Viterbo_Giuseppe_rlab04

May 14, 2023

[69]:
```r
install.packages('tidyverse')
install.packages('gridExtra')
install.packages('emdbook')
install.packages('ramify')

library(tidyverse)
library(gridExtra)
library(emdbook)
library(ramify)
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done


Attaching package: ‘ramify’


The following object is masked from ‘package:purrr’:

    flatten
```

```
The following object is masked from ‘package:tidyr’:

    fill


The following object is masked from ‘package:graphics’:

    clip
```

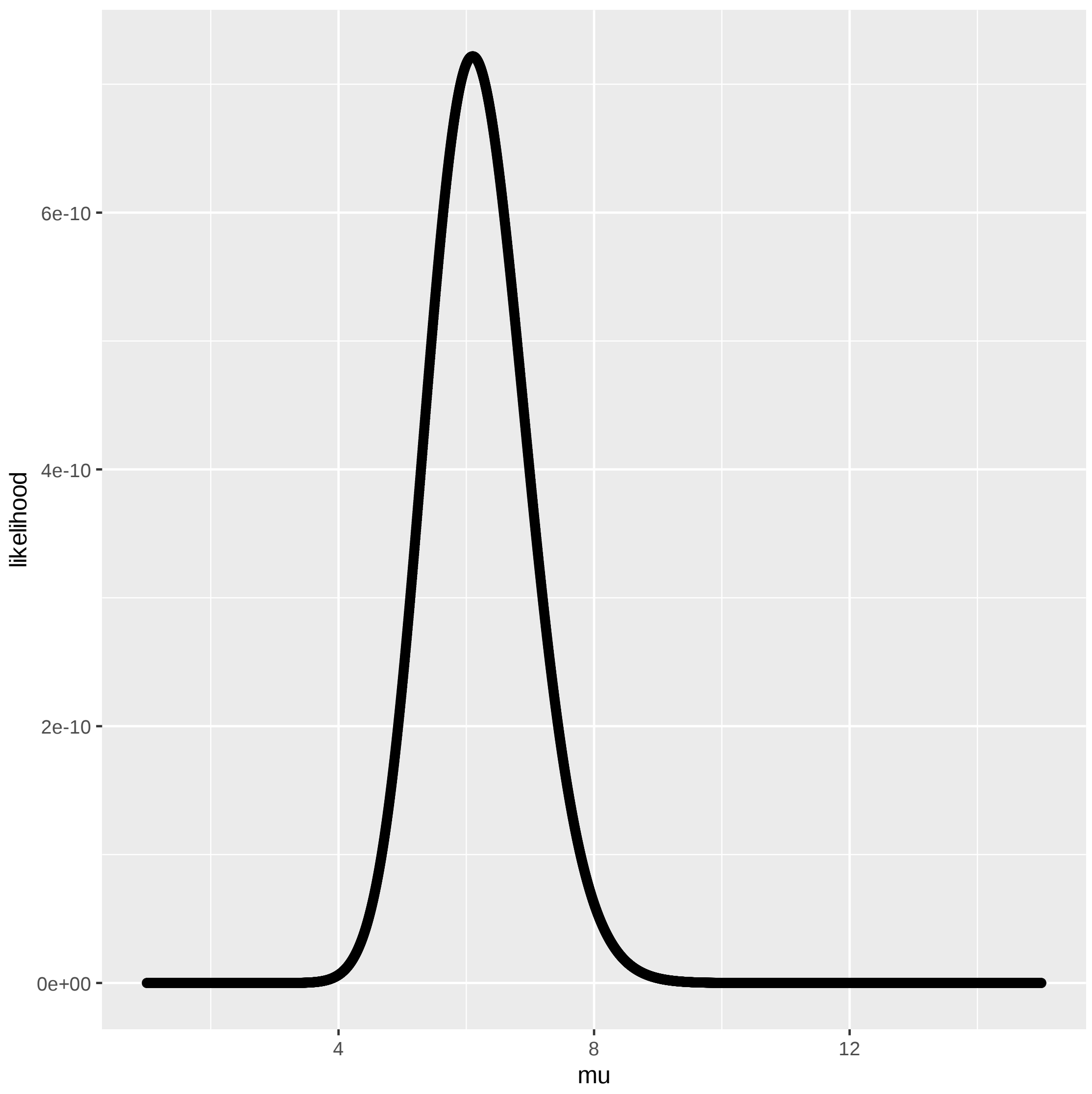
1 Ex.1

1.1 Poisson process

In the case of multiple measurment yhr likelihood is the product (atually is very similar to a Gamma distribution)

1.1.1 Likelihood

[2]:
```
y = c(5, 8, 4, 6, 11, 6, 6, 5, 6, 4)
n.sample <- 4000
delta.mu <- (15-1)/n.sample
mu <- seq(from= 1, to=15, length.out = n.sample)

likelihood <- rep(1, times=n.sample)
for (l in y){
    likelihood <- likelihood * dpois(x=l, lambda = mu)
    }

ggplot() + geom_point(aes(mu, likelihood))
```

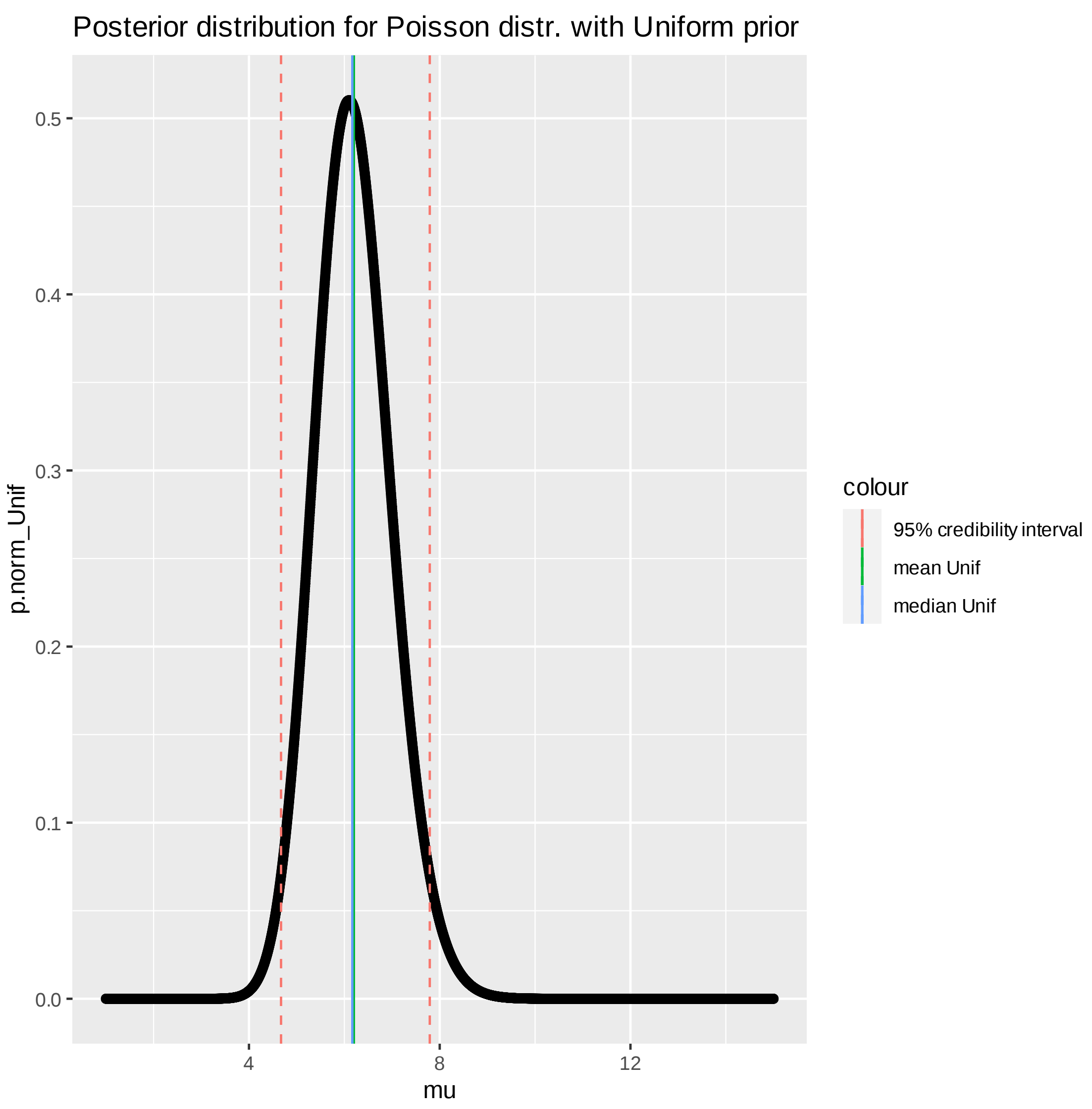


1.2 a) Uniform prior

[7]:
```
p.star_Unif <- likelihood * dunif(mu, min=1, max=15)
p.norm_Unif <- p.star_Unif / (delta.mu *sum(p.star_Unif))

#mean, variance, median
mean_Unif <- delta.mu * sum(mu*p.norm_Unif)
variance_Unif <- delta.mu * sum(((mean_Unif -mu)**2)*p.norm_Unif)
int_median_Unif <- 0
list_median <- c()
for (m in seq(1, n.sample)){
    list_median <- c(list_median, p.norm_Unif[m]*delta.mu)
    int_median_Unif <- sum(list_median)
    if (int_median_Unif > 0.5){
        median_Unif <- mu[m]
        return (median_Unif)
        break
        }
    }

Unif_result <- tibble(
                Mean = c(mean_Unif),
                Variance = c(variance_Unif),
                Median = c(median_Unif)
                )
Unif_result

#95% credibiility interval
lower_bound_Unif <- ncredint(mu, p.norm_Unif, leve=0.95)[['lower']]
upper_bound_Unif <- ncredint(mu, p.norm_Unif, level=0.95)[['upper']]

#plot
plt_Unif <- ggplot() + geom_point(aes(mu,p.norm_Unif))
plt_Unif <- plt_Unif + geom_vline(aes(xintercept=mean_Unif, color='mean Unif'))
plt_Unif <- plt_Unif + geom_vline(aes(xintercept=median_Unif, color='median␣
 ↪Unif'))
plt_Unif <- plt_Unif + geom_vline(aes(xintercept=lower_bound_Unif, color='95%␣
 ↪credibility interval'), linetype='dashed')
plt_Unif <- plt_Unif + geom_vline(aes(xintercept=upper_bound_Unif, color='95%␣
 ↪credibility interval'), linetype='dashed')
plt_Unif <- plt_Unif + labs(title='Posterior distribution for Poisson distr.␣
 ↪with Uniform prior')
plt_Unif
```

