mcmc-ladder

Hussein Al-Asadi

October 4, 2016

MCMC for univariate mixture of normals

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(mvtnorm)
library(plotly)
```

```
## Loading required package: ggplot2
```

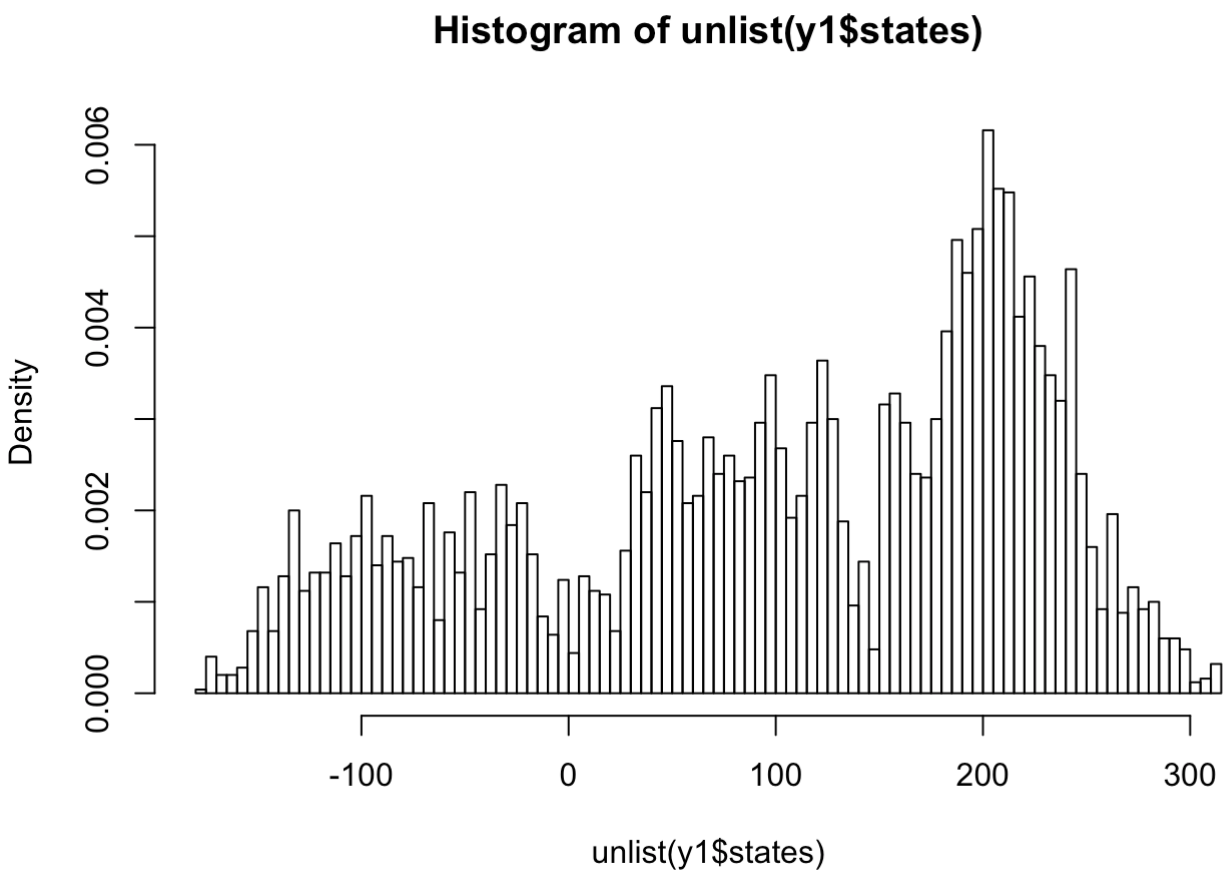
```
##
## Attaching package: 'plotly'
```

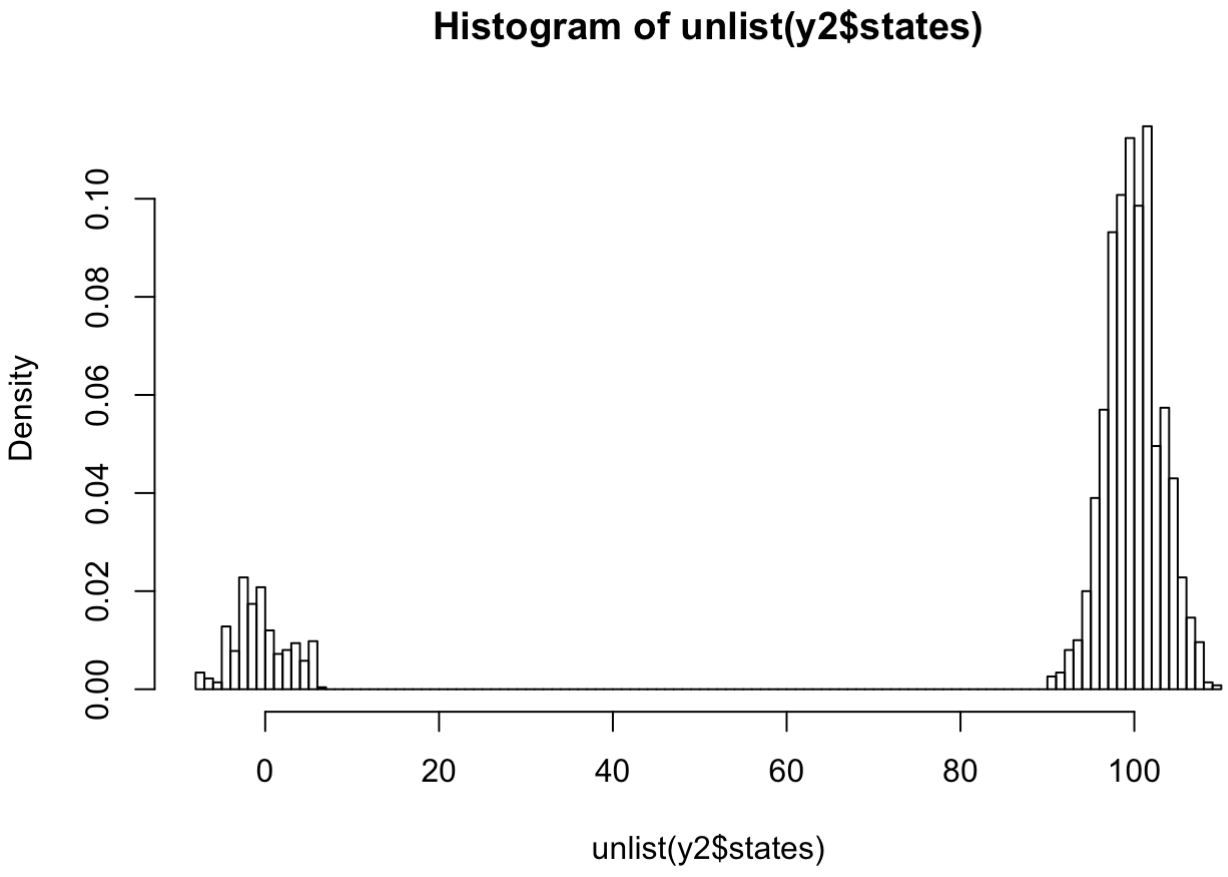
```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

