Bayesian Data Analysis HW02

I have executed these exercises on my own and written the answers in my own words. Signed: Shashi Shankar

1.

```
show( HairEyeColor ) # Show data
```

```
## , , Sex = Male
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8
##
## , , Sex = Female
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
```

The HairEyeColor data above is a 3-D array with two layers for male and female.

```
EyeHairFreq = apply( HairEyeColor, c("Eye","Hair"), sum ) # Sum across sex
EyeHairProp = EyeHairFreq / sum( EyeHairFreq ) # joint proportions, Table 4.1
show( round( EyeHairProp , 2 ) )
```

```
##        Hair
## Eye     Black Brown  Red Blond
##   Brown  0.11  0.20 0.04  0.01
##   Blue   0.03  0.14 0.03  0.16
##   Hazel  0.03  0.09 0.02  0.02
##   Green  0.01  0.05 0.02  0.03
```

The apply() function above applies the sum function to the HairEyeColor 3-D array, across sex dimension (i.e. the dimesnion excluding Eye and Hair). The next line divides by the total number of elements, to compute the proportion of the sample in each cell. The last line rounds each value to 2 decimal places.

```
HairFreq = apply( HairEyeColor , c("Hair") , sum ) # Sum across sex and eye
HairProp = HairFreq / sum( HairFreq ) # marginal proportions, Table 4.1
show( round( HairProp , 2 ) )
```

```
## Black Brown   Red Blond
##  0.18  0.48  0.12  0.21
```

The apply() function above applies the sum function to the HairEyeColor 3-D array, across sex and eye dimensions (i.e. the dimesnion excluding Hair). The next line divides by the total number of elements, to compute the marginal proportions of hair colors. The last line rounds each value to 2 decimal places.

```
EyeFreq = apply( HairEyeColor , c("Eye") , sum ) # Sum across sex and eye
EyeProp = EyeFreq / sum( EyeFreq ) # marginal proportions, Table 4.1
show( round( EyeProp , 2 ) )
```

```
## Brown  Blue Hazel Green
##  0.37  0.36  0.16  0.11
```

The above code snippet calculates marginal proportions of eye colors.

```
EyeHairProp["Blue",] / EyeProp["Blue"] # conditional prob, Table 4.2
```

```
##      Black      Brown        Red      Blond
## 0.09302326 0.39069767 0.07906977 0.43720930
```

The above line calculates the conditional probability of hair color given blue eyes by dividing the “Blue” row of EyeHairProp array by the marginal probability in the “Blue” cell.

```
# Hair colors given brown eyes:
# Notice location of comma!
EyeHairProp["Brown",] / EyeProp["Brown"]
```

```
##      Black      Brown        Red      Blond
## 0.30909091 0.54090909 0.11818182 0.03181818
```

The above line calculates the conditional probability of hair color given brown eyes by dividing the “Brown” row of EyeHairProp array by the marginal probability in the “Brown” cell.

```
# Eye colors given brown hair:
# Notice location of comma!
EyeHairProp[,"Brown"] / HairProp["Brown"]
```

```
##     Brown      Blue     Hazel     Green
## 0.4160839 0.2937063 0.1888112 0.1013986
```

The above line calculates the conditional probability of eye color given brown hair.

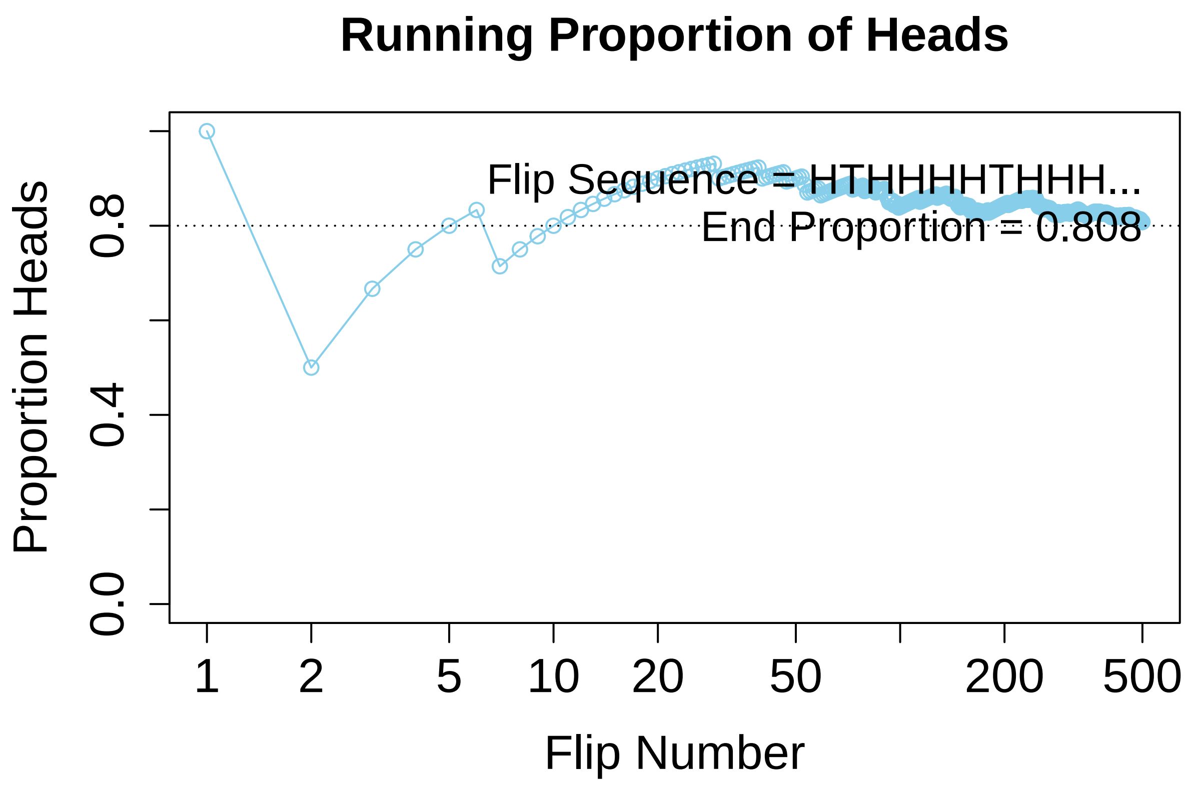
2.

```
setwd("C:\\Users\\hoosi\\Desktop\\BDA\\DBDA2Eprograms")
source("DBDA2E-utilities.R") # Definitions of openGraph, saveGraph, etc.
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************

## Loading required package: coda

## Linked to JAGS 4.2.0

## Loaded modules: basemod,bugs
```

```
N = 500 # Specify the total number of flips, denoted N.
pHeads = 0.8 # Specify underlying probability of heads.
# Generate a random sample of N flips (heads=1, tails=0):
flipSequence = sample( x=c(0,1), prob=c(1-pHeads,pHeads), size=N, replace=TRUE)
# Compute the running proportion of heads:
r = cumsum( flipSequence ) # Cumulative sum: Number of heads at each step.
```

```
n = 1:N # Number of flips at each step.
runProp = r / n # Component by component division.
# Graph the running proportion:
plot( n , runProp , type="o" , log="x" , col="skyblue" ,
xlim=c(1,N) , ylim=c(0.0,1.0) , cex.axis=1.5 ,
xlab="Flip Number" , ylab="Proportion Heads" , cex.lab=1.5 ,
main="Running Proportion of Heads" , cex.main=1.5 )
# Plot a dotted horizontal reference line:
abline( h=pHeads , lty="dotted" )
# Display the beginning of the flip sequence:
flipLetters = paste( c("T","H")[flipSequence[1:10]+1] , collapse="" )
displayString = paste0( "Flip Sequence = " , flipLetters , "..." )
text( N , .9 , displayString , adj=c(1,0.5) , cex=1.3 )
# Display the relative frequency at the end of the sequence.
text( N , .8 , paste("End Proportion =",runProp[N]) , adj=c(1,0.5) , cex=1.3 )
```



