# 440a3

phantomOfLaMancha

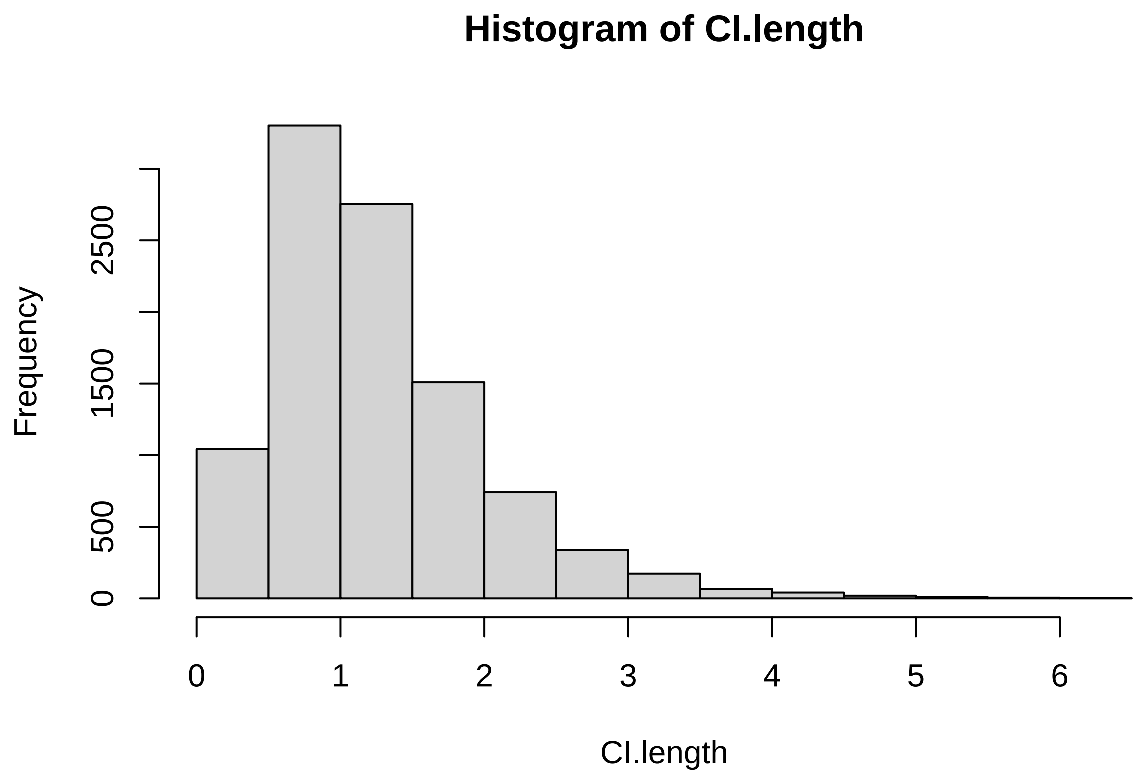
4/5/2021

1a

```r
# jackknife

M=10000


func = function(x){
  if (length(x)==25){
    return((x[19]-x[7])/1.34)
  }
  if (length(x)==24){
    return((0.75*x[18]+0.25*x[19]-x[6]*0.25-x[7]*0.75)/1.34)
  }

}



jackknife.est = function(x){
  x = sort(x)
  true.result = (qt(0.75,3)-qt(0.25,3))/1.34
  theta.hat = func(x)
  theta.minus = c(rep(func(x[-1]),6) ,func(x[-7]),  rep(func(x[-8]),11),
                  func(x[-19]), rep(func(x[-20]),6))
  theta.hat
  theta.minus
  se = sqrt(24/25*sum((theta.minus - mean(theta.minus))^2))
  CI = theta.hat + c(-1, 1)* qnorm(0.975) * se
  covered = ((true.result >= CI[1]) && (true.result <= CI[2]))
  return( list(covered = covered, CI.length = 2 * qnorm(0.975) * se))
}

covered = rep(0,M)
CI.length = rep(0,M)
set.seed(1)
for (i in 1:M){
  x = rt(25,3)
  j = jackknife.est(x)
  covered[i] = j[[1]]
  CI.length[i] = j[[2]]
}

sum(covered)
```

```
## [1] 8572
```

```
hist(CI.length)
```



```
mean(CI.length)
```

```
## [1] 1.258366
```

```
sd(CI.length)
```

```
## [1] 0.7389942
```

