Exercise 10 - Bayesian Non-parametrics and Bayesian Bootstrapping

Stefan Merdian

2025-01-22

Install

```
# packageurl <- "https://cran.r-project.org/src/contrib/Archive//DPpackage_1.1-7.tar.gz"
#
# install.packages(packageurl, repos=NULL, type="source")
```

We aim to replicate the first example from "https://pmc.ncbi.nlm.nih.gov/articles/PMC3142948/#S22", with slight adjustments to some values.

true conditional densities

```
dtrue <- function(grid, x) {
  exp(-2 * x) * dnorm(grid, mean = x, sd = sqrt(0.01)) +
    (1 - exp(-2 * x)) * dnorm(grid, mean = x^4, sd = sqrt(0.04))
}
```

simulation of the data

```
  set.seed(0)
  nrec <- 500
  x <- runif(nrec)
  y1 <- x + rnorm(nrec, 0, sqrt(0.01))
  y2 <- x^4 + rnorm(nrec, 0, sqrt(0.04))
  u <- runif(nrec)
  prob <- exp(-2*x)
  y <- ifelse(u<prob,y1,y2)
```

prior information

```
  w <- cbind(y,x)
  wbar <- apply(w,2,mean)
  wcov <- var(w)

  prior <- list(a0=10,
                b0=1,
                nu1=4,
                nu2=4,
```

```
                s2=0.5*wcov,
                m2=wbar,
                psiinv2=2*solve(wcov),
                tau1=6.01,
                tau2=3.01)
```

mcmc specification

```
mcmc <- list(nburn=5000,
                nsave=5000,
                nskip=3,
                ndisplay=100)
```

covariate values where the density

and mean function is evaluated

We have to adjust the values from the example with the values given in our task.

```
xpred <- c(0.1, 0.25, 0.5, 0.75)
```

fitting the model

It is necessary to include compute.band = TRUE and specify the type as "HDP" to obtain the 95% HPD interval.

```
library("DPpackage")
```

```
##
## DPpackage 1.1-7
##
## Copyright (C) 2006 - 2012, Alejandro Jara
## Department of Statistics
## P.U. Catolica de Chile
##
## Support provided by Fondecyt
## 11100144 grant.
##
```

```
fitWDDP <- DPcdensity(
  y = y,
  x = x,
  xpred = xpred,
  ngrid = 100,
  prior = prior,
  mcmc = mcmc,
  state = NULL,
```

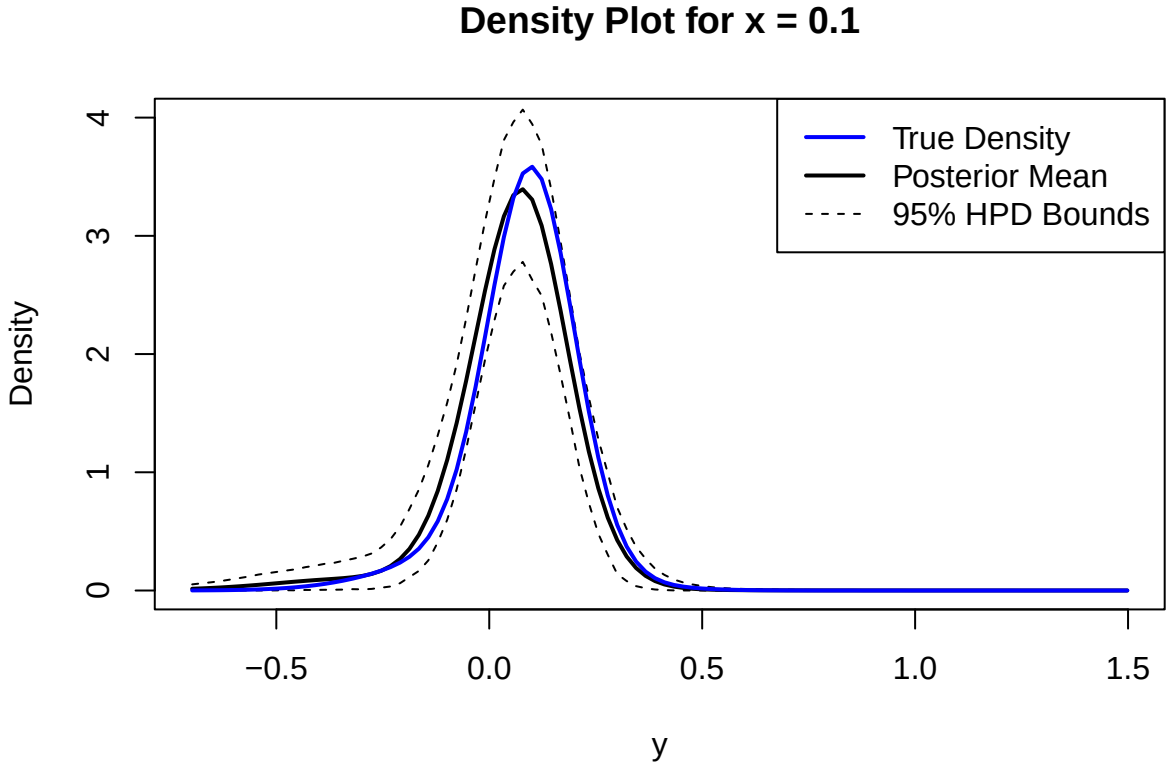
```
  status = TRUE,
  compute.band = TRUE,
  type.band = "HPD"
)
```

