Nicolai_ex6

Andrea Nicolai

12/5/2020

Exercise 1

The number of particles emitted by a radioactive source during a fixed interval of time ($\Delta t = 10$ s) follows a Poisson distribution on the parameter μ. The number of particles observed during consecutive time intervals is: 4, 1, 3, 1 and 3.

```
observations <- c(4,1,3,1,3)

delta.mu <- 0.01
mu <- seq(0, 6, by = delta.mu)

median <- function(mu, vector.prob) {
  index <- 1
  for (x in mu) {
    integral <- delta.mu*sum(vector.prob[1:index])
    if (integral > 0.50) break
    index <- index + 1
  }
  return (x)
}
```

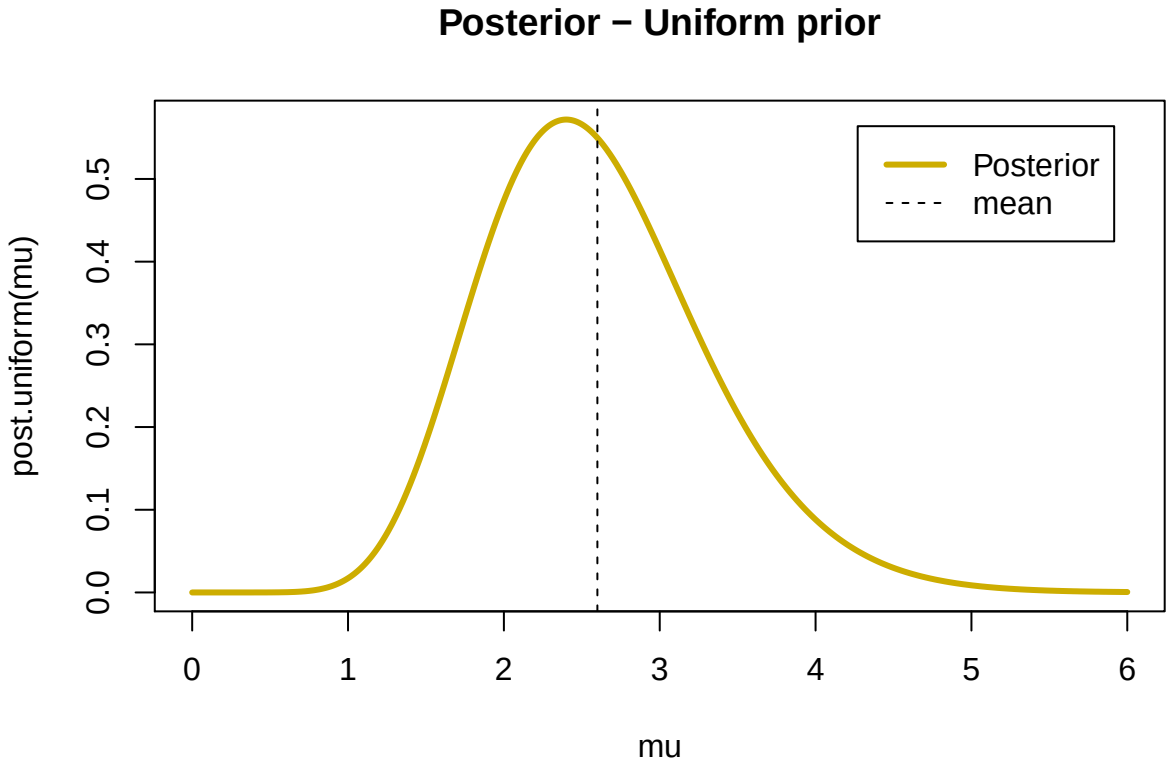
a) Suppose a uniform prior distribution for the parameter μ

Determine and draw the posterior distribution for μ, given the data

Evaluate mean, median and variance, both analytically and numerically in R

```
intervals <- length(observations)
tot_observations <- sum(observations)

alpha.post.unif  <- tot_observations + 1
lambda.post.unif <- intervals

post.uniform <- function(x) {
  return (dgamma(x, shape = alpha.post.unif , rate = lambda.post.unif ))
}


mean.analytical.u <- alpha.post.unif/lambda.post.unif
var.analytical.u  <- alpha.post.unif/(lambda.post.unif*lambda.post.unif)
median.analytical.u  <- qgamma(0.50, shape = alpha.post.unif , rate = lambda.post.unif )

mean.computat.u   <- delta.mu*sum(mu*post.uniform(mu))
E2 <- delta.mu*sum(mu*mu*post.uniform(mu))
var.computat.u <- E2 - mean.computat.u*mean.computat.u
```

```r
median.comput.u <- median(mu, post.uniform(mu))

message(sprintf("We obtain a mu of %1.2f for using the analytical formula, while a mu of %1.2f with the

## We obtain a mu of 2.60 for using the analytical formula, while a mu of 2.60 with the explicit computa
message(sprintf("We obtain a std of %1.2f for using the analytical formula, while a std of %1.2f with t

## We obtain a std of 0.52 for using the analytical formula, while a std of 0.52 with the explicit compu
message(sprintf("We obtain a median of %1.2f for using the analytical formula, while a median of %1.2f

## We obtain a median of 2.53 for using the analytical formula, while a median of 2.53 with the explici
plot( mu , post.uniform(mu), type ='l', main = "Posterior - Uniform prior" , col = 'gold3', lwd = 3)
abline (v= mean.analytical.u , col = 'black' , lty=2, xlab = 'mu', ylab = 'probability')
legend("topright", inset = 0.05 , c("Posterior", 'mean'), col = c('gold3', 'black') , lwd = c(3,1), lty
```



(b) Suppose a Jeffrey's prior for the parameter μ

Determine and draw the posterior distribution for μ, given the data. Evaluate mean, median and variance, both analytically and numerically in R.

```r
alpha.post.jeffrey  <- tot_observations + 1/2
lambda.post.jeffrey <- intervals

post.jeffrey <- function(x) {
  return (dgamma(x, shape = alpha.post.jeffrey , rate = lambda.post.jeffrey ))
}


mean.analytical.j <- alpha.post.jeffrey/lambda.post.jeffrey
var.analytical.j  <- alpha.post.jeffrey/(lambda.post.jeffrey*lambda.post.jeffrey)
```

```r
median.analytical.j  <- qgamma(0.50, shape = alpha.post.jeffrey , rate = lambda.post.jeffrey )

mean.computat.j   <- delta.mu*sum(mu*post.jeffrey(mu))
E2 <- delta.mu*sum(mu*mu*post.jeffrey(mu))
var.computat.j <- E2 - mean.computat.j*mean.computat.j
median.comput.j <- median(mu, post.jeffrey(mu))

message(sprintf("We obtain a mu of %1.2f for using the analytical formula, while a mu of %1.2f with the
```

```
## We obtain a mu of 2.50 for using the analytical formula, while a mu of 2.50 with the explicit computa
```

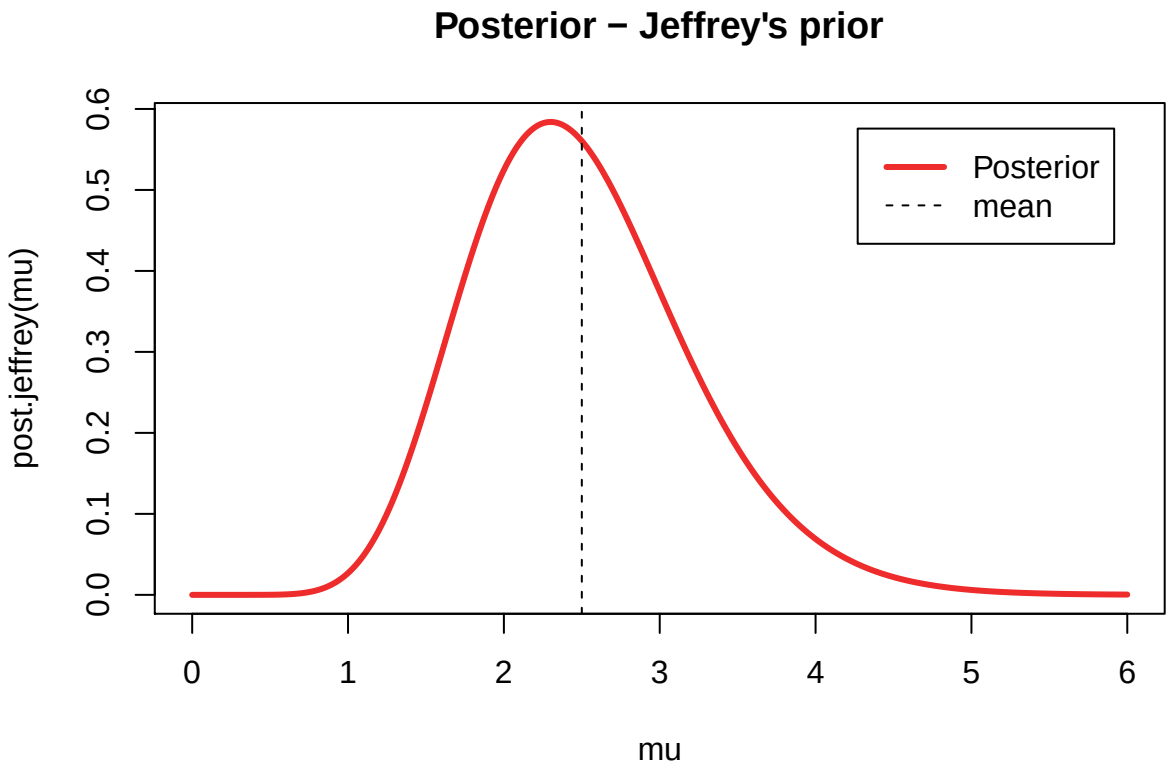
```r
message(sprintf("We obtain a std of %1.2f for using the analytical formula, while a std of %1.2f with t
```

```
## We obtain a std of 0.50 for using the analytical formula, while a std of 0.50 with the explicit compu
```

```r
message(sprintf("We obtain a median of %1.2f for using the analytical formula, while a median of %1.2f
```

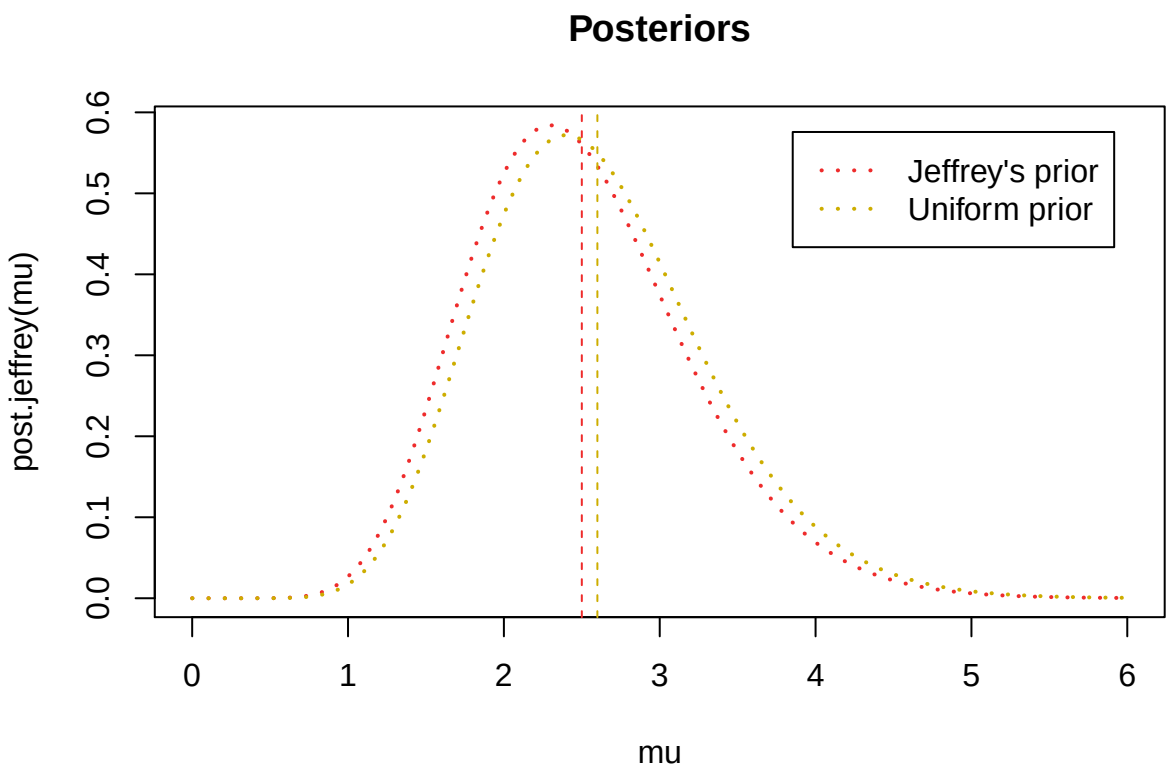
```
## We obtain a median of 2.43 for using the analytical formula, while a median of 2.43 with the explici
```

```r
plot( mu , post.jeffrey(mu), type ='l', main = "Posterior - Jeffrey's prior" , col = 'firebrick2', lwd =
abline (v= mean.analytical.j , col = 'black' , lty=2, xlab = 'mu', ylab = 'probability')
legend("topright", inset = 0.05 , c("Posterior", 'mean'), col = c('firebrick2', 'black') , lwd = c(3,1)
```



Let us compare the two priors.

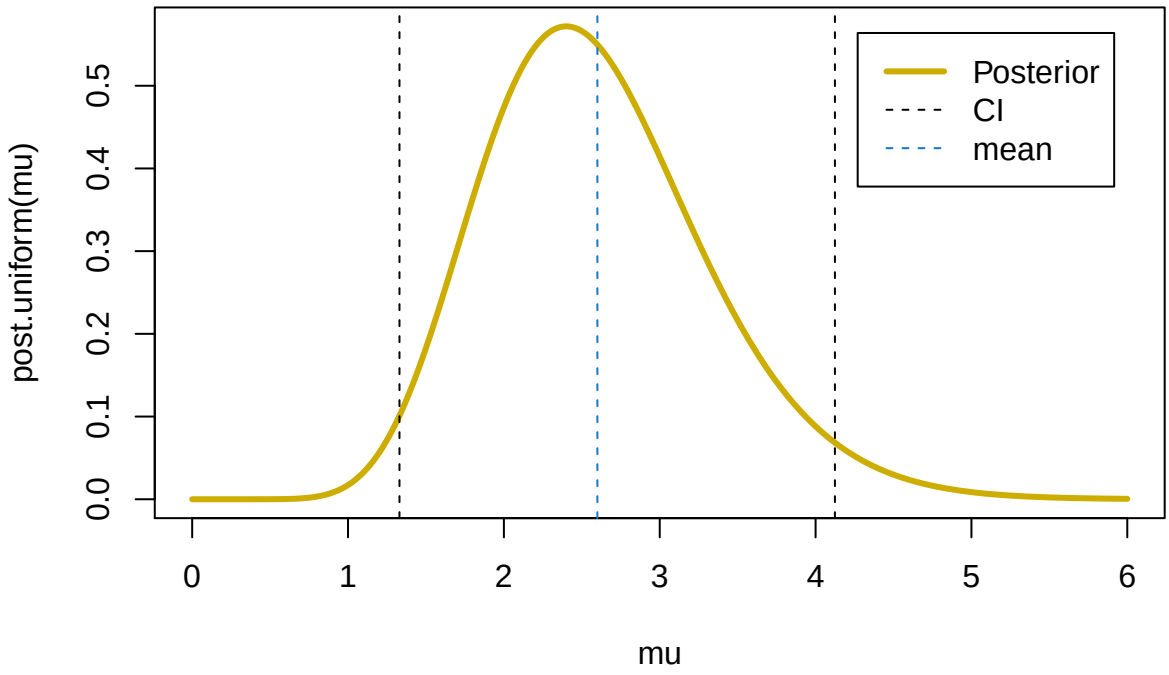
```r
plot( mu , post.jeffrey(mu), type ='l', main = "Posteriors" , col = 'firebrick2', lwd = 2, lty = 3)
abline (v= mean.analytical.j , col = 'firebrick2' , lty=2, lwd = 1)

lines( mu, post.uniform(mu), col = 'gold3', lwd = 2, lty = 3)
abline (v= mean.analytical.u , col = 'gold3' , lty=2, lwd = 1)

legend("topright", inset = 0.05 , c("Jeffrey's prior", 'Uniform prior'), col = c('firebrick2', 'gold3')
```



Evaluate a 95% credibility interval for the results obtained with both priors. Compare the result with that obtained using a normal approximation for the posterior distribution, with the same mean and standard deviation.