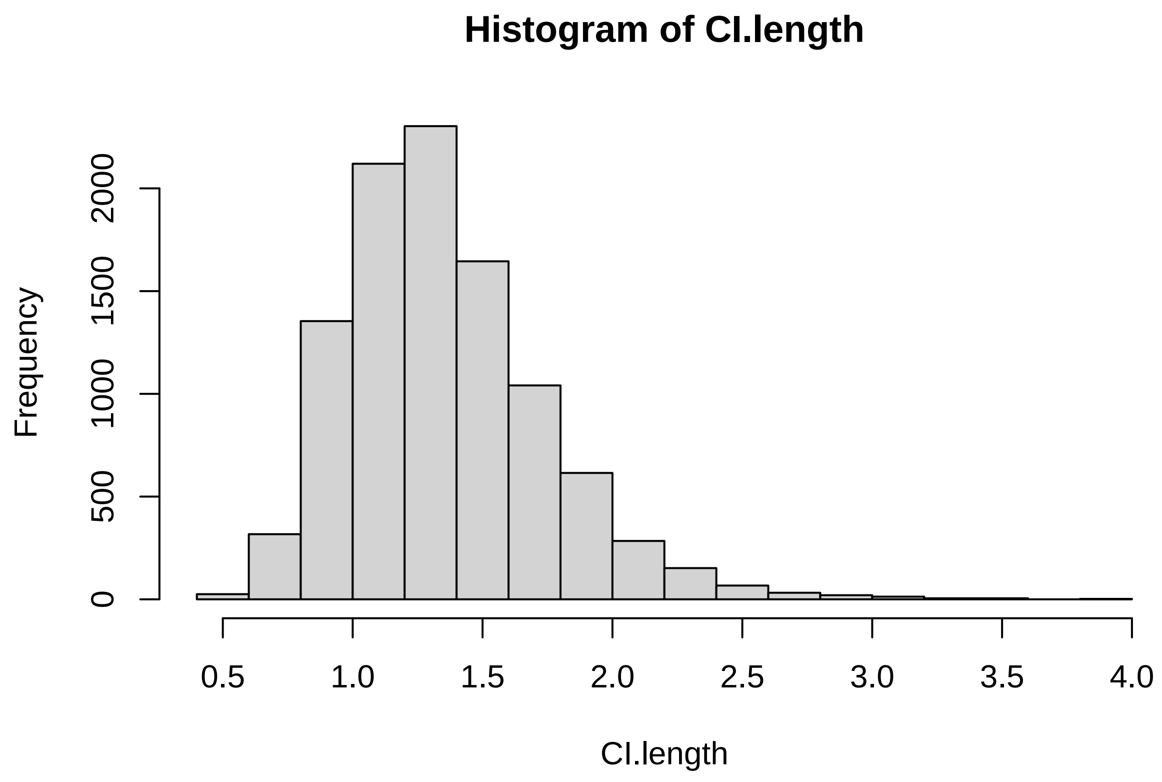
1b

```
M=10000
B=1000


covered = rep(0,M)
CI.length = rep(0,M)
true.result = (qt(0.75,3)-qt(0.25,3))/1.34
set.seed(1)
for (i in 1:M){
  x = sort(rt(25,3))
  # boot
  boots = rep(0,B)
  for (j in 1:B){
    bootx = sort(sample(x,25, replace=TRUE))
    boots[j] = func(bootx)
```

```r
  }
  sd = sd(boots)
  theta.hat = func(x)

  CI = theta.hat + c(-1, 1)* qnorm(0.975) * sd
  covered[i] = ((true.result >= CI[1]) && (true.result <= CI[2]))
  CI.length[i] = 2 * qnorm(0.975) * sd

}

sum(covered)
```

```
## [1] 9592
```

```r
hist(CI.length)
```



```r
mean(CI.length)
```

```
## [1] 1.352679
```

```r
sd(CI.length)
```

```
## [1] 0.3836484
```

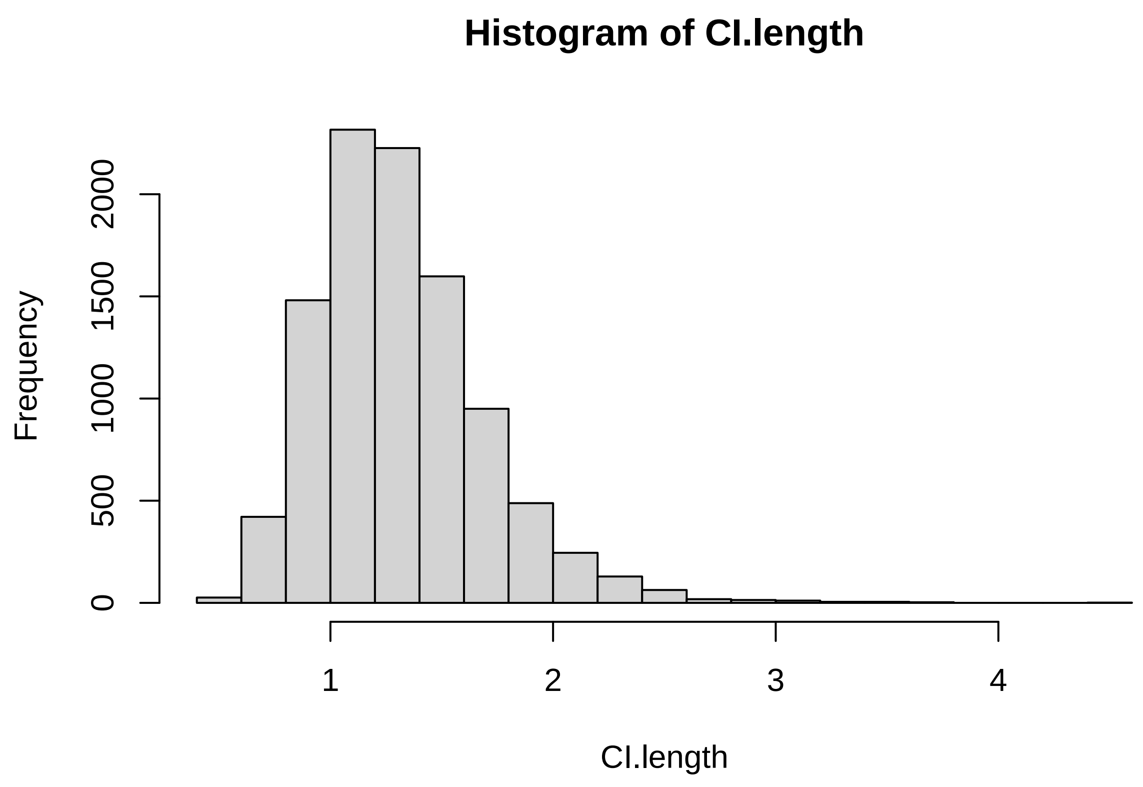
1c

```r
M=10000
B=1000
```

```r
covered = rep(0,M)
CI.length = rep(0,M)
true.result = (qt(0.75,3)-qt(0.25,3))/1.34
set.seed(1)
for (i in 1:M){
  x = sort(rt(25,3))
  theta.hat = func(x)
  # boot
  boots = rep(0,B)
  for (j in 1:B){
    boots[j] = func(sort(sample(x,25, replace=TRUE)))
  }
  boots = sort(boots)
  CI = quantile(boots, c(0.025, 0.975))
  covered[i] = ((true.result >= CI[1]) && (true.result <= CI[2]))
  CI.length[i] = CI[2]-CI[1]
}

sum(covered)
```

```
## [1] 9785
```

```r
hist(CI.length)
```



```r
mean(CI.length)
```

```
## [1] 1.316201
```

```r
sd(CI.length)
```

```
## [1] 0.3743622
```

```r
quantile(1:100, c(0.025,0.975))
```