3A.

```
# Given x = 9.9, mean = 10.0, sd = 0.2
x = 9.9
m = 10.0
sd = 0.2
dnorm(x,m,sd)
```

```
## [1] 1.760327
```

3B.

```
N = 100000 #Number of random points in the sample.
m = 10.0
sd = 0.2
heap0data = rnorm(N,m,sd)
```

3C.

```
sum( heap0data >= 9.8 & heap0data < 10.0 )/(N * 0.2)
```

```
## [1] 1.7083
```

The condition 'sum(heapOdata >= 9.8 & heapOdata < 10.0)' means the number of points in between 9.8 and 10.

4.

```
probaPositiveGivenDisease = 0.99 # True positive rate of test
probaPositiveGivenNoDisease = 0.05 # false positive rate of test
probaDisease = 0.001 # prior before test 1
#Applying Baye's rule for first test
probaDiseaseGivenPositive = ( probaPositiveGivenDisease * probaDisease / ( probaPositiveGivenDisease * p
probaDiseaseGivenPositive
```

```
## [1] 0.01943463
```

```
# Updating the prior before retest:
probaDisease = probaDiseaseGivenPositive

#Applying Baye's rule for retest
probaDiseaseGivenNegative = ( (1.0-probaPositiveGivenDisease) * probaDisease /
 ( (1.0-probaPositiveGivenDisease) * probaDisease
 + (1.0-probaPositiveGivenNoDisease) * (1.0-probaDisease) ) )

probaDiseaseGivenNegative
```

```
## [1] 0.0002085862
```

5A.

```
'freq(D=+,\theta=:))'
```

```
## [1] "freq(D=+,\theta=:))"
```

```
0.05 * (1.0 - 0.001) * 100000
```

```
## [1] 4995
```

```
'freq(D=-,\theta=:))'
```

```
## [1] "freq(D=-,\theta=:))"
```

```
(1.0-0.05) * (1.0-0.001) * 100000
```

```
## [1] 94905
```

```
'freq(D=+)'
```

```
## [1] "freq(D=+)"
```

```
4995+99
```

```
## [1] 5094
```

```
'freq(D=-)'
```

```
## [1] "freq(D=-)"
```

```
94905+1
```

```
## [1] 94906
```

5B.

Using natural frequencies from the table, 99 out of 5094 people are having the disease. Which is approximately 100 out of 5000 i.e. 2 percent. No it doesn't match the intuitive answer at all.

Exact ratio =

```
99/5094
```

```
## [1] 0.01943463
```

The above value exactly matches the value calculated using Baye's rule in part A.

5C.

In the left part of the tree, frequency of the 1st empty box from the top:

```
10000 * 0.99
```

```
## [1] 9900
```

frequency of the box at the bootom:

```
9900 * (1-0.99)
```

```
## [1] 99
```

In the right part of the tree, frequency of the 1st empty box from the top:

```
9990000 * 0.05
```

```
## [1] 499500
```

frequency of the box at the bottom:

```
499500 * (1-0.05)
```

```
## [1] 474525
```

5D.

Their proportion in the left branch of the tree:

```
99 / ( 99 + 474525 )
```

```
## [1] 0.0002085862
```

It exactly matches the result in question 4.

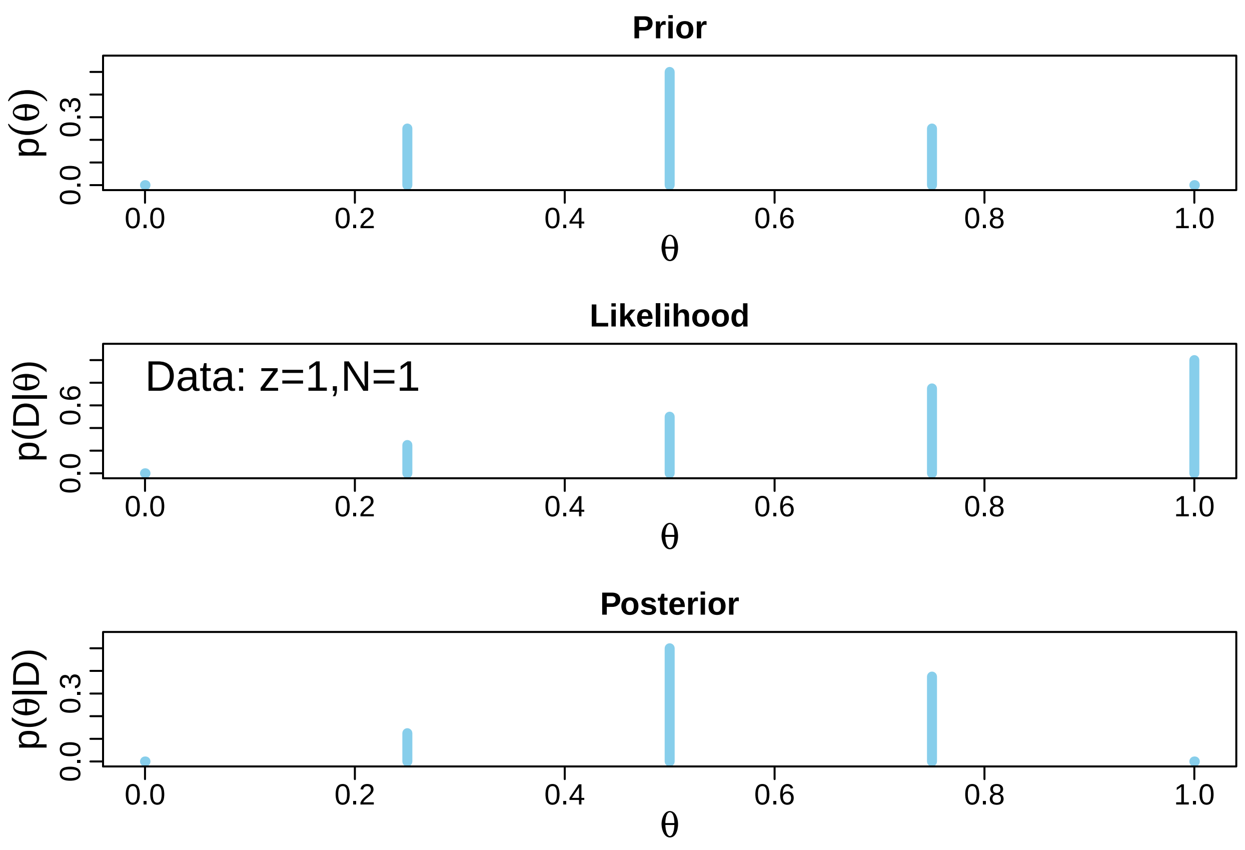
6.

```
setwd("C:\\Users\\hoosi\\Desktop\\BDA\\DBDA2Eprograms")
graphics.off()
source("DBDA2E-utilities.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
```

```
source("BernGrid.R")
#source("BernGridExample.R")
```