A tibble: 1 × 3

Mean <dbl>	Variance <dbl>	Median <dbl>
6.2	0.62	6.167292

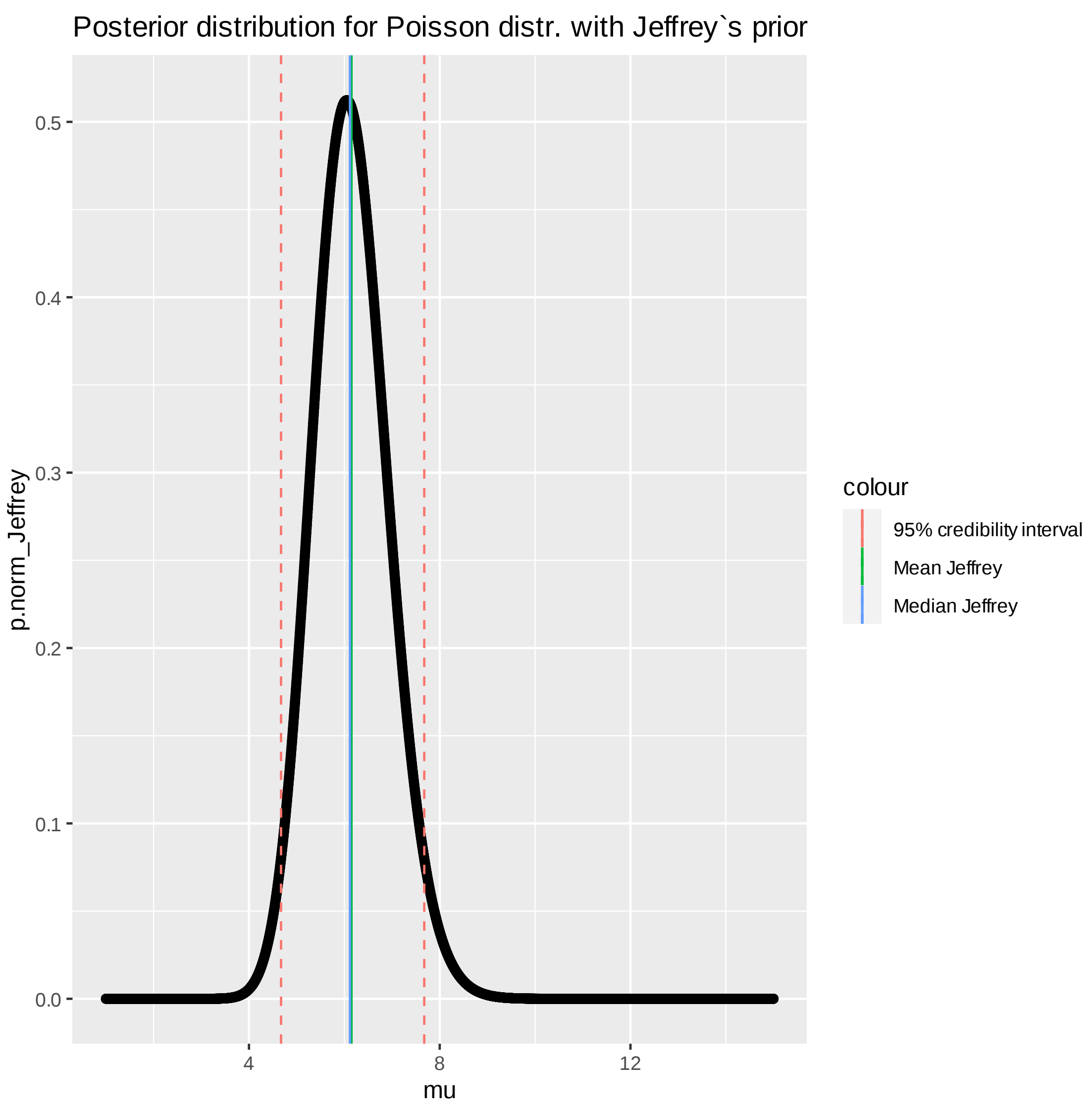


2 b) Jeffreys' prior

[8]:
```
p.star_Jeffrey <- likelihood * (1/sqrt(mu))
p.norm_Jeffrey <- p.star_Jeffrey / (delta.mu *sum(p.star_Jeffrey))

#mean, variance, median
mean_Jeffrey <- delta.mu * sum(mu*p.norm_Jeffrey)
variance_Jeffrey <- delta.mu * sum(((mean_Jeffrey - mu)**2)*p.norm_Jeffrey) ##␣
 ↪not sure
int_median_Jeffrey <- 0
list_median <- c()
for (m in seq(1, n.sample)){
    list_median <- c(list_median, p.norm_Jeffrey[m]*delta.mu)
    int_median_Jeffrey <- sum(list_median)
    if (int_median_Jeffrey > 0.5){
        median_Jeffrey <- mu[m]
        return (median_Jeffrey)
        break
        }
    }

Jeffrey_result <- tibble(
                Mean = c(mean_Jeffrey),
                Variance = c(variance_Jeffrey),
                Median = c(median_Jeffrey))
Jeffrey_result

#95% credibiility interval
lower_bound_Jeffrey <- ncredint(mu, p.norm_Jeffrey, leve=0.95)[['lower']]
upper_bound_Jeffrey <- ncredint(mu, p.norm_Jeffrey, level=0.95)[['upper']]

#plot
plt_Jeffrey <- ggplot() + geom_point(aes(mu,p.norm_Jeffrey))
plt_Jeffrey <- plt_Jeffrey + geom_vline(aes(xintercept=mean_Jeffrey,␣
 ↪color='Mean Jeffrey'))
plt_Jeffrey <- plt_Jeffrey + geom_vline(aes(xintercept=median_Jeffrey,␣
 ↪color='Median Jeffrey'))
plt_Jeffrey <- plt_Jeffrey + geom_vline(aes(xintercept=lower_bound_Jeffrey,␣
 ↪color='95% credibility interval'), linetype='dashed')
plt_Jeffrey <- plt_Jeffrey + geom_vline(aes(xintercept=upper_bound_Jeffrey,␣
 ↪color='95% credibility interval'), linetype='dashed')
plt_Jeffrey <- plt_Jeffrey + labs(title='Posterior distribution for Poisson␣
 ↪distr. with Jeffrey`s prior')
plt_Jeffrey
```

A tibble: 1 × 3

Mean	Variance	Median
<dbl>	<dbl>	<dbl>
6.15	0.615	6.11828



2.1 C) compare with a normal approximation

As parameters for the normal distribution I'm gonna use the mean of the parameters founded for the two different prior

[36]:
```
mean(mean_Unif, mean_Jeffrey)
mean(variance_Unif, variance_Jeffrey)
```

