# HW04

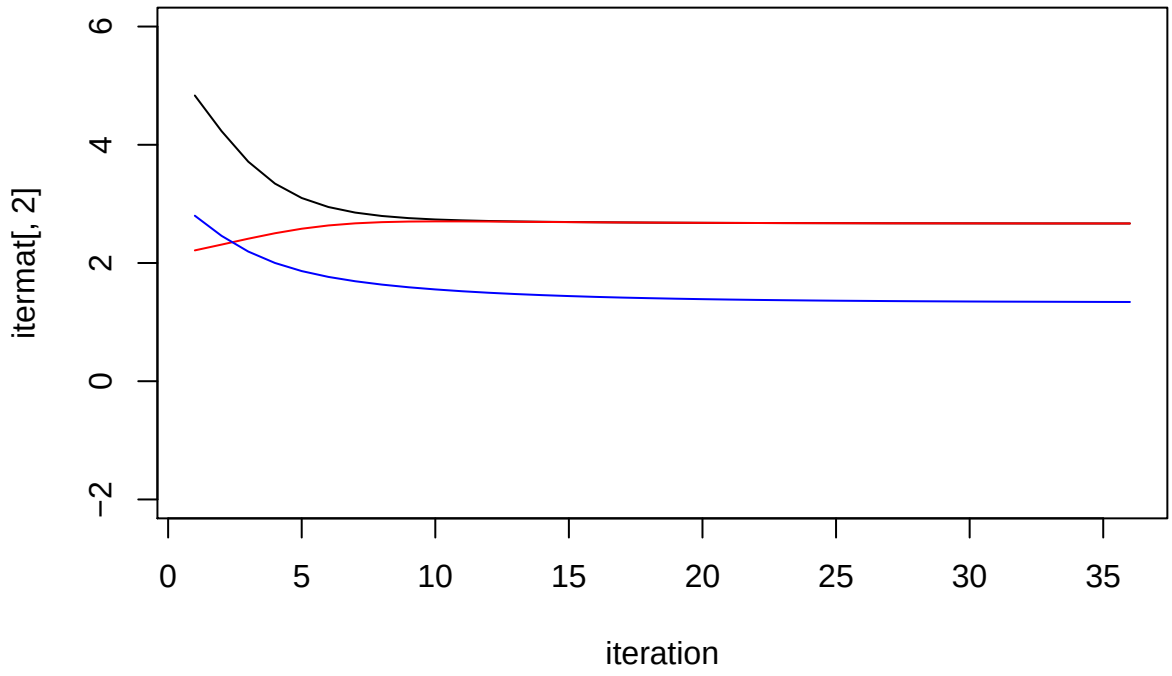
*Zach White*

*March 4, 2017*

## Problem 5.20

```
iter.n = 1
x1 = c(1,1,-1,-1,2,2,-2,-2,NA,NA,NA,NA)
x2 = c(1,-1,1,-1,NA,NA,NA,NA,2,2,-2,-2)
full.data = cbind(x1,x2)
cc.data = full.data[complete.cases(full.data),]
cc.n = nrow(cc.data)

r1 = as.numeric(!is.na(x1))
r1.not = which(is.na(x1))
r2 = as.numeric(!is.na(x2))
r2.not = which(is.na(x2))
r = cbind(r1,r2)
# Initial Values
sigma1.2 = 5
sigma2.2 = 2
sigma12 = 3
Sigmat = matrix(c(sigma1.2,sigma12,sigma12,sigma2.2),nrow = 2,ncol = 2)

full.data[is.na(full.data)] = 0
max.diff = 1
n = nrow(full.data)
itermat = NULL
# Second try
while(max.diff > .001){
      ex1 <- (r[,1]==1)*full.data[,1] + (r[,1]==0)*( Sigmat[1,2]*(full.data[,2])/Sigmat[2,2])
    ex2 <- (r[,2]==1)*full.data[,2] + (r[,2]==0)*(Sigmat[1,2]*(full.data[,1])/Sigmat[1,1])
    ex1.2 <- (r[,1]==1)*full.data[,1]^2 + (r[,1]==0)*(Sigmat[1,1]-Sigmat[1,2]^2/Sigmat[2,2]+ ex1^2)
    ex2.2 <- (r[,2]==1)*full.data[,2]^2 + (r[,2]==0)*(Sigmat[2,2]-Sigmat[1,2]^2/Sigmat[1,1]+ ex2^2)
    ex1x2 <- (r[,1]*r[,2]*full.data[,1]*full.data[,2] + r[,1]*(1-r[,2])*full.data[,1]*ex2
         + (1-r[,1])*r[,2]*ex1*full.data[,2])

     mu.new = c(mean(ex1),mean(ex2))
  s11 = mean(ex1.2) - mu.new[1]^2
  s22 = mean(ex2.2) - mu.new[2]^2
  s12 = mean(ex1x2) - mu.new[1]*mu.new[2]
  # Define these as the previous parameters
  sigma.mat = c(s11,s22,s12)

  sigma1.2.diff = abs(s11 - sigma1.2)
  sigma2.2.diff = abs(s22 - sigma2.2)
  sigma12.diff = abs(s12 - sigma12)
  max.diff = max(sigma12.diff,sigma2.2.diff,sigma1.2.diff)
  itermat = rbind(itermat,c(iter.n,sigma.mat))
```

```
  sigma1.2 = sigma.mat[1]
  sigma2.2 = sigma.mat[2]
  sigma12 = sigma.mat[3]

  Sigmat = matrix(c(sigma1.2,sigma12,sigma12,sigma2.2),nrow = 2,ncol = 2)

  iter.n = iter.n + 1

}
plot(itermat[,1],itermat[,2], type = "l", xlab = "iteration",ylim = c(-2,6))
lines(itermat[,1],itermat[,3], type = "l" , col = "red")
lines(itermat[,1],itermat[,4], type = "l" , col = "blue")
```