We want to choose the interval in order symmetric with respect to the mean but adding a small scale "to the right", in order to not discard at all the fact that the distribution is asymmetric (Skewness $\simeq 0.55$).

```
asymmetry.factor <- 1.20
confidence <- c()


#uniform case
for (single_value in mu){
  integral <- (pgamma(mean.analytical.u  + single_value*asymmetry.factor, alpha.post.unif, lambda.post.u
  if (integral > 0.95) {
    confidence <- c(confidence, single_value)
    break
    }
}
#mean.analytical.u
#median.analytical.u

lowerbound.u <- mean.analytical.u - confidence
upperbound.u <- mean.analytical.u + confidence*asymmetry.factor

plot( mu , post.uniform(mu), type ='l', main = "Uniform prior - CL interval" , col = 'gold3', lwd = 3)
abline (v= lowerbound.u , col = 'black' , lty=2)
abline (v= upperbound.u , col = 'black' , lty=2)
abline (v= mean.analytical.u , col = 'dodgerblue3' , lty=2)
legend("topright", inset = 0.05 , c("Posterior", 'CI', 'mean'), col = c('gold3', 'black', 'dodgerblue3'
```




```
message(sprintf("For the uniform prior we obtain a 95percent confidence interval that is [%1.2f, %1.2f]
```

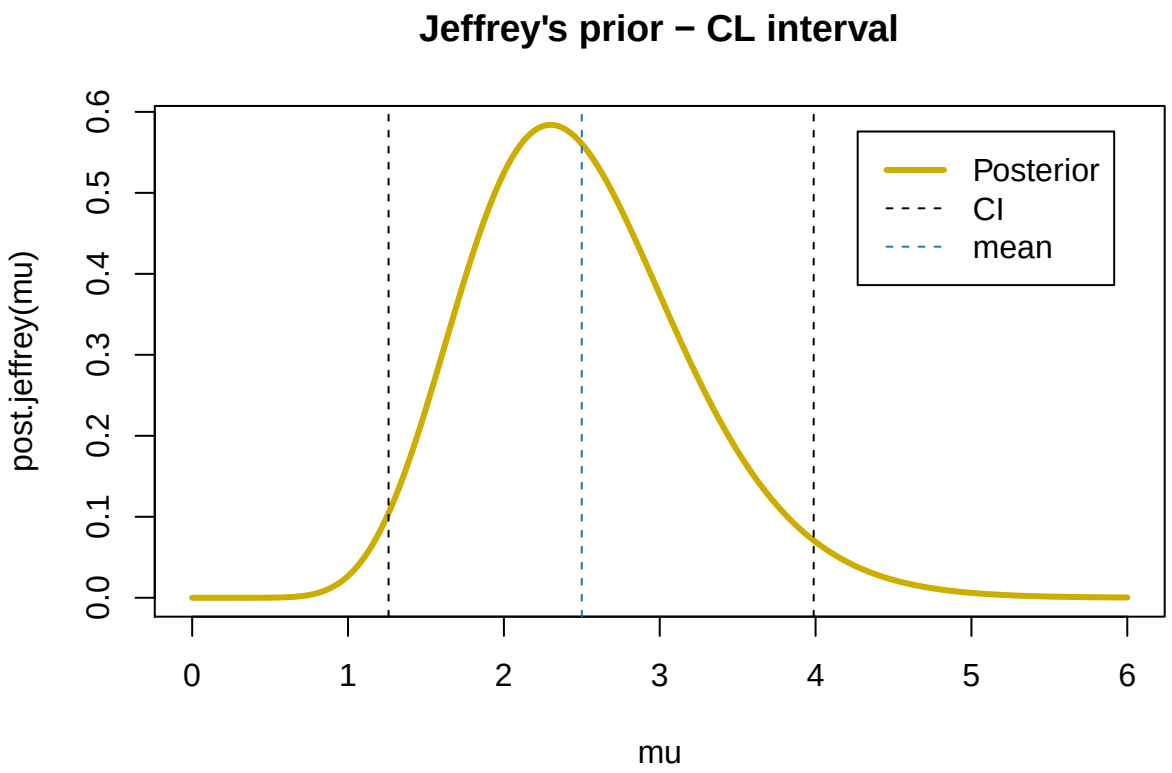
```
## For the uniform prior we obtain a 95percent confidence interval that is [1.33, 4.12]
```

```
message(sprintf("\nFor a Normal distribution we know that the 95percent interval is centered on the mea
```

```
##
## For a Normal distribution we know that the 95percent interval is centered on the mean and is fairly
```

```
asymmetry.factor <- 1.20
confidence <- c()


#uniform case
for (single_value in mu){
  integral <- (pgamma(mean.analytical.j  + single_value*asymmetry.factor, alpha.post.jeffrey, lambda.pos
  if (integral > 0.95) {
    confidence <- c(confidence, single_value)
    break
    }
}
#mean.analytical.j
#median.analytical.u

lowerbound.j <- mean.analytical.j - confidence
upperbound.j <- mean.analytical.j + confidence*asymmetry.factor

plot( mu , post.jeffrey(mu), type ='l', main = "Jeffrey's prior - CL interval" , col = 'gold3', lwd = 3
abline (v= lowerbound.j , col = 'black' , lty=2)
abline (v= upperbound.j , col = 'black' , lty=2)
abline (v= mean.analytical.j , col = 'dodgerblue3' , lty=2)
```

```
legend("topright", inset = 0.05 , c("Posterior", 'CI', 'mean'), col = c('gold3', 'black', 'dodgerblue3'
```




```
message(sprintf("For the Jeffrey's prior we obtain a 95percent confidence interval that is [%1.2f, %1.2
```

```
## For the Jeffrey's prior we obtain a 95percent confidence interval that is [1.26, 3.99]
```

```
message(sprintf("\nFor a Normal distribution we know that the 95percent interval is centered on the mea
```

```
##
## For a Normal distribution we know that the 95percent interval is centered on the mean and is fairly
```

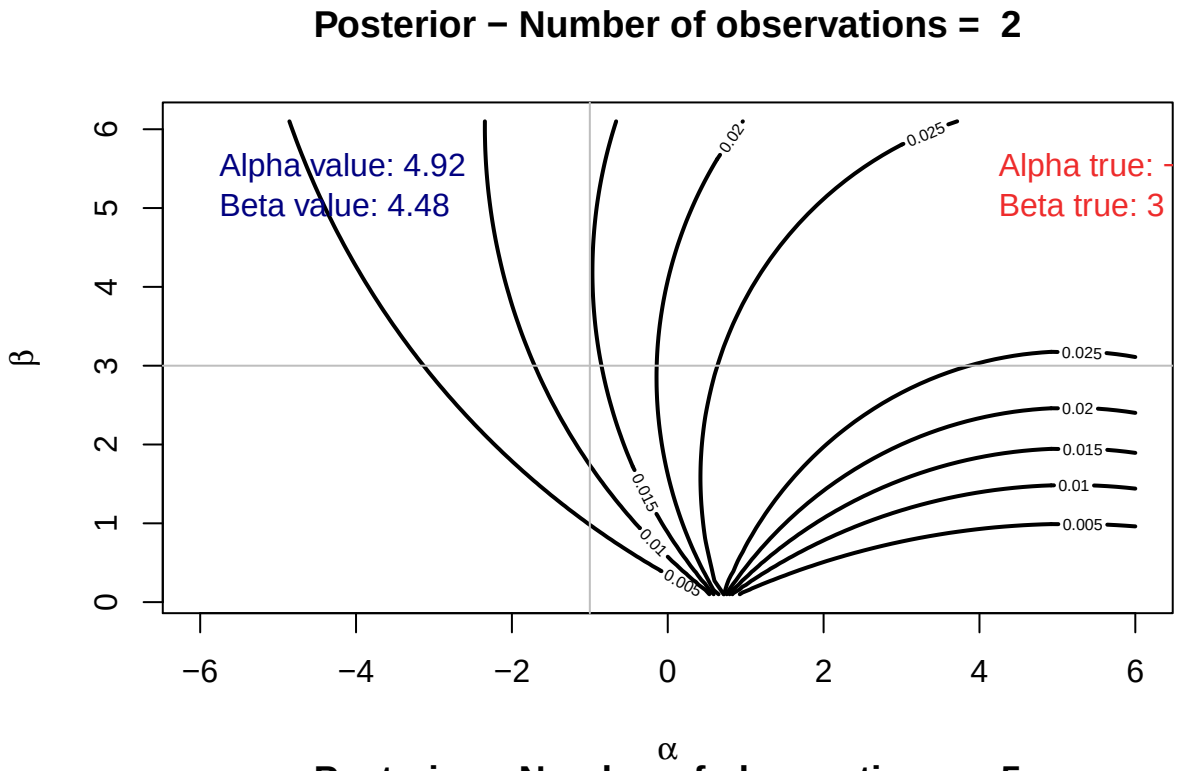
As a conclusion we can note that the normal distribution is an approximation, because it does not take into account the fact that both posteriors are asymmetric.

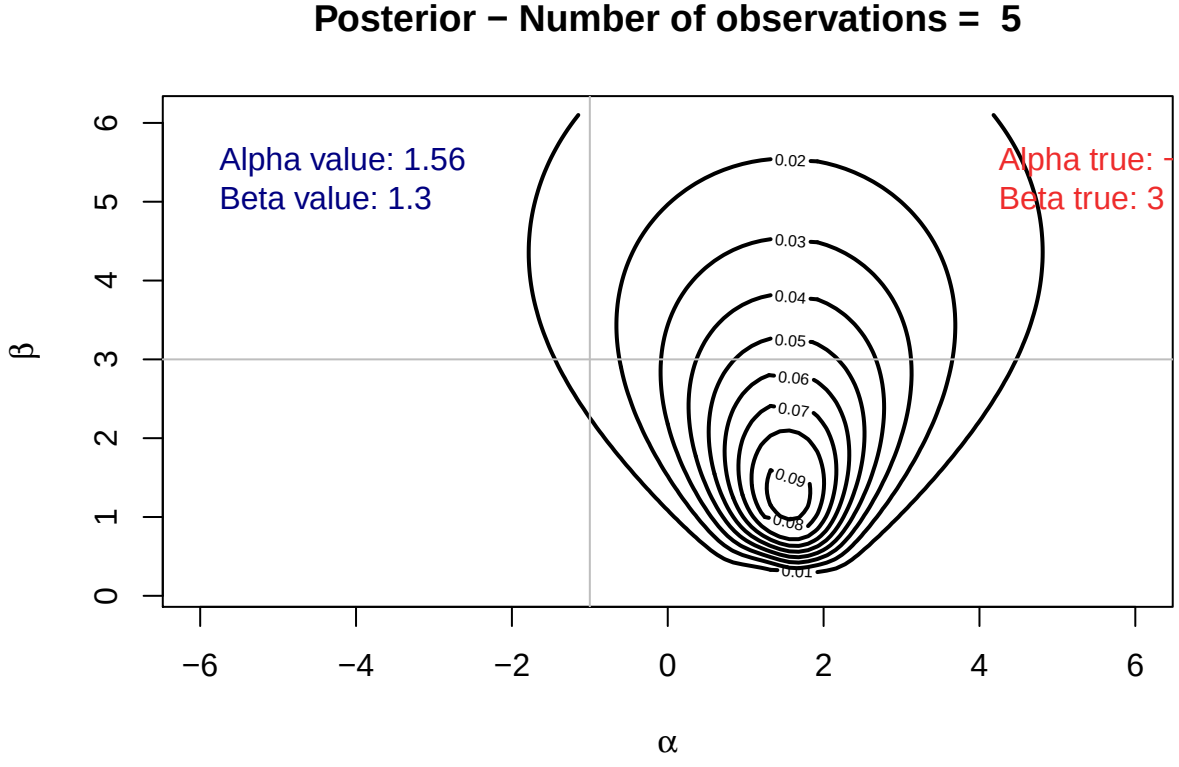
Exercise 2

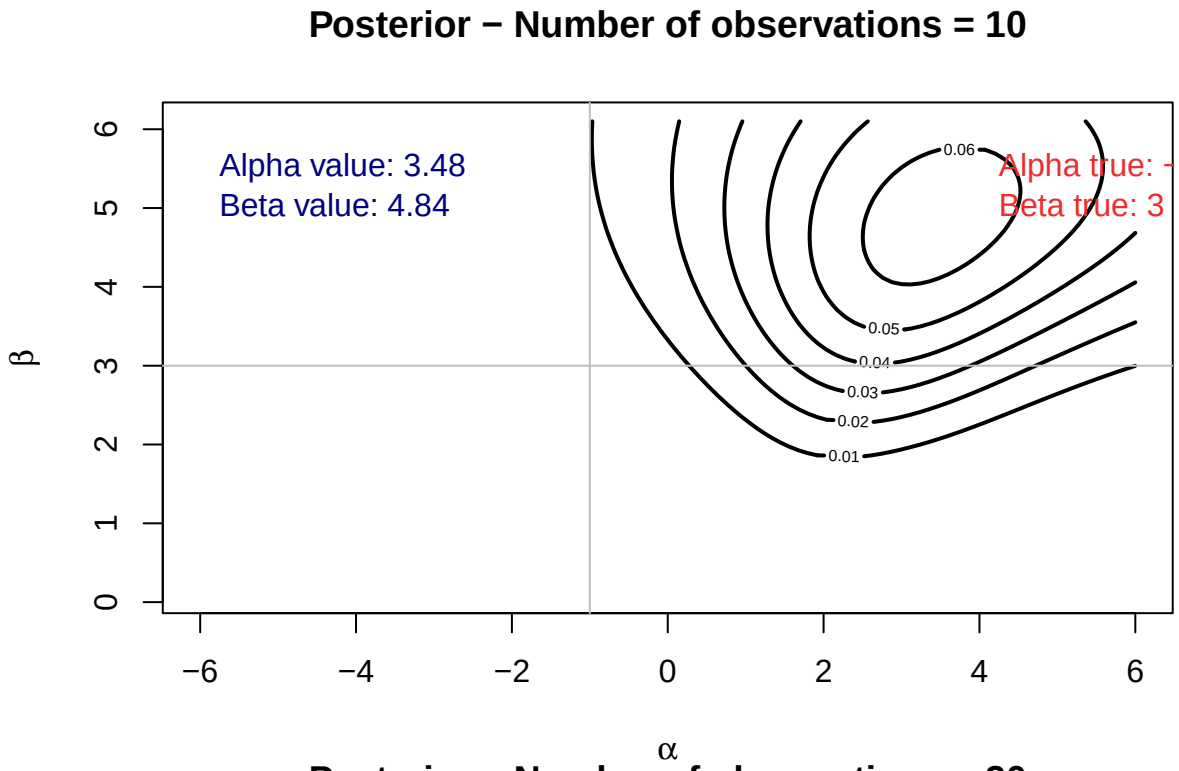
Given the problem of the lightouse discussed last week, study the case in which both the position the shore (α) and the distance Parameter out at sea estimation (β) are unknown.

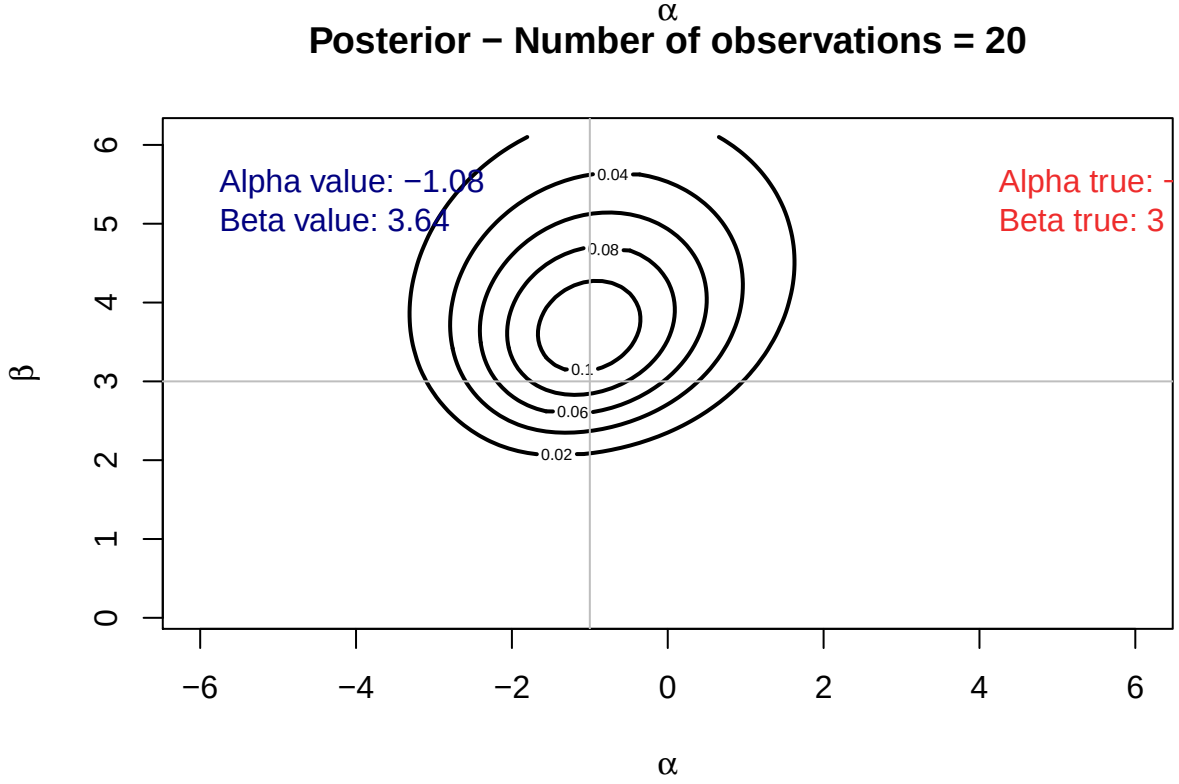
```
seed <- 1968
set.seed(seed)
true.alpha <- -1
true.beta <- 3

observations <- 250
data <- rcauchy(observations, true.alpha, true.beta) #+ rnorm(observations, 0, 0.4)
#data <- data[data > -6 & data < 6]

n <- c(2,5,10,20,35,50,100,150, 200)

post.likelihood.fun <- function(data, alpha, beta){
  if(alpha < -6 || alpha > 6 || beta < 0) { return ( 0 )}
```