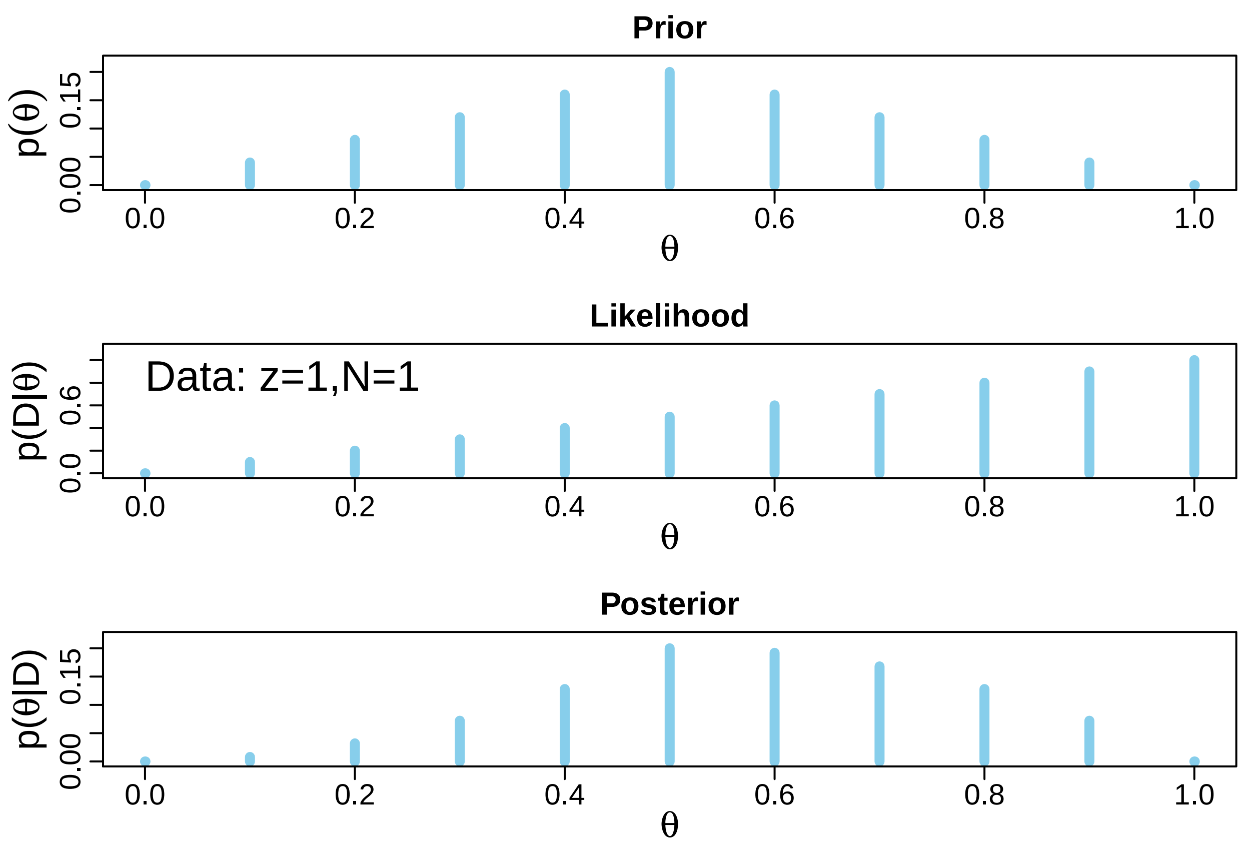
```
Theta = seq( 0 , 1 , length=5 )  # Sparse teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,0),rep(1,1))      # Single flip with 1 head

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="None" , showHDI=FALSE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample0",type="eps")
```

This plot shows the reallocation of credibility for a small number of candidate parameter values and a single flip of coin.