It's clear that these all converge very quickly. With the starting values of $\sigma^2_{11} = 1$, $\sigma^2_{22} = 2$, $\sigma^2_{21} = 3$, they converge to 2.6683567, 2.6683567, and 1.3405923

# Problem 26

## Part A

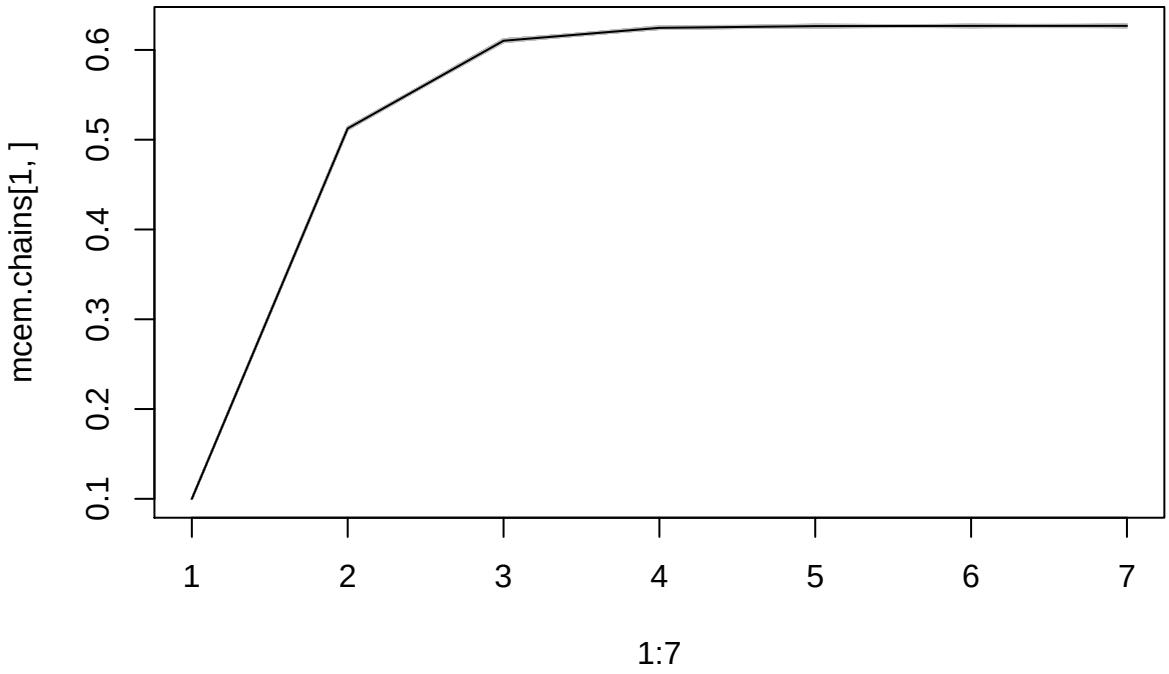
```
x = c(125,18,20,34)

start.val = chain = current = diff = .1
iter.count = 2
```

```
while(diff > .0001){
  new.val = ((current * x[1] / (2+ current)) + x[4]) / ((current*x[1] / (2 + current)) + x[2] + x[3] + x
  chain = c(chain,new.val)
  diff = abs(current - chain[iter.count])
  current = chain[iter.count]
  iter.count = iter.count + 1
}
```