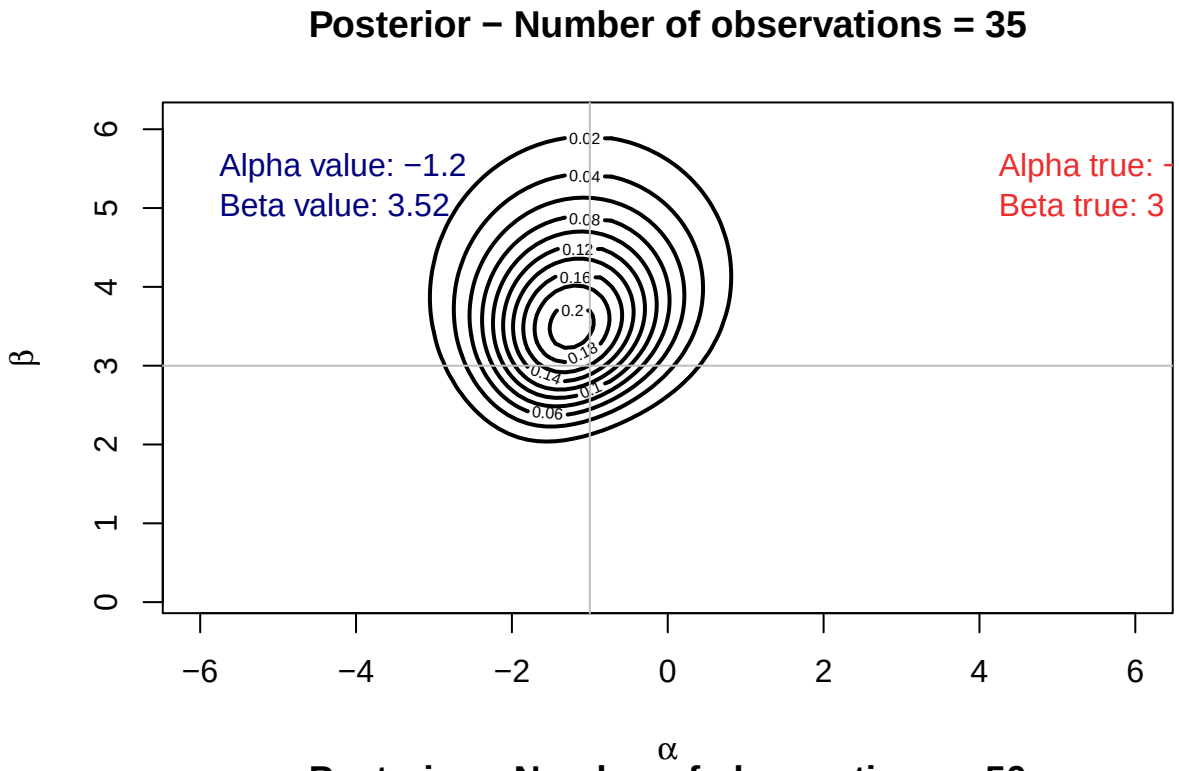
```
  likelihood <- 1
  for (x in data) {
    likelihood <- likelihood*dcauchy(x, alpha, beta)
  }
  return (likelihood)
}


n.sample <- 100
x.min <- -6; x.max <- +6
h <- (x.max - x.min )/n.sample

dist.min <- 0.1; dist.max <- 6.1
g <- (dist.max - dist.min)/n.sample

alpha <- seq(from= x.min      , by=h  , length.out = n.sample +1)
beta  <- seq(from= dist.min   , by=g  , length.out = n.sample +1)


for (samples in n){

dt <- data[1:samples]

z <- matrix (data = NA , nrow = length(alpha), ncol = length(beta))
  for(j in 1: length(alpha)) {
    for(k in 1: length(beta)) {
      z[j,k] <- post.likelihood.fun(dt, alpha[j], beta[k])
    }
  }
z.norm <- h*g*sum(z)
z <- z/z.norm

contour (alpha, beta, z, nlevels = 8,labcex = 0.5,lwd = 2, xlab=expression(alpha), ylab=expression(beta)
abline (v=true.alpha, h=true.beta, col="grey")
ind <- which(z == max(z), arr.ind = TRUE)
alpha.max <- alpha[ind[1]]
beta.max <- beta[ind[2]]
text(x = -6, y = 5.5 , col="navy ", lwd = 2, pos=4, paste("Alpha value: ", alpha.max , sep=""))
text(x = -6, y = 5.0 , col="navy ", lwd = 2, pos=4, paste("Beta value: ", beta.max , sep=""))

text(x = +4, y = 5.5 , col="firebrick2 ", lwd = 2, pos=4, paste("Alpha true: ", true.alpha , sep=""))
text(x = +4, y = 5.0 , col="firebrick2 ", lwd = 2, pos=4, paste("Beta true: ", true.beta , sep=""))
}
```

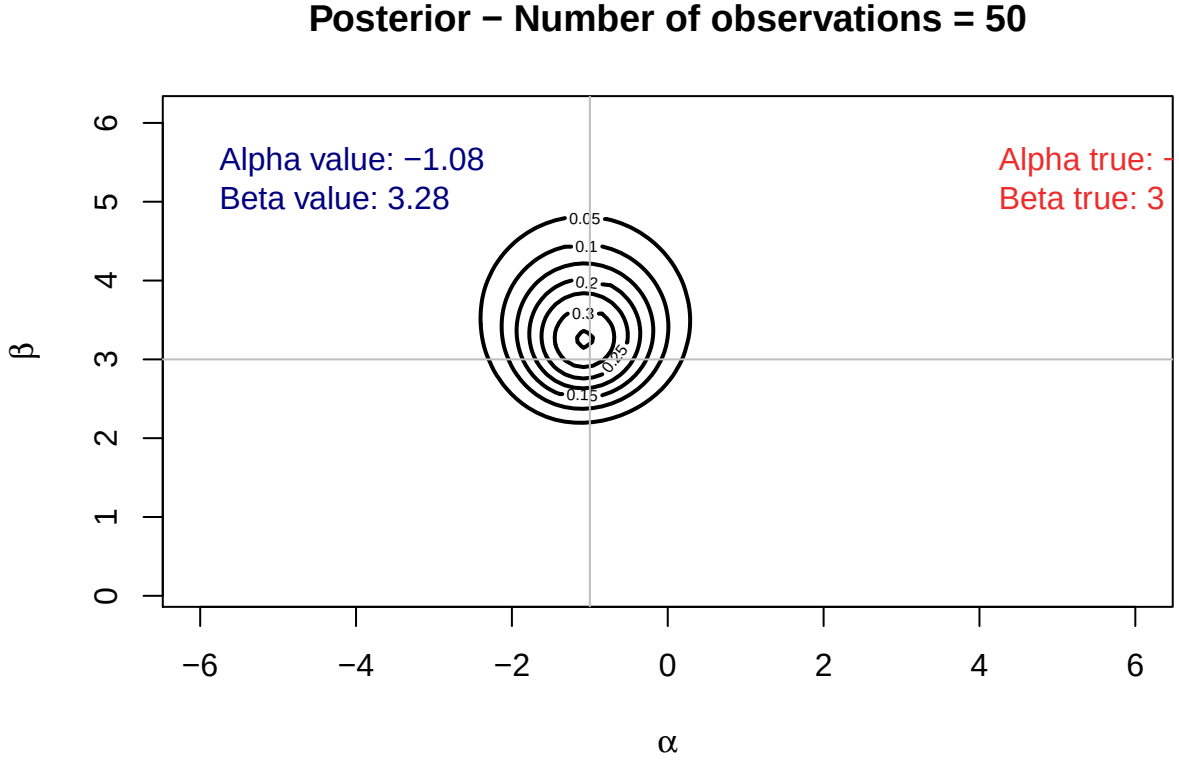

Posterior − Number of observations = 2
Alpha value: 4.92
Beta value: 4.48
Alpha true: -
Beta true: 3
0.025
0.02
0.015
0.01
0.005
β
α

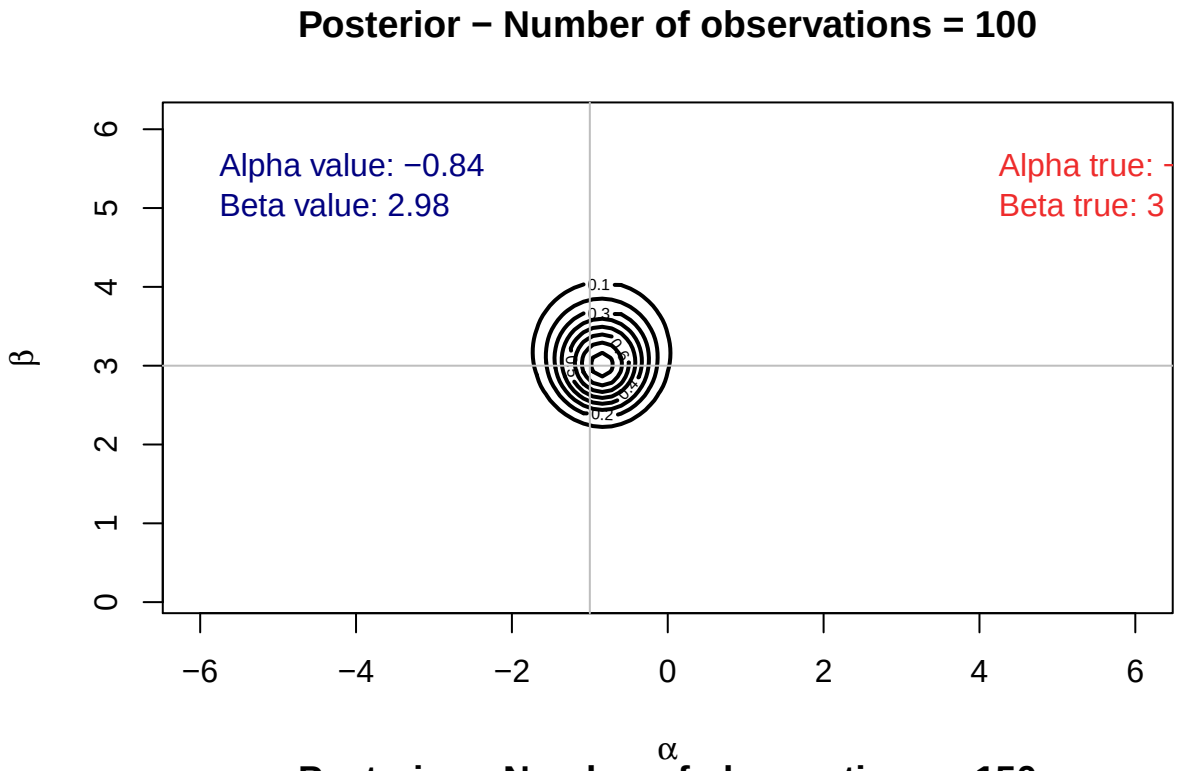

Posterior − Number of observations = 5
Alpha value: 1.56
Beta value: 1.3
Alpha true: -
Beta true: 3
0.02
0.03
0.04
0.05
0.06
0.07
0.09
0.08
0.01
β
α

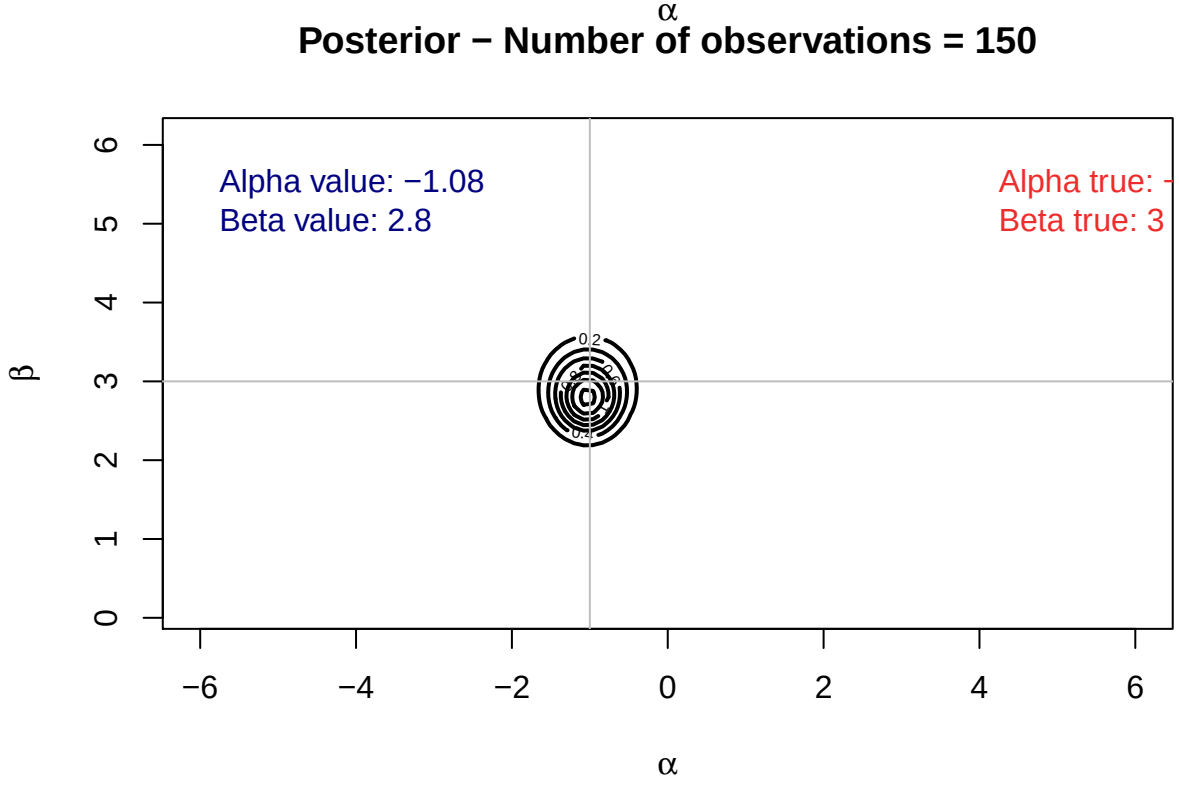

Posterior − Number of observations = 10
Alpha value: 3.48
Beta value: 4.84
Alpha true: -
Beta true: 3
0.06
0.05
0.04
0.03
0.02
0.01
β
α
0
1
2
3
4
5
6
−6
−4
−2
0
2
4
6

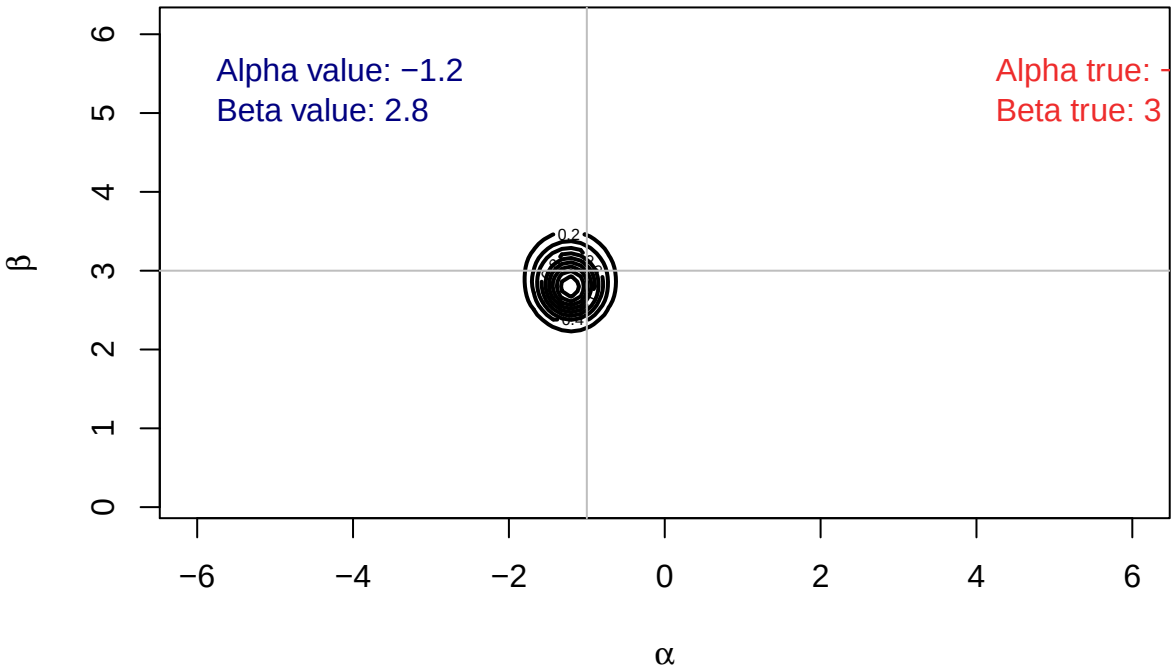

Posterior − Number of observations = 20
Alpha value: −1.08
Beta value: 3.64
Alpha true: -
Beta true: 3
0.04
0.08
0.1
0.06
0.02
β
α
0
1
2
3
4
5
6
−6
−4
−2
0
2
4
6


Posterior − Number of observations = 35
Alpha value: −1.2
Beta value: 3.52
Alpha true: -
Beta true: 3
0.02
0.04
0.08
0.12
0.16
0.2
0.18
0.14
0.1
0.06
β
α
−6
−4
−2
0
2
4
6
0
1
2
3
4
5
6


Posterior − Number of observations = 50
Alpha value: −1.08
Beta value: 3.28
Alpha true: -
Beta true: 3
0.05
0.1
0.2
0.3
0.25
0.15
β
α
−6
−4
−2
0
2
4
6
0
1
2
3
4
5
6