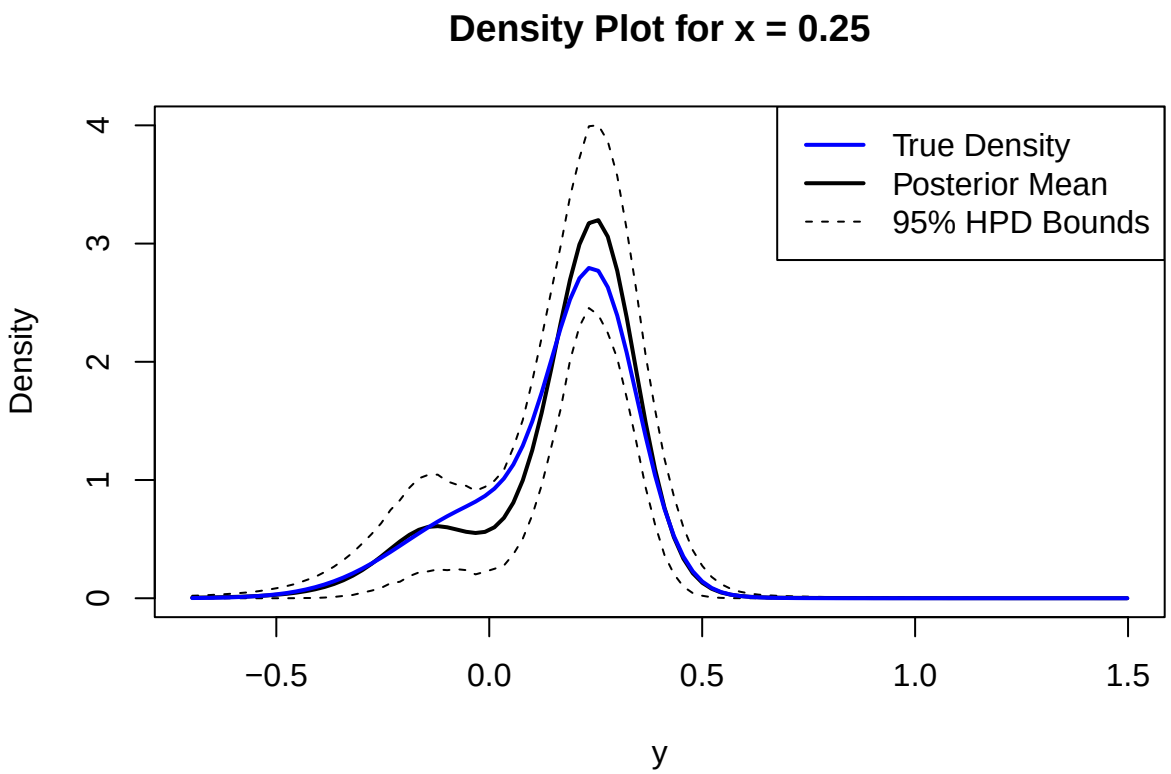
```
##
## MCMC scan 100 of 5000 (CPU time: 9.802 s)
## MCMC scan 200 of 5000 (CPU time: 12.698 s)
## MCMC scan 300 of 5000 (CPU time: 15.671 s)
## MCMC scan 400 of 5000 (CPU time: 18.645 s)
## MCMC scan 500 of 5000 (CPU time: 21.570 s)
## MCMC scan 600 of 5000 (CPU time: 24.459 s)
## MCMC scan 700 of 5000 (CPU time: 27.405 s)
## MCMC scan 800 of 5000 (CPU time: 30.264 s)
## MCMC scan 900 of 5000 (CPU time: 33.459 s)
## MCMC scan 1000 of 5000 (CPU time: 36.532 s)
## MCMC scan 1100 of 5000 (CPU time: 39.440 s)
## MCMC scan 1200 of 5000 (CPU time: 42.404 s)
## MCMC scan 1300 of 5000 (CPU time: 45.271 s)
## MCMC scan 1400 of 5000 (CPU time: 48.187 s)
## MCMC scan 1500 of 5000 (CPU time: 51.207 s)
## MCMC scan 1600 of 5000 (CPU time: 54.081 s)
## MCMC scan 1700 of 5000 (CPU time: 56.984 s)
## MCMC scan 1800 of 5000 (CPU time: 59.841 s)
## MCMC scan 1900 of 5000 (CPU time: 62.766 s)
## MCMC scan 2000 of 5000 (CPU time: 65.555 s)
## MCMC scan 2100 of 5000 (CPU time: 68.466 s)
## MCMC scan 2200 of 5000 (CPU time: 71.437 s)
## MCMC scan 2300 of 5000 (CPU time: 74.363 s)
## MCMC scan 2400 of 5000 (CPU time: 