6.2

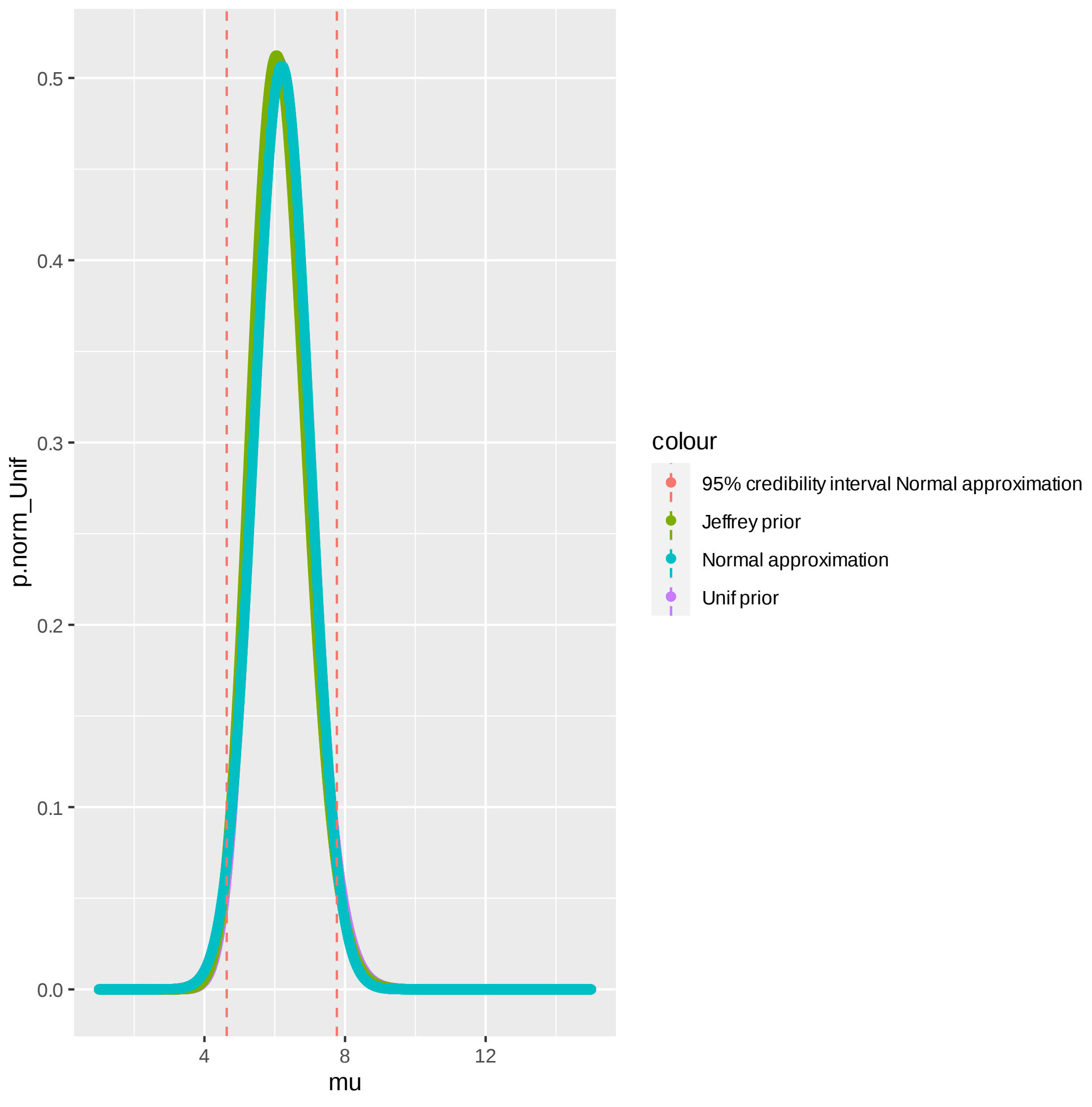
0.619999999999999

[27]:
```
mean_normal_approx <- mean(mean_Unif, mean_Jeffrey)
std_normal_approx <- sqrt(mean(variance_Unif, variance_Jeffrey))
```

```r
p.normal_approx <- dnorm(mu, mean = mean_normal_approx, sd = std_normal_approx)
lower_bound_normal_approx <- ncredint(mu, p.normal_approx, leve=0.95)[['lower']]
upper_bound_normal_approx <- ncredint(mu, p.normal_approx, level=0.
 ↪95)[['upper']]

int_median_approx <- 0
list_median <- c()
for (m in seq(1, n.sample)){
    list_median <- c(list_median, p.normal_approx[m]*delta.mu)
    int_median_approx <- sum(list_median)
    if (int_median_approx > 0.5){
        median_normal_approx <- mu[m]
        return (median_normal_approx)
        break
        }
    }

#Creating dataframe for comparison
uniform = c(median_Unif, mean_Unif, variance_Unif, lower_bound_Unif,␣
 ↪upper_bound_Unif)
jeffrey = c(median_Jeffrey, mean_Jeffrey, variance_Jeffrey,␣
 ↪lower_bound_Jeffrey, upper_bound_Jeffrey)
approx= c(median_normal_approx, mean_normal_approx, mean(variance_Unif,␣
 ↪variance_Jeffrey), lower_bound_normal_approx, upper_bound_normal_approx)

df = data.frame(Uniform = uniform, Jeffrey=jeffrey, Gauss=approx,
          row.names=c('Median','First moment', 'Second moment', 'Lower limit␣
 ↪(95% credibility)', 'Upper limit (95% credibility)'))
df

ggplot() +
geom_point(aes(mu, p.norm_Unif, color='Unif prior')) +

geom_point(aes(mu, p.norm_Jeffrey, color='Jeffrey prior')) +

geom_point(aes(mu, p.normal_approx, color='Normal approximation')) +
geom_vline(aes(xintercept=lower_bound_normal_approx, colour='95% credibility␣
 ↪interval Normal approximation'), linetype='dashed') +
geom_vline(aes(xintercept=upper_bound_normal_approx, colour='95% credibility␣
 ↪interval Normal approximation'), linetype='dashed')
```

A data.frame: 5 × 3

	Uniform <dbl>	Jeffrey <dbl>	Gauss <dbl>
Median	6.167292	6.118280	6.198800
First moment	6.200000	6.150000	6.200000
Second moment	0.620000	0.615000	0.620000
Lower limit (95% credibility)	4.672418	4.672418	4.633908
Upper limit (95% credibility)	7.791698	7.676169	7.767192



2.2 Ex.2

- a well established and diffused method for detecting a disease in blood fails to detect the presence of disease in 15% of the patients that actually have the disease.

- A young UniPD startUp has developed an innovative method of screening. During the qualification phase, a random sample of n = 75 patients known to have the disease is screened using the new method.

(a) what is the probability distribution of y, the number of times the new method fails to detect the disease ?

ANSWER: It is a Bernullian distribution

(b) on the n = 75 patients sample, the new method fails to detect the disease in y = 6 cases. What is the frequentist estimator of the failure probability of the new method ?

[28]:
```
p_freq = 6/75
cat('Frequentist estimation', 6/75)
```

```
Frequentist estimation 0.08
```

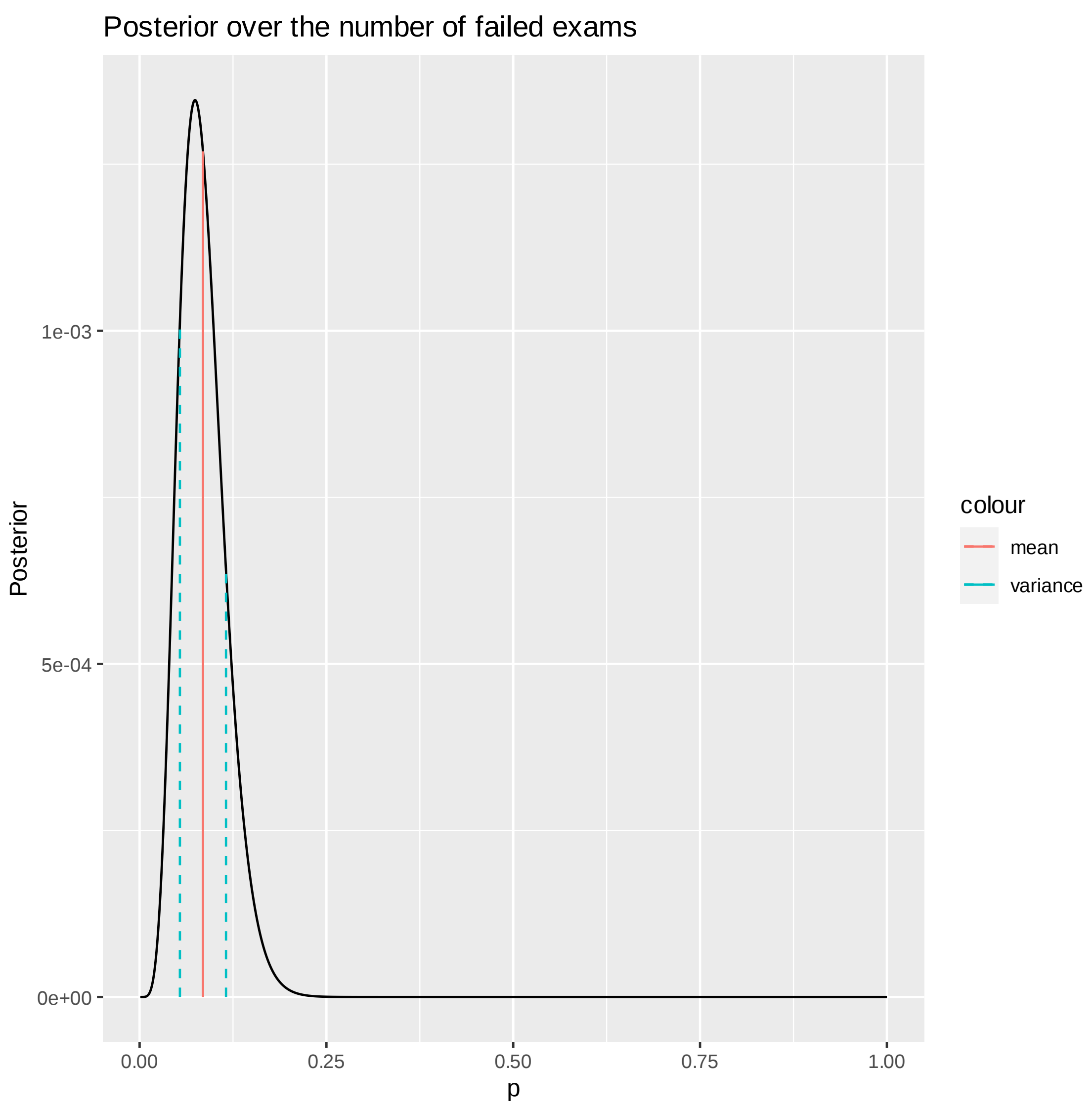
(c) setup a bayesian computation of the posterior probability, assuming a beta distribution with mean value 0.15 and standard deviation 0.14. Plot the posterior distribution for y, and mark on the plot the mean value and variance

[42]:
```
n_succ = 6
n_trial = 75
n_fail= seq(0,75, by= 1)

mean <- 0.15
std <- 0.14
var <- std*std

alpha <- mean*((mean*(1-mean))/var-1)
beta <- (1-mean)*((mean*(1-mean)/var-1))

priorB = function(p){g<-dbeta(p, alpha, beta)
                     return (g)}

#likelihood
lhB = function(p){g <-dbinom(6, n_trial, p)
                 return (g)}

Z= integrate(function(x){lhB(x)*priorB(x)}, lower=0, upper=1)$value

#posterior
postC = function(p){i <- lhB(p)*priorB(p)/Z
                   return (i)}

p <-  seq(0.001, 1, by=0.0001)
pC = postC(p)
```

[61]:
```
moment1C = sum(p*pC/sum(pC))
moment2C = sum((moment1C-p)^2 * pC/sum(pC))
t <- tibble('Mean' = moment1C,
            'Variance' = sqrt(moment2C))
t

ggplot() + geom_line(aes(x=p, y=pC/sum(pC)))+
geom_segment(aes(x = moment1C, y=0, xend=moment1C, yend=postC(moment1C)/
 ↪sum(pC),color='mean'))+
geom_segment(aes(x = moment1C-sqrt(moment2C) , y = 0, xend =␣
 ↪moment1C-sqrt(moment2C), yend = postC((moment1C-sqrt(moment2C)))/sum(pC),␣
 ↪color='variance'),linetype='dashed')+
geom_segment(aes(x = moment1C+sqrt(moment2C) , y = 0, xend =␣
 ↪moment1C+sqrt(moment2C), yend = postC((moment1C+sqrt(moment2C)))/sum(pC),␣
 ↪color='variance'),linetype='dashed')+
labs(x='p', y='Posterior', title='Posterior over the number of failed exams')
```

A tibble: 1 × 2

Mean <dbl>	Variance <dbl>
0.08478674	0.03085551



(d) Perform a test of hypothesis assuming that if the probability of failing to the detect the desease in ill patients is greater or equal than 15%, the new test is no better that the traditional method. Test the sample at a 5% level of significance in the Bayesian way.

```
[64]: #Parameter of the hypothesis testing
      p0 <- 0.15 #threshold of acceptance
      alpha <- 0.05 #confidence parameter


      #BAYESIAN APPROACH
      int = sum(pC[p>0.15]*0.0001)
      cat('Value of the posterior integral:', int, ', is smaller than alpha, than we␣
       ↪conclude the new method is better than the old one')
```

```
Value of the posterior integral: 0.03119761 , is smaller than alpha, than we
conclude the new method is better than the old one
```

(e) Perform the same hypothesis test in the classical frequentist way

[65]:
```
#FREQUENTIST APPROACH
#H0: the probability to fail the test is less or
#equal than 0.15


P0 = 0.15
alpha = 0.05

p6 = pbinom(6, size=75, prob=0.15)

cat('P-value:', p6, '\n')
cat('p-value<alpha:',p6<alpha, '. We accept the null hypotesis, the new test is␣
 ↪no better than the old one')
```

