mixmodels_nlp_technote - Akash Gupta

STATS

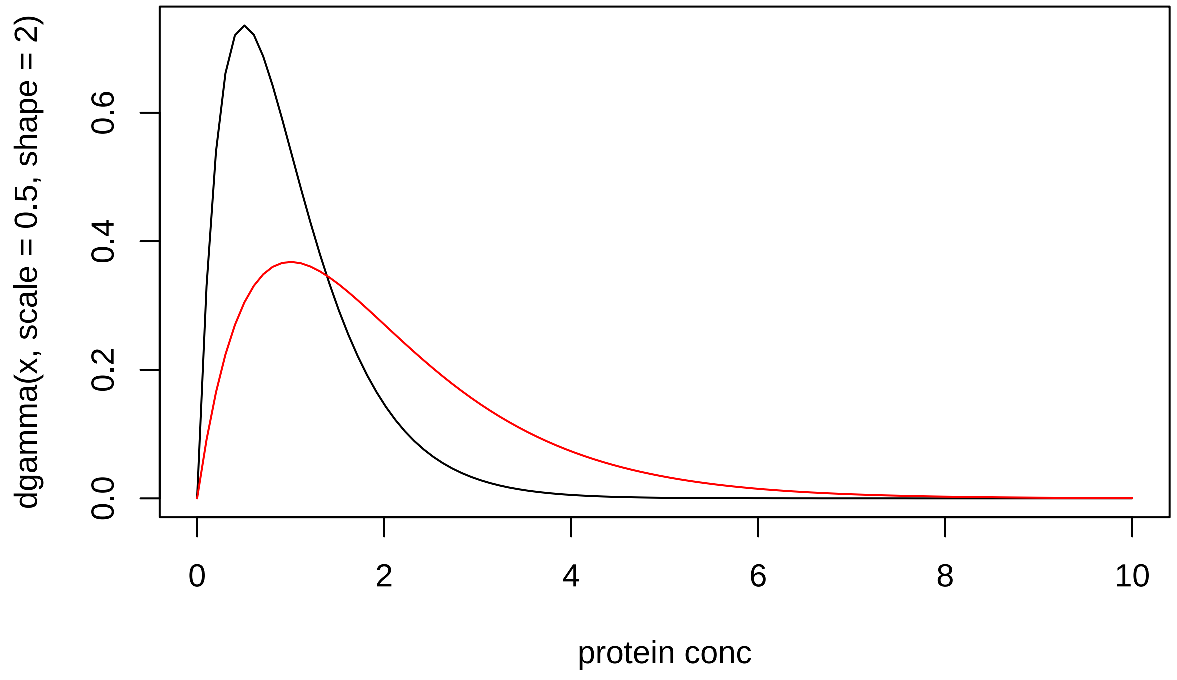
Each data observation contains some information about where it came from. Statistical approaches are used to essentially quantify this information and come to conslusions on the basis of that. Note that **likelihood ratio** is the ratio of probability of data being generated from two different models.

```r
x <- c(1,0,1,0,0,1)
fS <- c(0.40, 0.12, 0.21, 0.12, 0.02, 0.32)
fF <- c(0.80, 0.20, 0.11, 0.17, 0.23, 0.25)
L <- function(f, x){prod(f^x*(1-f)^(1-x))}
LR <- L(fS, x)/L(fF, x)
print(LR)
```

```
## [1] 1.81359
```