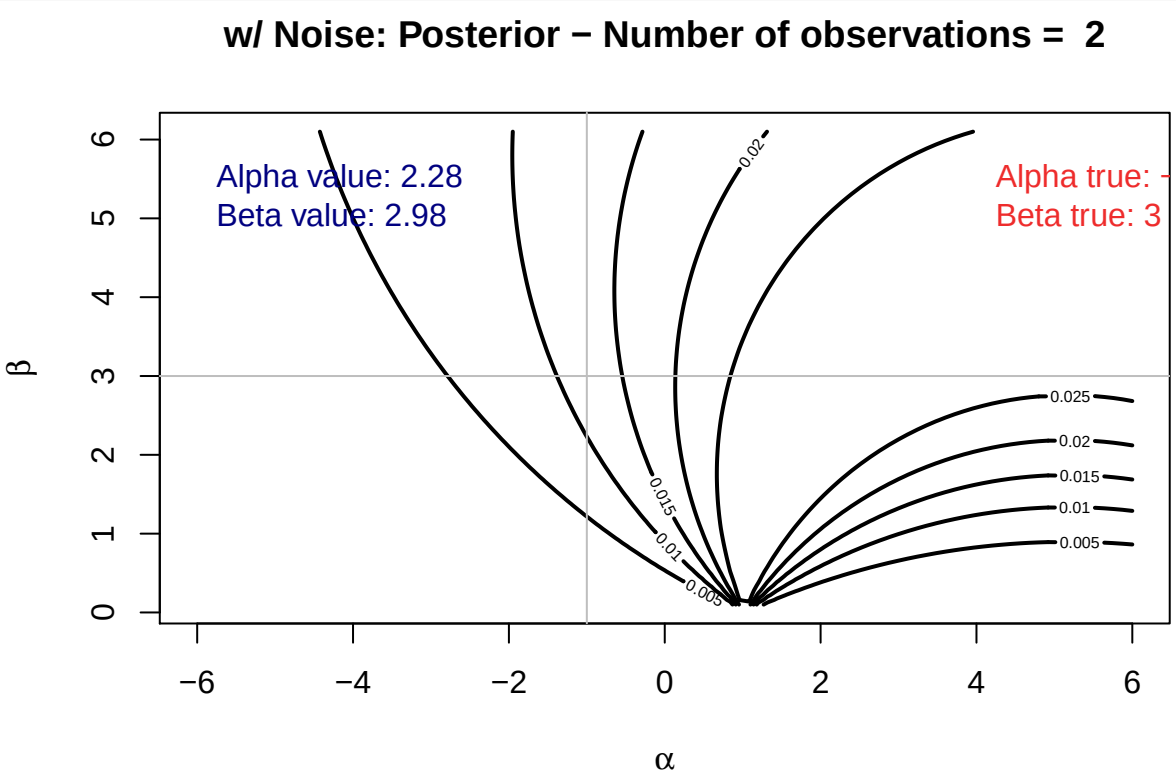

Posterior − Number of observations = 100
Alpha value: −0.84
Beta value: 2.98
Alpha true: -
Beta true: 3
β
α
0
1
2
3
4
5
6
−6
−4
−2
0
2
4
6

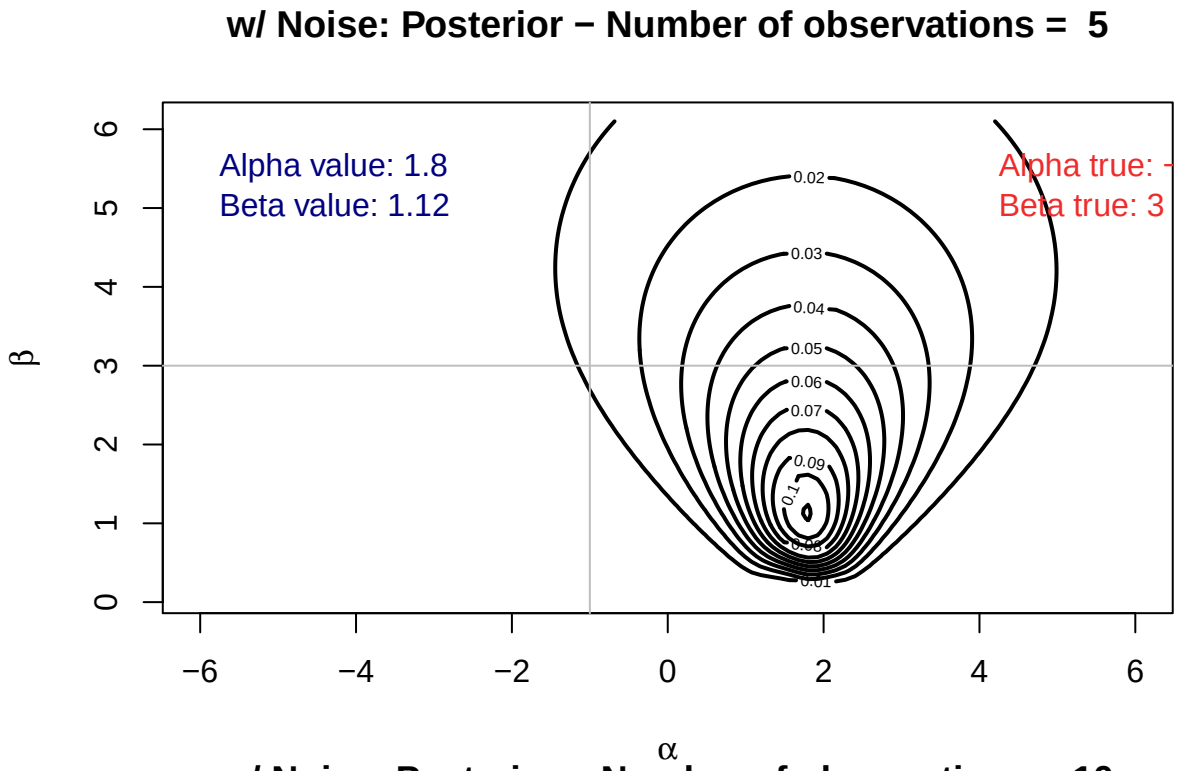

Posterior − Number of observations = 150
Alpha value: −1.08
Beta value: 2.8
Alpha true: -
Beta true: 3
β
α
0
1
2
3
4
5
6
−6
−4
−2
0
2
4
6

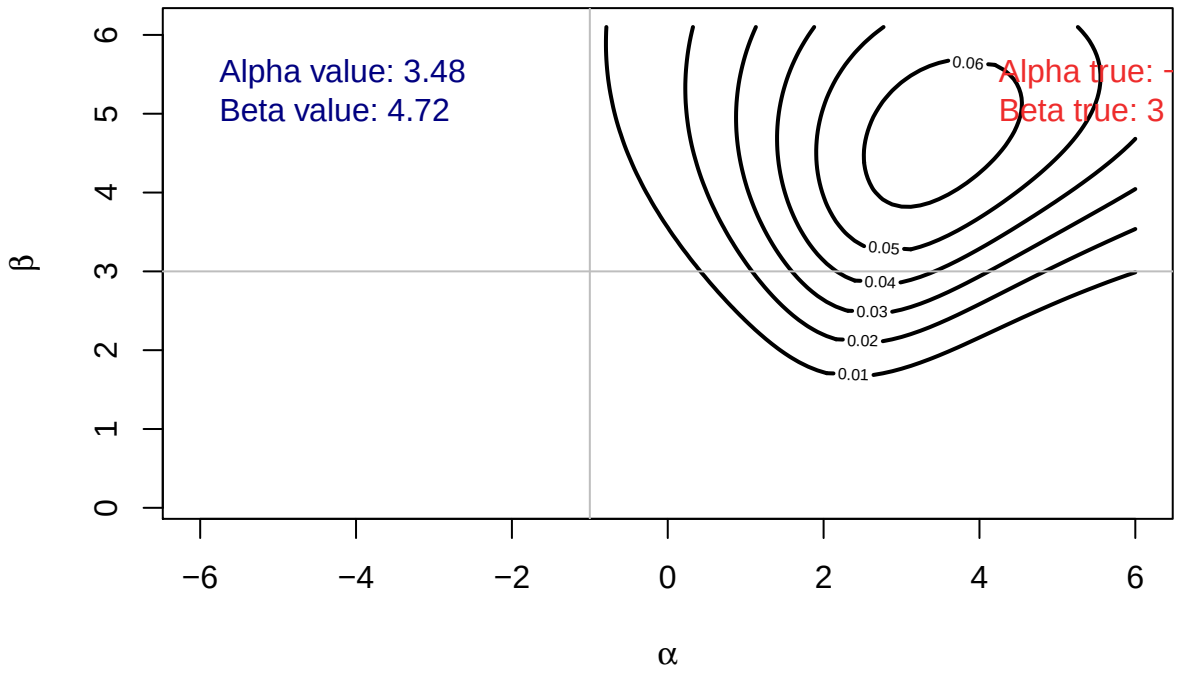


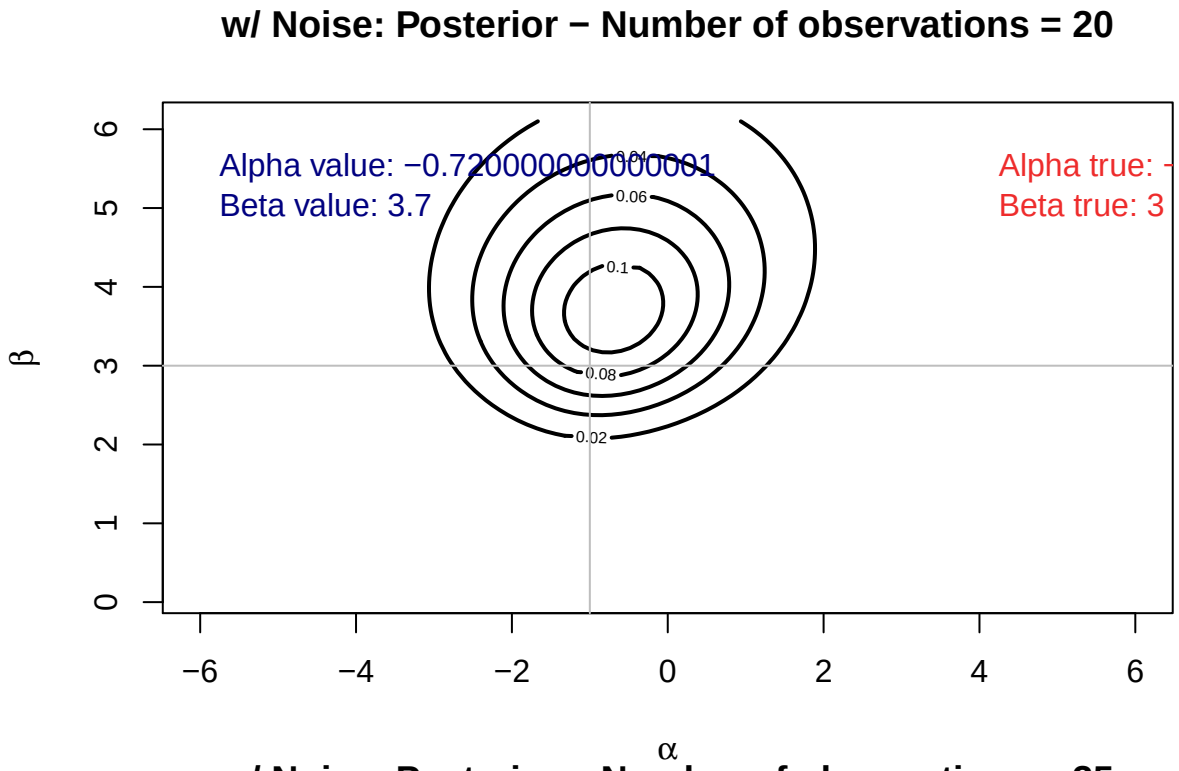

```r
set.seed(seed)
observations <- 250
data <- rcauchy(observations, true.alpha, true.beta) + rnorm(observations, 0, 0.5)
#data <- data[data > -6 & data < 6]

n <- c(2,5,10,20,35,50,100,150, 200)

post.likelihood.fun <- function(data, alpha, beta){
  if(alpha < -6 || alpha > 6 || beta < 0) { return ( 0 )}
  likelihood <- 1
  for (x in data) {
    likelihood <- likelihood*dcauchy(x, alpha, beta)
  }
  return (likelihood)
}


n.sample <- 100
x.min <- -6; x.max <- +6
h <- (x.max - x.min )/n.sample

dist.min <- 0.1; dist.max <- 6.1
g <- (dist.max - dist.min)/n.sample

alpha <- seq(from= x.min      , by=h  , length.out = n.sample +1)
beta  <- seq(from= dist.min   , by=g  , length.out = n.sample +1)


for (samples in n){
```

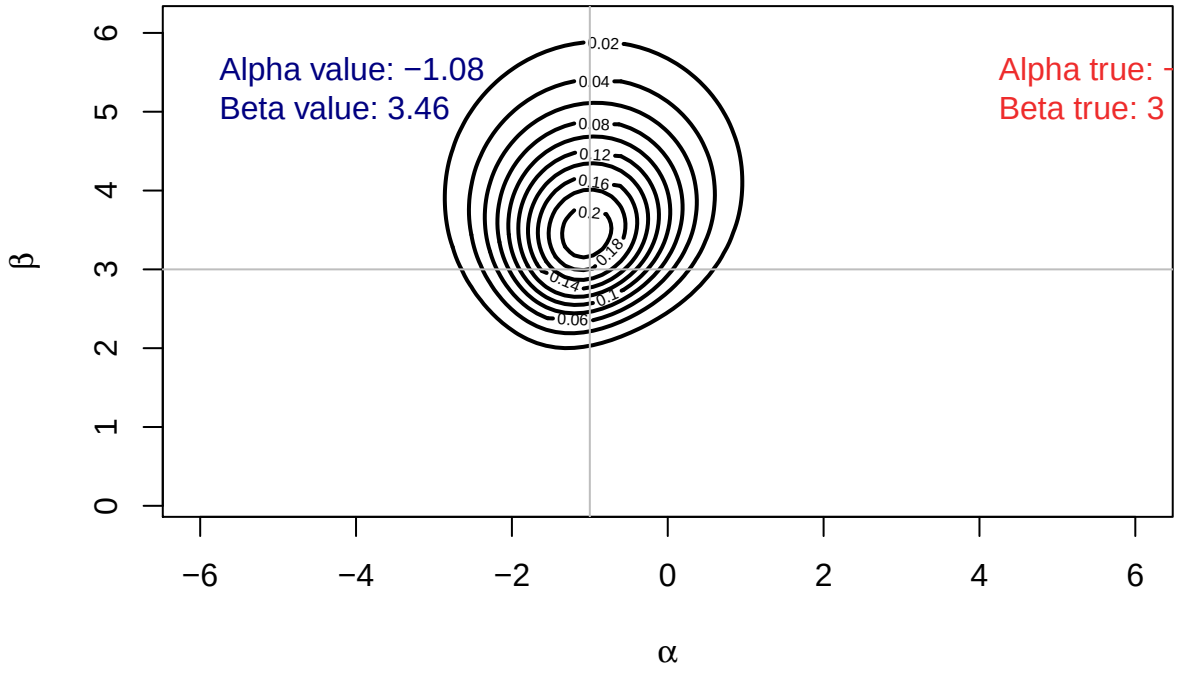
```r
dt <- data[1:samples]

z <- matrix (data = NA , nrow = length(alpha), ncol = length(beta))
  for(j in 1: length(alpha)) {
    for(k in 1: length(beta)) {
      z[j,k] <- post.likelihood.fun(dt, alpha[j], beta[k])
    }
  }
z.norm <- h*g*sum(z)
z <- z/z.norm

contour (alpha, beta, z, nlevels = 8,labcex = 0.5,lwd = 2, xlab=expression(alpha), ylab=expression(beta)
abline (v=true.alpha, h=true.beta, col="grey")
ind <- which(z == max(z), arr.ind = TRUE)
alpha.max <- alpha[ind[1]]
beta.max <- beta[ind[2]]
text(x = -6, y = 5.5 , col="navy ", lwd = 2, pos=4, paste("Alpha value: ", alpha.max , sep=""))
text(x = -6, y = 5.0 , col="navy ", lwd = 2, pos=4, paste("Beta value: ", beta.max , sep=""))

text(x = +4, y = 5.5 , col="firebrick2 ", lwd = 2, pos=4, paste("Alpha true: ", true.alpha , sep=""))
text(x = +4, y = 5.0 , col="firebrick2 ", lwd = 2, pos=4, paste("Beta true: ", true.beta , sep=""))
}
```

