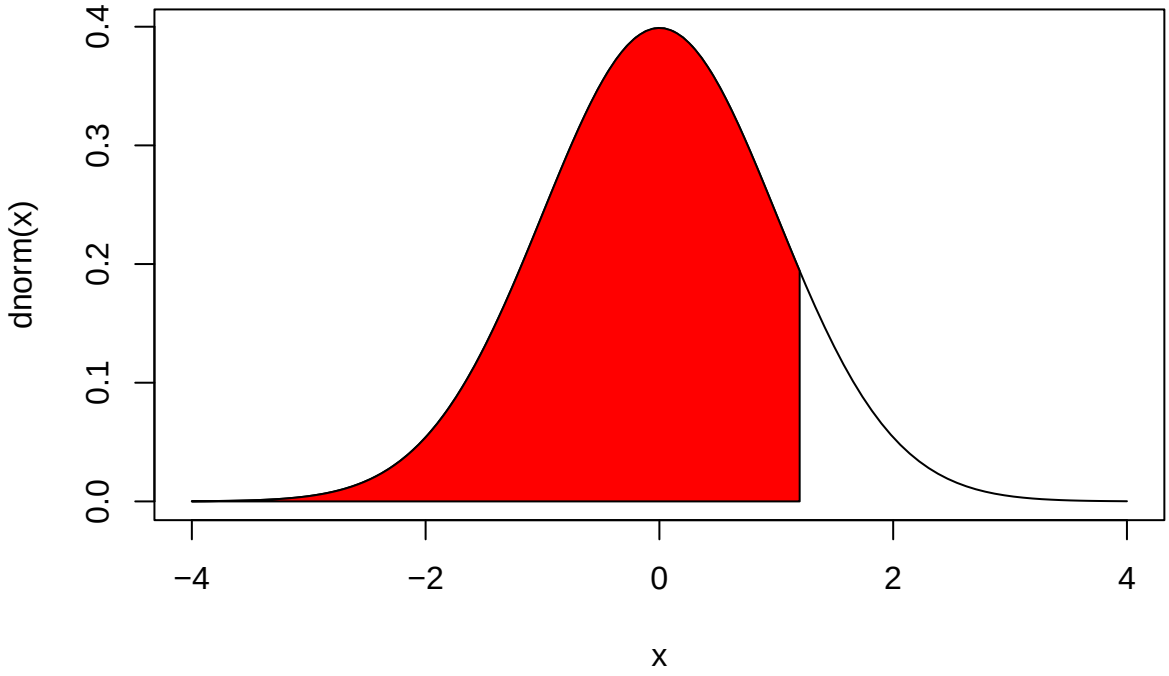
```
areaplotter(1,2)
```



The red area shows the probability that a standard Gaussian random variable lies between 1 and 2.

In R, the way to calculate areas is `pnorm()`. Function `pnorm(x)` gives the area to the left of point x. Suppose we want the probability that $X < 1.2$ [it is usual to refer to a random variable itself using upper-case letters]:

```
areaplotter(-4,1.2)
```




```
pnorm(1.2)
```

```
## [1] 0.8849303
```

Cultivate the habit of guessing before evaluating.

We can verify by numerical sampling:

```
table(rnorm(1e6) < 1.2)/1e6
```

```
##
##    FALSE     TRUE
## 0.114753 0.885247
```