77.325 s)
## MCMC scan 2500 of 5000 (CPU time: 80.144 s)
## MCMC scan 2600 of 5000 (CPU time: 83.031 s)
## MCMC scan 2700 of 5000 (CPU time: 86.079 s)
## MCMC scan 2800 of 5000 (CPU time: 88.969 s)
## MCMC scan 2900 of 5000 (CPU time: 92.053 s)
## MCMC scan 3000 of 5000 (CPU time: 95.007 s)
## MCMC scan 3100 of 5000 (CPU time: 97.944 s)
## MCMC scan 3200 of 5000 (CPU time: 100.871 s)
## MCMC scan 3300 of 5000 (CPU time: 103.771 s)
## MCMC scan 3400 of 5000 (CPU time: 106.694 s)
## MCMC scan 3500 of 5000 (CPU time: 109.635 s)
## MCMC scan 3600 of 5000 (CPU time: 112.558 s)
## MCMC scan 3700 of 5000 (CPU time: 115.544 s)
## MCMC scan 3800 of 5000 (CPU time: 118.530 s)
## MCMC scan 3900 of 5000 (CPU time: 121.478 s)
## MCMC scan 4000 of 5000 (CPU time: 124.396 s)
## MCMC scan 4100 of 5000 (CPU time: 127.338 s)
## MCMC scan 4200 of 5000 (CPU time: 130.342 s)
## MCMC scan 4300 of 5000 (CPU time: 133.300 s)
## MCMC scan 4400 of 5000 (CPU time: 136.222 s)
## MCMC scan 4500 of 5000 (CPU time: 139.116 s)
## MCMC scan 4600 of 5000 (CPU time: 142.066 s)
## MCMC scan 4700 of 5000 (CPU time: 145.060 s)
## MCMC scan 4800 of 5000 (CPU time: 147.915 s)
```