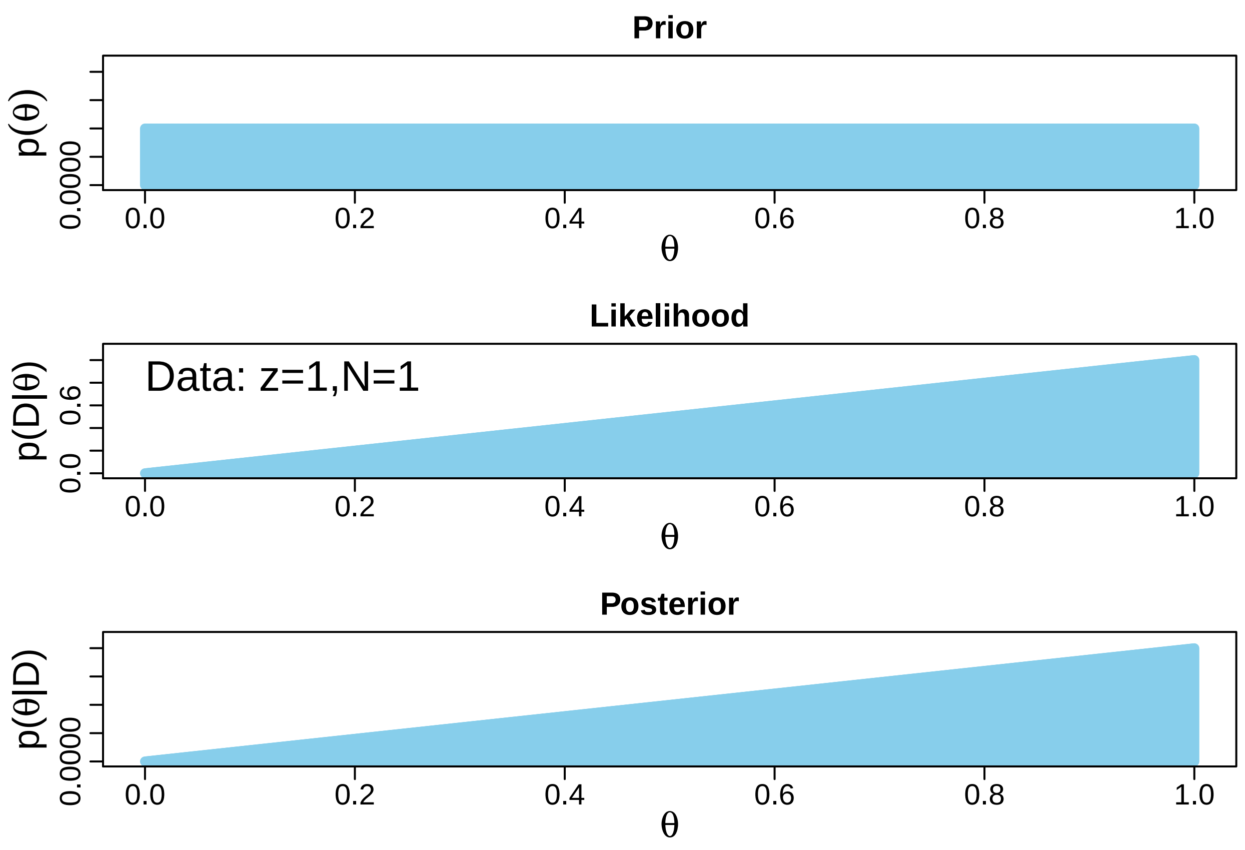
```
Theta = seq( 0 , 1 , length=11 )  # Sparse teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,0),rep(1,1))      # Single flip with 1 head

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="None" , showHDI=FALSE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample1",type="eps")
```

This plot shows the reallocation of credibility for a larger number of candidate parameter values than the first plot and a single flip of coin.

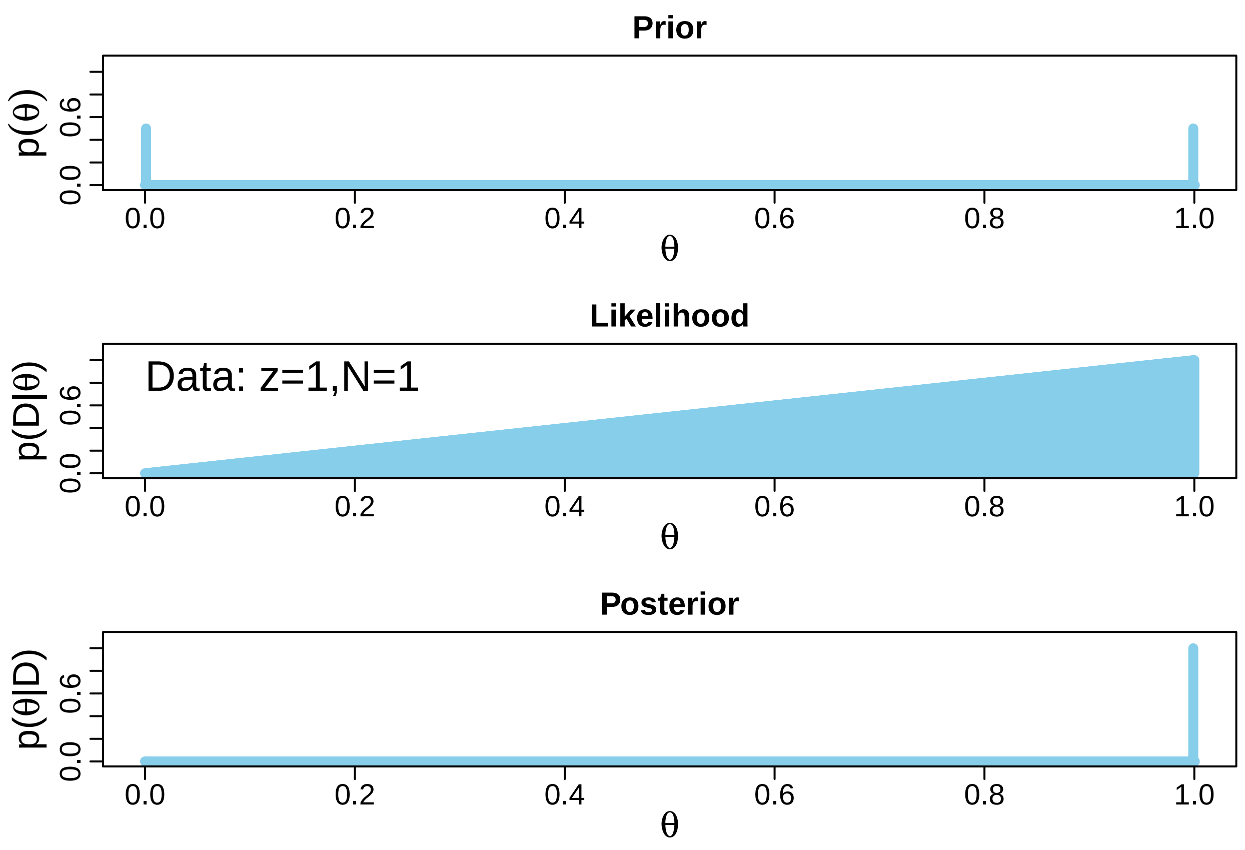
```
Theta = seq( 0 , 1 , length=1001 ) # Fine teeth for Theta.
pTheta = rep(1,length(Theta))      # Uniform (horizontal) shape for pTheta.
pTheta = pTheta/sum(pTheta)        # Make pTheta sum to 1.0
Data = c(rep(0,0),rep(1,1))        # Single flip with 1 head

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="None" , showHDI=FALSE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample2",type="eps")
```

It shows that when the prior is uniform, Posterior looks like likelihood.