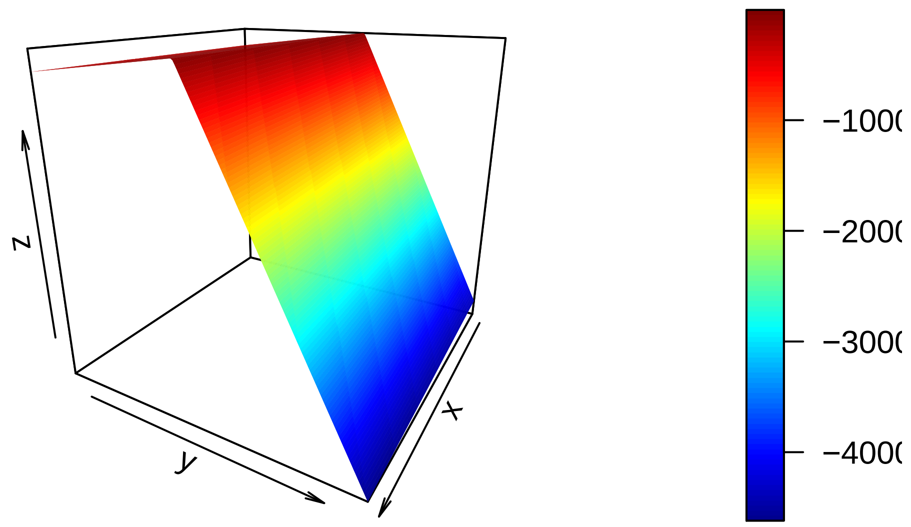
```
##   2.5%  97.5%
##  3.475 97.525
```

2

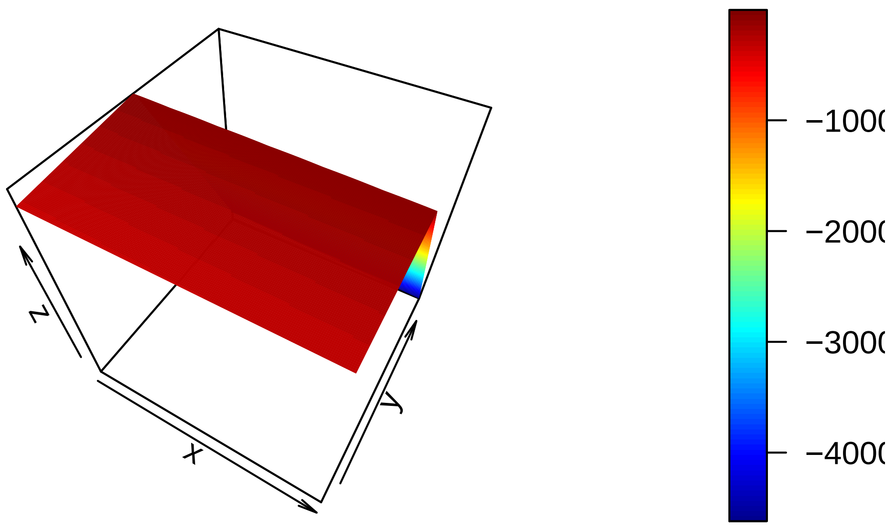
```r
x=c(53,57,58,63,66,67,67,67,68,69,70,70,70,70,72,73,75,75,76,76,78,79,81)
y=c(2,1,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,2,0,0,0,0,0)

llikehd = function(v){
  a = v[1]
  b = v[2]
  a*sum(y)+b*sum(x*y)-6*sum(log(1+exp(a+b*x)))
}

neg.llikehd = function(v){
  a = v[1]
  b = v[2]
  -(a*sum(y)+b*sum(x*y)-6*sum(log(1+exp(a+b*x))))
}

get.first.dr = function(v){
  a = v[1]
  b = v[2]
  c(sum(y)-6*sum(exp(a+b*x)/(1+exp(a+b*x))),
    sum(x*y)-6*sum(x*exp(a+b*x)/(1+exp(a+b*x))))
}


get.H.matrix = function(v){
  a = v[1]
  b = v[2]
  aa = -6*sum(exp(a+b*x)/(1+exp(a+b*x))^2 )
  ab = -6*sum(x*exp(a+b*x)/(1+exp(a+b*x))^2 )
  bb = -6*sum(x^2*exp(a+b*x)/(1+exp(a+b*x))^2 )
  matrix(c(aa,ab,ab,bb), nrow = 2, ncol = 2)
}



library(plot3D)
```

```
## Warning: package 'plot3D' was built under R version 4.0.4
```

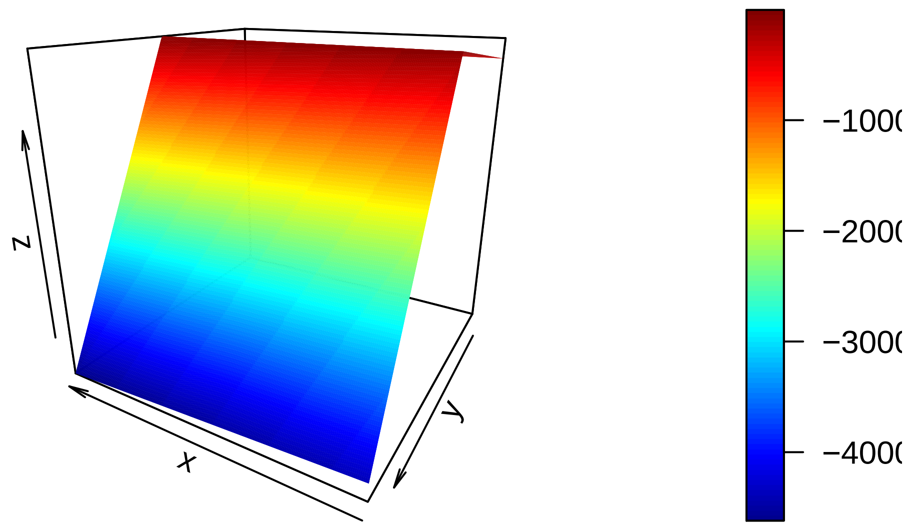
```r
# create grid matrices
 X       <- seq(-10, 10, length.out = 200)
 Y       <- seq(-5, 5, length.out = 200)
 M       <- mesh(X, Y)
 Z = matrix(0, nrow = 200, ncol = 200)
 X = M$x
 Y = M$y
 for (i in 1:200){
   for (j in 1:200){
     Z[i,j] = llikehd(c(X[i,j],Y[i,j]))
   }
 }

surf3D(X, Y, Z, colvar = Z, colkey = TRUE,
       box = TRUE, bty = "b", phi = 20, theta = 120)
```