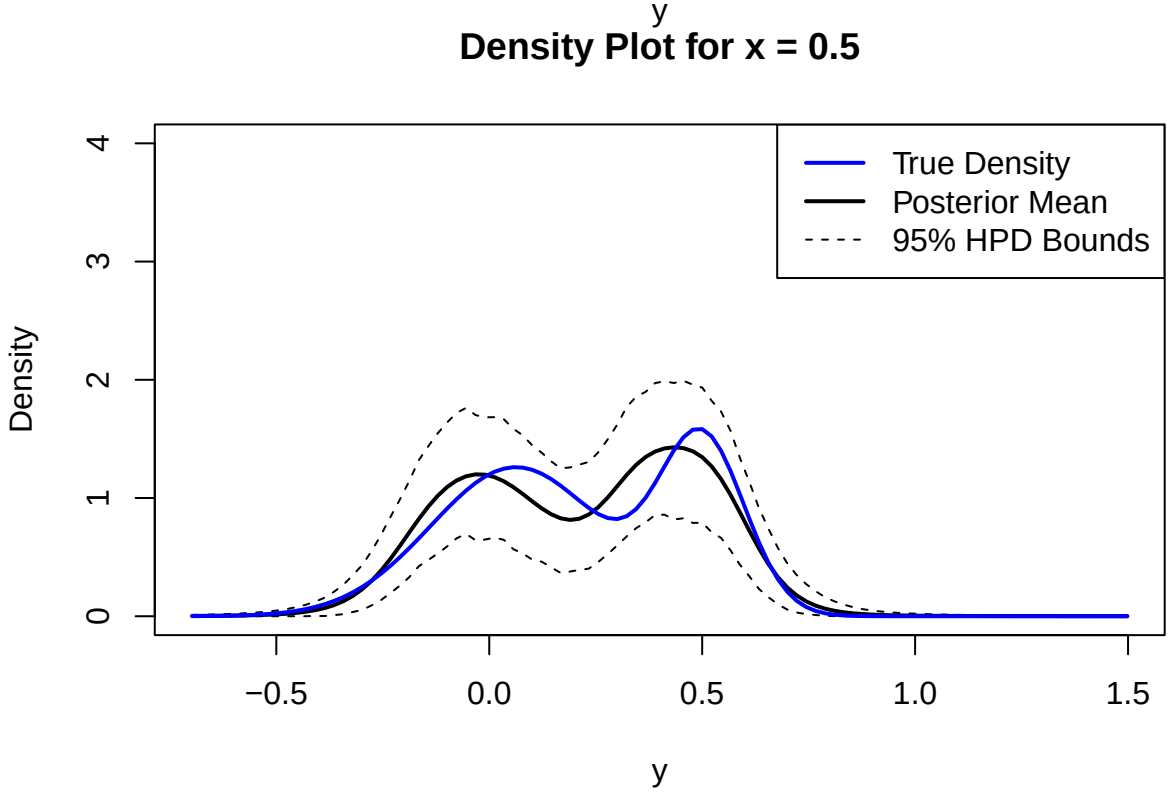
```
## MCMC scan 4900 of 5000 (CPU time: 150.910 s)
## MCMC scan 5000 of 5000 (CPU time: 153.984 s)
```