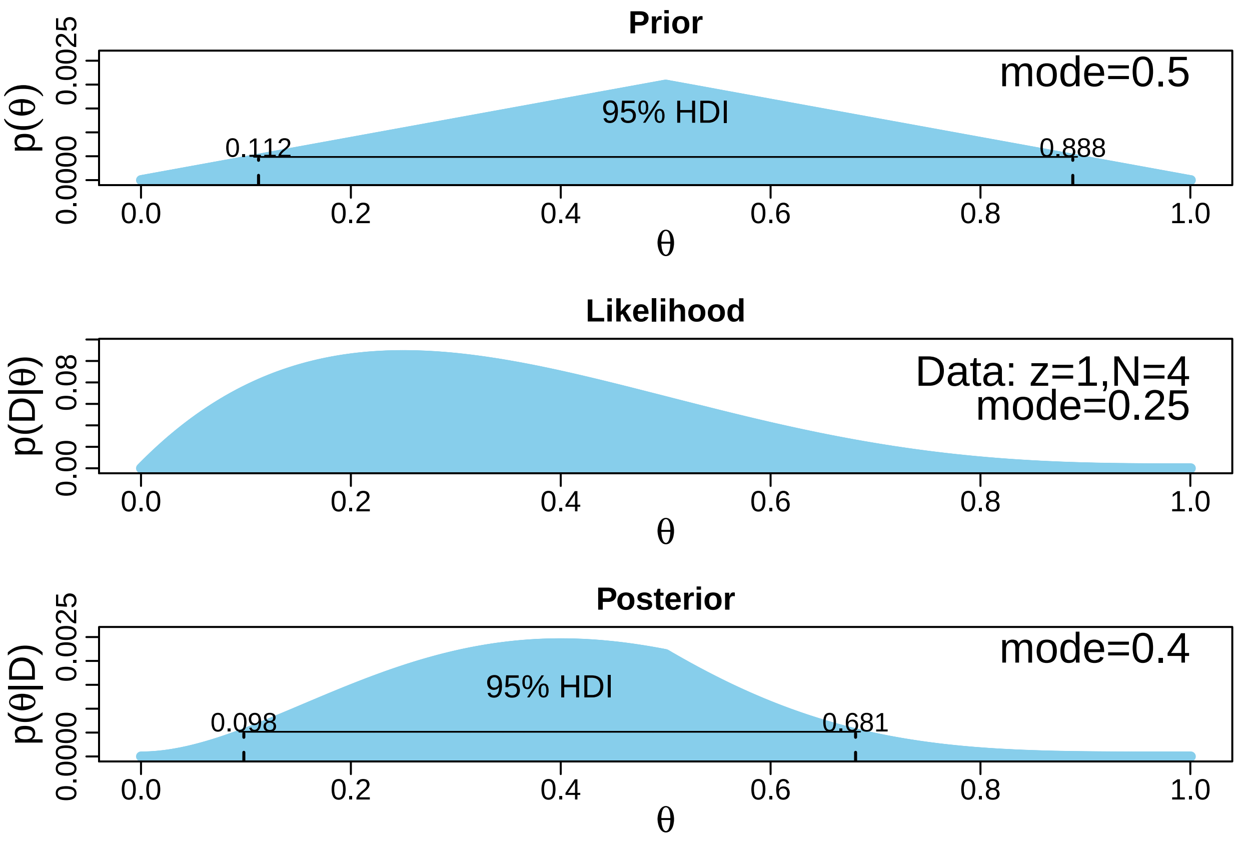
```
Theta = seq( 0 , 1 , length=1001 ) # Fine teeth for Theta.
pTheta = rep(0,length(Theta))      # Only extremes are possible!
pTheta[2] = 1                      # Only extremes are possible!
pTheta[length(pTheta)-1] = 1
pTheta = pTheta/sum(pTheta)        # Make pTheta sum to 1.0
Data = c(rep(0,0),rep(1,1))        # Single flip with 1 head

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="None" , showHDI=FALSE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample3",type="eps")
```

Prior allows only two values of parameter theta (either 0 or 1). A single flip of coin shifts the credibility value to theta=1.

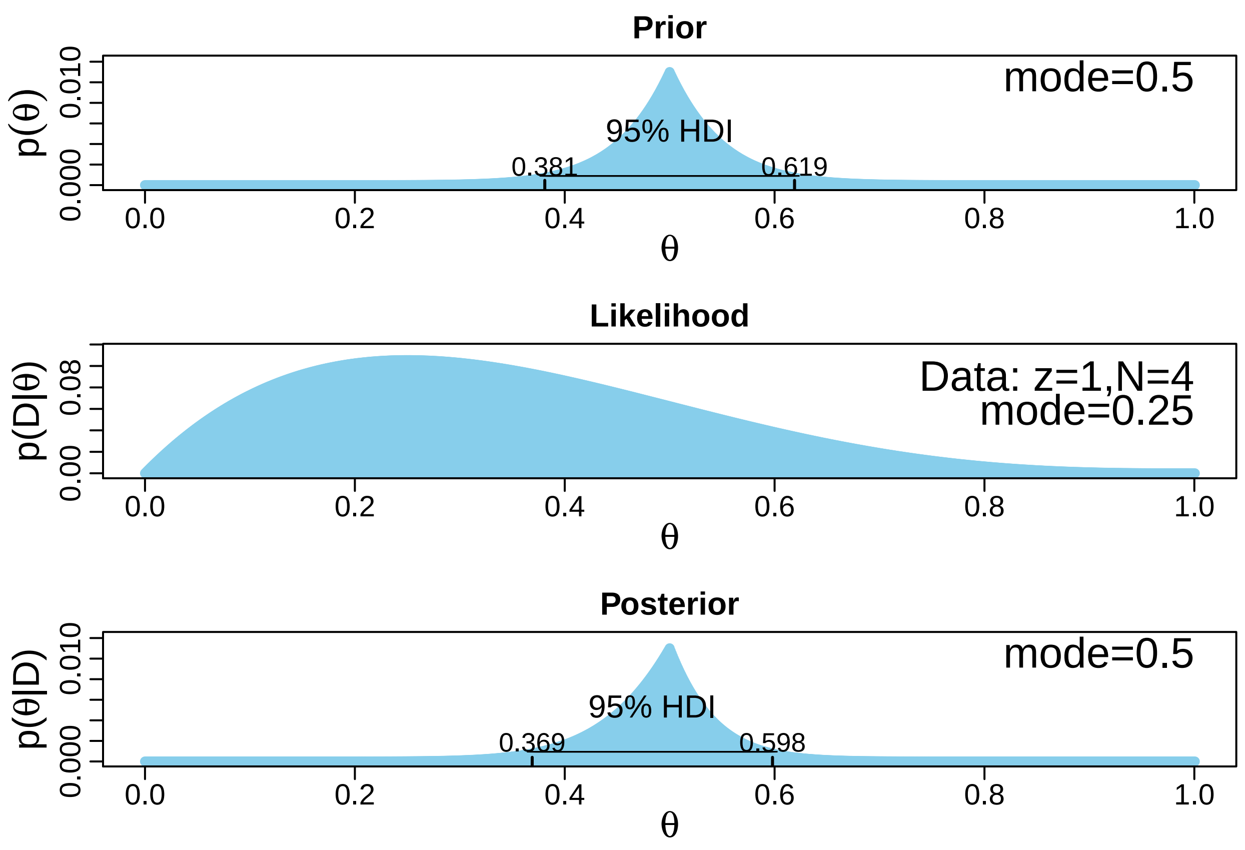
```
Theta = seq( 0 , 1 , length=1001 )  # Fine teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,3),rep(1,1))      # 25% heads, N=4

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="Mode" , showHDI=TRUE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample4",type="eps")
```

This plot shows that the credibilities don't shift much when data is small and the prior has significant influence on the posterior.