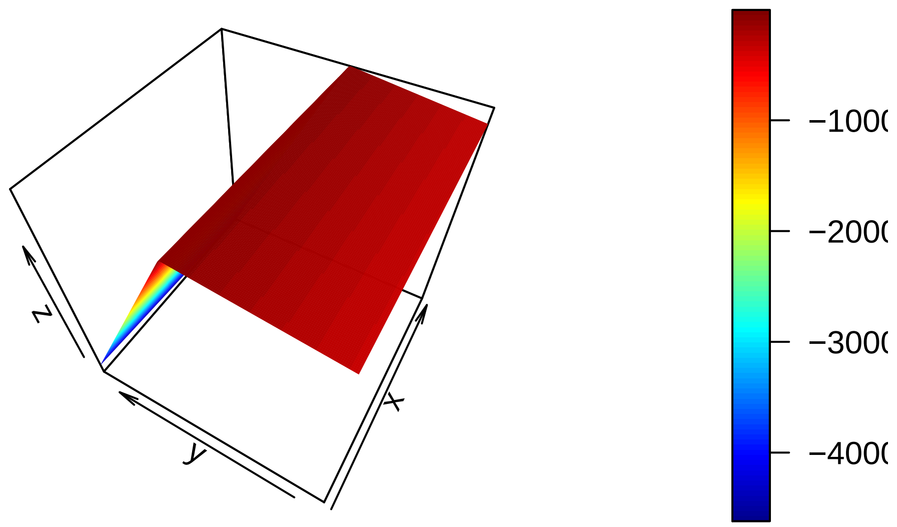
```r
surf3D(X, Y, Z, colvar = Z, colkey = TRUE,
       box = TRUE, bty = "b", phi = 50, theta = 30)
```



```r
surf3D(X, Y, Z, colvar = Z, colkey = TRUE,
       box = TRUE, bty = "b", phi = 20, theta = 210)
```



```r
surf3D(X, Y, Z, colvar = Z, colkey = TRUE,
       box = TRUE, bty = "b", phi = 50, theta = 300)
```



```
result = optim(par = c(0, 0), neg.llikehd, method = "Nelder-Mead")
result$par
```

```
## [1]  5.0849369 -0.1156042
```

c

```
v = c(-2,0)
H = get.H.matrix(v)
s = get.first.dr(v)

newton = function(v){
  it = 0
  while (TRUE) {
    s = solve(get.H.matrix(v), -get.first.dr(v))
    if ( max(abs(s))<0.01){
      return(v)
    }
    if (it > 10000){
      stop("over 10000 iterations")
    }
    v = v+s
  }
}

newton(v)
```

```
## [1]  5.084908 -0.115600
```

d

we wish to determine expectation of second order derivatives, we observe 2nd order derivatives depend only on x (temperature), we do not know distribution of temperature, so we do bootstrap

```
boot.H.matrix = function(v){
  a = v[1]
  b = v[2]

  aa = rep(0,B)
  ab = rep(0,B)
  bb = rep(0,B)
  for (i in 1:B){
    newx = sample(x,length(x),replace=TRUE)
    aa[i] = 6*sum(exp(a+b*newx)/(1+exp(a+b*newx))^2)
    ab[i] = 6*sum(newx*exp(a+b*newx)/(1+exp(a+b*newx))^2 )
    bb[i] = 6*sum(newx^2*exp(a+b*newx)/(1+exp(a+b*newx))^2 )
  }
  aaa=mean(aa)
  bbb=mean(bb)
  aba=mean(ab)

  matrix(c(aaa,aba,aba,bbb), nrow = 2, ncol = 2)
}

B=1000

fisher = function(v){
  v = c(-2,0)# delete this
  it = 0
  while (TRUE) {
    s = solve(boot.H.matrix(v), get.first.dr(v))
    if ( max(abs(s))<0.01){
      return(v)
    }
    if (it > 10000){
      stop("over 10000 iterations")
    }
    v = v+s
    it = it + 1
  }
}

fisher(c(0,0))
```