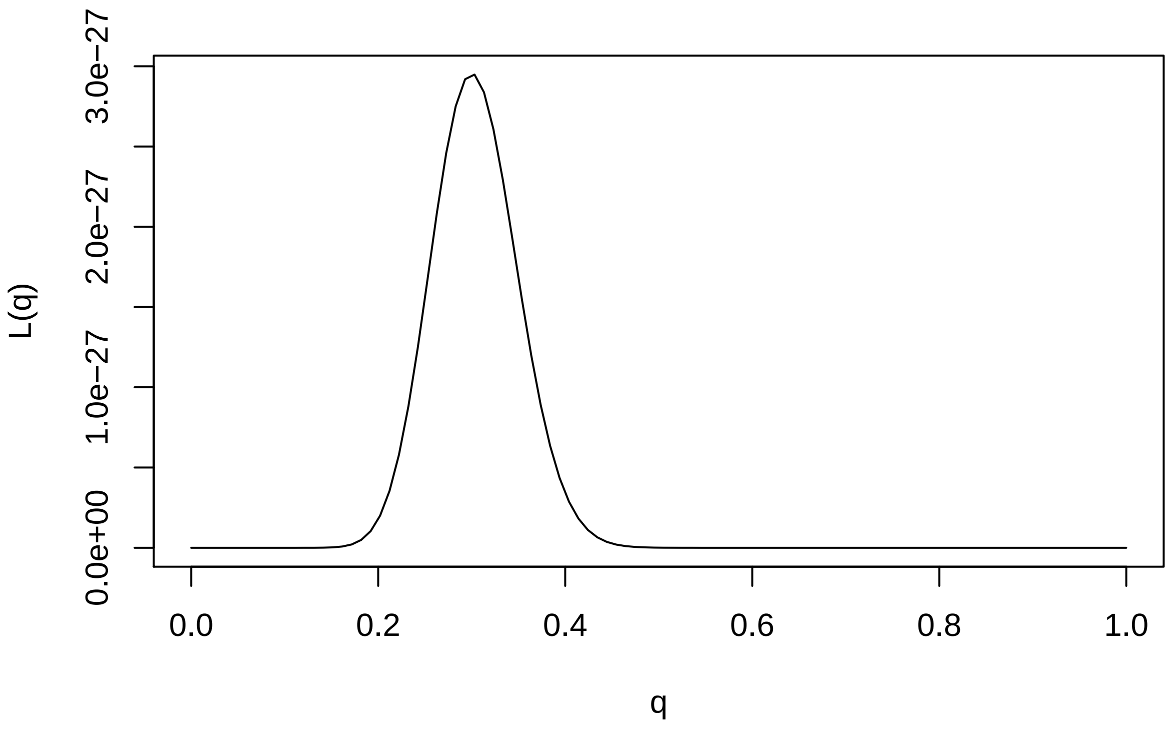
A likelihood ratio of 1.8 essentially means that the data is more likely to have come from model **fS** by a factor of 1.8 than from model **fF**. Now we will see plots of random variables that are continuous in nature.

```r
x = seq(0,10,length = 100)
plot(x, dgamma(x,scale = 0.5,shape = 2),type = "l",xlab = "protein conc")
lines(x, dgamma(x,scale = 1,shape = 2), type = "l", col = "red")
```



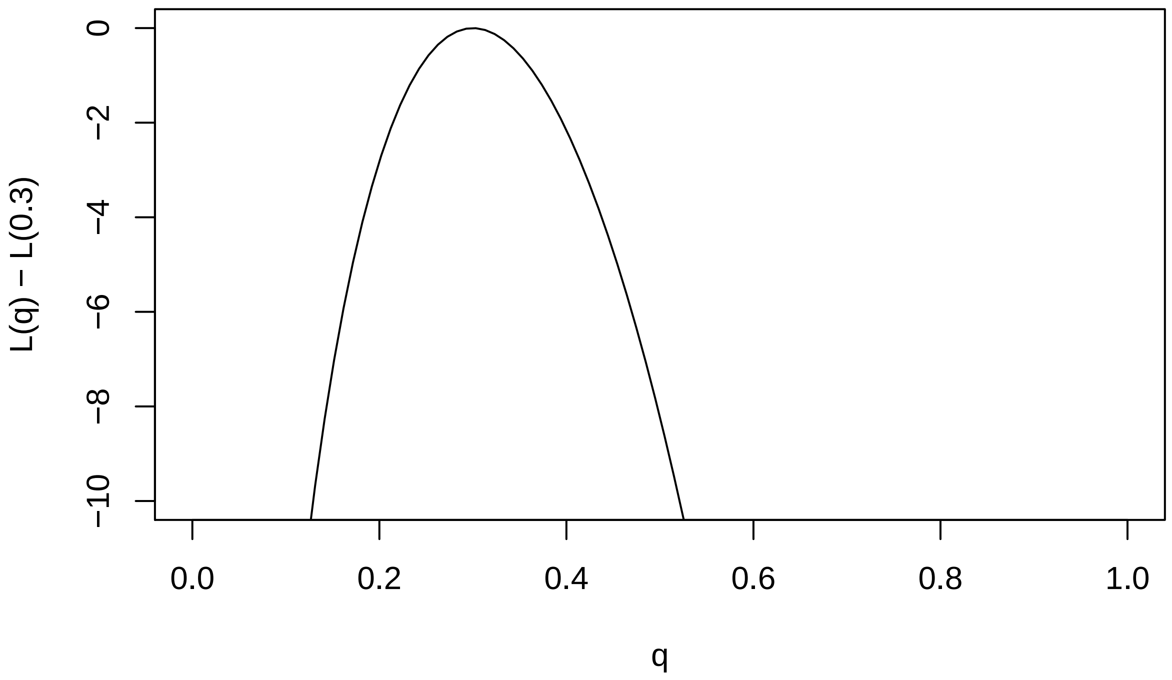
Plot a likelihood function to see what it looks like.

```r
q = seq(0,1,length = 100)
L <- function(q){q^30 * (1-q)^70}
plot(q,L(q),type = "l")
```