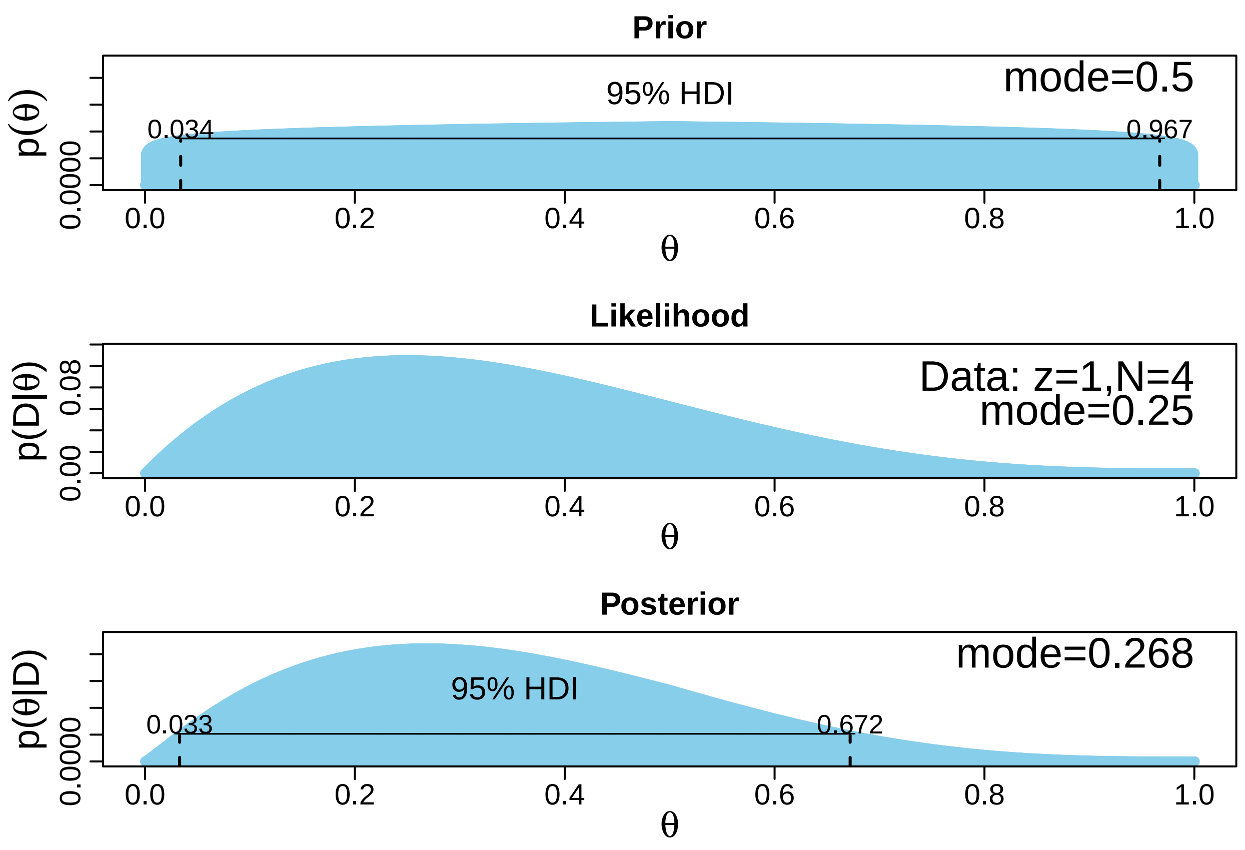
```
Theta = seq( 0 , 1 , length=1001 )  # Fine teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
pTheta = pTheta^10               # Sharpen pTheta !
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,3),rep(1,1))      # 25% heads, N=4

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="Mode" , showHDI=TRUE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample5",type="eps")
```

Strong prior is dominating the posterior here.

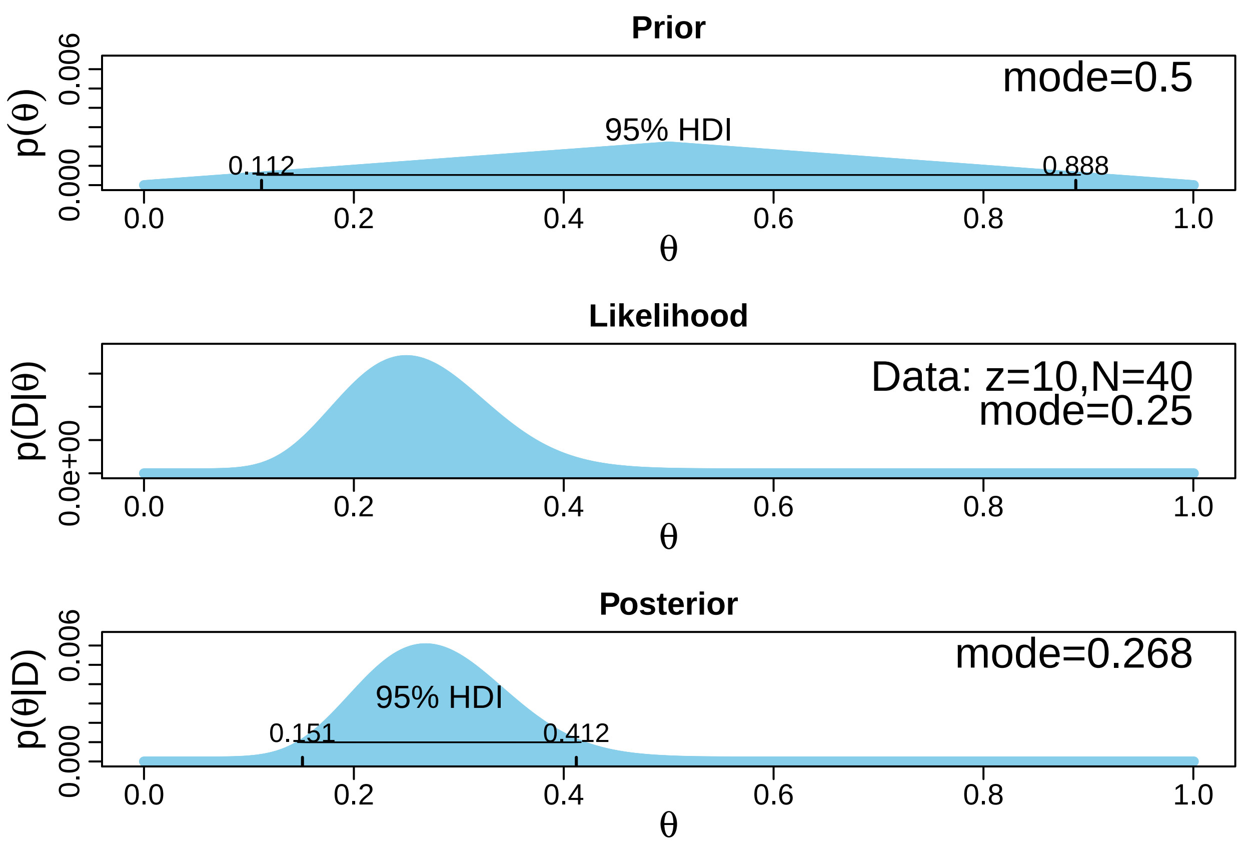
```
Theta = seq( 0 , 1 , length=1001 )  # Fine teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
pTheta = pTheta^0.1              # Flatten pTheta !
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,3),rep(1,1))      # 25% heads, N=4

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="Mode" , showHDI=TRUE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample6",type="eps")
#------------------------------------------------------------------------------
```