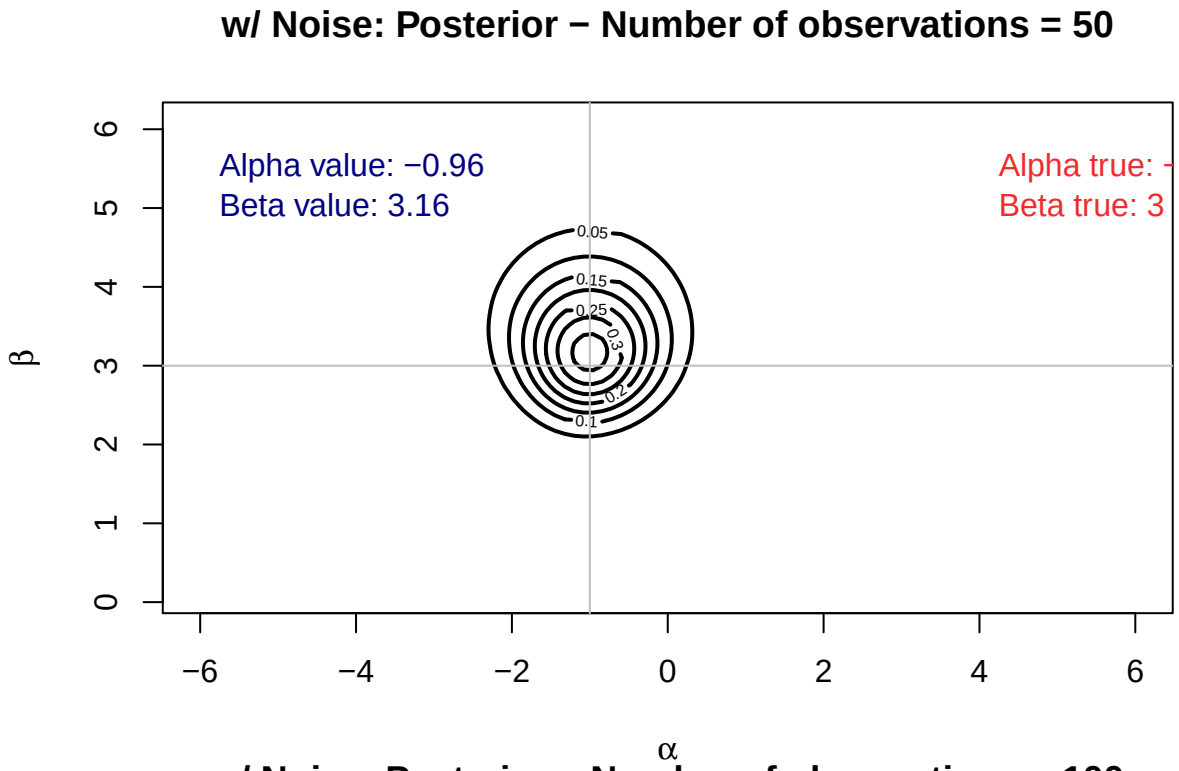


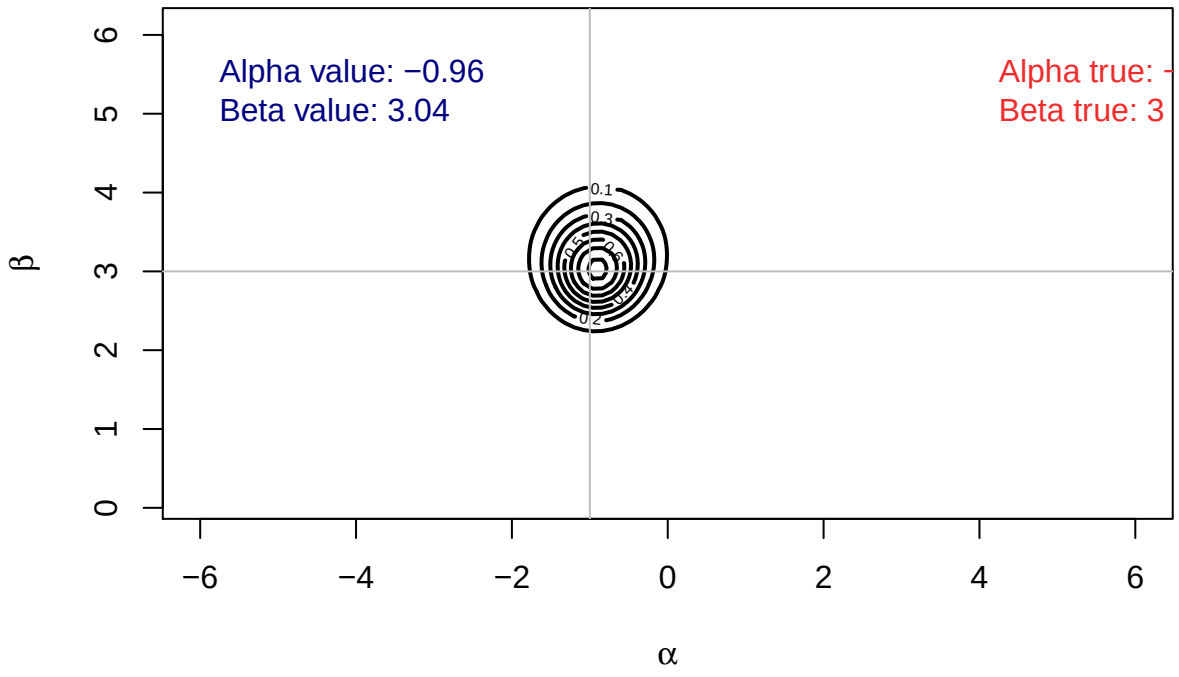

w/ Noise: Posterior − Number of observations = 5
Alpha value: 1.8
Beta value: 1.12
Alpha true: -
Beta true: 3
0.02
0.03
0.04
0.05
0.06
0.07
0.09
0.1
0.01
β
α
−6
−4
−2
0
2
4
6
0
1
2
3
4
5
6

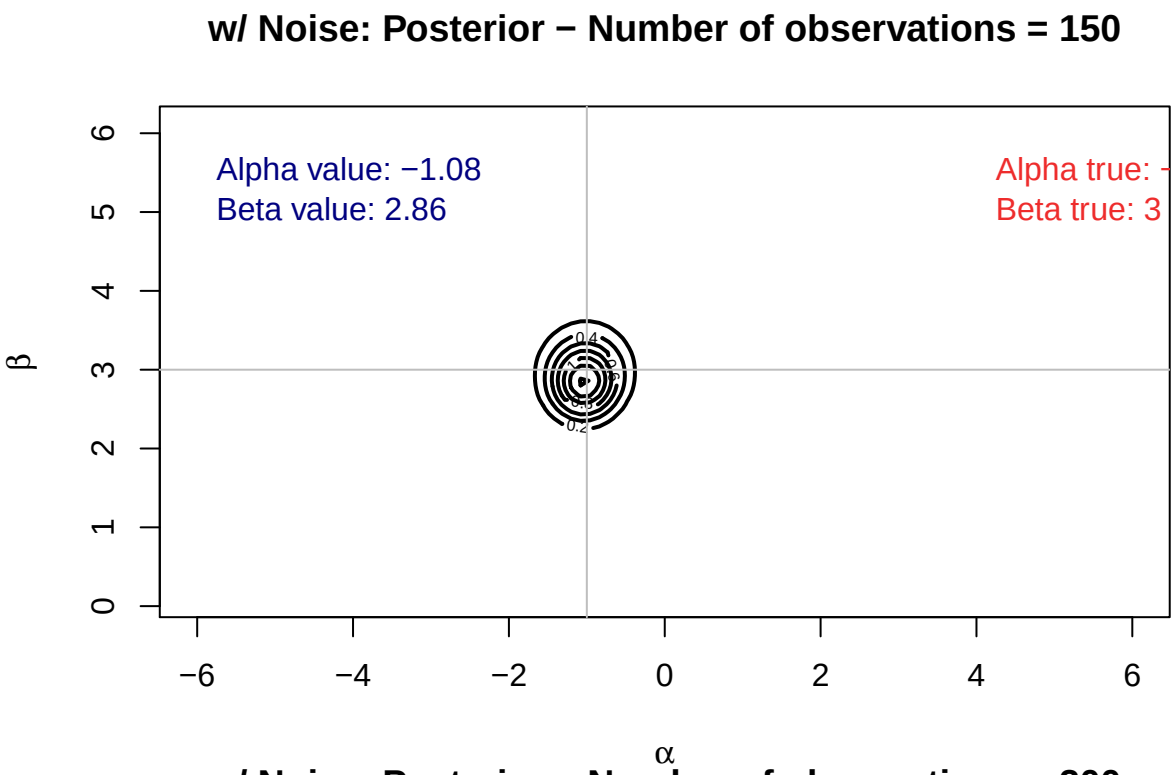

w/ Noise: Posterior − Number of observations = 10
Alpha value: 3.48
Beta value: 4.72
Alpha true: -
Beta true: 3
0.06
0.05
0.04
0.03
0.02
0.01
β
α
−6
−4
−2
0
2
4
6
0
1
2
3
4
5
6

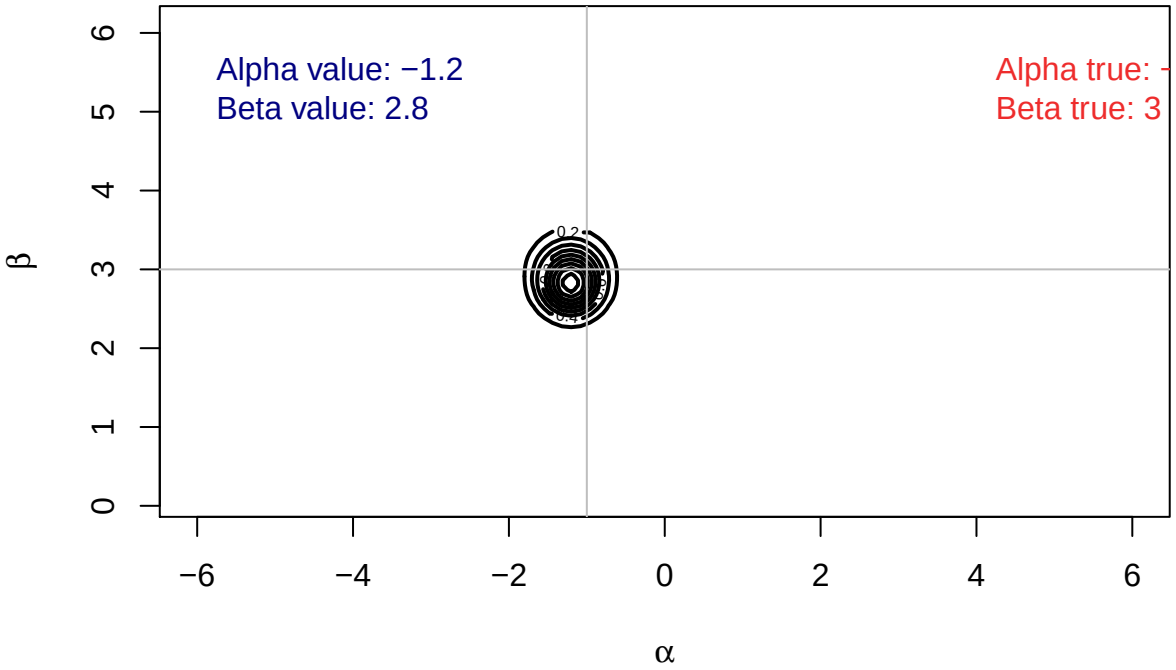

w/ Noise: Posterior − Number of observations = 20
Alpha value: −0.720000000000001
Beta value: 3.7
Alpha true: -
Beta true: 3
0.04
0.06
0.1
0.08
0.02
β
α
0
1
2
3
4
5
6
−6
−4
−2
0
2
4
6


w/ Noise: Posterior − Number of observations = 35
Alpha value: −1.08
Beta value: 3.46
Alpha true: -
Beta true: 3
0.02
0.04
0.08
0.12
0.16
0.2
0.18
0.14
0.1
0.06
β
α
0
1
2
3
4
5
6
−6
−4
−2
0
2
4
6


w/ Noise: Posterior − Number of observations = 50
Alpha value: −0.96
Beta value: 3.16
Alpha true: -
Beta true: 3
0.05
0.15
0.25
0.3
0.2
0.1
β
α
−6
−4
−2
0
2
4
6
0
1
2
3
4
5
6


w/ Noise: Posterior − Number of observations = 100
Alpha value: −0.96
Beta value: 3.04
Alpha true: -
Beta true: 3
0.1
0.3
0.2
β
α
−6
−4
−2
0
2
4
6
0
1
2
3
4
5
6





It is interesting as the random noise biases the results only for fewobservations, but in the long run it does not affect our estimates of parameters alpha and beta. Moreover our confidence interval quite always (also when changing seeds) contains the true values.

Exercise 3

Given the Signal over Background example discussed last week, analyze and discuss the following cases:

(a) Vary the sampling resolution of used to generate the data, keeping the same sampling range:

```
# - Generative model
signal <- function (x, a, b, x0, w, t) {
t*(a*exp (-(x-x0)^2/(2*w^2)) + b)
}

# Define model
x0 <- 0 #Signal peak
w  <-1 #Signal width

#parameters
A.true  <- 2 # Signal amplitude
B.true  <- 1 # Background amplitude
Delta.t <- 8 # Exposure time

#Grid for evaluating the posterior
alim  <- c(0.0, 4.0)
blim  <- c(0.5, 2)
Nsamp <- 200
uniGrid <- seq(from=1/(2*Nsamp), to=1-1/(2*Nsamp), by=1/Nsamp)
delta_a <- diff(alim)/Nsamp
delta_b <- diff(blim)/Nsamp
a <- alim[1] + diff(alim)*uniGrid
b <- blim[1] + diff(blim)*uniGrid
```

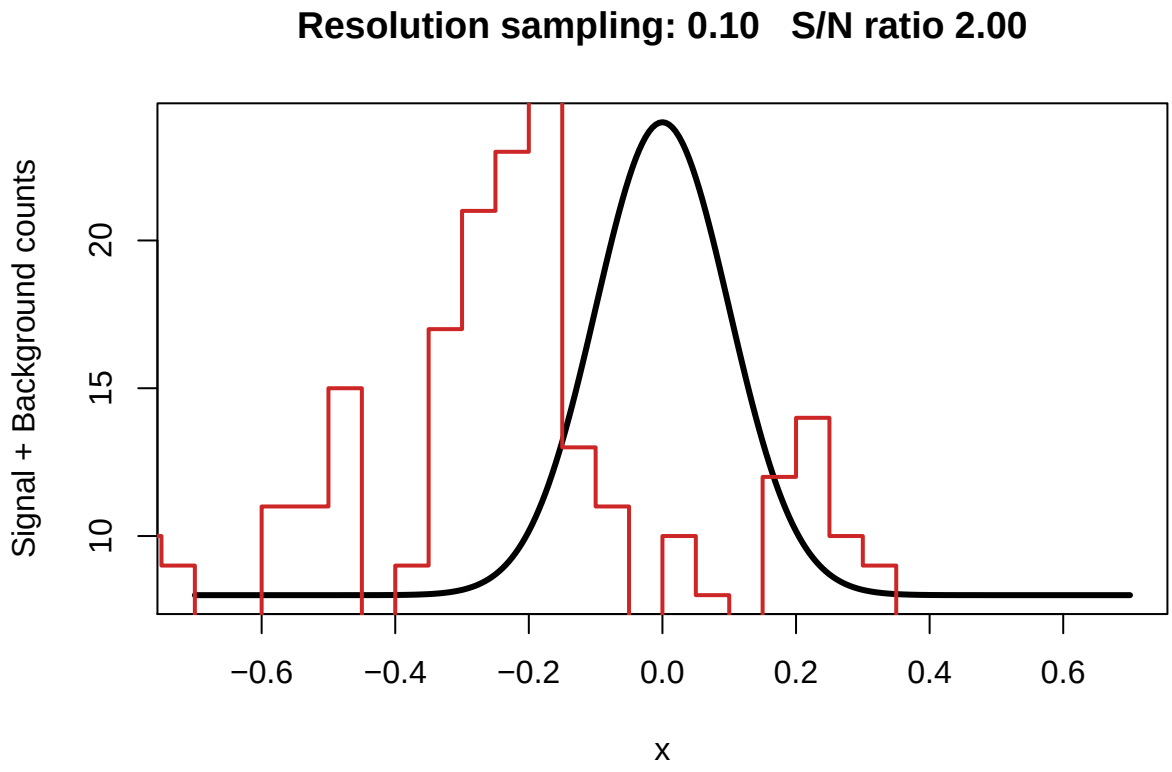
Change the resolution w = {0.1, 0.25, 1, 2, 3}. And check the effect on the results

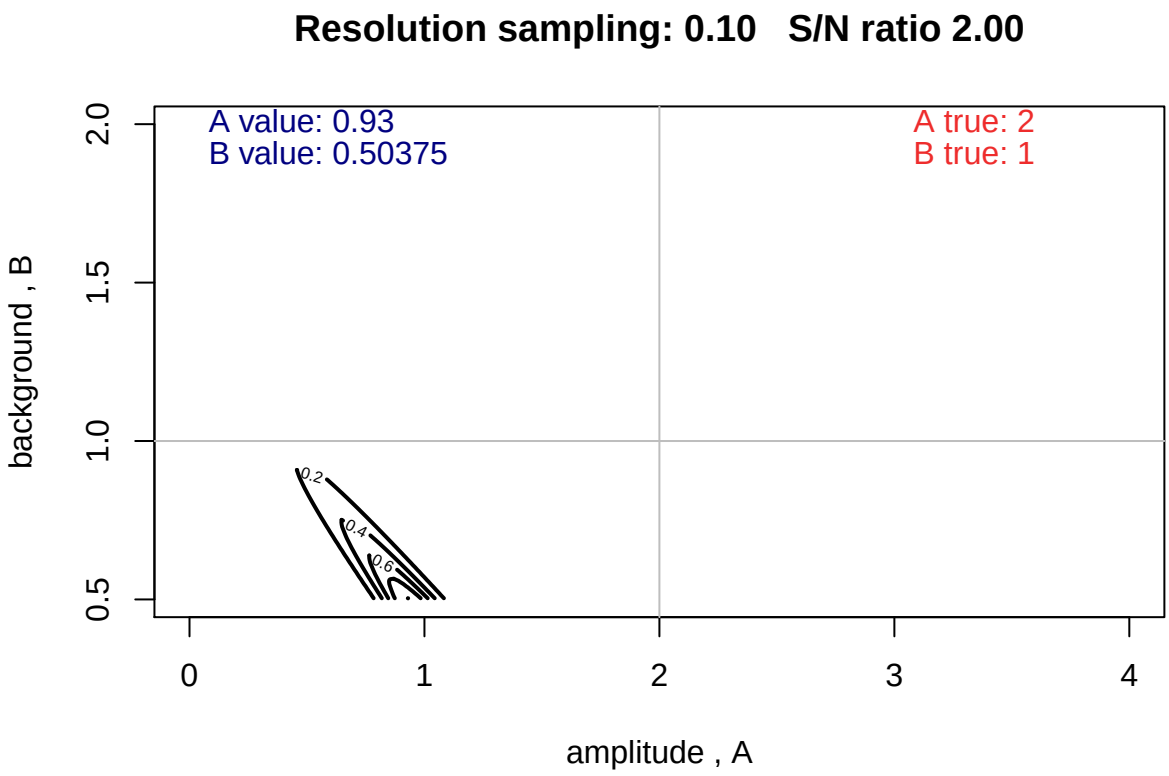
```
# Log posterior
log.post <- function (d, x, a, b, x0, w, t) {
# prior is: both a and b must be positive
  if(a<0 || b <0) { return (-Inf )}
  sum( dpois(d, lambda = signal (x, a, b, x0, w, t), log=TRUE ))
}
```

```
SB_analysis <- function (value, A = A.true, B = B.true){

xdat <- seq(from=-7*value, to=7*value, by=0.5*value)

set.seed(205)
#true signal
s.true <- signal (xdat , A , B, x0, value, Delta.t)
ddat <- rpois( length (s.true), s.true)

xplot <- seq(from=min(xdat), to=max(xdat), by=0.05*value)
splot <- signal(xplot , A , B , x0, value, Delta.t)
plot(xplot , splot , xlab="x", ylab=" Signal + Background counts ", type = 'l', lwd = 3, main = sprintf
xdat.off <- xdat-0.25
lines(xdat.off , ddat , type="s",col="firebrick 3", lwd=2,xlim=range ( xplot), ylim= range (c(splot , d

# Compute log unnormalized posterior , z = ln P^*(a,b|D), on a regular grid
z <- matrix (data = NA , nrow = length (a), ncol =length (b))
for(j in 1: length (a)) {
  for(k in 1: length (b)) {
    z[j,k] <- log.post(ddat , xdat , a[j], b[k], x0, w, Delta.t)
  }
```