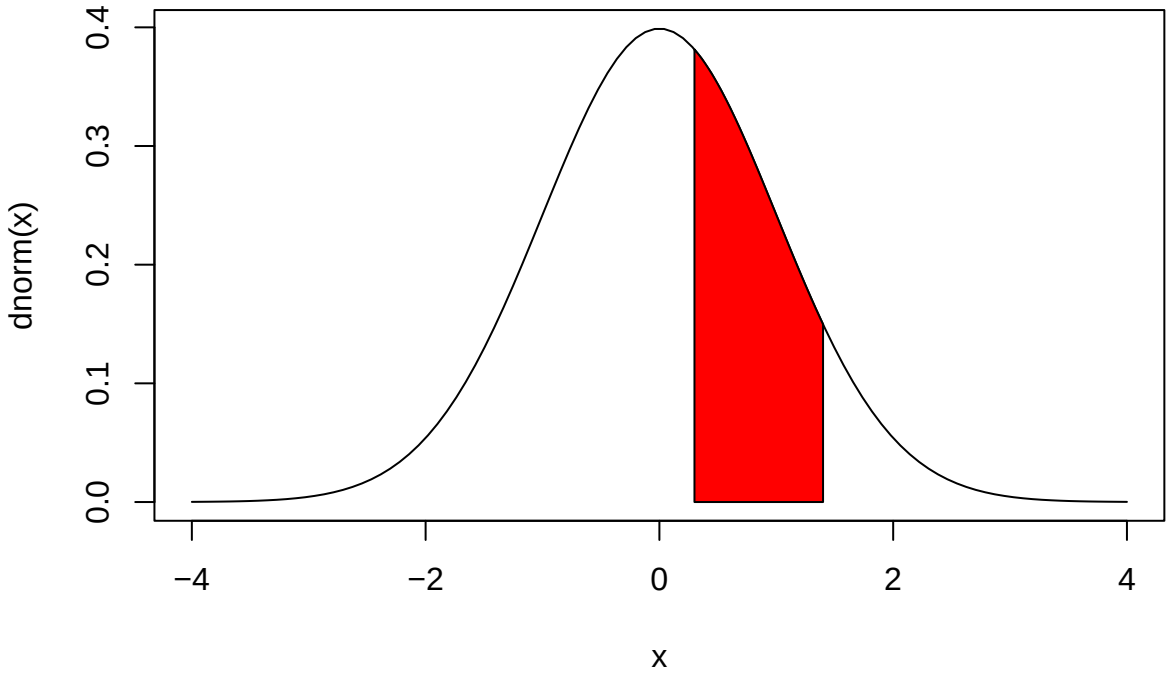
```
table(rnorm(1e6) < 1.2)/1e6
```

```
##
##   FALSE    TRUE
## 0.11514 0.88486
```

Showing that the 88% given by `pnorm()` is quite accurate.

To get values in a range, subtract one value of `pnorm()` from another. Thus to find the probability that X lies in the range 0.3-1.4:

```
areaplotter(0.3,1.4)
```




```
pnorm(1.4) - pnorm(0.3)
```

```
## [1] 0.3013319
```

Numerical verification is slightly harder:

```
x <- rnorm(1e6)
table((x>0.3) & (x<1.4))/1e6
```

```
##
##    FALSE     TRUE
## 0.698762 0.301238
```

```
table((x>0.3) & (x<1.4))/1e6
```

```
##
##    FALSE     TRUE
## 0.698762 0.301238
```

(why do the two evaluations agree exactly? How can we get a handle on the random variability?)

The quantile function

Suppose we wanted to find the poisition on the x axis for which $P(X < x) = 0.8$. That is, the probability that our observation is less than x is 80%.

We can use `pnorm()` in a trial-and-error fashion:

```
pnorm(1.1)   # initial guess
```

```
## [1] 0.8643339
```

```
pnorm(0.9)   # too high
```

```
## [1] 0.8159399
```

```
pnorm(0.8)   # etc
```

```
## [1] 0.7881446
```

```
pnorm(0.85)
```

```
## [1] 0.8023375
```

But it is much better to use special function `qnorm()` which does everything much faster:

```
qnorm(0.8)
```

```
## [1] 0.8416212
```

It is important to verify this numerically:

```
table(rnorm(1e6) < qnorm(0.8)) / 1e6
```

```
##
##    FALSE     TRUE
## 0.199956 0.800044
```

```
table(rnorm(1e6) < qnorm(0.8)) / 1e6
```

```
##
##    FALSE     TRUE
## 0.200451 0.799549
```

Further questions on the Gaussian distribution

(below, X refers to a standard Gaussian random variable)

- What is the probability that $X < 0.3$? Verify numerically and visually
- What is the probability that $X < 4.5$?
- (harder) What is the probability that $X < 14.5$? Hint: look at the help page and use the `lower.tail` argument
- What value x has $Prob(X < x) = 0.7$?