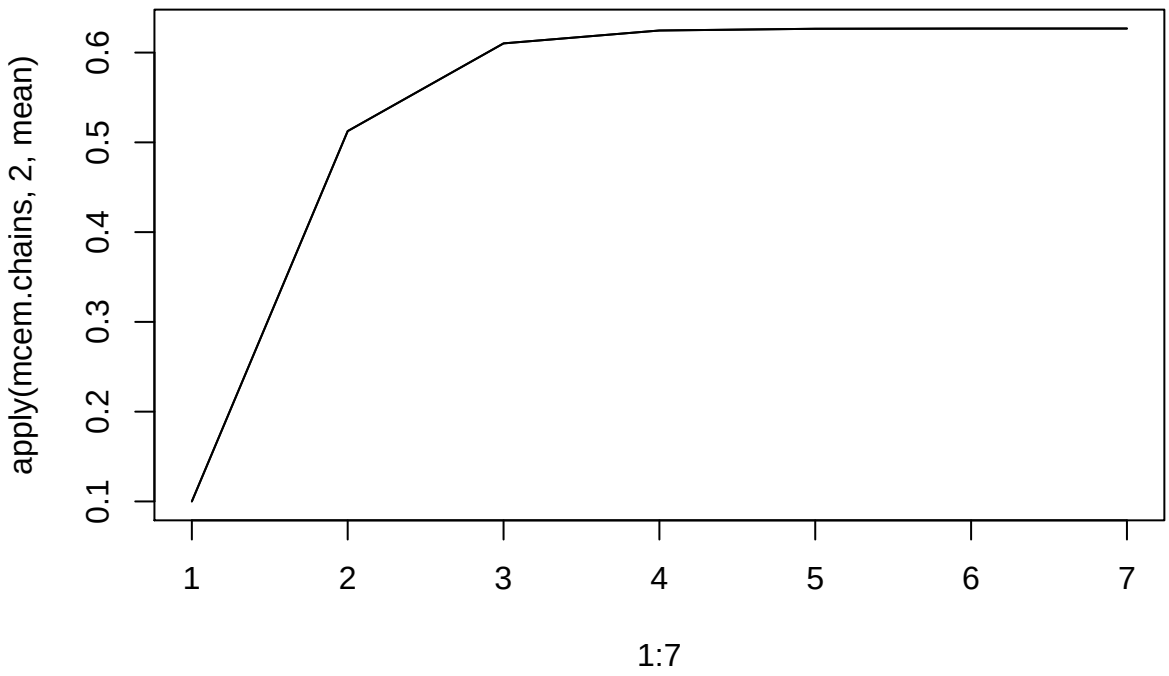
## Part B

I now use Monte Carlo EM algorithm. This means we will simulate some binomial draws and then insert an approximated value into a part of the equation

```
M = 1000
mcem.chains = matrix(start.val, length(chain), nrow = 500)
for(i in 2:length(chain)){
  mcem.chains[,i] = 1/(1+(x[2]+x[3])/(x[4]+rbinom(500,M*x[1],
1/(1+2/mcem.chains[,i-1]))/M))
}
plot(1:7,mcem.chains[1,], type = "l")
for(i in 2:500){
  lines(1:7, mcem.chains[i,],type = "l",col = "grey")
}
lines(1:7, chain)
```



```
plot(1:7, apply(mcem.chains,2,mean), type = "l")
lines(1:7,chain)
```




Both of these methods converge very quickly. Even under the Monte Carlo method, our converges very quickly. I assume it is partly because of the number of iterations we choose. If that were smaller, the variance would be larger, and convergence would be slower.