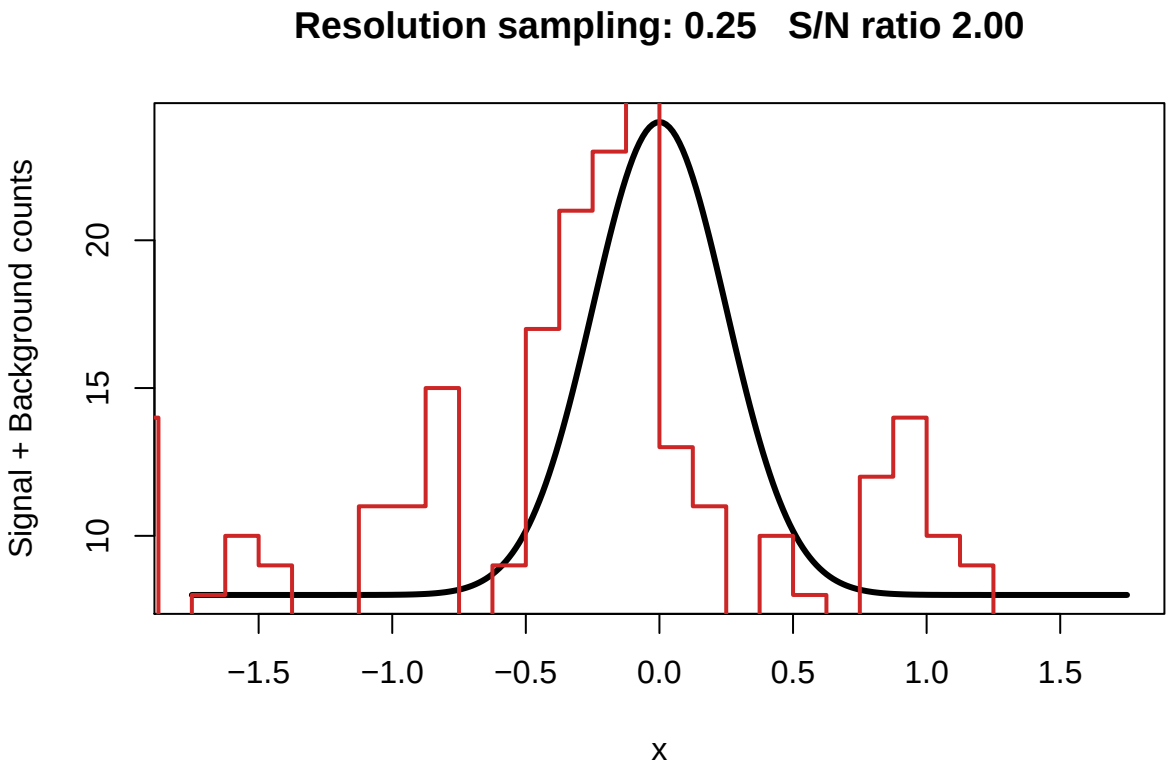
```
}
z <- z - max(z) # set maximum to zero

# Plot unnormalized 2D posterior as contours .
contour (a, b, exp(z), nlevels = 5, labcex = 0.5, lwd = 2, xlab="amplitude , A", ylab="background , B",
abline (v = A, h = B, col = "grey")

ind <- which(z == max(z), arr.ind = TRUE)
A.max <- a[ind[1]]
B.max <- b[ind[2]]

text(x = 0, y = 2 , col="navy ", lwd = 2, pos=4, paste("A value: ", A.max , sep=""))
text(x = 0, y = 1.9 , col="navy ", lwd = 2, pos=4, paste("B value: ", B.max , sep=""))

text(x = +3, y = 2 , col="firebrick2 ", lwd = 2, pos=4, paste("A true: ", A , sep=""))
text(x = +3, y = 1.9 , col="firebrick2 ", lwd = 2, pos=4, paste("B true: ", B , sep=""))

}
```