The data is dominating the posterior in case of small data set with vague prior.

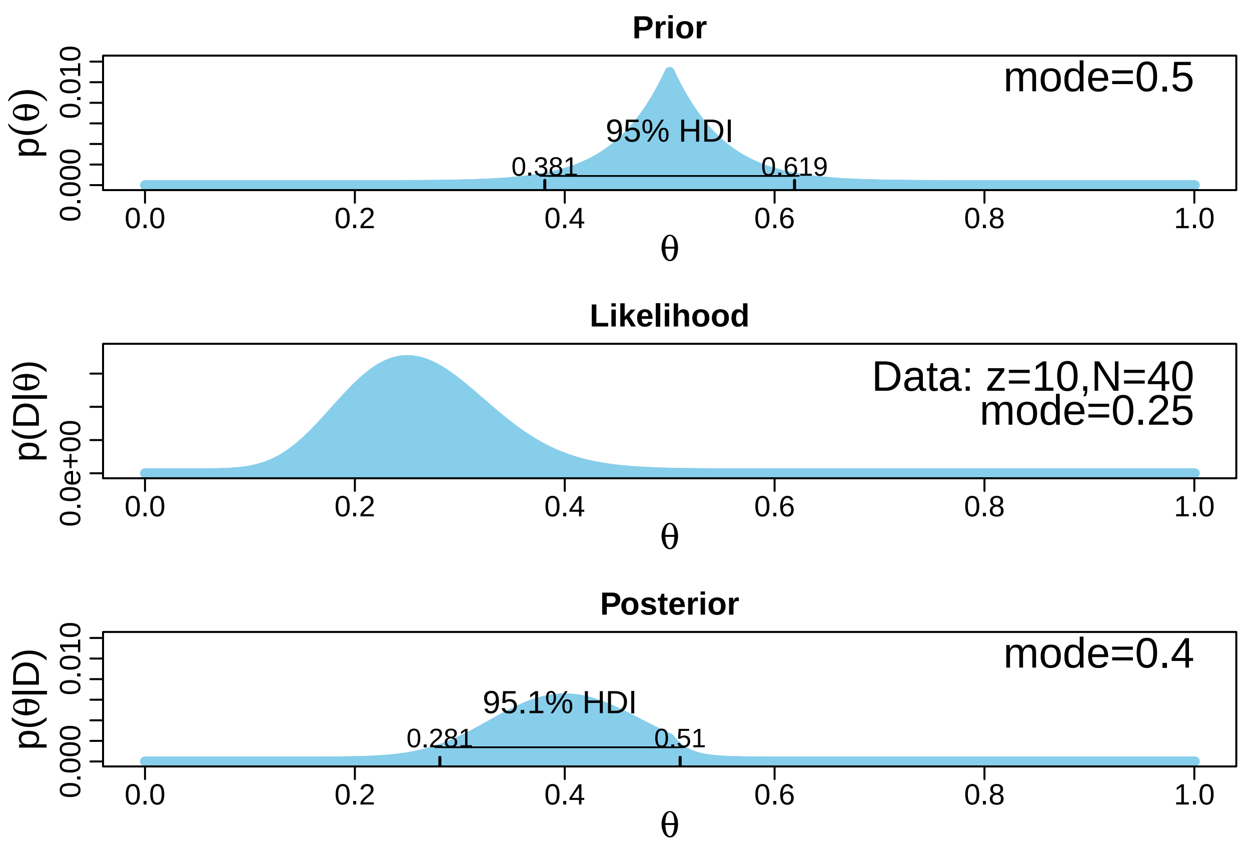
```
Theta = seq( 0 , 1 , length=1001 )  # Fine teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,30),rep(1,10))    # 25% heads, N=40

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="Mode" , showHDI=TRUE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample7",type="eps")
```

The modest amount of data with somewhat informed prior is enough to mostly dominate the prior.