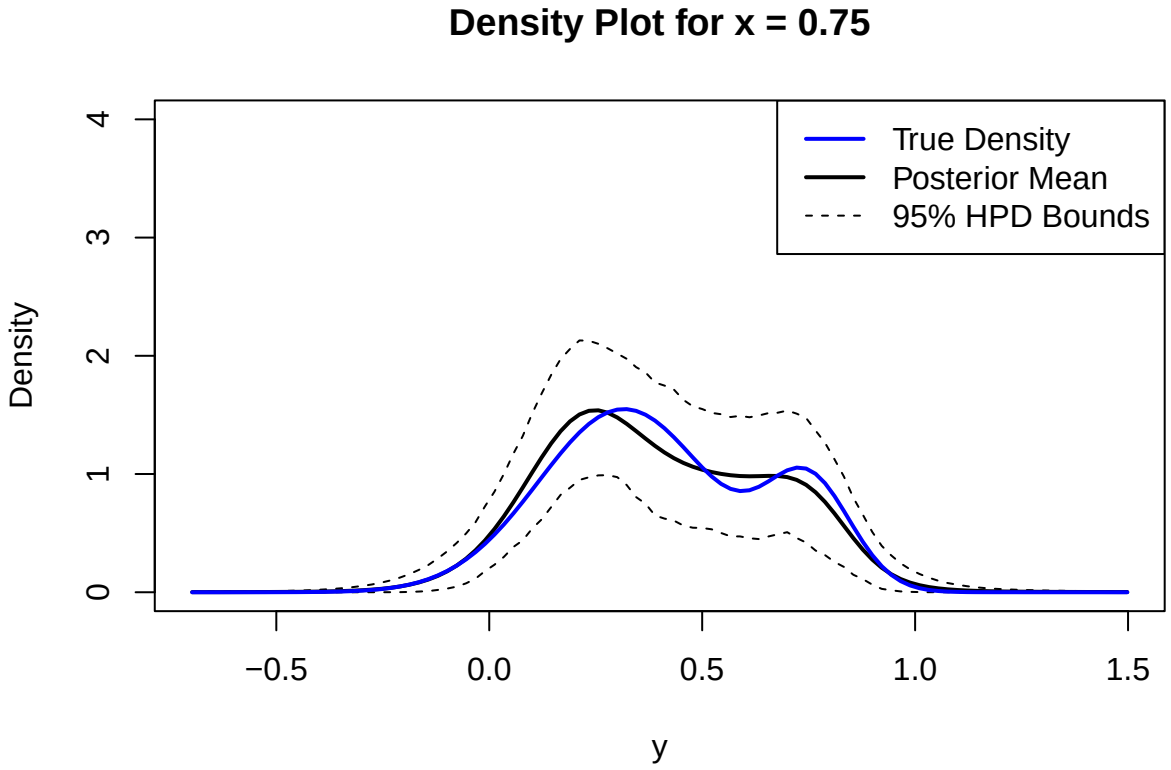
```r
library(ggplot2)

visualize_density <- function(model_output, x_val, idx) {
  par(mfrow = c(1, 1))

  #  95% HPD interval
  plot(model_output$grid, model_output$densp.h[idx, ], type = "l", lwd = 1, lty = 2,
       main = paste("Density Plot for x =", x_val),
       xlab = "y", ylab = "Density", ylim = c(0, 4))

  # lower 95% HPD interval
  lines(model_output$grid, model_output$densp.l[idx, ], lwd = 1, lty = 2)

  # posterior mean density estimate
  lines(model_output$grid, model_output$densp.m[idx, ], lwd = 2, lty = 1)

  # true conditional density
  lines(model_output$grid, dtrue(model_output$grid, x_val), col = "blue", lwd = 2, lty = 1)

  legend("topright",
         legend = c("True Density", "Posterior Mean", "95% HPD Bounds"),
         col = c("blue", "black", "black"), lty = c(1, 1, 2), lwd = c(2, 2, 1))
}
```

```r
for (i in seq_along(xpred)) {
  visualize_density(fitWDDP, xpred[i], i)
}
```




Density Plot for x = 0.25
True Density
Posterior Mean
95% HPD Bounds
Density
0
1
2
3
4
−0.5
0.0
0.5
1.0
1.5
y


Density Plot for x = 0.5
True Density
Posterior Mean
95% HPD Bounds
Density
0
1
2
3
4
−0.5
0.0
0.5
1.0
1.5
y


Density Plot for x = 0.75
True Density
Posterior Mean
95% HPD Bounds
Density
0
1
2
3
4
−0.5
0.0
0.5
1.0
1.5
y