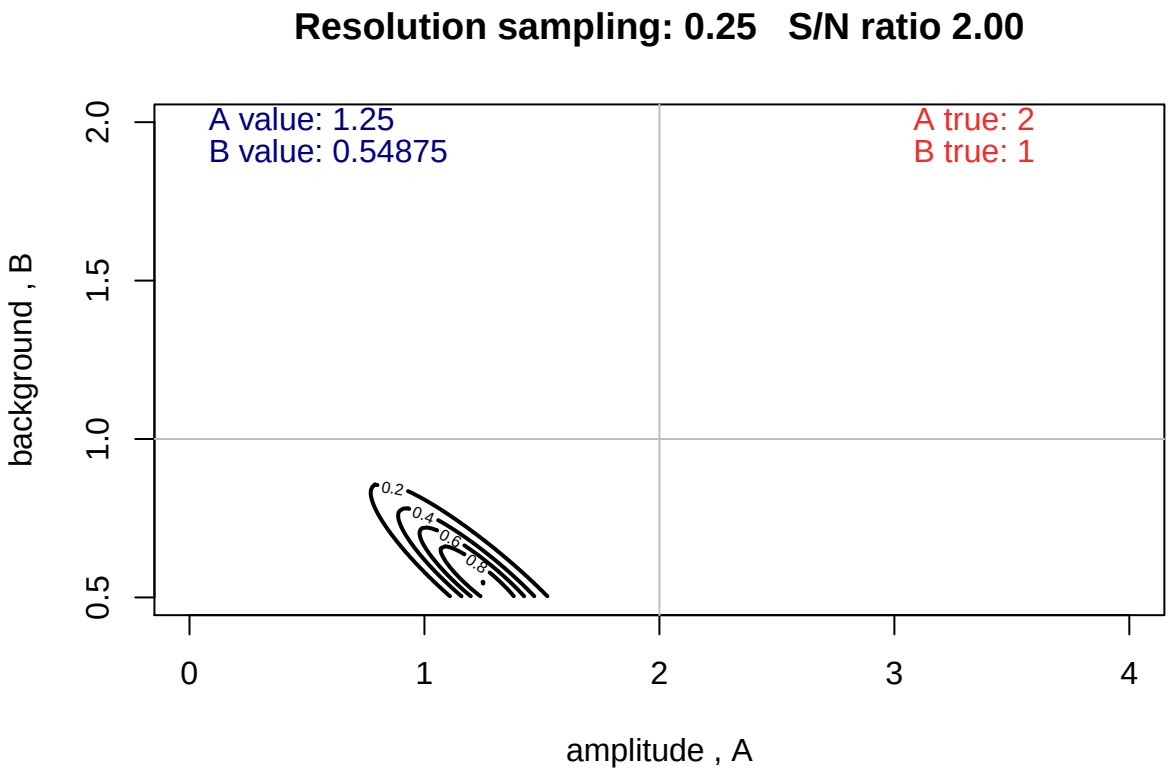
```
SB_analysis(0.1)
```



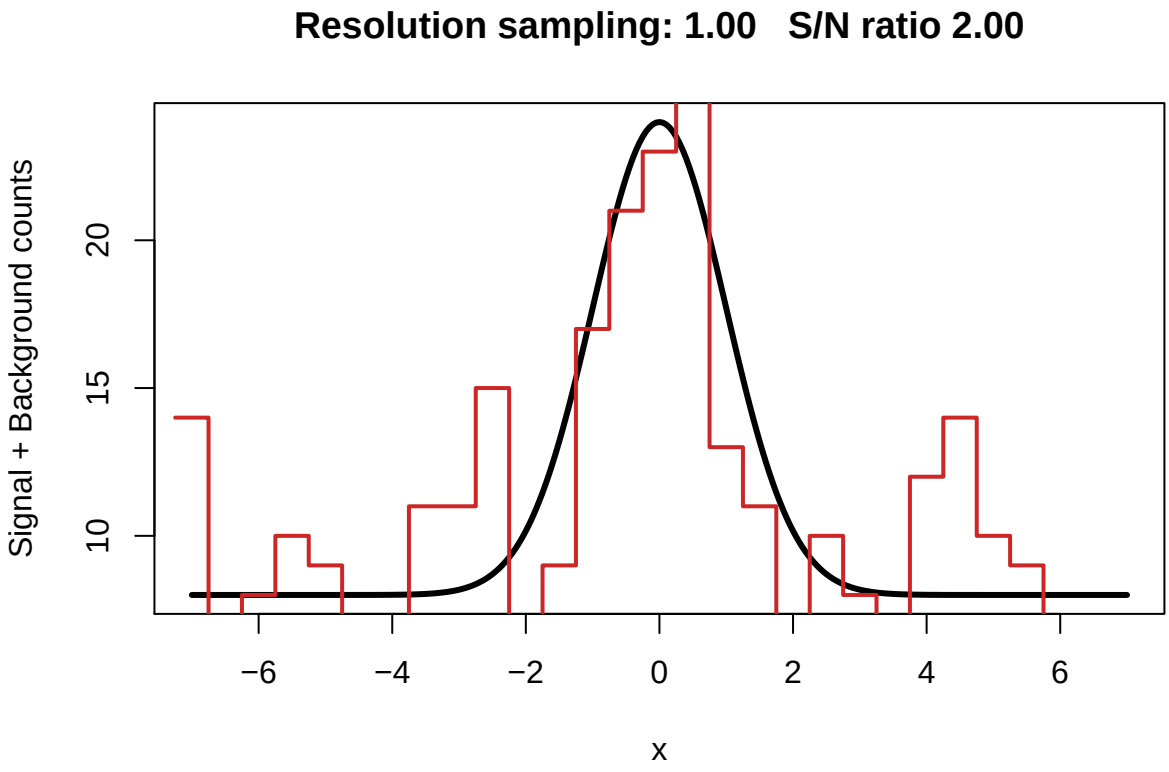


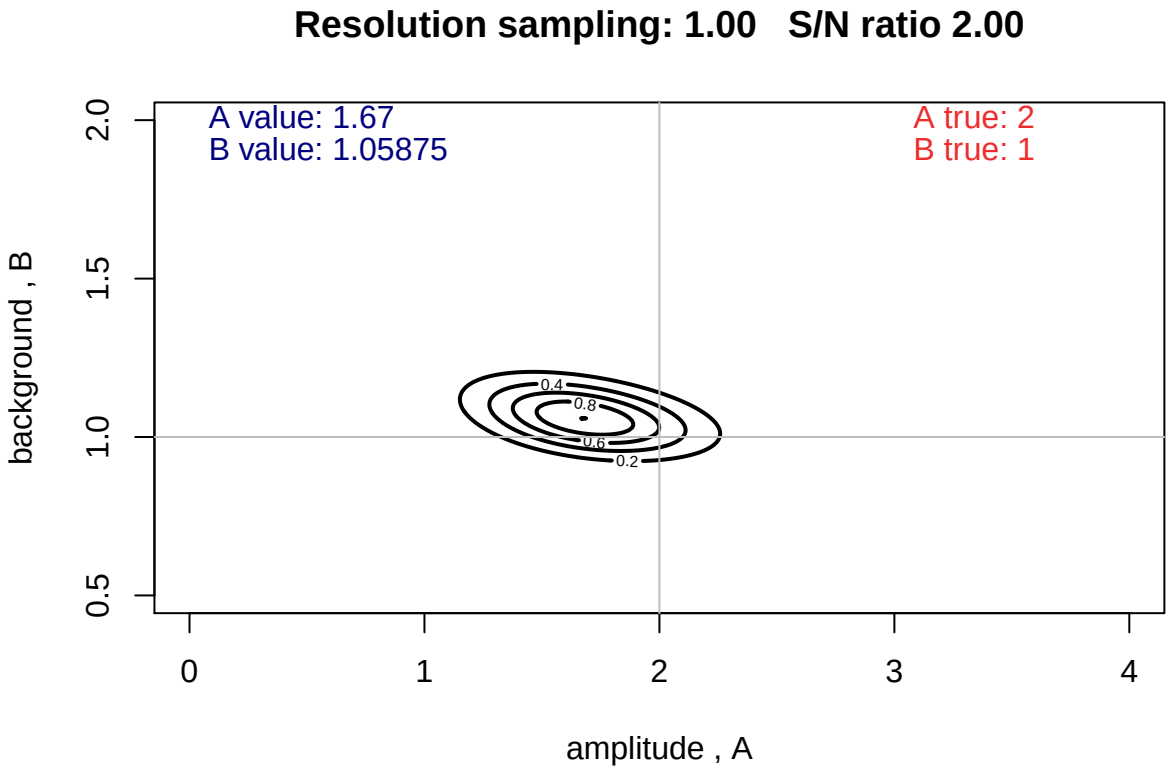

```
SB_analysis(0.25)
```



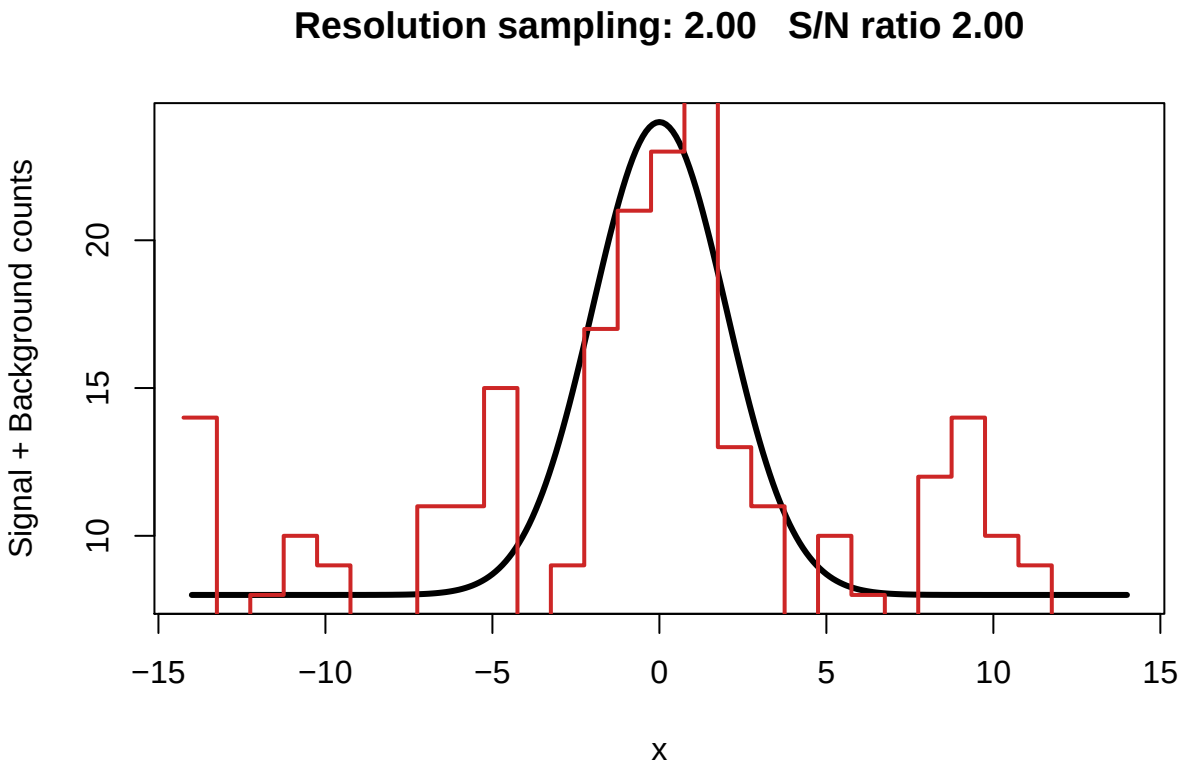


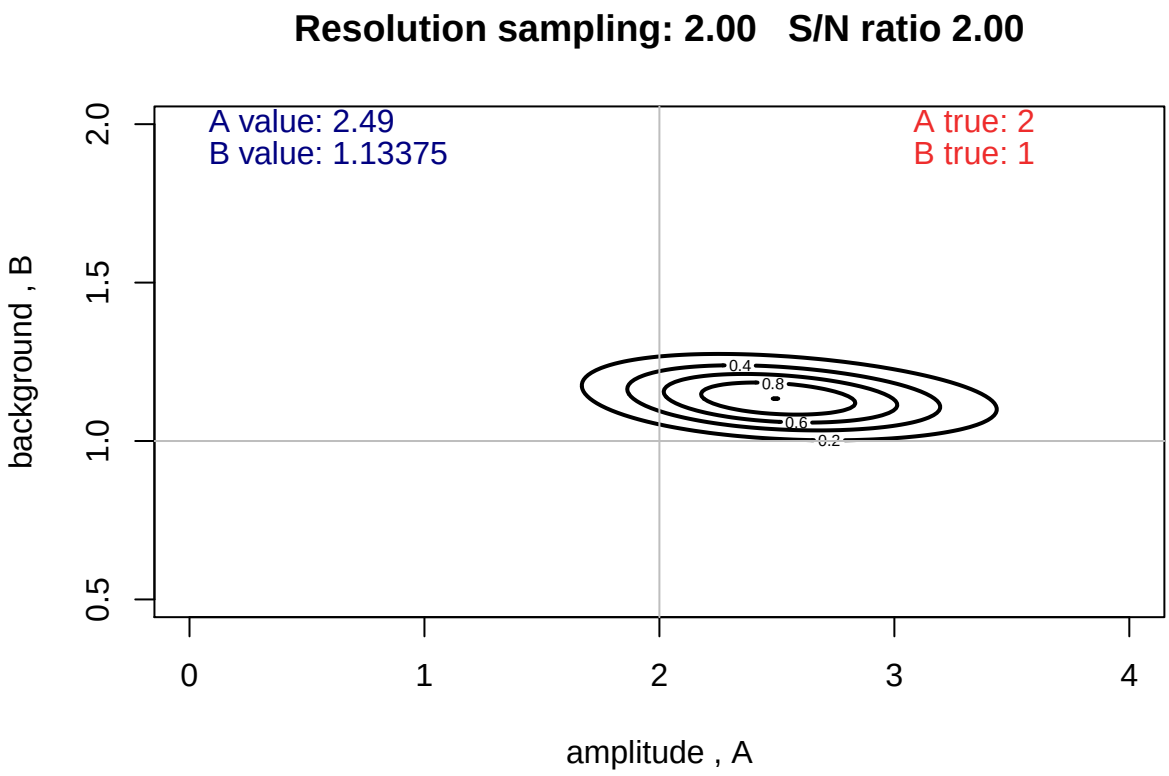

```
SB_analysis(1)
```



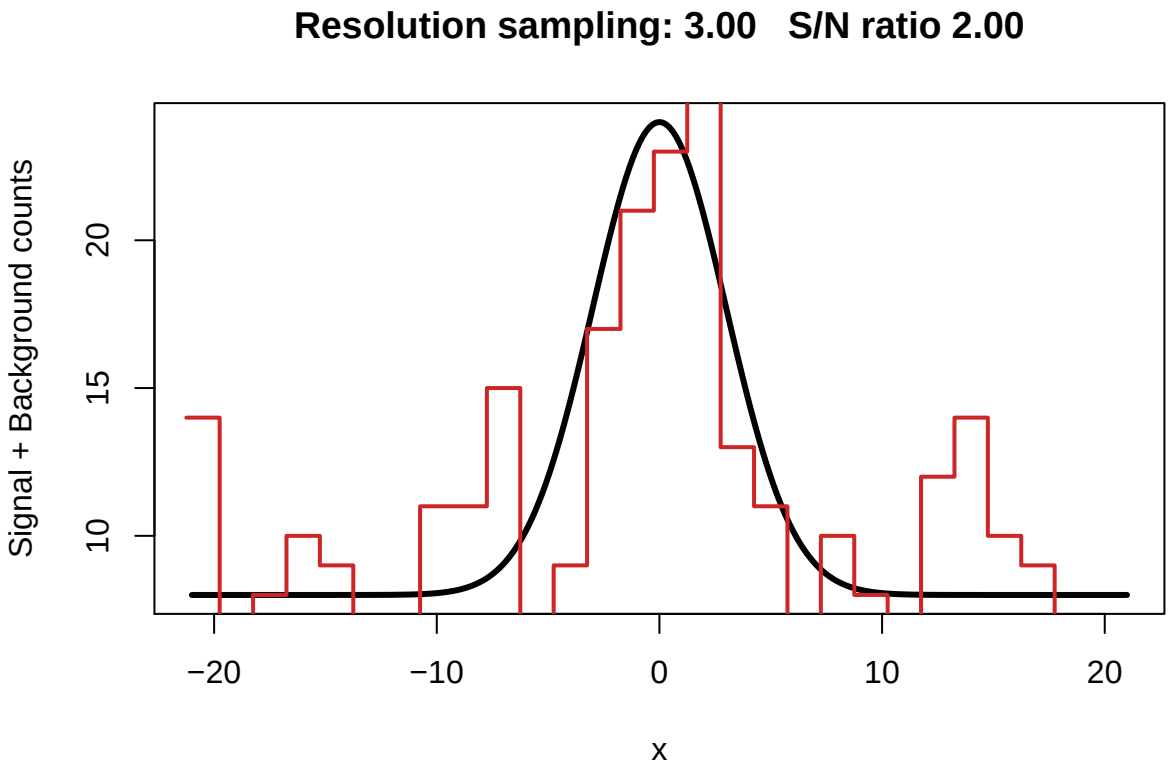


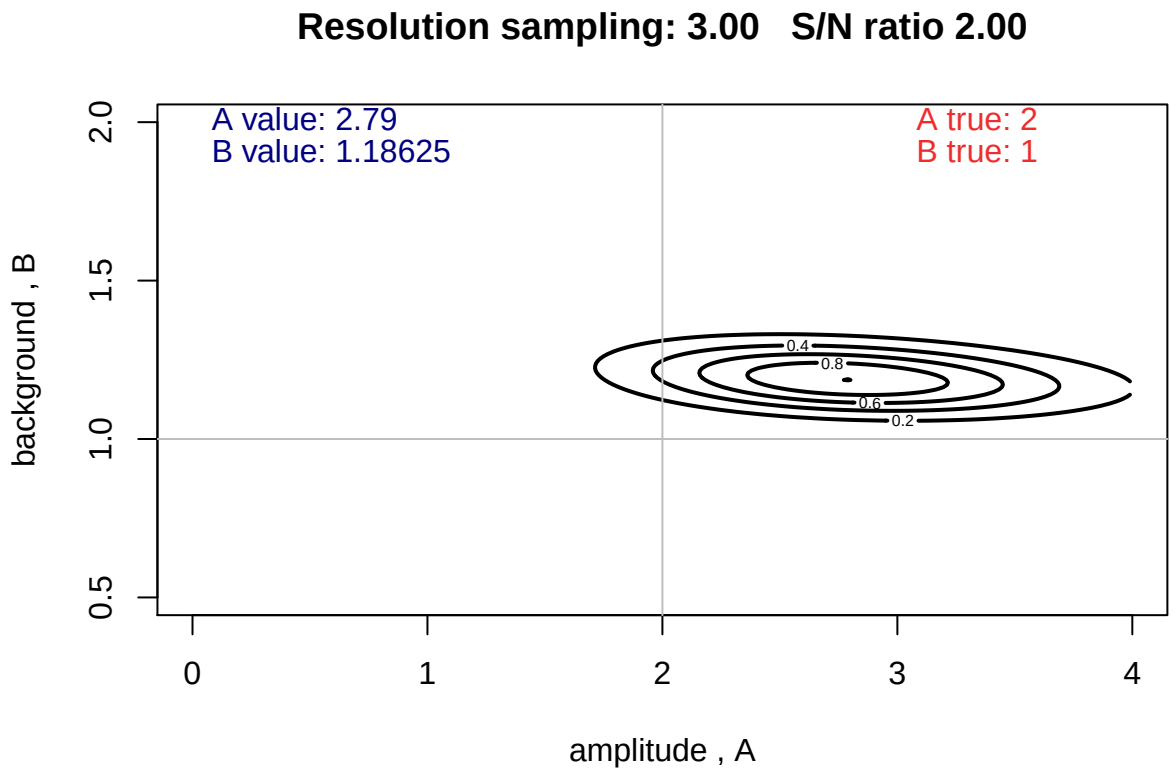

```
SB_analysis(2)
```





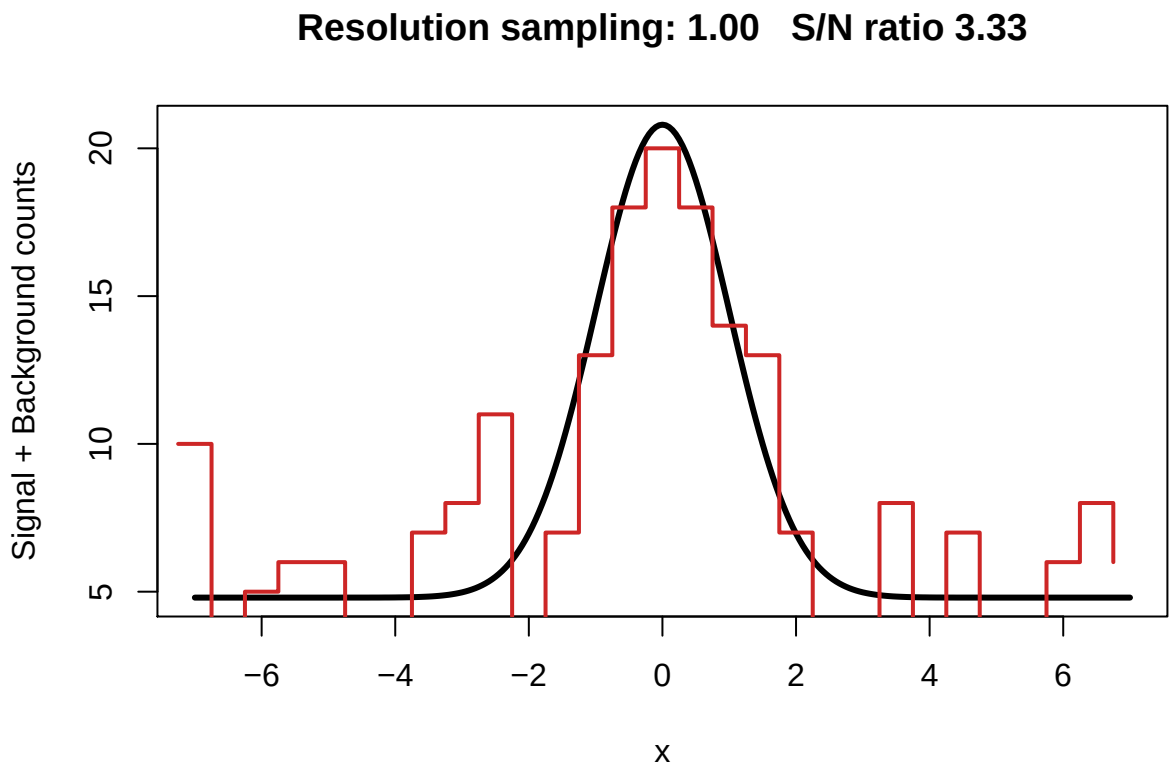

```
SB_analysis(3)
```

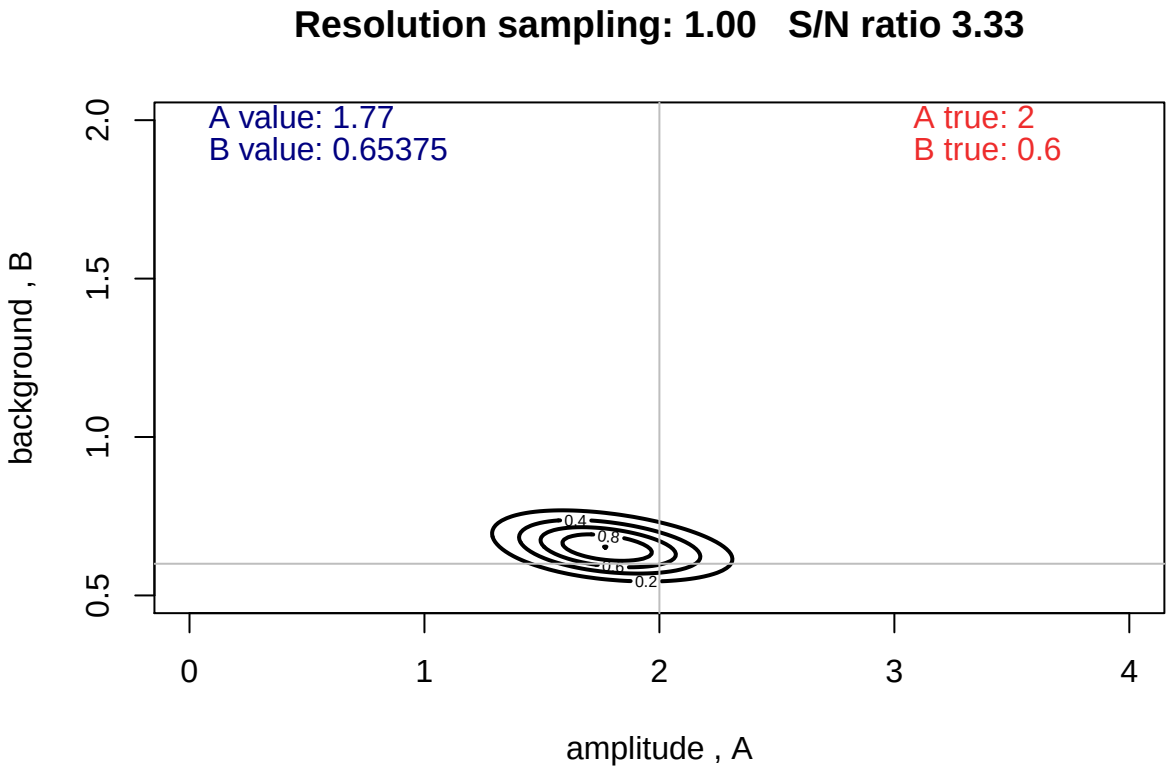




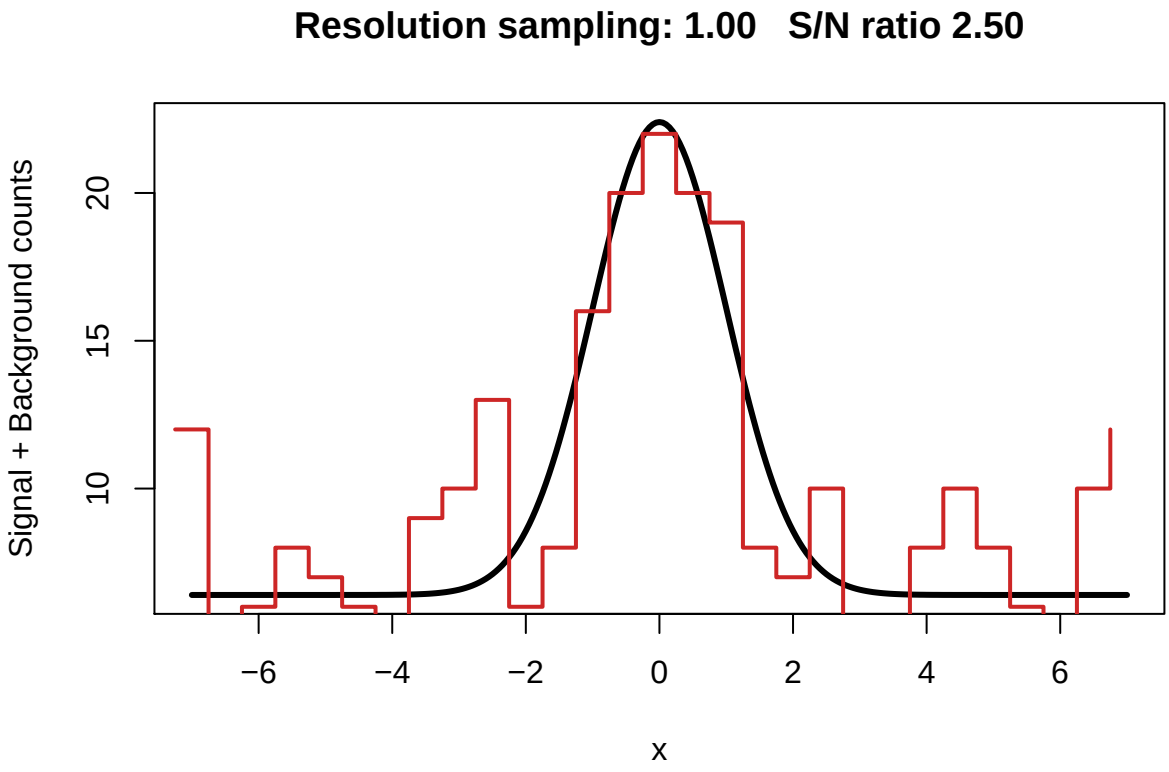
(b) Change the ratio A/B used to simulate the data (keeping both positive in accordance with the prior). Check the effect on the results