Gaussian approximation to the binomial

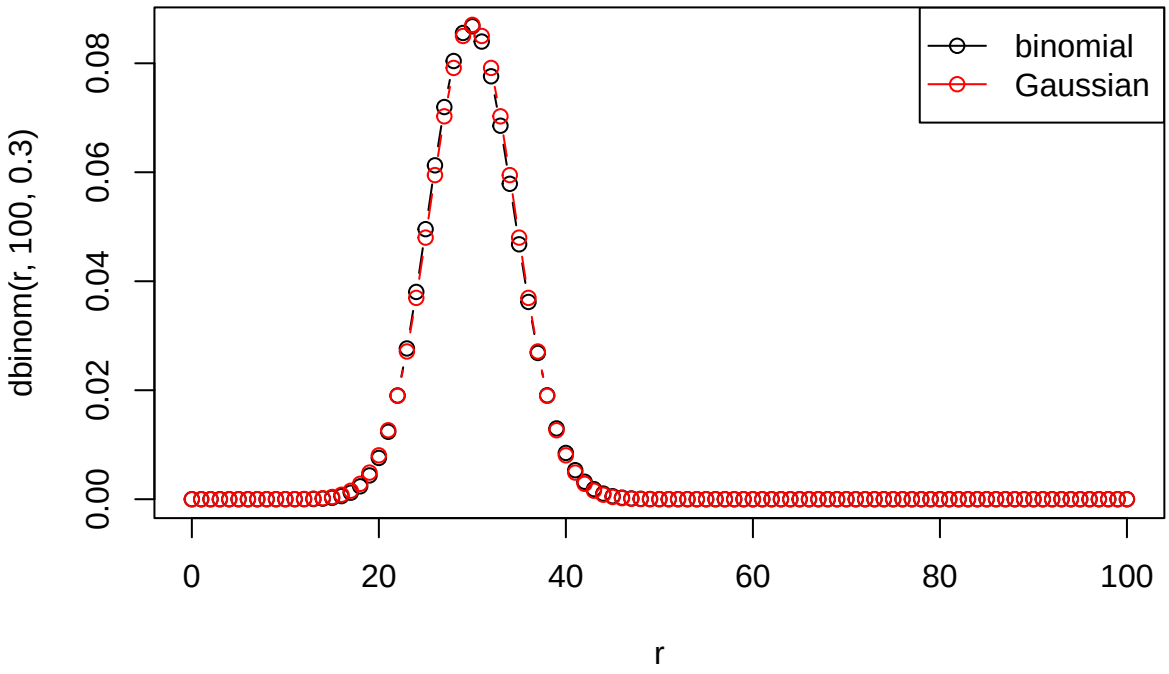
We know from lectures that if $X \sim \text{Bin}(n, p)$, then a good approximation for large n and moderate p would be

$$X \sim \text{Norm}\left(\mu = np, \sigma = \sqrt{np(1-p)}\right)$$

We will verify this using R, with $n = 100$ and $p = 0.3$. Then the mean would be $np = 30$ and $\sigma = \sqrt{100 * 0.3 * 0.7} \simeq 4.58$.

First we will plot a graph of the two probability distributions:

```
r <- 0:100
plot(r,dbinom(r,100,0.3),type='b')
points(r,dnorm(r,mean=30,sd=sqrt(21)),type='b',col='red')
legend("topright",col=c('black','red'),legend=c("binomial","Gaussian"),lty=1,pch=1)
```



(in the above, see how I have written `sqrt(21)` for the standard deviation as shorthand for `sqrt(100*0.3*0.7)`. Of course, the next step is to figure out how good the approximation is. Can you quantify the difference between the Gaussian and the binomial using R?

We can also calculate probabilities of falling inside a particular region. Suppose we want to calculate Prob($22 \leq r \leq 35$). We can estimate this using exact calculations:

```
sum(dbinom(22:35,100,0.3))
```

```
## [1] 0.8550901
```

```
diff(pbinom(c(21,35),100,0.3))
```

```
## [1] 0.8550901
```

See how I give two different ways to answer it. Which one is preferable and why? The Gaussian approximation would be

```
pnorm(35,30,sqrt(21))-pnorm(22,30,sqrt(21))
```

```
## [1] 0.8219554
```

the approximation is not bad, but not super accurate, because of the difference between `<` and `<=`. Google for "continuity correction" to see how to improve the approximation.

Now try different values of n and p with a view to:

- making the Gaussian approximation as good as possible
- making the Gaussian approximation as bad as possible.

Points to ponder

- In the above, the mean and standard deviation were known exactly. What would happen if you did not know the exact values of the mean and standard deviation?
- How would you *verify* that the standard Gaussian is symmetric about zero?
- Does the standard Gaussian have an upper limit?
- What is the density of the standard Gaussian at $x = 0$?
- How would you *verify* that `dnorm()` has an area under the curve of exactly one?
- The Gaussian density function is of the form $e^{-x^2/2}$. What would densities like $e^{-|x|}$ look like?
- With mean μ and standard deviation σ, I assert that the steepest point of the PDF is at $x = \mu \pm \sigma$: that is, the points of highest gradient are distance σ from the mean μ. Verify this.
- We know that if $X \sim \text{Bin}(n, p)$ is approximately Gaussian with mean np and variance npq if n is large and p, q moderate. Verify this statement and quantify the error incurred for different values of n, p.