# Problem 2

## Part B

```
y1 = c(125,18,20,34)
y2 = c(14,0,1,5)

y1.like = function(theta.y){
  (2+theta.y)^(125) * (1-theta.y)^(38) * theta.y^34
}
log.y1 = function(theta.y){
  -log(y1.like(theta.y))
}
y2.like = function(theta.y2){
  (2+theta.y2)^(14) * (1-theta.y2)^(1) * theta.y2^(5)
}
log.y2 = function(theta.y2){
```

```
  -log(y2.like(theta.y2))
}
#xs = seq(0,1,length = 1000)
#plot(xs,y2.like(xs))


# summary(ms( ~ log.y1(theta.y2), start = c(theta.y2 = c(.1,.2,.3,.4,.6,.8))))
# optim(.1,log.y2)
# optim(.1,y1.like)
# summary(ms( ~ ng(th),start=c(th=0.5))) # R users: optim(0.5,ng)

newton<-function(error=1e-7, theta = .1)
{
  #theta<- 0.2
  k<-1
  while(k<=10000)
  {
    k<-k+1
    theta[k]<-theta[k-1]-((125/(2+theta[k-1])-38/(1-theta[k-1])+34/theta[k-1]))/
      (-125*((2+theta[k-1])^(-2))-38*((1-theta[k-1])^(-2))-34*((theta[k-1])^(-2)))
    if(abs(theta[k]-theta[k-1])<error)
      break
  }
 cat("theta",theta)
 list(theta[k],k)
}
newton(theta = .1)
```

```
## theta 0.1 0.202813 0.3965553 0.6154252 0.6270131 0.6268216 0.6268215
```

```
## [[1]]
## [1] 0.6268215
##
## [[2]]
## [1] 7
```

```
newton(theta = .2)
```

```
## theta 0.2 0.3917428 0.6130034 0.6270999 0.6268216 0.6268215 0.6268215
```

```
## [[1]]
## [1] 0.6268215
##
## [[2]]
## [1] 7
```

```
newton(theta = .3)
```

```
## theta 0.3 0.5367543 0.6340136 0.6269043 0.6268215 0.6268215
```

```
## [[1]]
## [1] 0.6268215
##
## [[2]]
## [1] 6
```

```r
newton(theta = .4)
```

```
## theta 0.4 0.6170669 0.6269629 0.6268215 0.6268215
```

```
## [[1]]
## [1] 0.6268215
##
## [[2]]
## [1] 5
```

```r
newton(theta = .6)
```

```
## theta 0.6 0.6278042 0.626823 0.6268215 0.6268215
```

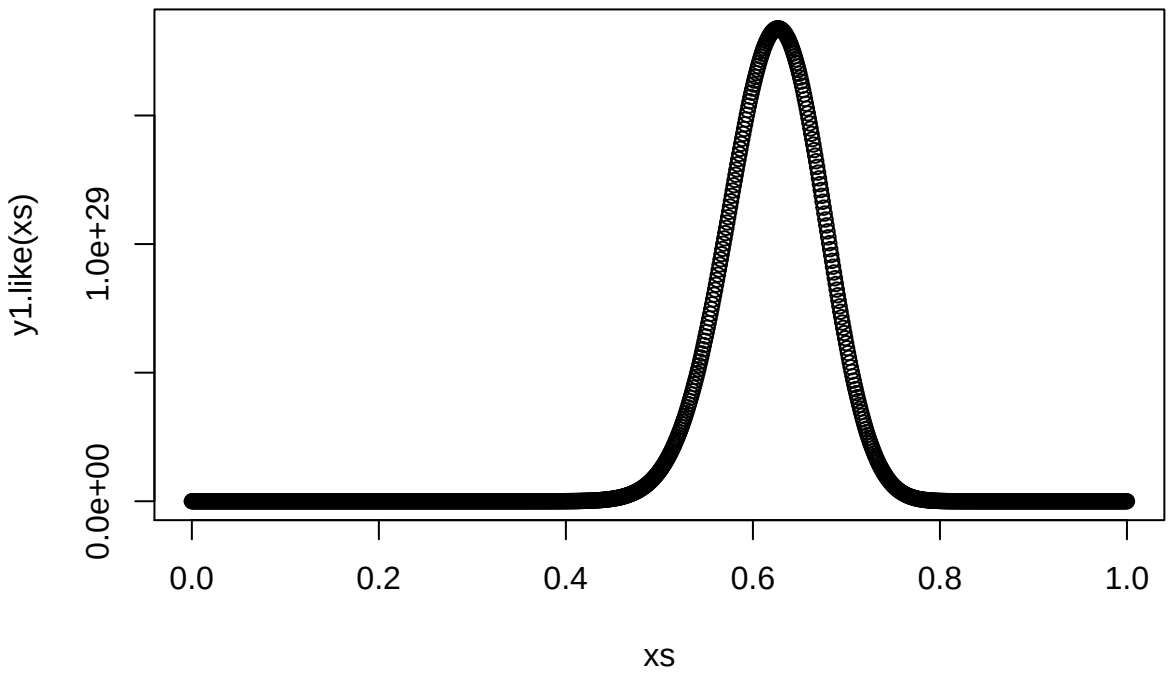
```
## [[1]]
## [1] 0.6268215
##
## [[2]]
## [1] 5
```

```r
newton(theta = .8)
```

```
## theta 0.8 0.6990675 0.6372008 0.6269962 0.6268215 0.6268215
```

```
## [[1]]
## [1] 0.6268215
##
## [[2]]
## [1] 6
```

```r
xs = seq(0,1,length = 1000)
plot(xs,y1.like(xs))
```