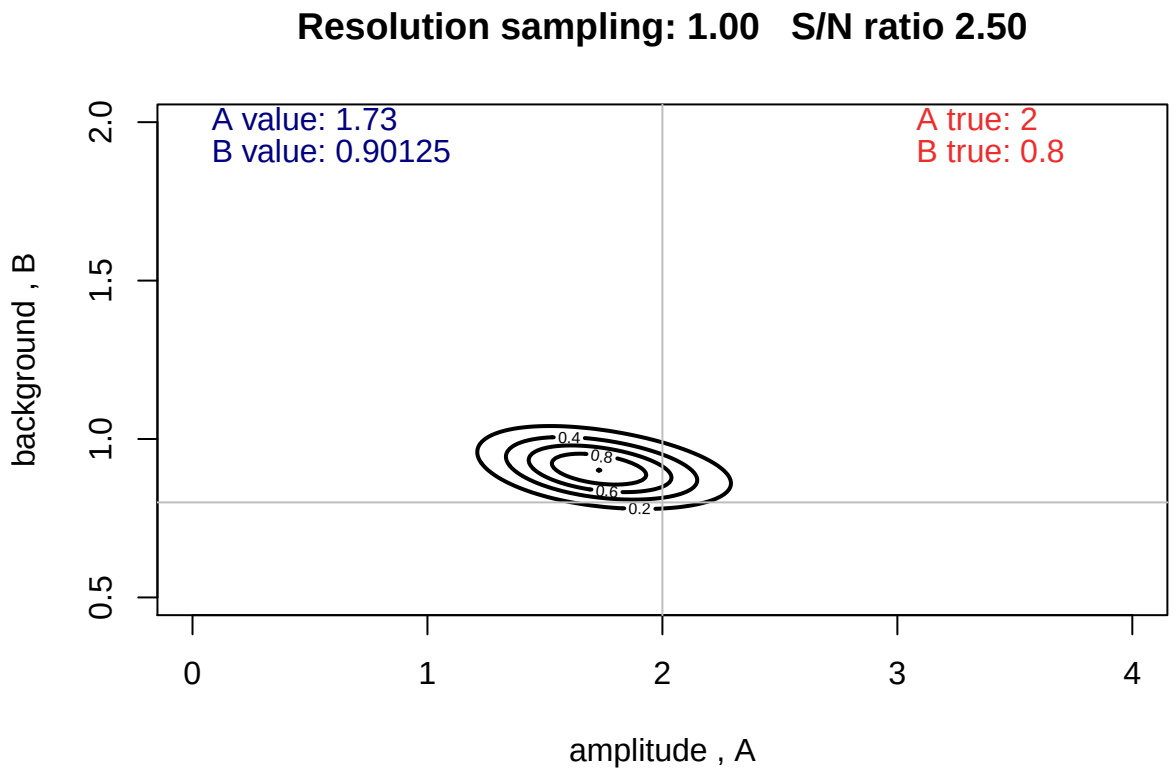
```
SB_analysis(1, A = 2, B = 0.6)
```



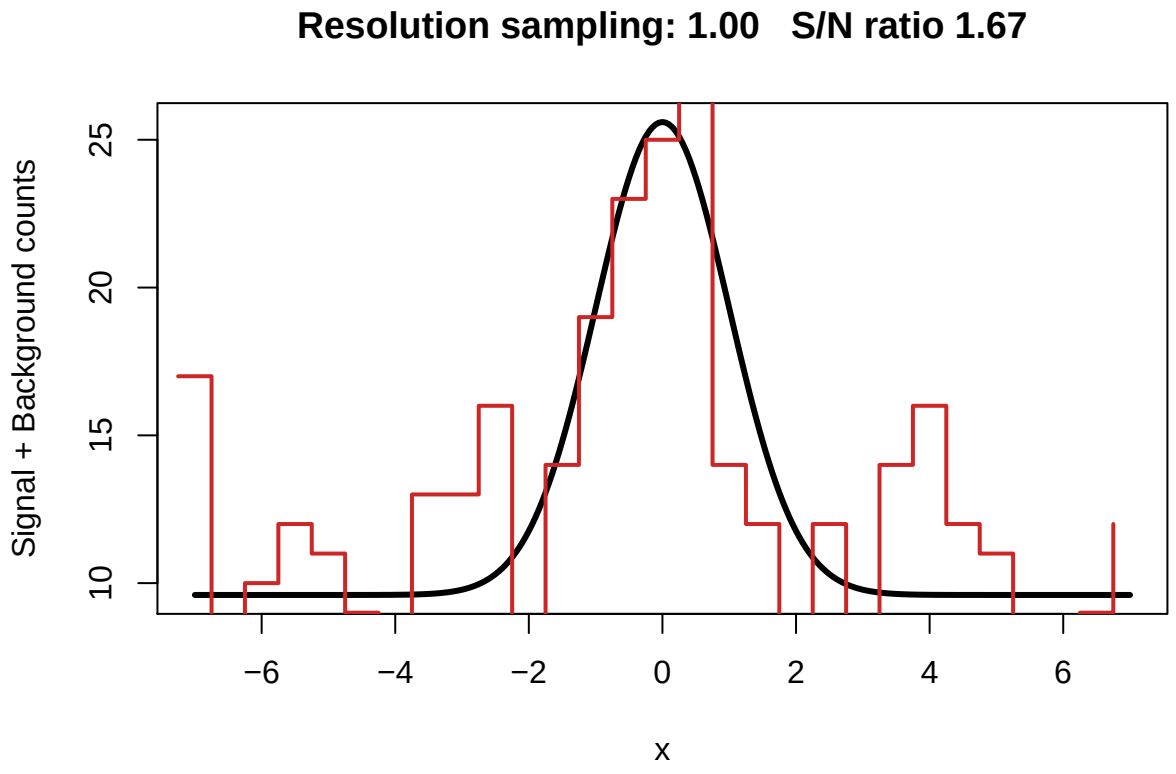


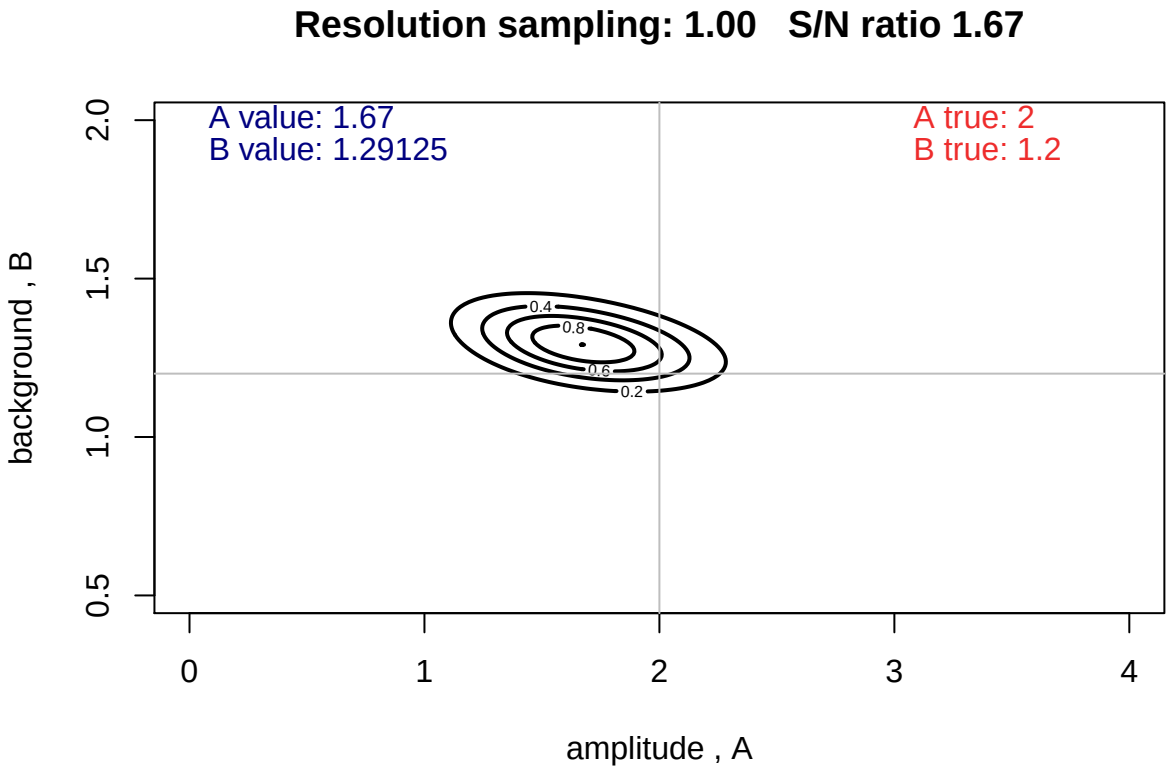

```
SB_analysis(1, A = 2, B = 0.8)
```



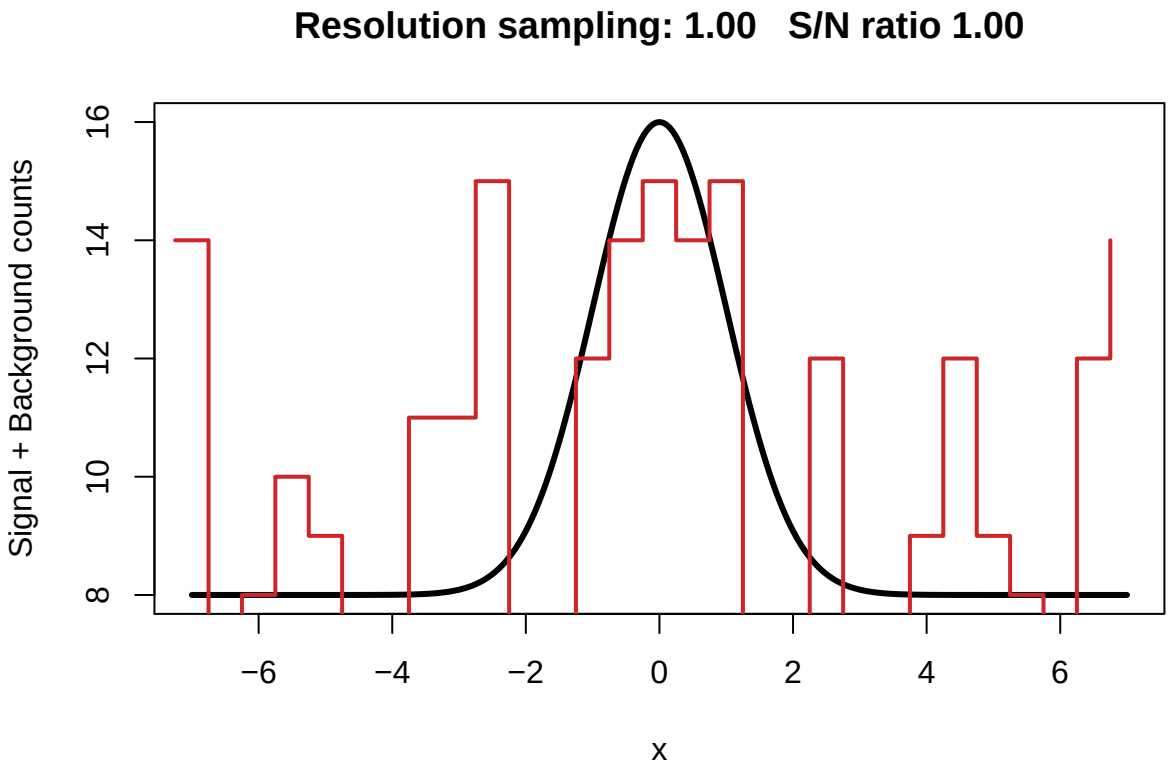


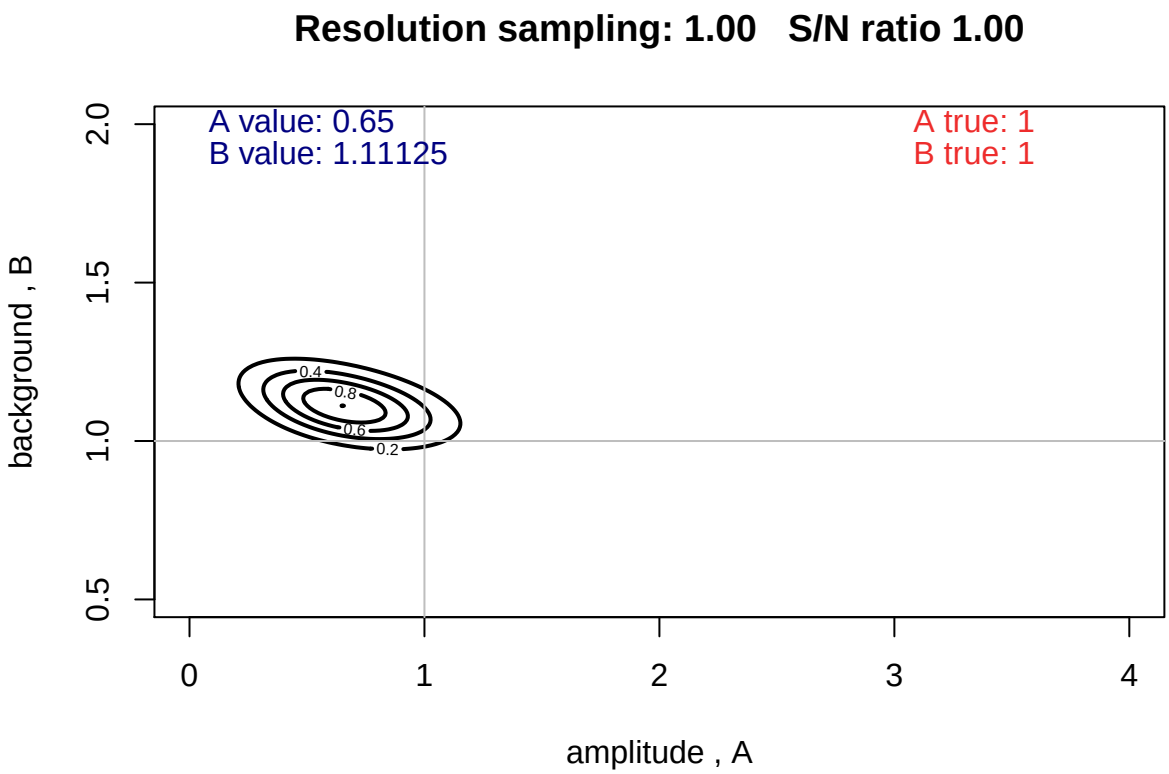

```
SB_analysis(1, A = 2, B = 1.2)
```



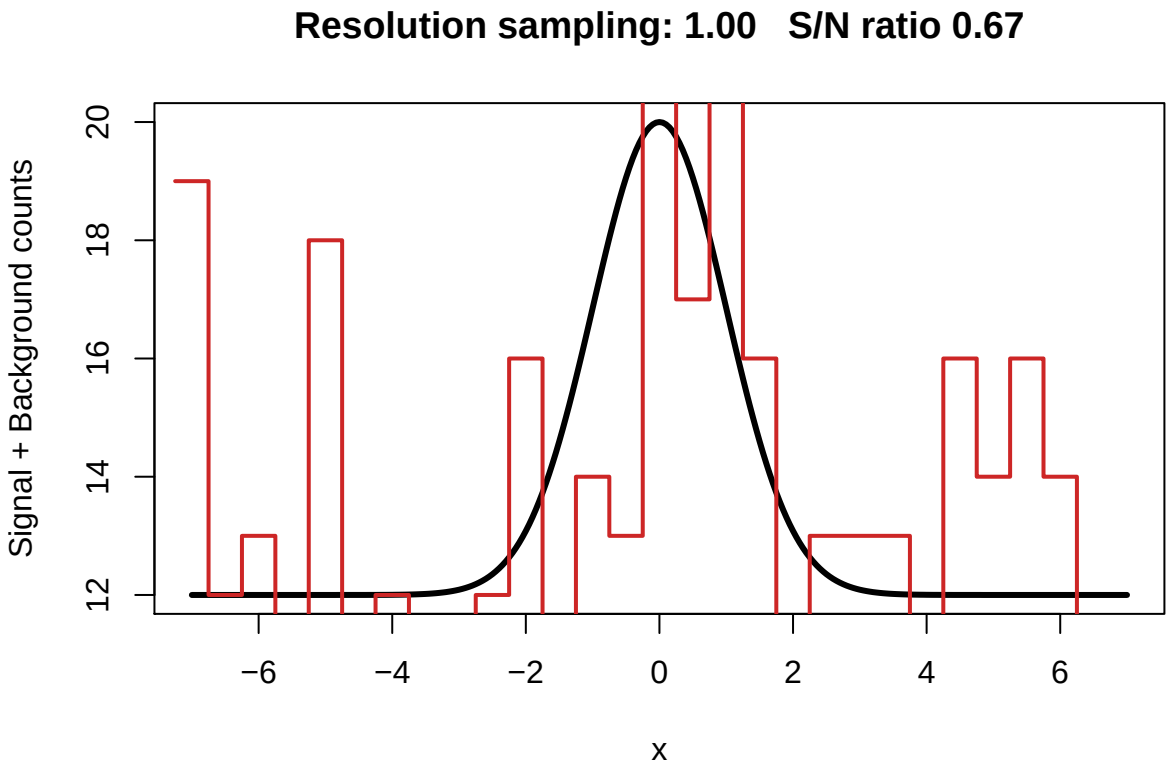


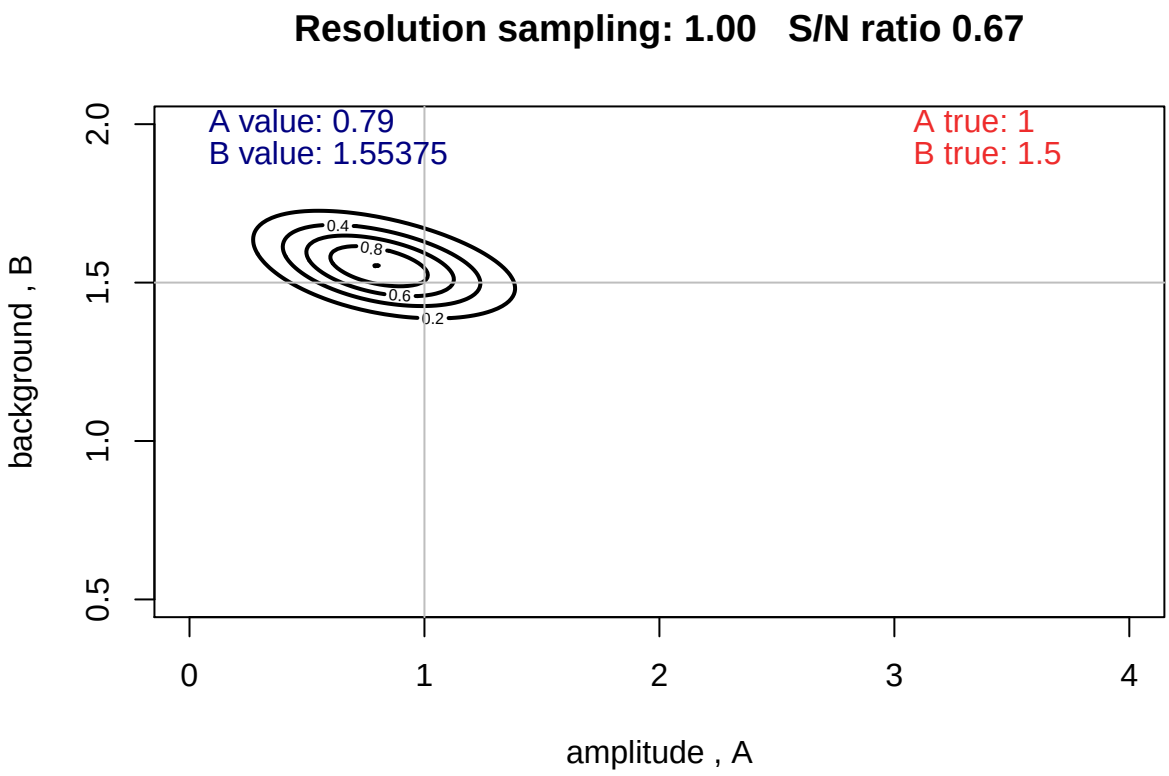

```
SB_analysis(1, A = 1, B = 1)
```






```
SB_analysis(1, A = 1, B = 1.5)
```




Resolution sampling: 1.00 S/N ratio 0.67
A value: 0.79
B value: 1.55375
A true: 1
B true: 1.5
0.4
0.8
0.6
0.2
background , B
2.0
1.5
1.0
0.5
0
1
2
3
4
amplitude , A