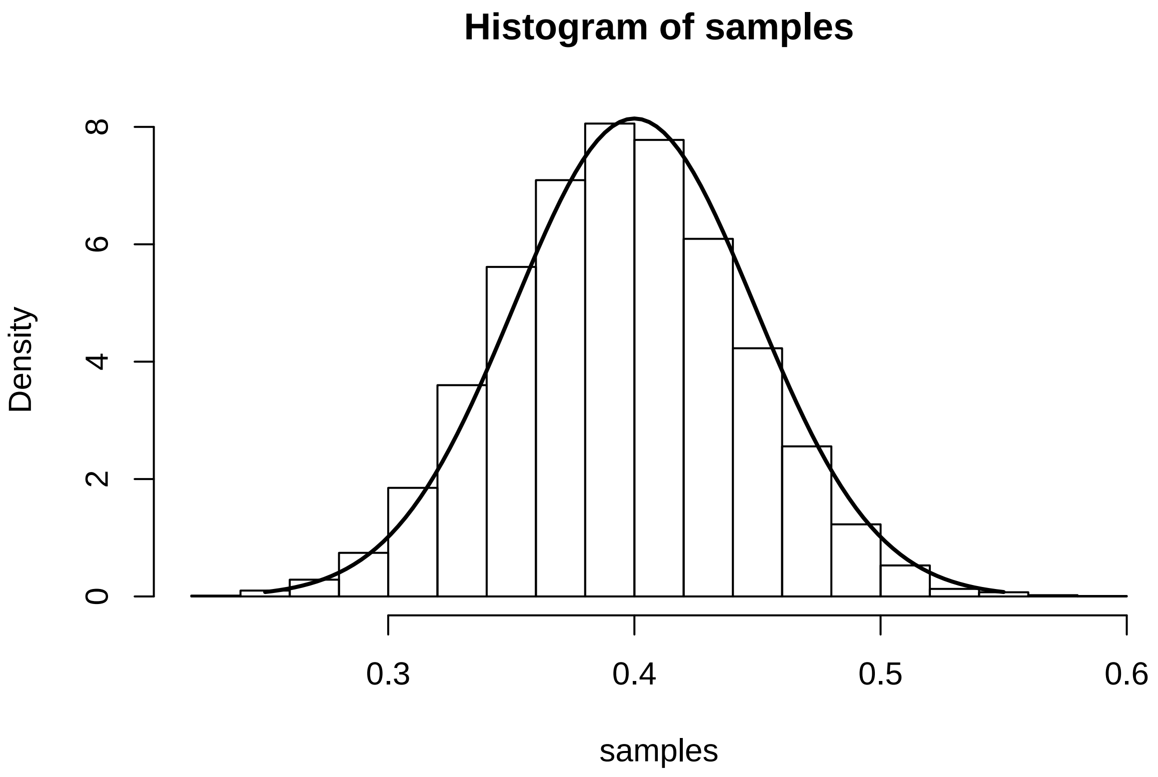
Computing the same distribution using log likelihood function.

```
q = seq(0,1,length = 100)
L <- function(q){30*log(q) + 70*log(1-q)}
plot(q,L(q)-L(0.3),type = "l",ylim = c(-10,0))
```