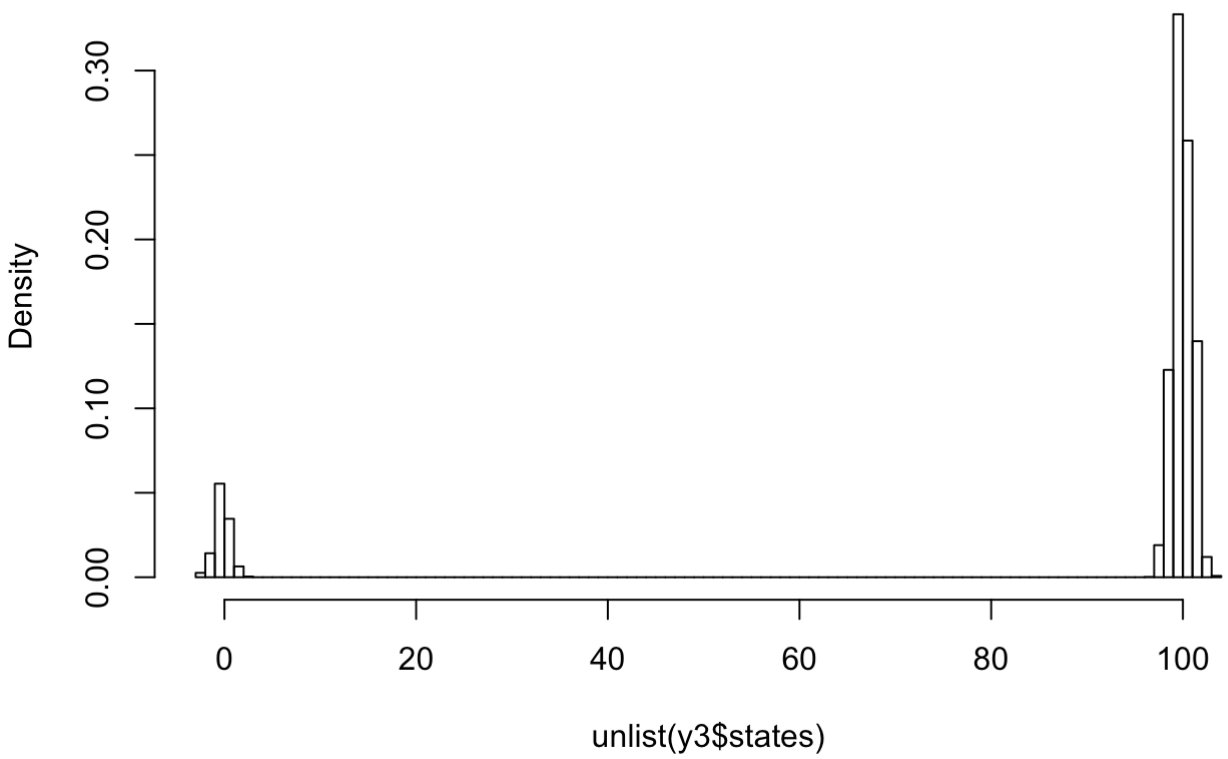
```
## The following object is masked from 'package:stats':
##
##     filter
```