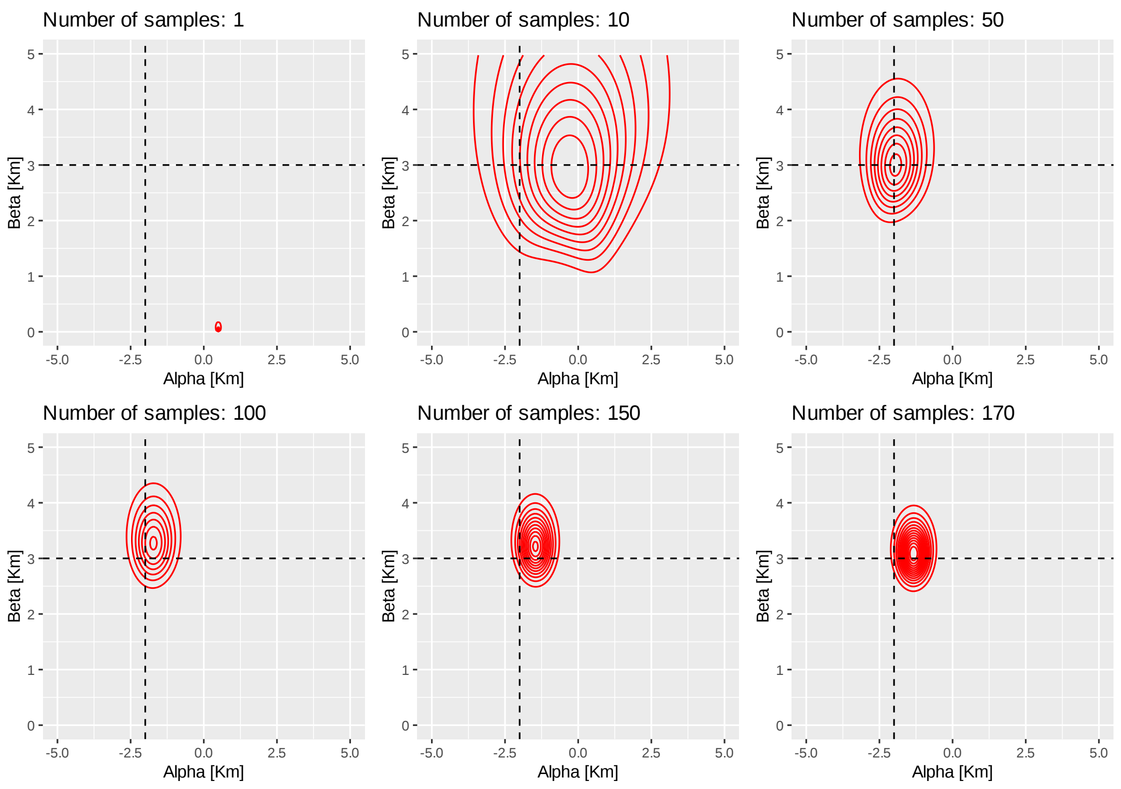
```
P-value: 0.0543533
p-value<alpha: FALSE . We accept the null hypotesis, the new test is no better
than the old one
```

2.3 Ex.3

- given the problem of the lightouse discussed last week, study the case in which both the position along the shore ( ) and the distance out at sea ( ) are unknown.

[73]:
```
# for each observation
func = function(obs){
    set.seed(12345)

    # true value of the lighthouse
    alpha_true <- -2
    beta_true <- 3

    # positions of the light sightings
    x <- function(alpha = alpha_true, beta = beta_true, n) {
        angle <- runif(n, min = -pi/2, max = pi/2)
        return(alpha + beta * tan(angle))
    }

    # total number of observations to base the analysis
    tot_obs = c(1, 3, 5, 10, 50, 100, 150, 175)

    # compute the sightings positions
    light_obervations <- x(n = max(tot_obs))

    # log posterior
    posterior <- function(a, b, x_pos) {
        logL <- 0
        for (x in x_pos) {
            logL <- logL + log(b/(pi * (b^2 + (x - a)^2)))
        }
        return(logL)
    }

    # integration variables
    n_points <- 200

    x_min <- -5
    x_max <- 5

    y_min <- 0
    y_max <- 5

    dx <- (x_max - x_min)/n_points
    dy <- (y_max - y_min)/n_points
    dxy <- (x_max - x_min) * (y_max - y_min)/(n_points^2)

    alpha <- seq(from = x_min, by = dx, length.out = n_points)
    beta <- seq(from = y_min, by = dy, length.out = n_points)

    # palette and graphic stuff
    cols <- rev(hcl.colors(10, "Reds"))
    par(mfrow = c(4, 2), cex.main = 2, cex.axis = 1.5)
    options(repr.plot.width = 16, repr.plot.height = 32)
    log_post <- matrix(data = NA, nrow = n_points, ncol = n_points)  # create␣
 ↪an empty matrix

    # Compute log unnormalized posterior , z = ln P^*(a,b|D), on a regular grid
    df = expand_grid(alpha, beta)
    log_post = c()
    for (i in 1:length(pull(df['alpha']))){log_post = c(log_post,␣
 ↪posterior(pull(df['alpha'])[i], pull(df['beta'])[i], light_obervations[1:
 ↪obs]))}
    log_post=log_post-max(log_post)
    df = df |> add_column(post = exp(log_post)/(dxy * sum(exp(log_post))))

    ggplot(df, aes(x = alpha, y=beta,␣
 ↪z=post))+geom_contour(color='red')+labs(x='Alpha [Km]', y='Beta [Km]',␣
 ↪title=paste0('Number of samples: ', obs))+xlim(x_min, x_max)+ylim(y_min,␣
 ↪y_max)+
    geom_vline(xintercept = alpha_true,␣
 ↪linetype='dashed')+geom_hline(yintercept = beta_true, linetype='dashed')
}
```

[74]:
```
i_ls <-c(1, 10, 50, 100, 150, 170)
options(repr.plot.width = 10, repr.plot.height =3)
plots = lapply(i_ls, function(.i_ls){func(.i_ls)})
options(repr.plot.width = 10, repr.plot.height=7)
do.call(grid.arrange, c(grobs=plots, nrow=2))
```



2.4 EX.4

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

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

- change the resolution `w = {0.1, 0.25, 1, 2, 3}`

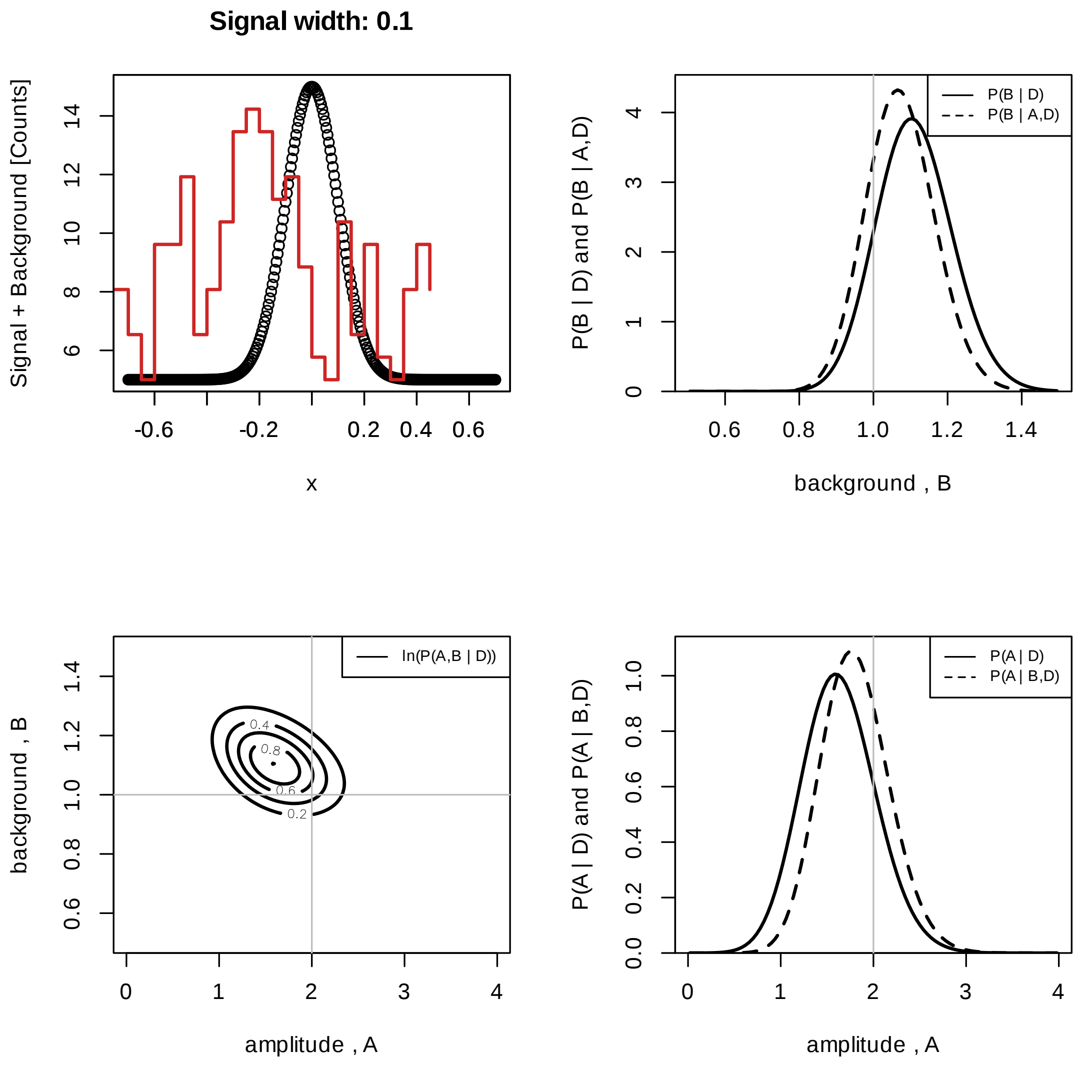
(note: Jupyter notebook dosen't seem to adapt well when using the plot function (no ggplot). For this reasons the plots for each `w` have been put into 2x2 matrix.)