The MLE estimate can be thought of as the parameter value which is the mean of the distribution of parameter values derived from sampling a dataset many times. We will in the next example take 7000 samples of a dataset and draw 100 data observations from a **bernoulli distribution** with **p = 0.4**.

```r
num_iterations = 7000
p_truth = 0.4
num_samples = 100
samples = numeric(num_iterations)
for(iter in seq_len(num_iterations)){
  samples[iter] <- mean(rbinom(num_samples,1,p_truth))
}
hist(samples,freq = F)
curve(dnorm(x,mean=p_truth,sd=sqrt((p_truth*(1-p_truth)/num_samples))),0.25,0.55,
            lwd=2,add = T)
```



Now we will compute likelihood that data observations came from any of **K models**.

```r
x <- c(1,0,1,0,0,1)
ref_freqs <- rbind(
  c(0.39,0.14,0.22,0.12,0.03,0.38),
  c(0.41,0.10,0.18,0.12,0.02,0.28),
  c(0.40,0.11,0.22,0.11,0.01,0.3),
  c(0.75,0.25,0.11,0.18,0.25,0.25),
  c(0.85,0.15,0.11,0.16,0.21,0.26)
)
normalize <- function(x){return(x/sum(x))}
posterior_prob <- function(L_vec,pi_vec){return(normalize(L_vec*pi_vec))}
L = function(f,x){prod(f^x*(1-f)^(1-x))}
L_vec = apply(ref_freqs,1,L,x=x)
print(L_vec)
```

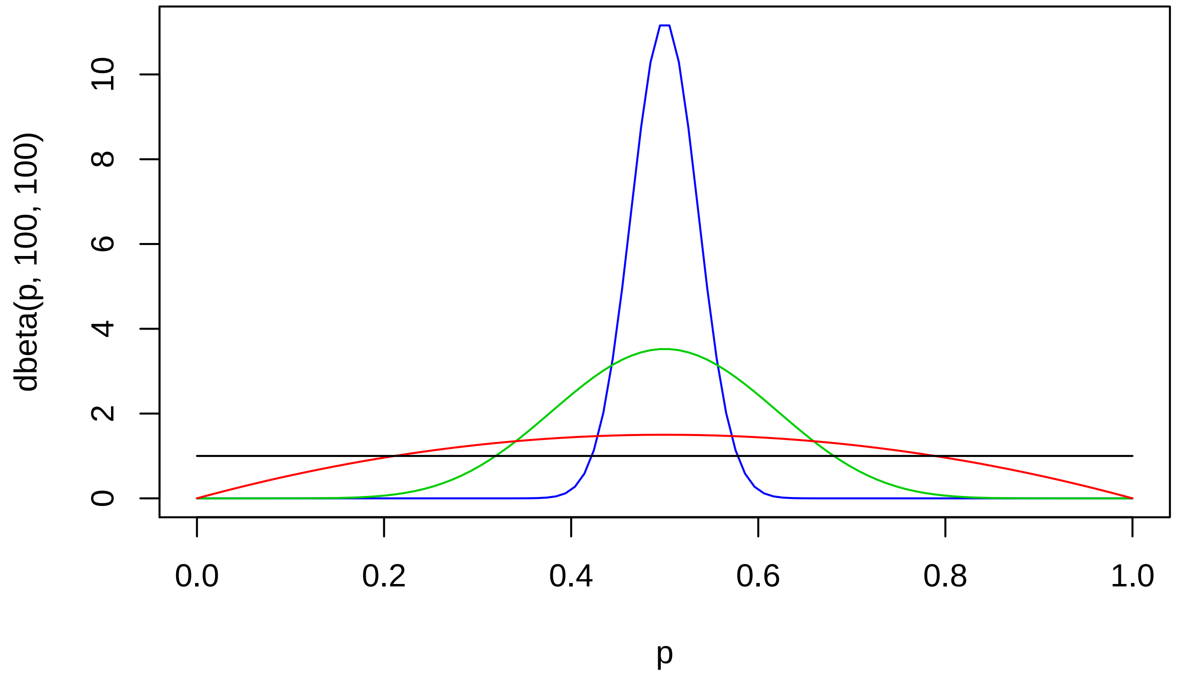
```
## [1] 0.023934466 0.016038570 0.020702326 0.009513281 0.013712299
```

```r
posterior_prob(L_vec, c(0.2,0.2,0.2,0.2,0.2))
```

```
## [1] 0.2852705 0.1911608 0.2467472 0.1133871 0.1634344
```