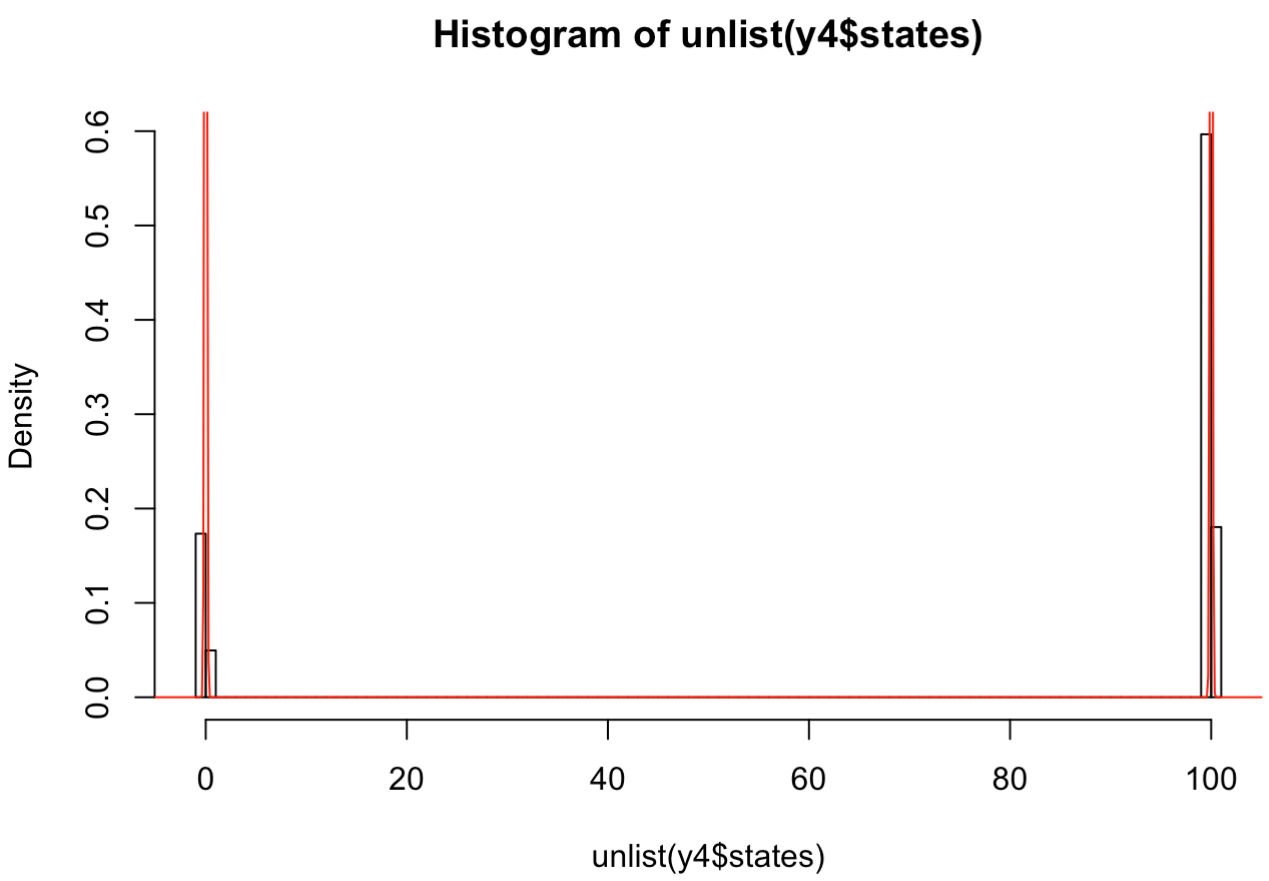
```
## The following object is masked from 'package:graphics':
##
##     layout
```