The starting values that produce the smallest number of iterations is .4 and .6. The other starting values converge around 6 or 7 iterations.

### Part C

```
newton2<-function(error=1e-7, theta = .1)
{
  #theta<- 0.2
  k<-1
  while(k<=10000)
  {
    k<-k+1
    theta[k]<-theta[k-1]-((14/(2+theta[k-1])-1/(1-theta[k-1])+5/theta[k-1]))/
      (-14*((2+theta[k-1])^(-2))-1*((1-theta[k-1])^(-2))-5*((theta[k-1])^(-2)))
    if(abs(theta[k]-theta[k-1])<error)
      break
  }
 cat("theta",theta)
 list(theta[k],k)
}
newton2(theta = .1)
```

```
## theta 0.1 0.2101399 0.4553638 0.9534414 0.9289039 0.9098902 0.9038433 0.9034417 0.9034401 0.9034401
```

```
## [[1]]
## [1] 0.9034401
```

```
##
## [[2]]
## [1] 10
```

```
newton2(theta = .2)
```

```
## theta 0.2 0.4326185 0.9157311 0.9049171 0.9034611 0.9034401 0.9034401
```

```
## [[1]]
## [1] 0.9034401
##
## [[2]]
## [1] 7
```

```
#newton2(theta = .3)
newton2(theta = .4)
```

```
## theta 0.4 0.8571429 0.9220472 0.9068548 0.9035526 0.9034402 0.9034401 0.9034401
```

```
## [[1]]
## [1] 0.9034401
##
## [[2]]
## [1] 8
```

```
#newton2(theta = .6)
newton2(theta = .8)
```

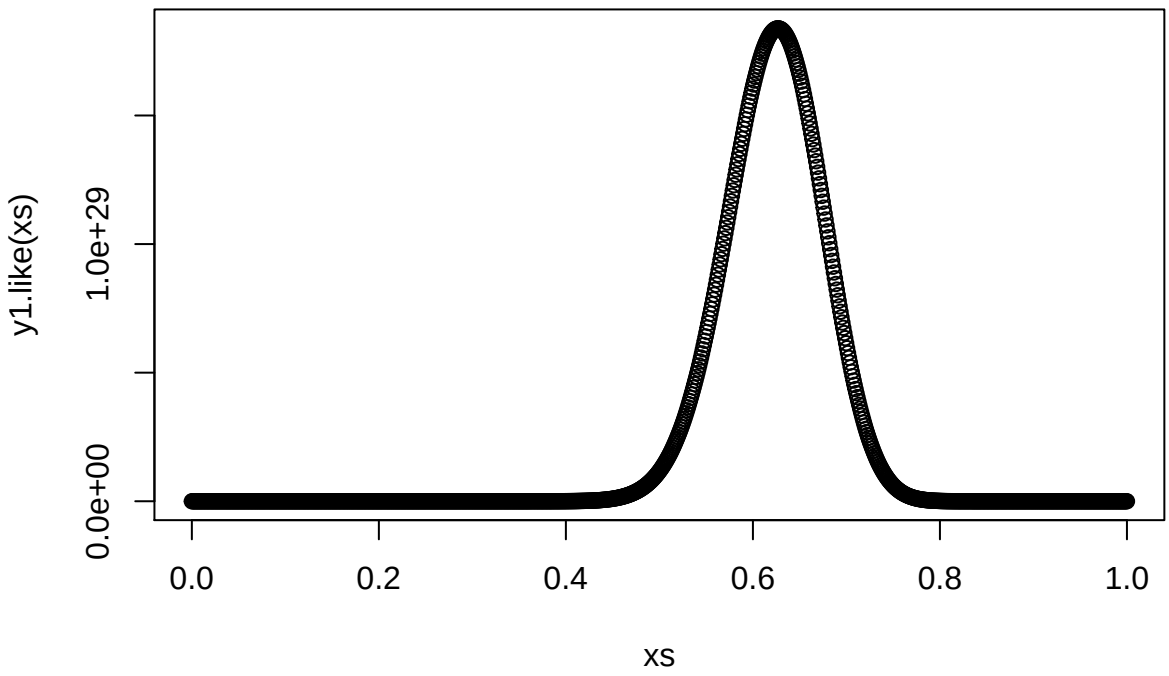
```
## theta 0.8 0.9806452 0.9649996 0.9423241 0.9186965 0.9057253 0.9034904 0.9034401 0.9034401
```

```
## [[1]]
## [1] 0.9034401
##
## [[2]]
## [1] 9
```