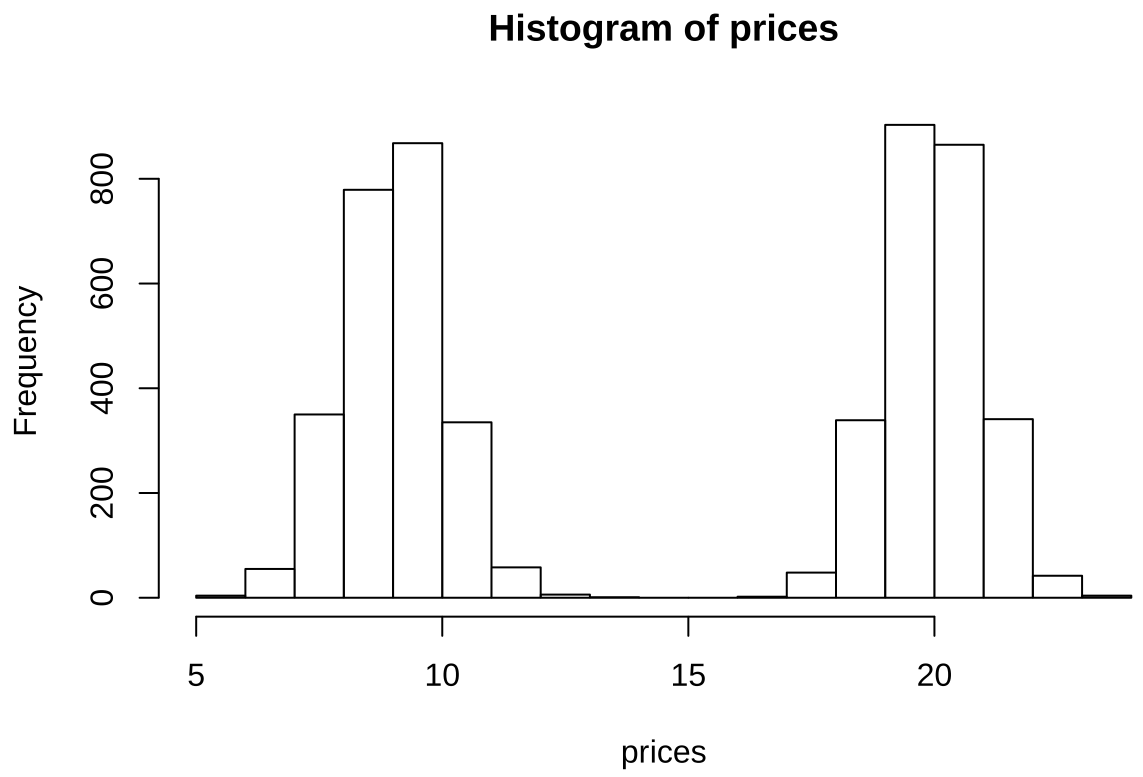
Before moving further into this topic, a quick look into the **beta distribution**.

```r
p = seq(0,1,length = 100)
plot(p, dbeta(p,100,100),type = "l",col = 4)
lines(p,dbeta(p,10,10),type = "l", col = 3)
lines(p, dbeta(p,2,2), type = "l", col = 2)
lines(p, dbeta(p,1,1), type = "l", col = 1)
```