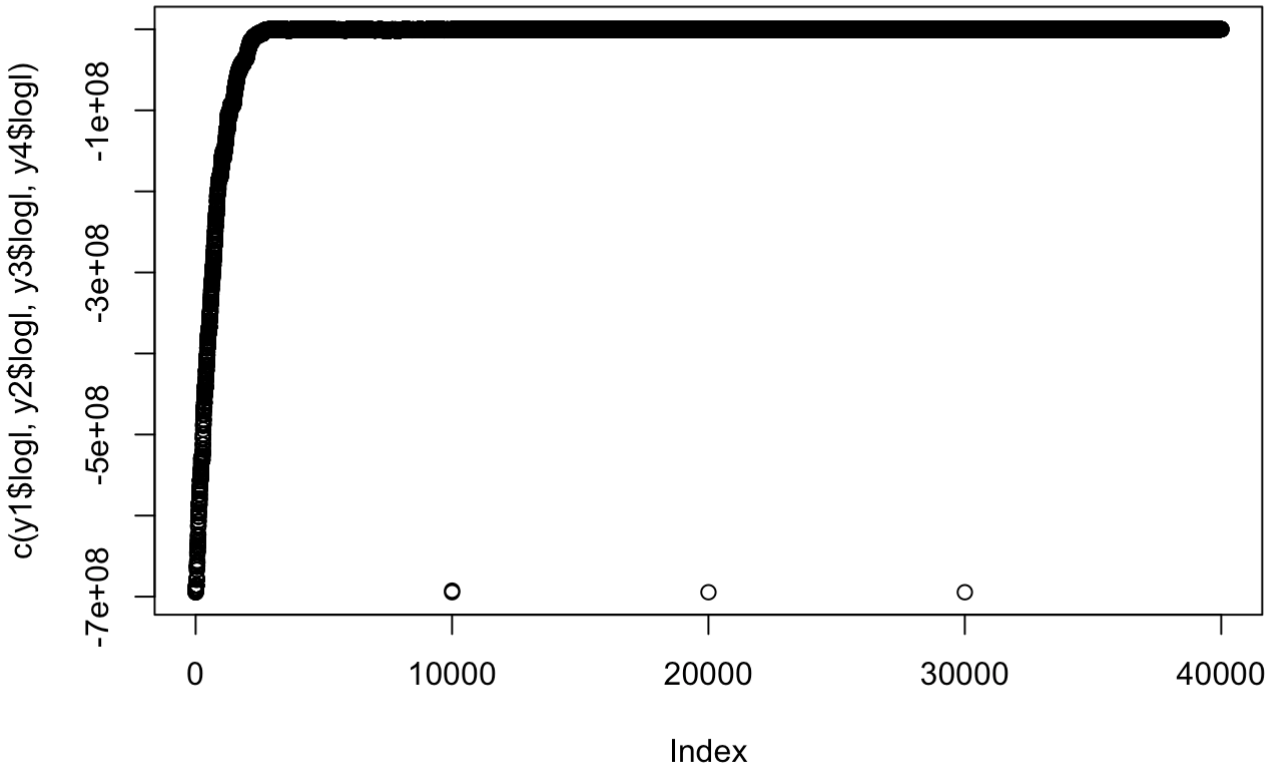
```
# find the log of the mean of x, from log(x)
# i.e. log(mean(exp(lx)))
lmean=function(lx){
  m = max(lx)
  m + log(mean(exp(lx-m)))
}

# this is the log(mixture density)  ie the log of the target we are going to sample fro
m, 50/50
# i.e. x ~ 0.5 * N + 0.5 * N
lmixnorm = function(x){
  x1 = dnorm(x,0,.1,log=TRUE)
  x2 = dnorm(x,100,.1,log=TRUE)
  # compute the log-likelihood of a 50/50 mixture of normals
  return(lmean(c(x1,x2)))
}

run_mcmc = function(burnin, niters, temp, prev.chain, s0, pi, sym_proposal_dis){
  states <- list()
  logl <- rep(NA, niters)
  states[[1]] <- s0
  logl[1] <- pi(states[[1]])

  for (i in 2:niters){
    u <- runif(1)

    # make long range proposal
    if (u < 0.5){
      loga = -Inf
      if (length(prev.chain$states) > 0){
        new.state <- sample(prev.chain$states, 1)
        new.state <- new.state[[1]]
        loga <- ((1/temp) - (1/prev.chain$temp)) * (pi(new.state) -pi(states[[i-1]]))
      }

    # make local proposal
    }else{
      new.state <- states[[i-1]] + sym_proposal_dis()
      loga = (pi(new.state) - pi(states[[i-1]]))/temp
    }

    # accept or reject step
    u = runif(1)
    if (log(u) < min(0, loga)){
      states[[i]] <- new.state
    } else{
      states[[i]] <- states[[i-1]]
    }
    logl[i] <- pi(states[[i]])
  }

  return(list(states=states[burnin:niters],temp=temp, logl = logl))
}
```