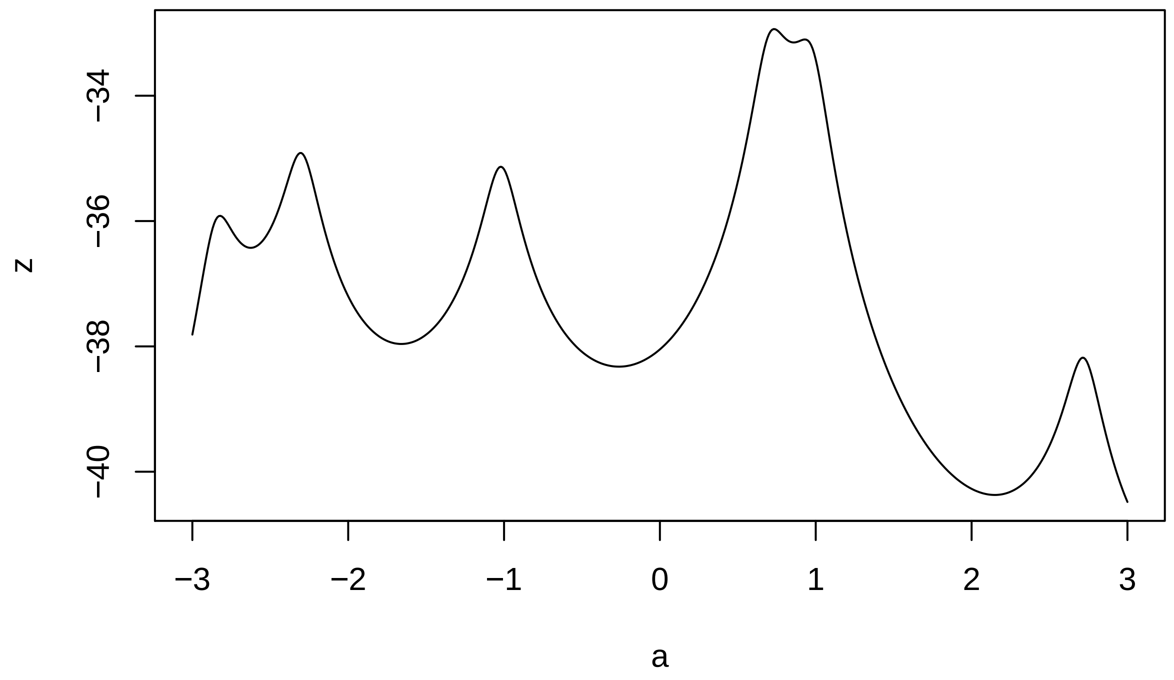
```
## [1]  5.0848490 -0.1155994
```

```
result = optim(par = c(0, 0), neg.llikehd, method = "BFGS")
result$par
```

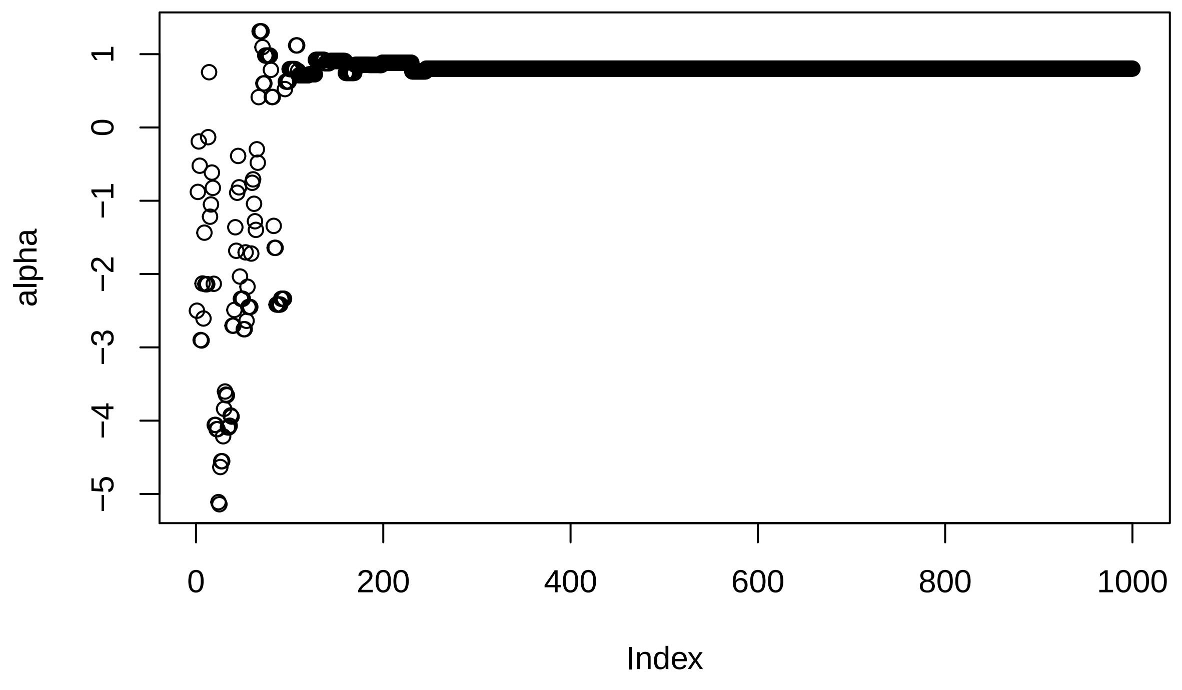
```
## [1]  5.1244166 -0.1162063
```

3

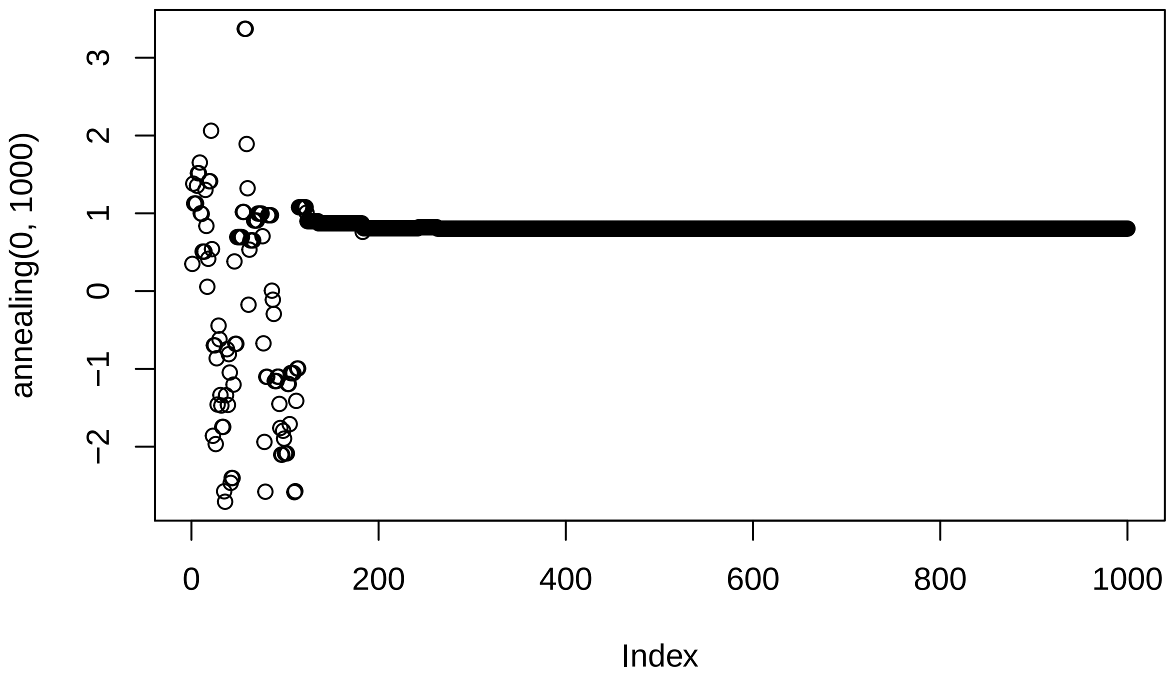
```r
x = c(-4.2, -2.85, -2.3, -1.02, 0.7, 0.98, 2.72, 3.5)
n = length(x)

llikehd = function(a, b){
  n*log(b) - n*log(pi) - sum(log(b^2 + (x-a)^2) )
}

plot.llikehd = function(b){
  a = seq(-3,3,0.01)
  m = length(a)
  z = rep(0, m)
  for (i in 1:m){
    z[i] =llikehd(a[i],b)
  }
  plot(a, z, type = 'l')
}

plot.llikehd(0.1)
```