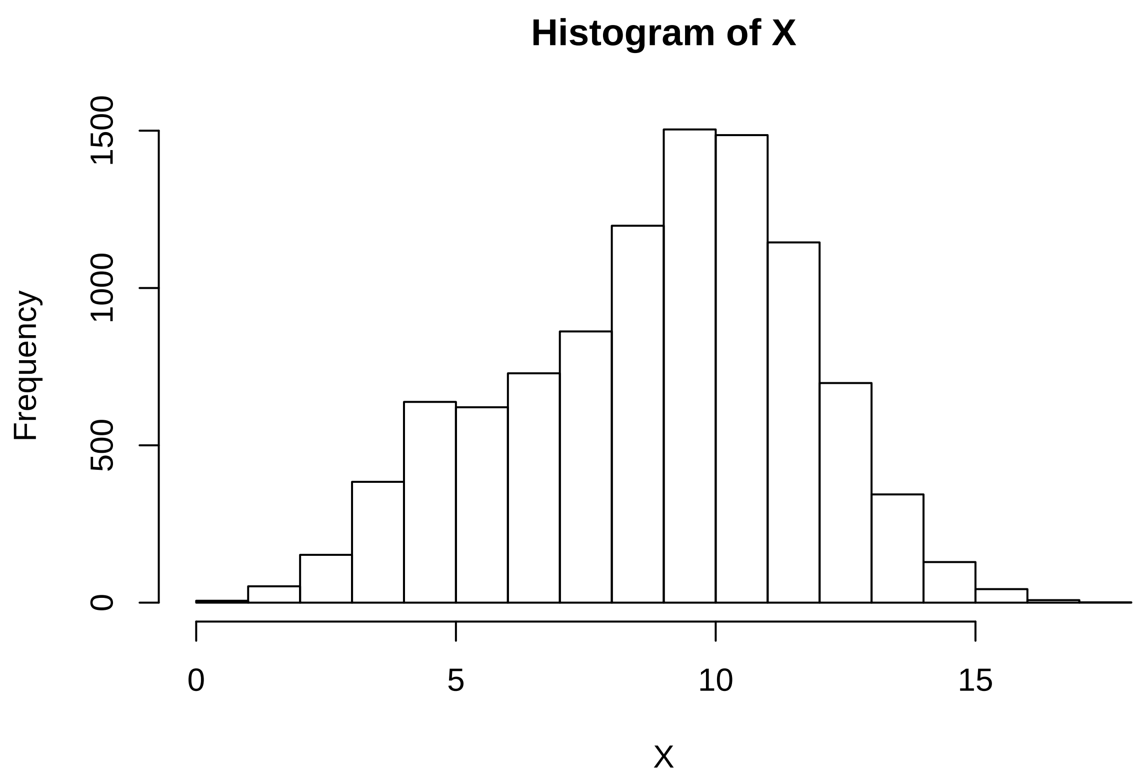
Now we will illustrate the concept of **mixture models**.

```r
num_samples <- 5000
prices <- numeric(num_samples)
for(i in seq_len(num_samples)){
  z_i <- rbinom(1,1,0.5)
  if(z_i == 0){prices[i] <- rnorm(1,mean=9,sd=1)}
  else{prices[i] <- rnorm(1,mean=20,sd=1)}
}
hist(prices)
```



Now we will emulate an **expectation maximization** algorithm with the aim of finding the maximum likelihood estimates of the mixture proportions.

```
mu_true <- c(5,10)
sigma_true <- c(1.5,2)
Z <- rbinom(500,1,0.75)
X <- rnorm(10000, mean = mu_true[Z+1], sd = sigma_true[Z+1])
hist(X,breaks = 15)
```



Now write down the log likelihood function that will be the criteria for convergence.

```
compute_log_like <- function(L,w){
  L[,1] = L[,1]*w[1]
  L[,2] = L[,2]*w[2]
  return(sum(log(rowSums(L))))
}
L = matrix(NA, nrow = length(X), ncol = 2)
L[,1] = dnorm(X, mean = mu_true[1], sd = sigma_true[1])
L[,2] = dnorm(X, mean = mu_true[2], sd = sigma_true[2])
```

Now to write down the EM algorithm.

```
mixtureEM <- function(w.init, L){
  w.curr <- w.init

  log_liks <- c()
  ll <- compute_log_like(L, w.curr)
  log_liks <- c(log_liks, ll)
  delta_ll <- 1

  while(delta_ll > 1e-5){
    w.curr <- EM.iter(w.curr, L)
    ll <- compute_log_like(L, w.curr)
    log_liks <- c(log_liks, ll)
    delta_ll <- log_liks[length(log_liks)] - log_liks[length(log_liks)-1]
  }
  return(list(w.curr, log_liks))
}

EM.iter <- function(w.curr, L, ...){
```

```r
  z_ik <- L
  for(i in seq_len(ncol(L))){
    z_ik[,i] <- w.curr[i]*z_ik[,i]
  }
  z_ik <- z_ik/rowSums(z_ik)

  w.next <- colSums(z_ik)/sum(z_ik)
  return(w.next)
}
```

Now we will perform the EM algorithm.

```r
ee <- mixtureEM(w.init = c(0.5,0.5), L)
print(ee)
```

```
## [[1]]
## [1] 0.2270206 0.7729794
##
## [[2]]
##  [1] -25436.93 -24303.71 -24238.19 -24233.60 -24233.26 -24233.23 -24233.23
##  [8] -24233.23 -24233.23 -24233.23
```