Histogram of unlist(y1$states)
Density
0.000
0.002
0.004
0.006
-100
0
100
200
300
unlist(y1$states)


Histogram of unlist(y2$states)
Density
0.00
0.02
0.04
0.06
0.08
0.10
0
20
40
60
80
100
unlist(y2$states)


Histogram of unlist(y3$states)
Density
0.00
0.10
0.20
0.30
0
20
40
60
80
100
unlist(y3$states)


Histogram of unlist(y4$states)
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
20
40
60
80
100
unlist(y4$states)

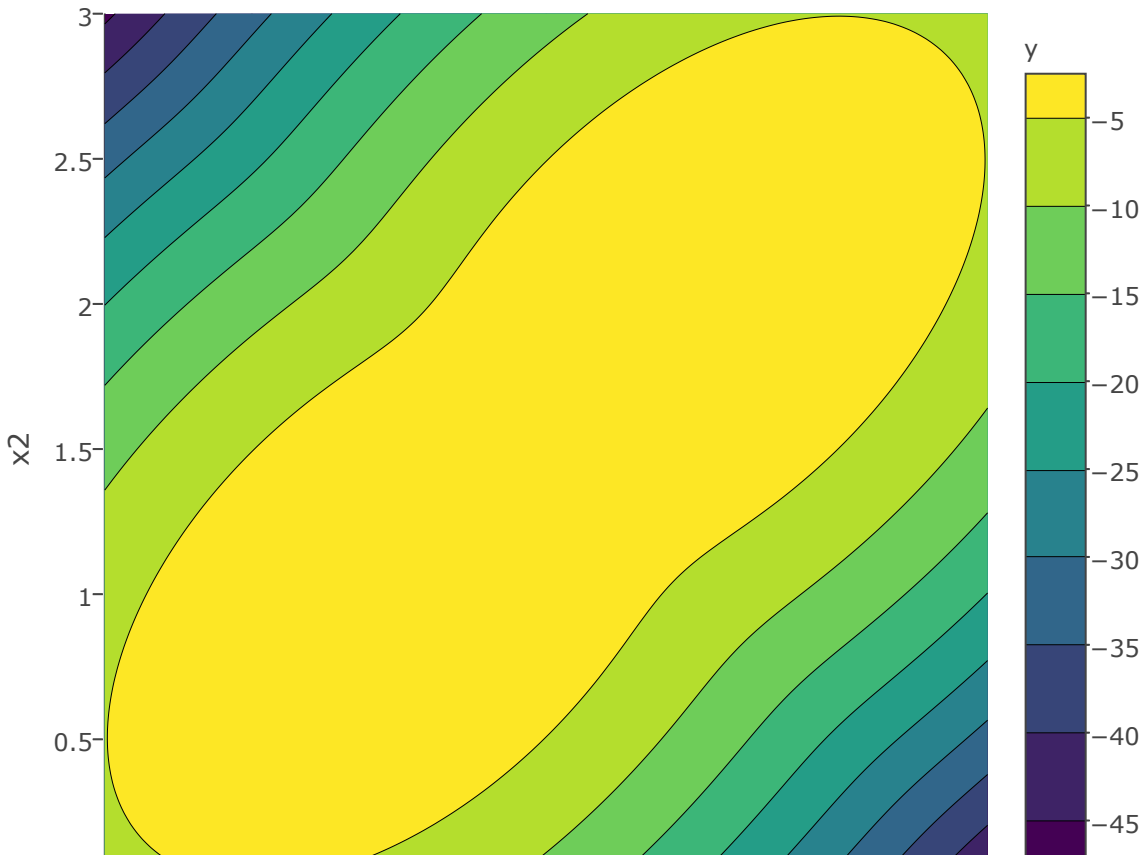

c(y1$logl, y2$logl, y3$logl, y4$logl)
-7e+08
-5e+08
-3e+08
-1e+08
0
10000
20000
30000
40000
Index

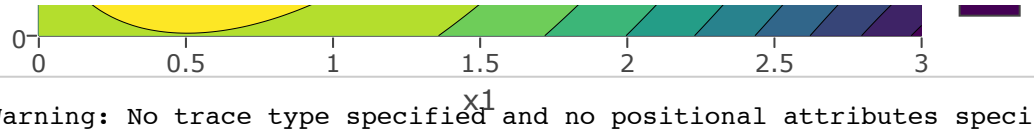
mixture of bivariate normals

```
# this is the log(mixture density)  ie the log of the target we are going to sample fro
m, 50/50
# i.e. x ~ 0.5 * N + 0.5 * N
lmixmvnorm = function(x){
  m1 <- c(1, 1)
  m2 <- c(2, 2)
  S1 <- matrix(c(0.1, 0.05, 0.05, 0.1), nrow = 2, ncol = 2)
  S2 <- S1
  x1 = dmvnorm(x,mean = m1, sigma = S1,log=TRUE)
  x2 = dmvnorm(x,mean = m2, sigma = S2,log=TRUE)
  # compute the log-likelihood of a 50/50 mixture of normals
  return(lmean(c(x1,x2)))
}
```

plot the mixture of bivariate normals

```
x1 = seq(0,3,length=100)
x2 = seq(0,3,length=100)
twod.grid <- expand.grid(x1 = x1, x2 = x2)
y = apply(twod.grid, 1, lmixmvnorm)
mtrx <- cbind(twod.grid, y)

plot_ly(mtrx, x = ~x1, y = ~x2, z = ~y, type = "contour") %>% layout(autosize = F, width
 = 600, height = 500)
```






```
## Warning: No trace type specified and no positional attributes specified
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#scatter
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```