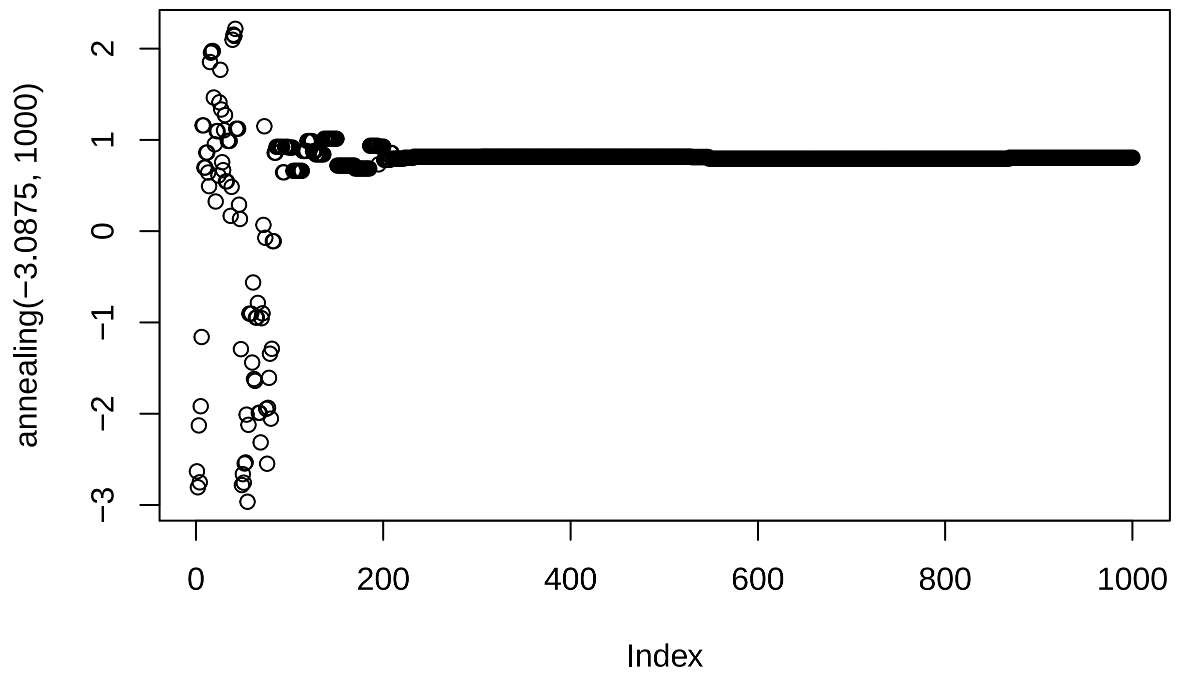
```r
annealing = function(a,m){
  i=1
  b = 0.2
  T0 = 10
  Tf = 10^(-10)
  olda = a
  amonitor = rep(0,m)

  for (i in 1:m){
    Ti = T0*(Tf/T0)^((i-1)/m)
    newa = rnorm(1, mean = olda)
    r = exp((llikehd(newa, b)- llikehd(olda, b) )/Ti )
    if(r < runif(1)){
      olda = olda
    }else{
      olda = newa
    }
    amonitor[i] = olda
  }
  amonitor
}

alpha = annealing(-2.5,1000)
plot(alpha)
```



```
plot(annealing(0, 1000))
```