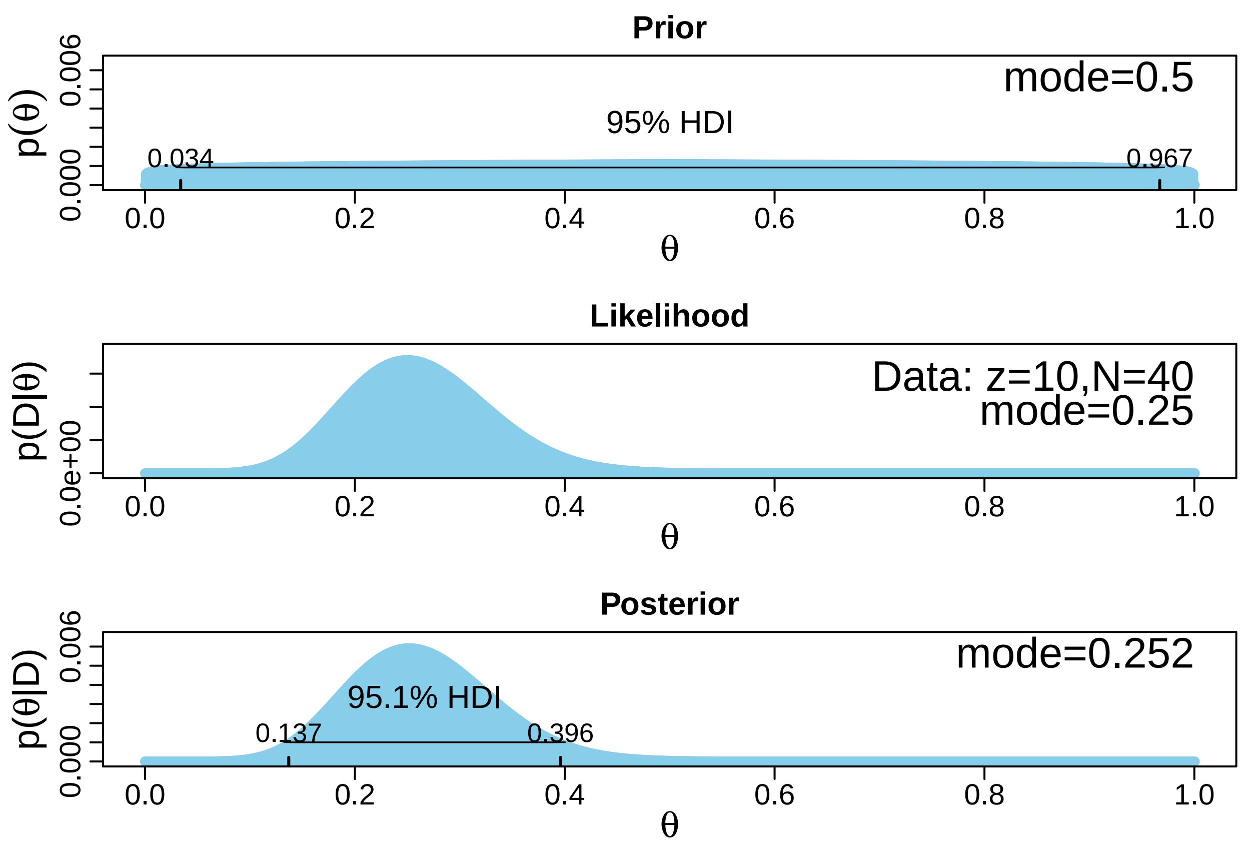
```
#------------------------------------------------------------------------------

Theta = seq( 0 , 1 , length=1001 )  # Fine teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
pTheta = pTheta^10               # Sharpen pTheta !
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,30),rep(1,10))    # 25% heads, N=40

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="Mode" , showHDI=TRUE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample8",type="eps")
#------------------------------------------------------------------------------
#---------------------------------------------------
```

Even the modest amount of data is not enough to overwhelm a strong prior.

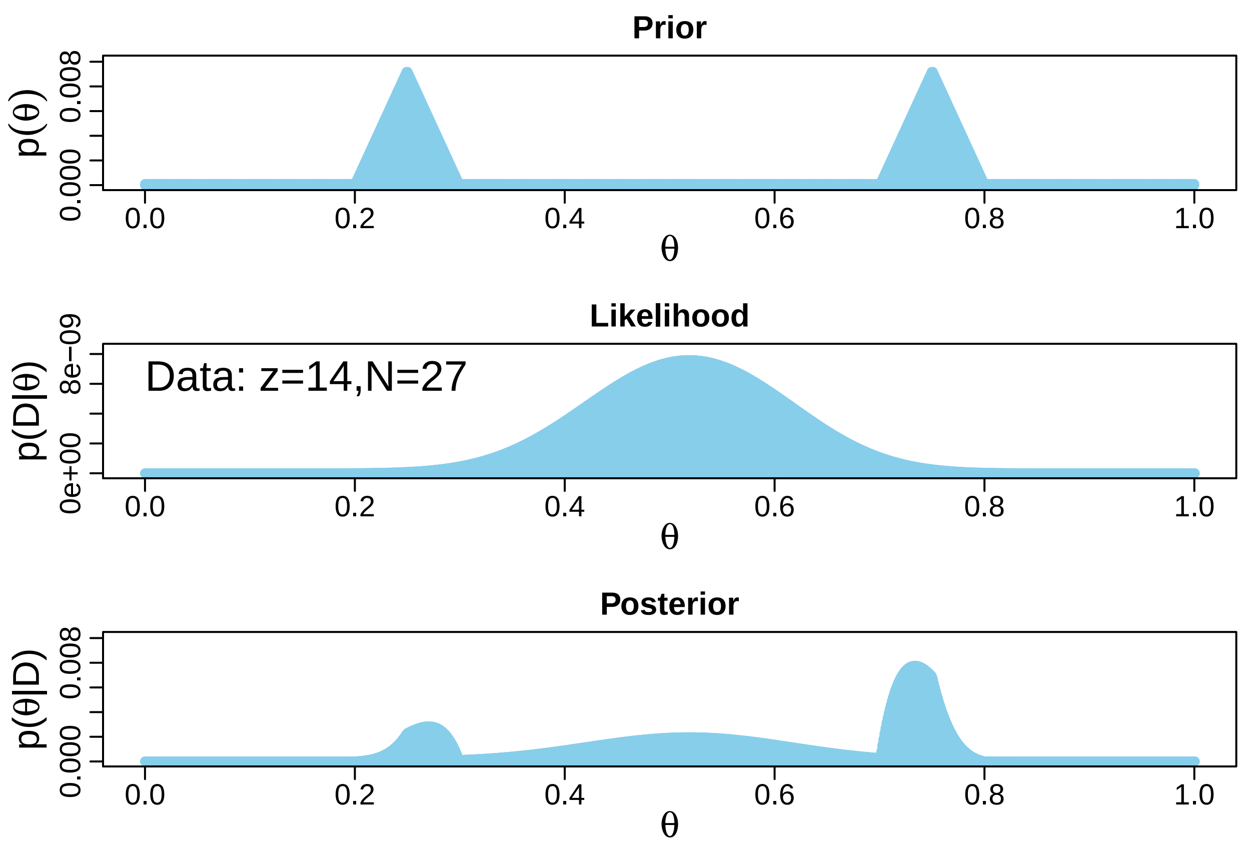
```
Theta = seq( 0 , 1 , length=1001 )  # Fine teeth for Theta.
pTheta = pmin( Theta , 1-Theta ) # Triangular shape for pTheta.
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
pTheta = pTheta^0.1              # Flatten pTheta !
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,30),rep(1,10))    # 25% heads, N=40

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="Mode" , showHDI=TRUE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample9",type="eps")
#------------------------------------------------------------------------------
```

The posterior is only slightly affected by vague prior.

```
Theta = seq( 0 , 1 , length=1000 )  # Fine teeth for Theta.
# Two triangular peaks on a small non-zero floor:
pTheta = c( rep(1,200),seq(1,100,length=50),seq(100,1,length=50),rep(1,200) ,
            rep(1,200),seq(1,100,length=50),seq(100,1,length=50),rep(1,200) )
pTheta = pTheta/sum(pTheta)      # Make pTheta sum to 1.0
Data = c(rep(0,13),rep(1,14))

openGraph(width=5,height=7)
posterior = BernGrid( Theta, pTheta , Data , plotType="Bars" ,
                      showCentTend="None" , showHDI=FALSE , showpD=FALSE )
```



```
saveGraph(file="BernGridExample10",type="eps")
```

The posterior is a compromise between prior and likelihood because of modest amount of data. (z=14, N=27)