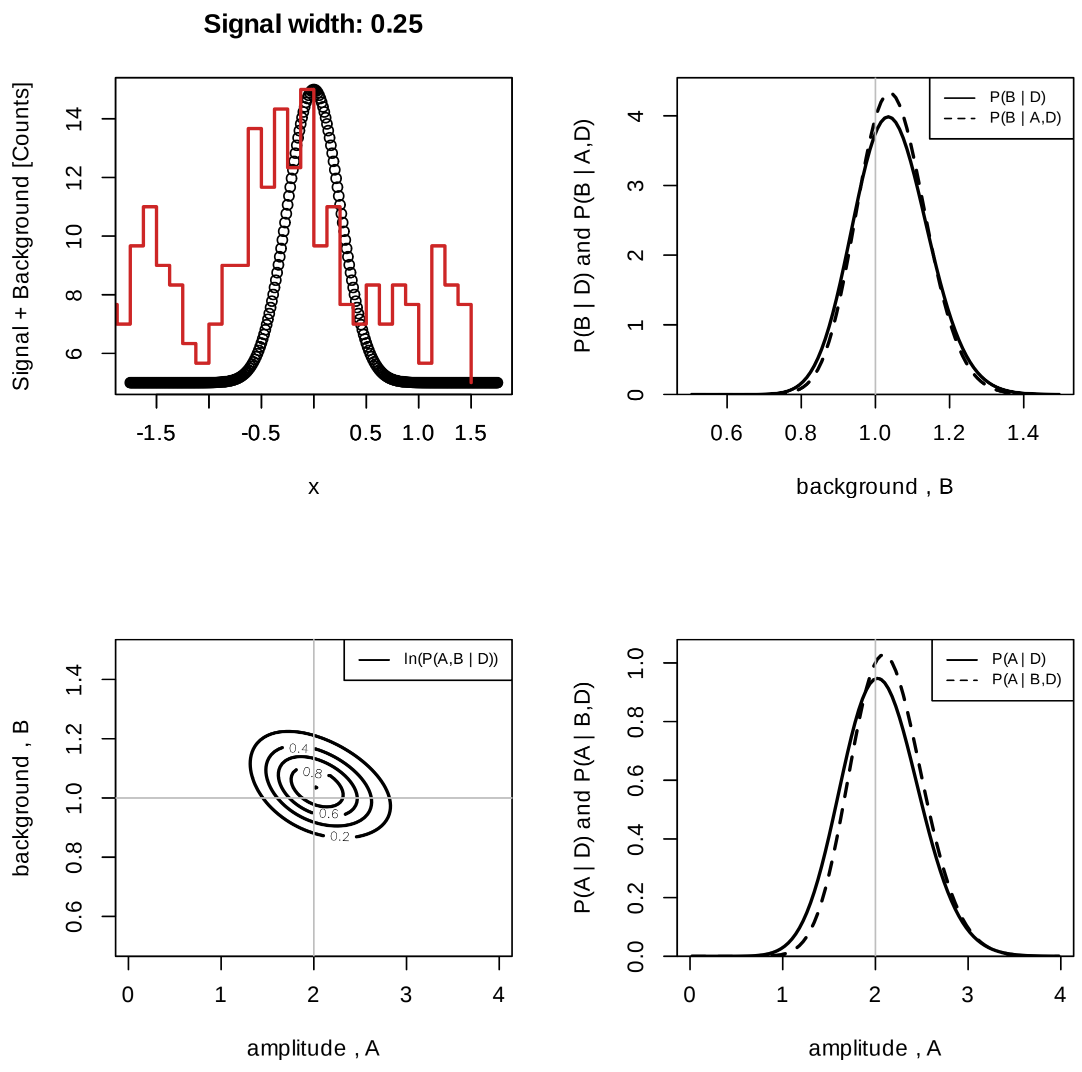
[83]:
```
# - Generative model
signal <- function (x, a, b, x0, w, t) {
t * (a*exp(-(x-x0)^2/(2*w^2)) + b)
}
```

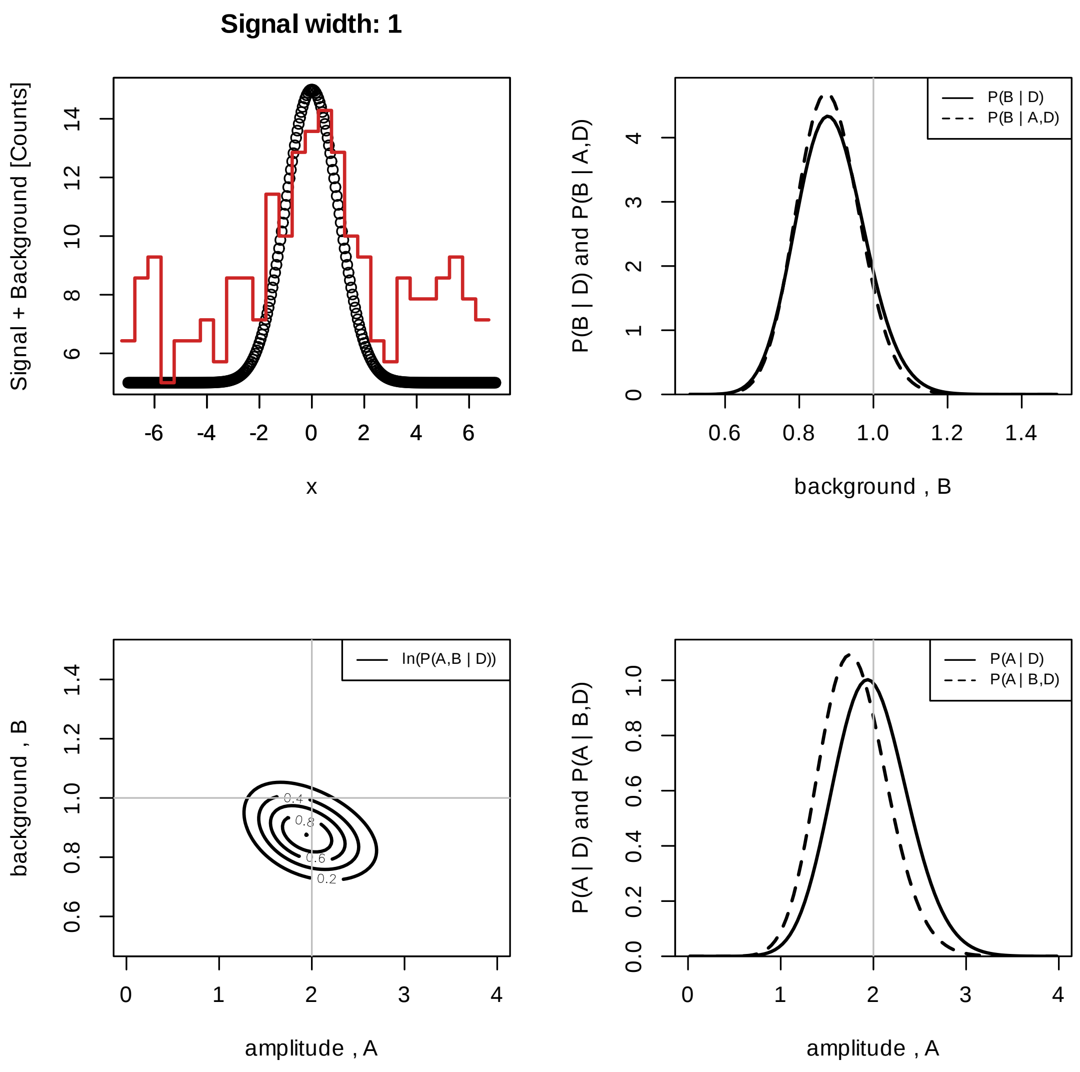
```r
# Define model parameters
x0 <- 0 # Signal peak
w <-1 # Signal width
A.true <- 2 # Signal amplitude
B.true <- 1 # Background amplitude
Delta.t <- 5 # Exposure time

set.seed(205)

omega <- c(0.1, 0.25, 1, 2, 3)

for(w in omega){

    layout((matrix(1:4, nrow = 2, ncol = 2)), respect = FALSE)

    #Plotting Signal and Signal+Background
    xdat <- seq(from=-7*w, to=7*w, by=0.5*w)
    s.true <- signal (xdat , A.true , B.true , x0, w, Delta.t)
    ddat <- rpois(length (s.true), s.true)
    xplot <- seq(from=min(xdat), to=max(xdat), by=0.05*w)
    splot <- signal (xplot , A.true , B.true , x0, w, Delta.t)
    plot(xplot , splot, xlab="x", ylab="Signal + Background [Counts]", main =␣
 ↪paste("Signal width:", w))
    par(new=TRUE)
    xdat.off <- xdat -0.25
    plot(xdat.off , ddat , type='s',col='firebrick 3',
        lwd=2, xlim=range(xplot), ylim=range(c(splot , ddat)), ylab='',␣
 ↪xlab='', yaxt="n")

    # - Sampling grid for computing posterior
    alim <- c(0.0, 4.0)
    blim <- c(0.5, 1.5)
    Nsamp <- 100
    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


    # Log posterior
    log.post <- function (d, x, a, b, x0, w, t) {
        if(a<0 || b <0) {return(-Inf )} # the effect of the prior
        sum(dpois(d, lambda=signal(x, a, b, x0, w, t), log=TRUE))
        }
```

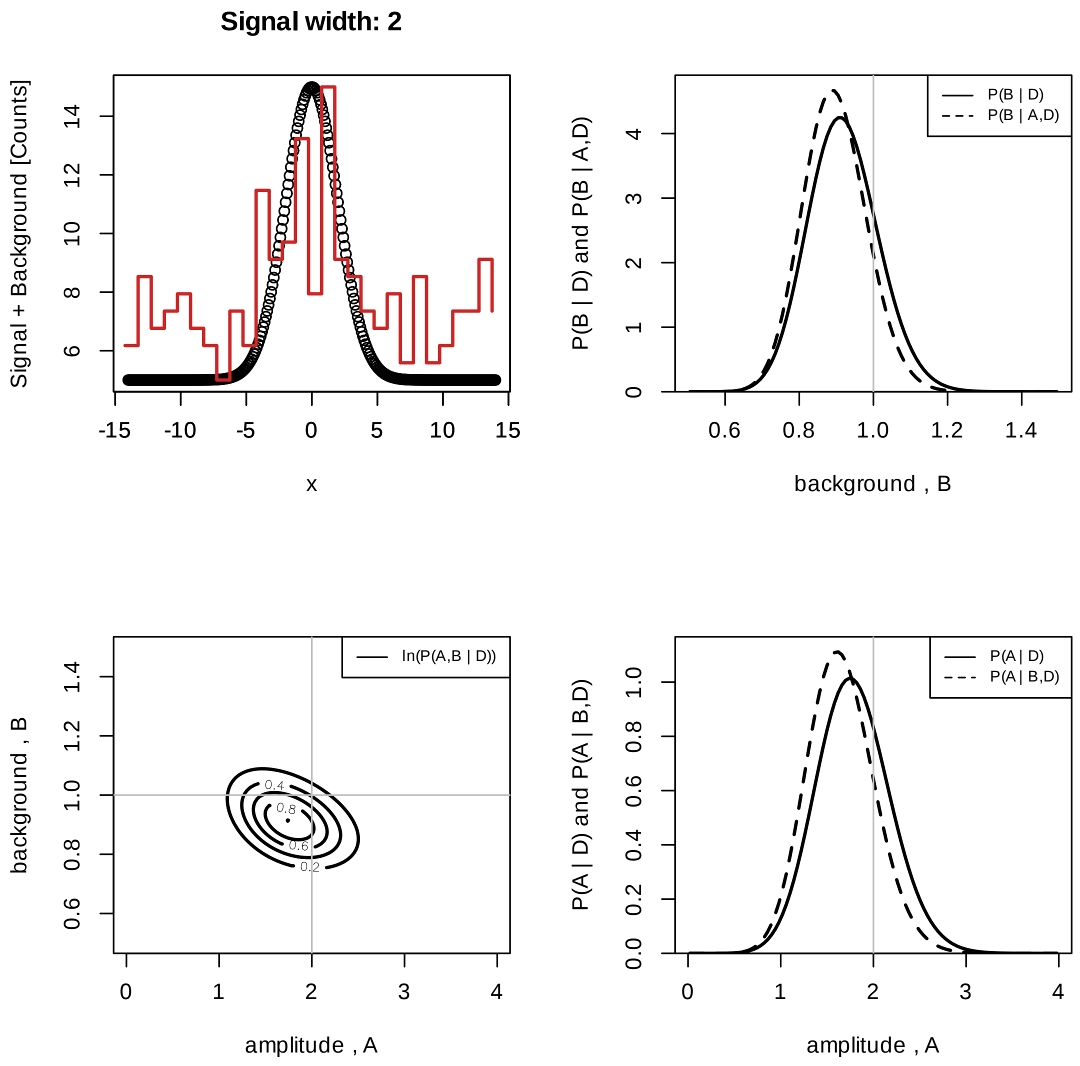
```
# Compute log unnormalized posterior , z = ln P^*(a,b|D), on a regular grid
z <- matrix(data=NA , nrow= length (a), ncol=length(b))
for(j in 1:length(a)) {
    for(k in 1:length(b)) {
        z[j,k] <- log.post(ddat , xdat , a[j], b[k], x0, w, Delta.t)
    }
}

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")
legend(x = 'topright', legend=c('ln(P(A,B | D))'), lty=c(1), cex = 0.7, box.
↪lty=1)

abline(v=2,h=1,col="grey")

# Compute normalized marginalized posteriors , P(a|D) and P(b|D)
# by summing over other parameter . Normalize by gridding .
p_a_D <- apply(exp(z), 1, sum)
p_a_D <- p_a_D/( delta_a*sum(p_a_D))
p_b_D <- apply(exp(z), 2, sum)
p_b_D <- p_b_D/( delta_b*sum(p_b_D))

# Compute normalized conditional posteriors , P(a|b,D) and P(b|a,D)
# using true values of conditioned parameters . Vectorize (func , par)
# makes a vectorized function out of func in the parameter par.
p_a_bD <- exp( Vectorize (log.post , "a")(ddat , xdat , a, B.true ,
        x0, w, Delta.t))
p_a_bD <- p_a_bD/( delta_a*sum(p_a_bD))
p_b_aD <- exp( Vectorize (log.post , "b")(ddat , xdat , A.true , b,
        x0, w, Delta.t))
p_b_aD <- p_b_aD/( delta_b*sum(p_b_aD))

#par(mfrow=c(2,2), mgp=c(2,0.8,0), mar=c(3.5,3.5,1,1), oma=0.1*c(1,1,1,1))
# Plot the 1D marginalized posteriors
plot(b, p_b_D, xlab="background , B", yaxs="i",
    ylim=1.05*c(0,max(p_b_D, p_b_aD)), ylab="P(B | D) and P(B | A,D)",
    type="l", lwd=2)
lines(b, p_b_aD , lwd=2, lty=2)
abline(v=B.true , col="grey")
```