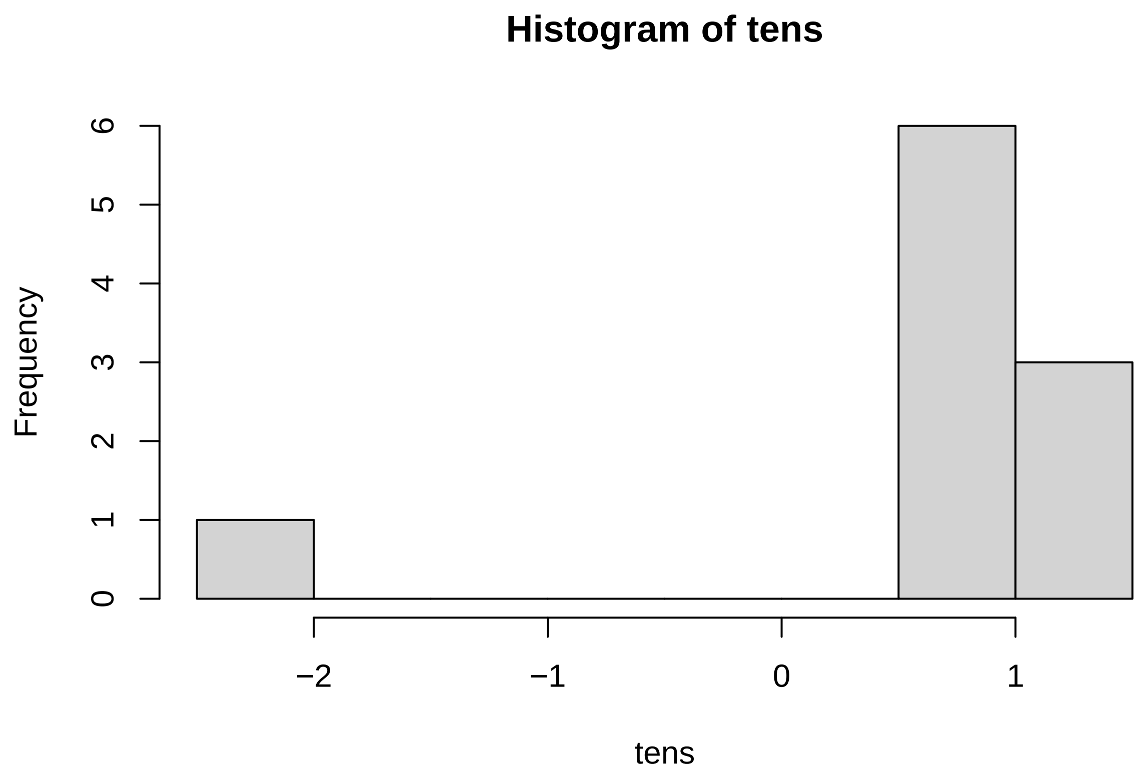
```
plot(annealing(-3.0875,1000))
```



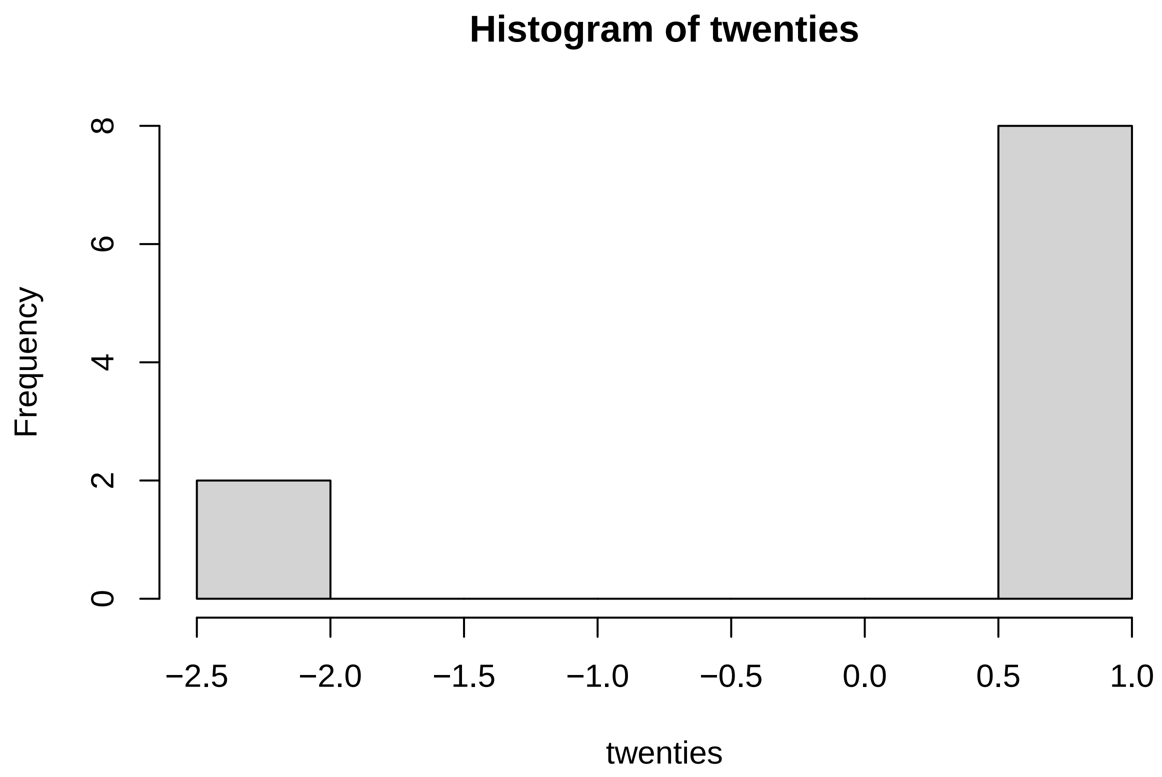
```
num_to_tag = function(num){

  num=num*2^(Dig-5)
  tag = rep(0, Dig)
  if (num < 0){
    tag[1] = 1
  }
  num = round(abs(num))
  for (i in 2:Dig){
    tag[i] = num%%2
    num = floor(num/2)
    if(num==0){
      break
    }
  }
  tag[2:Dig] = rev(tag[2:Dig])
  tag
}


tag_to_number = function(tag){
  sign = (-1)^(tag[1])
  tag = tag[2:length(tag)]
  tag = rev(tag)
  num = 0
  for (i in 1:length(tag)){
    num = num+2^(i-1)*tag[i]
  }
  num*sign/2^15
}



selection = function(tags, scores){
  N = length(scores)
  scores = scores - min(scores)
  prob = scores/ sum(scores)
  rand = runif(N)
  out = rep(0, N)
  for (r in 1:N){
    for(x in 1:N){
      rand[r] = rand[r] - prob[x]
      if(rand[r]<0){
        out[r] = x
        break
      }
    }
  }
  tags[out,]
}


crossover = function(tags){
  m = nrow(tags)/2
  for (i in 1:m){
    v1 = tags[i*2-1,]
    v2 = tags[i*2,]
    diff = v1!=v2
    childgene = round(runif(sum(diff)))
    childgene2 = round(runif(sum(diff)))
    tags[i*2-1, diff] = childgene
    tags[i*2, diff] = childgene2
  }
  tags
}

mutate = function(tags){
  r = nrow(tags)
  c = ncol(tags)
  mr = sample(1:r,2)
  mc = sample(1:c,2)
  tags[mr[1],mc[1]] = 1- tags[mr[1],mc[1]]
  tags[mr[2],mc[2]] = 1- tags[mr[2],mc[2]]
  tags
}


Dig=20
# start with some random number in (-3, 3)
```

```r
N=40

gene = function(N){
  nums = runif(N,-3, 3)
  # get their tag
  tags = matrix(0, nrow = N, ncol = Dig)
  for(i in 1:N){
    tags[i,] = num_to_tag(nums[i])
  }

  scores = rep(0,N)
  for(k in 1:1000){
    # calculate score
    for(i in 1:N){
      scores[i] =  llikehd(nums[i],0.1)
    }
    if (length(unique(scores)) == 1){
      # we've converged
      print("converge")
      return(nums[1])
    }

    # update tags base on score
    tags = selection(tags, scores)
    tags = crossover(tags)
    tags = mutate(tags)

    # update number base on tag
    for(i in 1:N){
      nums[i] = tag_to_number(tags[i,])
    }
  }

    for(i in 1:N){
      scores[i] =  llikehd(nums[i],0.1)
    }

  nums[which.max(scores)]

}
```