Under this second likelihood, the convergence is generally slower. And newton raphson doesn't actually occur with the starting values .3 and .6.

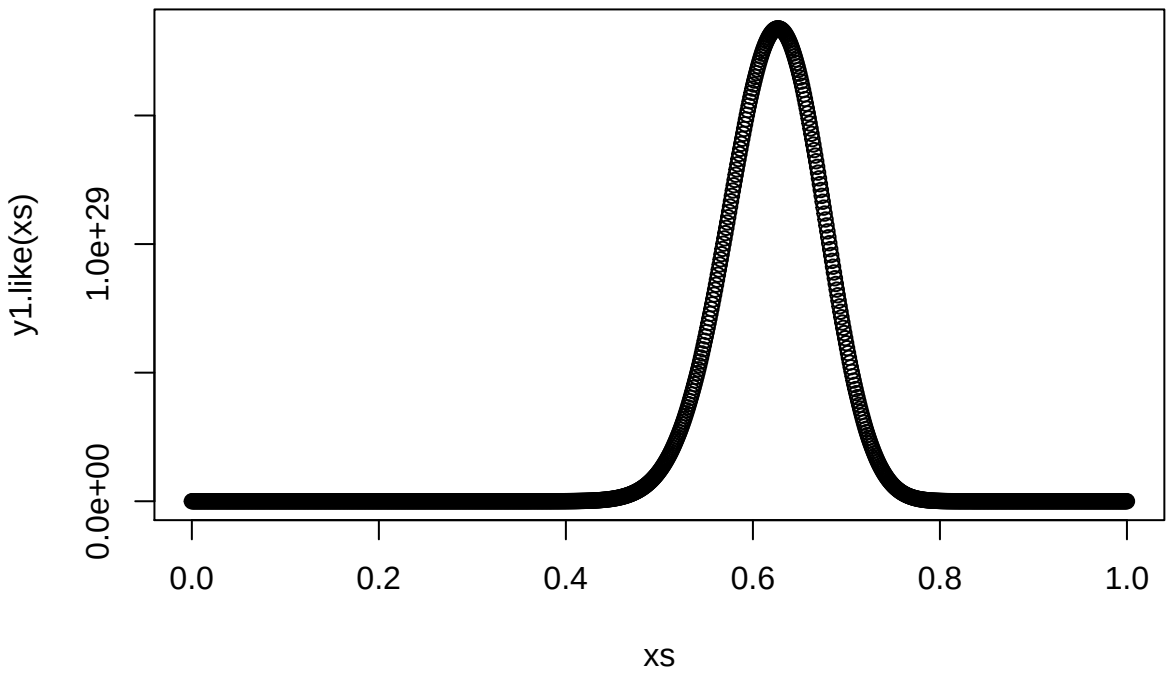
## Part D

```
xs = seq(0,1,length = 1000)
plot(xs,y1.like(xs))
```




### Part E

```
xs = seq(0,1,length = 1000)
plot(xs,y1.like(xs))
```


y1.like(xs)
0.0e+00
1.0e+29
0.0
0.2
0.4
0.6
0.8
1.0
xs