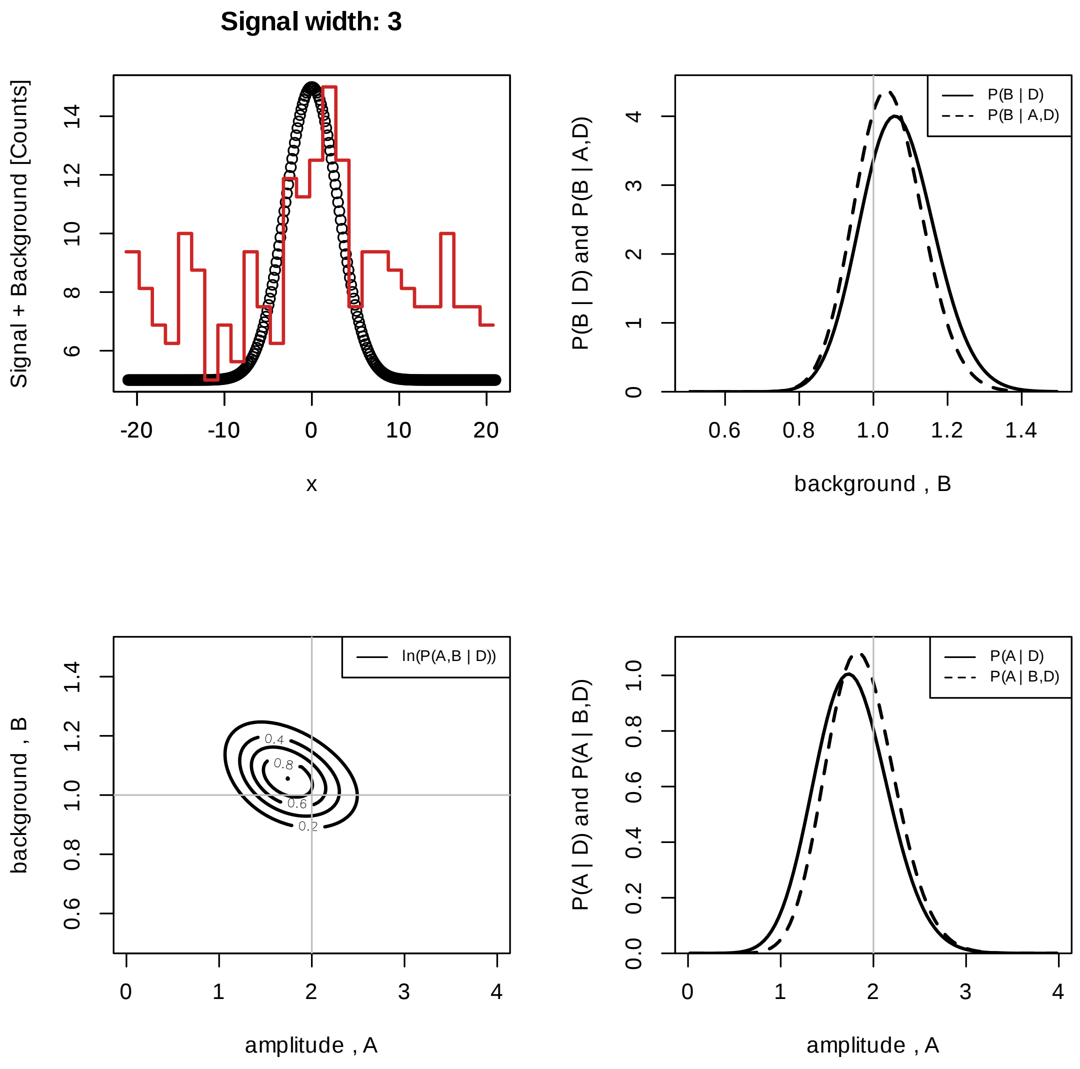
```
    legend(x = 'topright',legend=c('P(B | D)', 'P(B | A,D)' ), lty=c(1, 2), cex␣
↪= 0.7, box.lty=1)

    plot(a, p_a_D, xlab="amplitude , A", yaxs="i",
        ylim=1.05*c(0,max(p_a_D, p_a_bD)), ylab="P(A | D) and P(A | B,D)",
        type="l", lwd=2)
    lines(a, p_a_bD , lwd=2, lty=2)
    abline(v=A.true , col="grey")
    legend( x = 'topright', legend=c('P(A | D)', 'P(A | B,D)'), lty=c(1, 2),␣
↪cex = 0.7, box.lty=1)
}
```




Signal width: 0.25
Signal + Background [Counts]
x
P(B | D) and P(B | A,D)
P(B | D)
P(B | A,D)
background , B
ln(P(A,B | D))
background , B
amplitude , A
P(A | D) and P(A | B,D)
P(A | D)
P(A | B,D)
amplitude , A


Signal width: 1
Signal + Background [Counts]
x
P(B | D) and P(B | A,D)
P(B | D)
P(B | A,D)
background , B
ln(P(A,B | D))
0.2
0.4
0.6
0.8
background , B
amplitude , A
P(A | D) and P(A | B,D)
P(A | D)
P(A | B,D)
amplitude , A

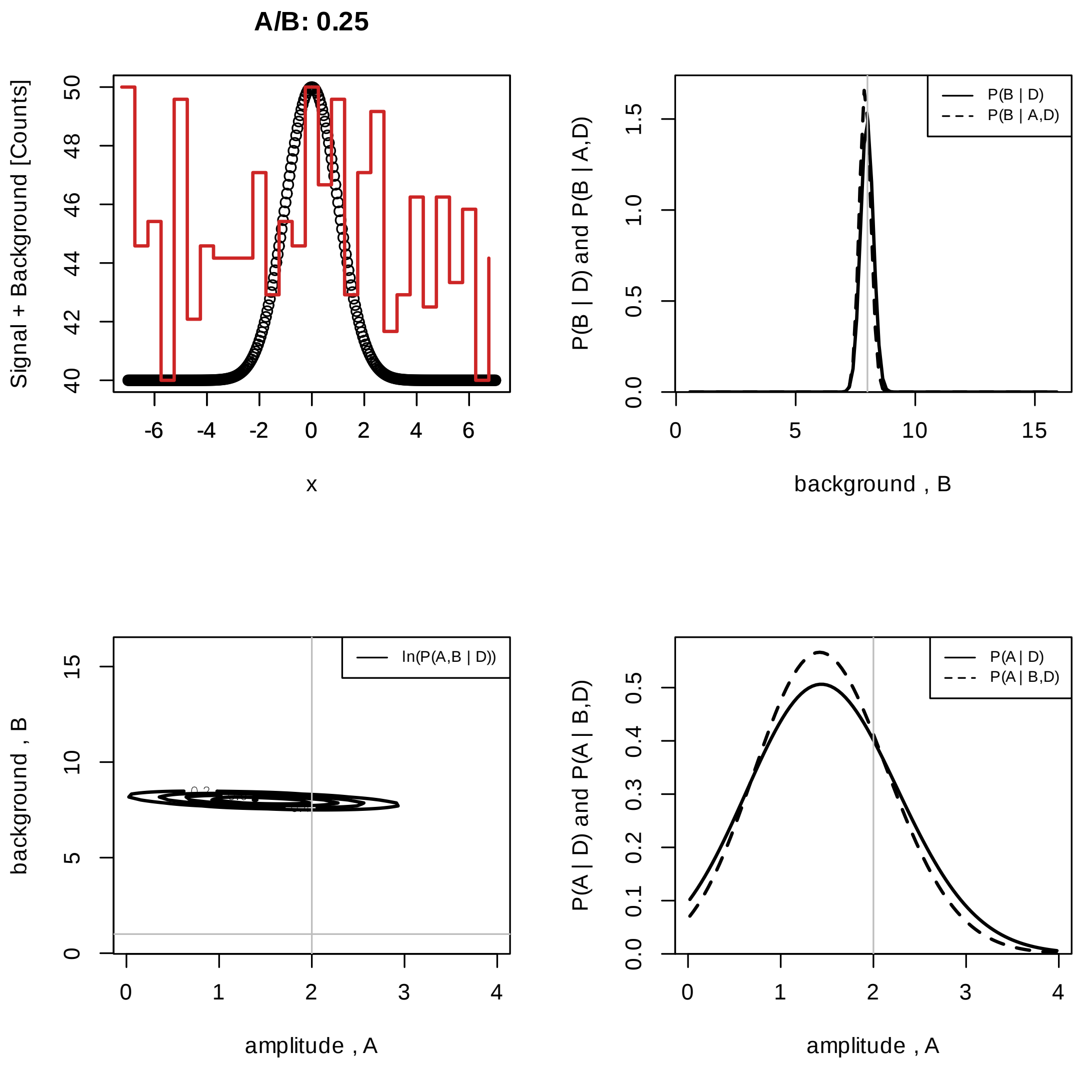

Signal width: 2
Signal + Background [Counts]
x
P(B | D) and P(B | A,D)
P(B | D)
P(B | A,D)
background , B
background , B
ln(P(A,B | D))
0.2
0.4
0.6
0.8
amplitude , A
P(A | D) and P(A | B,D)
P(A | D)
P(A | B,D)
amplitude , A