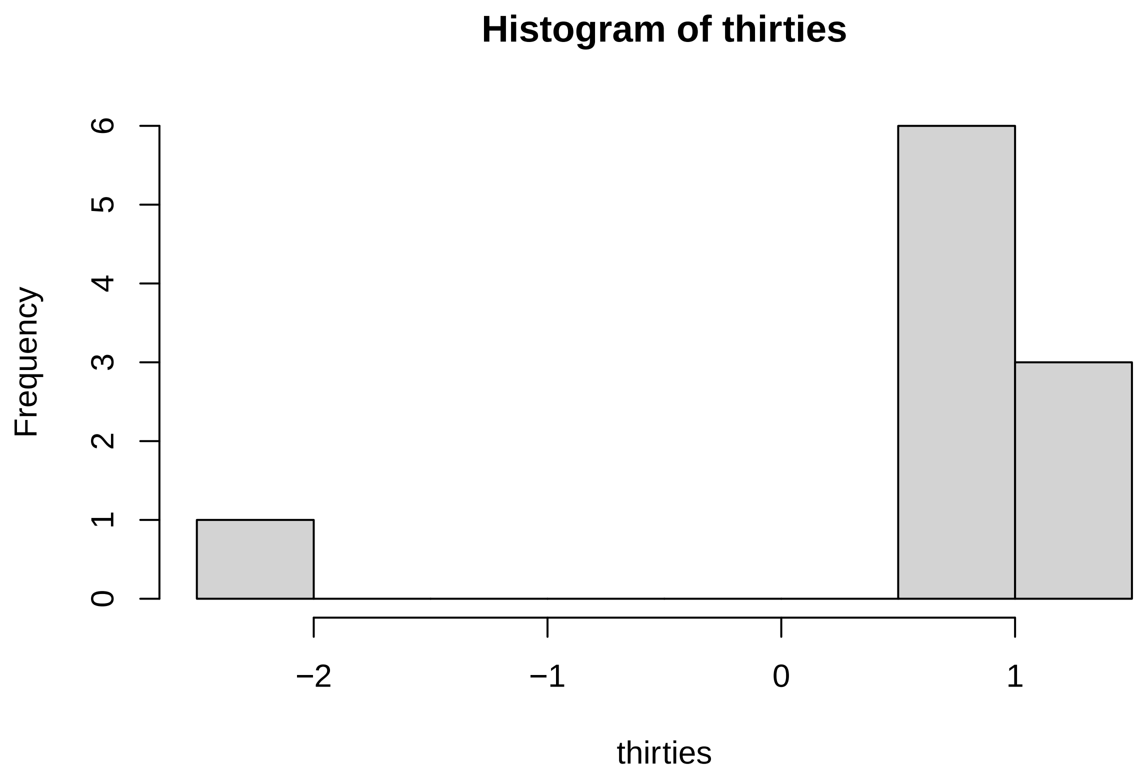
```r
tens = rep(0,10)
twenties= rep(0,10)
thirties= rep(0,10)
for(i in 1:10){
  tens[i] = gene(10)
  twenties[i] = gene(20)
  thirties[i] = gene(30)

}

hist(tens)
```



```r
hist(twenties)
```



```
hist(thirties)
```



we find population of size 10,20,30 all have good probablity of converge around 0.7, the global maximum. roughly speaking, the higher the polulation size, the more accurate our result.

4

```
x = c(24,18,21,5,5,11,11,17,6,7,20,13,4,16,19,21,4,22,8,17)
getw = function(p, l1, l2){
  p*dpois(x, l1) / (p*dpois(x, l1)+(1-p)*dpois(x, l2)  )
}


getQ = function(p, l1, l2){
  w = getw(p,l1,l2)
  sum( w*(x*log(l1) - l1 -log(factorial(x))) +
      (1-w)*(x*log(l2)-l2 -log(factorial(x))) +
        w*log(p) +(1-w)*log(1-p)
  )
}

getphat = function(p, l1, l2){
  w = getw(p,l1,l2)
  mean(w)
}

getl1hat = function(p, l1, l2){
  w = getw(p,l1,l2)
  sum(x*w)/sum(w)
}

getl2hat = function(p, l1, l2){
  w = getw(p,l1,l2)
  sum(x*(1-w))/sum(1-w)
}


it=0
E=c()

p=0.5
l1 = 5
l2 = 5
it=0

while (TRUE) {
  it = it+1
  E[it] = getQ(p, l1, l2)
  p = getphat(p, l1, l2)
  l1 = getl1hat(p, l1, l2)
  l2 = getl2hat(p, l1, l2)
  if(it>1){
    if (abs(E[it] - E[it-1])<0.0001){
      break
    }
  }
  if (it >10000){
    stop("no")
  }
```

```r
}

c(p, l1, l2)
```

```
## [1]  0.6066057 18.1301708  6.2333260
```

we did not get the optimum value, but a close to optimum value, note p=0.61, lambda1=18.1 lambda2 = 6.2 is equivalent to p =0.39, lambda1=6.2, lambda2=18.1