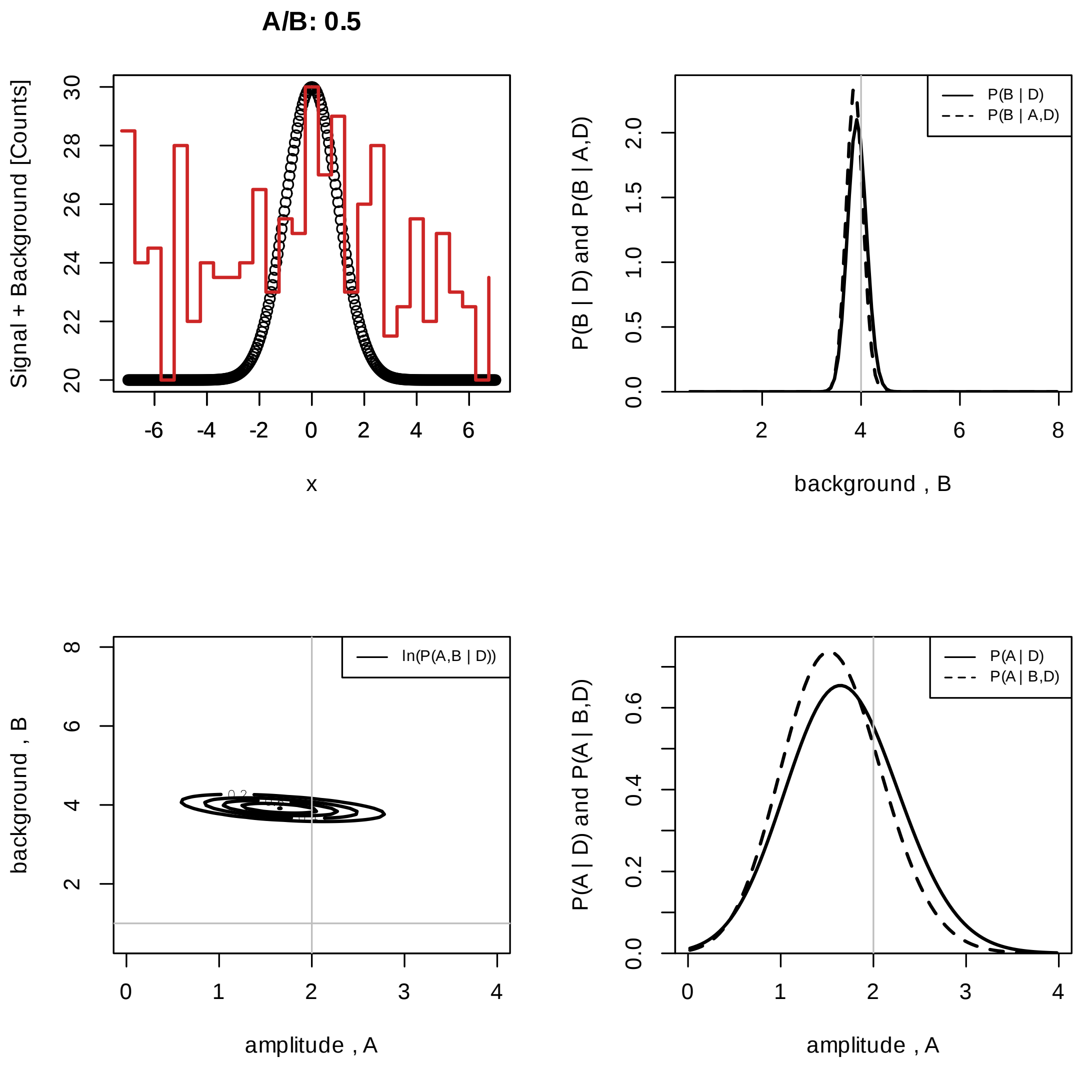


(b) change the ratio A/B used to simulate the data (keeping both positive in accordance with the prior) The possible `A/B = {0.1, 0.25, 1, 2, 3}`. A has been kept costant while B was obtained from B and A/B.

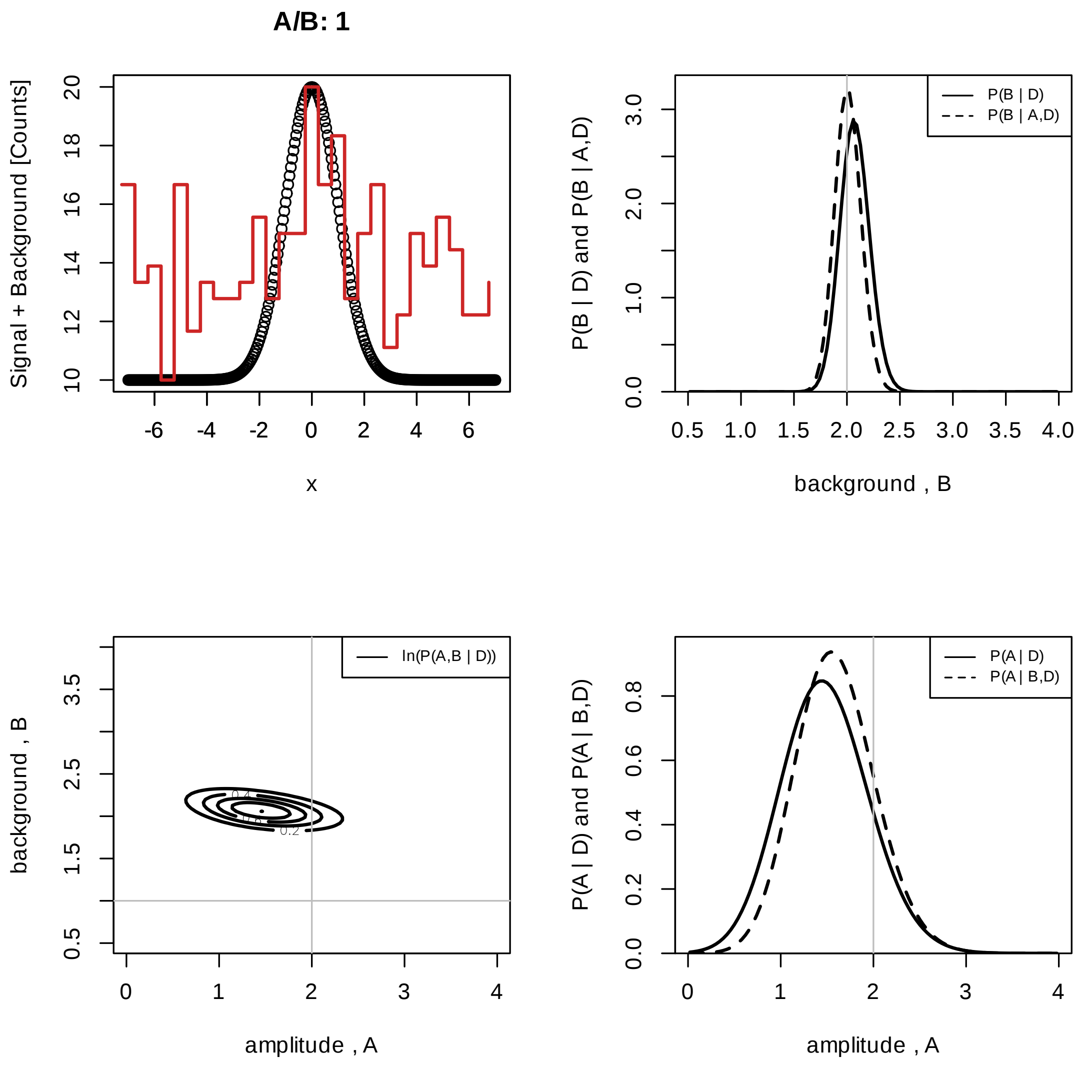
[94]:
```
ratio_AB <- c(1/4, 1/2, 1, 2, 4)

for (ratio in ratio_AB){

    layout((matrix(1:4, nrow = 2, ncol = 2)), respect = FALSE)

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

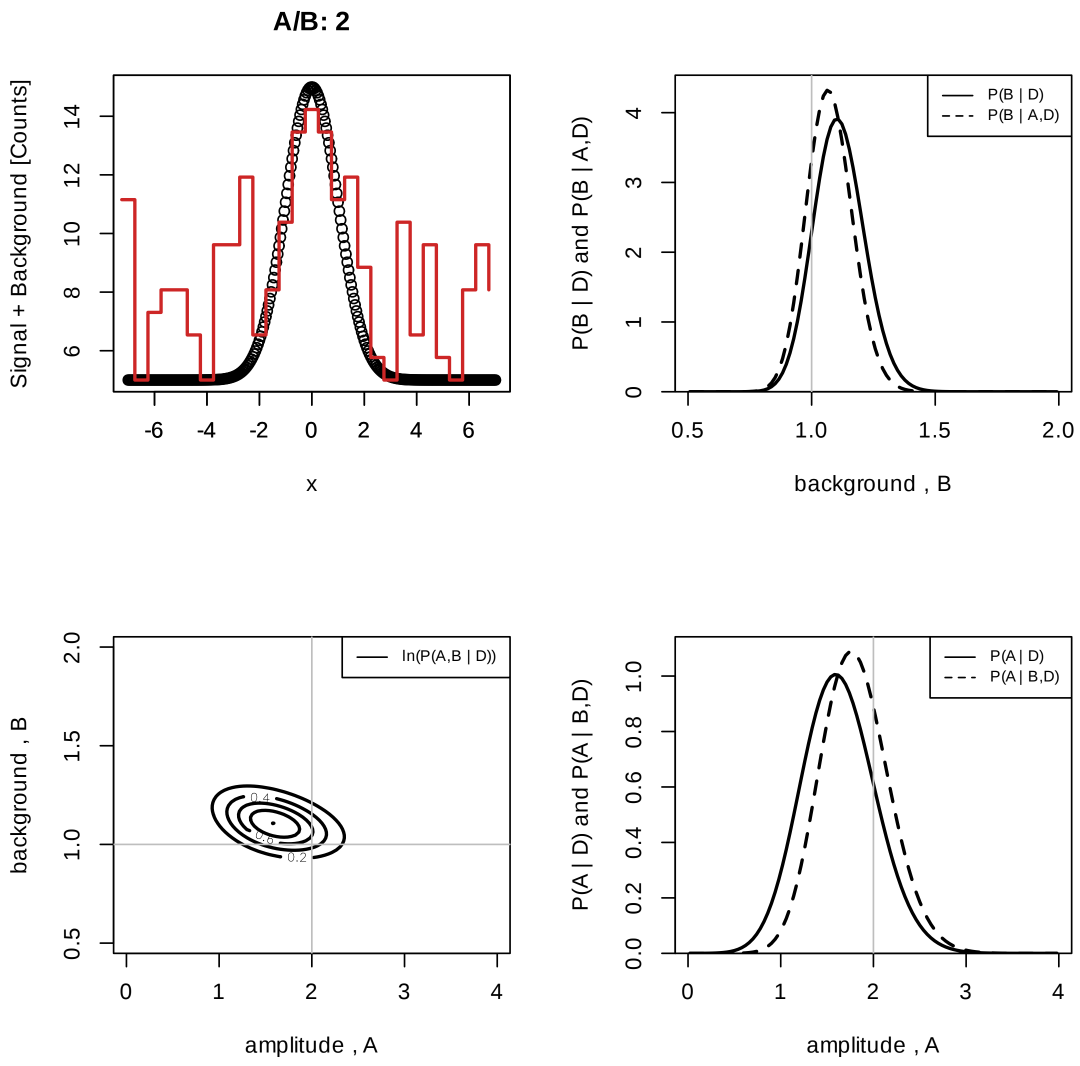
```
}

# Define model parameters
x0 <- 0 # Signal peak
w <-1 # Signal width
A.true <- 2 # Signal amplitude
B.true <- A.true * ratio^(-1) # Background amplitude
Delta.t <- 5 # Exposure time

set.seed(205)

#Plotting Signal and Signal+Background
xdat <- seq(from=-7*w, to=7*w, by=0.5*w)
s.true <- signal (xdat , A.true , B.true , x0, w, Delta.t)
ddat <- rpois(length (s.true), s.true)
xplot <- seq(from=min(xdat), to=max(xdat), by=0.05*w)
splot <- signal (xplot , A.true , B.true , x0, w, Delta.t)
plot(xplot , splot, xlab="x", ylab="Signal + Background [Counts]", main =␣
↪paste("A/B:", ratio))
par(new=TRUE)
xdat.off <- xdat -0.25
plot(xdat.off , ddat , type='s',col='firebrick 3',
    lwd=2, xlim=range(xplot), ylim=range(c(splot , ddat)), ylab='',␣
↪xlab='', yaxt="n")

# - Sampling grid for computing posterior
alim <- c(0.0, 2*A.true)
blim <- c(0.5, 2*B.true)
Nsamp <- 100
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


# Log posterior
log.post <- function (d, x, a, b, x0, w, t) {
    if(a<0 || b <0) {return(-Inf )} # the effect of the prior
    sum(dpois(d, lambda=signal(x, a, b, x0, w, t), log=TRUE))
    }

# 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")
    legend(x = 'topright', legend=c('ln(P(A,B | D))'), lty=c(1), cex = 0.7, box.
↪lty=1)
    abline(v=2,h=1,col="grey")

    # Compute normalized marginalized posteriors , P(a|D) and P(b|D)
    # by summing over other parameter . Normalize by gridding .
    p_a_D <- apply(exp(z), 1, sum)
    p_a_D <- p_a_D/( delta_a*sum(p_a_D))
    p_b_D <- apply(exp(z), 2, sum)
    p_b_D <- p_b_D/( delta_b*sum(p_b_D))
    # Compute normalized conditional posteriors , P(a|b,D) and P(b|a,D)
    # using true values of conditioned parameters . Vectorize (func , par)
    # makes a vectorized function out of func in the parameter par.
    p_a_bD <- exp( Vectorize (log.post , "a")(ddat , xdat , a, B.true ,
            x0, w, Delta.t))
    p_a_bD <- p_a_bD/( delta_a*sum(p_a_bD))
    p_b_aD <- exp( Vectorize (log.post , "b")(ddat , xdat , A.true , b,
            x0, w, Delta.t))
    p_b_aD <- p_b_aD/( delta_b*sum(p_b_aD))

    #par(mfrow=c(2,2), mgp=c(2,0.8,0), mar=c(3.5,3.5,1,1), oma=0.1*c(1,1,1,1))
    # Plot the 1D marginalized posteriors
    plot(b, p_b_D, xlab="background , B", yaxs="i",
        ylim=1.05*c(0,max(p_b_D, p_b_aD)), ylab="P(B | D) and P(B | A,D)",
        type="l", lwd=2)
    lines(b, p_b_aD , lwd=2, lty=2)
    abline(v=B.true , col="grey")
    legend(x = 'topright',legend=c('P(B | D)', 'P(B | A,D)' ), lty=c(1, 2), cex␣
↪= 0.7, box.lty=1)

    plot(a, p_a_D, xlab="amplitude , A", yaxs="i",
        ylim=1.05*c(0,max(p_a_D, p_a_bD)), ylab="P(A | D) and P(A | B,D)",
        type="l", lwd=2)
    lines(a, p_a_bD , lwd=2, lty=2)
    abline(v=A.true , col="grey")
    legend( x = 'topright', legend=c('P(A | D)', 'P(A | B,D)'), lty=c(1, 2),␣
  ↪cex = 0.7, box.lty=1)
}
```

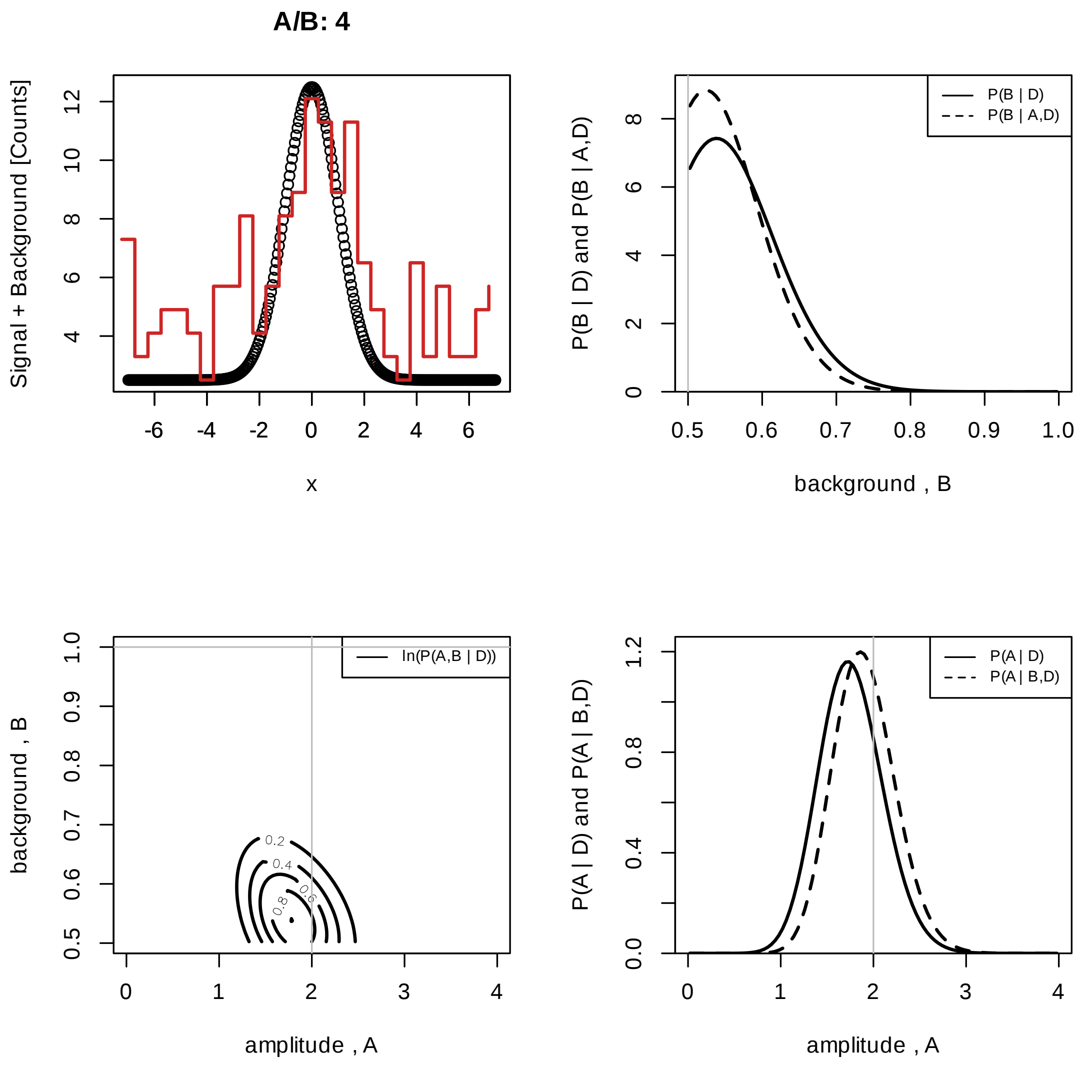




A/B: 0.5
Signal + Background [Counts]
x
P(B | D) and P(B | A,D)
background , B
P(B | D)
P(B | A,D)
background , B
amplitude , A
ln(P(A,B | D))
P(A | D) and P(A | B,D)
amplitude , A
P(A | D)
P(A | B,D)


A/B: 1
Signal + Background [Counts]
x
P(B | D) and P(B | A,D)
background , B
P(B | D)
P(B | A,D)
background , B
amplitude , A
ln(P(A,B | D))
P(A | D) and P(A | B,D)
amplitude , A
P(A | D)
P(A | B,D)


A/B: 2
Signal + Background [Counts]
x
P(B | D) and P(B | A,D)
background , B
P(B | D)
P(B | A,D)
background , B
amplitude , A
ln(P(A,B | D))
0.2
0.4
0.6
P(A | D) and P(A | B,D)
amplitude , A
P(A | D)
P(A | B,D)



Varying the Signal to Noise ratio we can cleary see how strong it impact the marginal distribution, especially when the ration is below 1, like in the two cases `A/B = {0.25, 0.5}`.

```
[ ]:
```