Mixture Modeling with JAGS

```
## Loading required package: knitr
```

This page is about estimating a mixture models using JAGS via R.

Two Component Mixture of Normals

The first mixture model examined considers a distribution comprised of two normal distributions. First things first, we create the data, pooling two distinct normally-distributed variables – `x1`, with a mean of 3 and standard deviation of 2; and `x2`, with a mean of 10 and standard deviation of 2. The last puts the data into the list format that JAGS requires.

```
require(coda)
require(rjags)
require(reshape)
x1 <- rnorm(25, 3, 2)
x2 <- rnorm(75, 10, 2)
x <- c(x1,x2)
data <- list(x=x,n=length(x))
```

The JAGS code for a two-component mixture of normals is:

```
model{
  p ~ dbeta(1,1)
  mu1 ~ dnorm(10, .01)
  mu2 ~ dnorm(50, .01)
  tau <- pow(sigma, -2)
  sigma ~ dunif(0,100)
  for (i in 1:n)
      {
      z[i] ~ dbern(p)
      mu[i] <- z[i]*mu1 + (1-z[i])*mu2
      x[i] ~ dnorm(mu[i], tau)
      }
}
```

The parameters we are estimating are the means of the two component distributions, `mu1` and `mu2`. The model results also provide values `z[i]` for each observation's probability of belonging to one of the two component distributions.

We pass initial values to JAGS in order to initialize the MCMC simulation. Note that each `x[i]` gets its own mean, but all observations share the variance, which in this case is set to equal 1.

```
init <- list(mu=array(rnorm(1*data$n)),mu1=rnorm(1), mu2=rnorm(1),sigma=1)
```

We pass the model to JAGS using the `textConnection()` function. The text is entered into `R` as the object `modelstring`. I prefer to run the MCMC simulation using the `coda.samples()` function in the **coda** package. Another popular package for doing so is **R2Jags**.

Of particular interest is whether the model can recover the (known) means of the two component distributions (`mu1` = 3, and `mu2` = 10). Figure 1 plots the simulated values for these parameters. Despite using highly uninformative priors, the means are easily recovered.

```
model <- jags.model(textConnection(modelstring), data=data, n.adapt=1e5)
# Run samples using coda package sampler
out <- coda.samples(model=model, init=init, variable.names=c("mu1", "mu2", "z"), n.iter=1e5, thin=20)
# Extract elements from list: our estimates of the two means
mu1 <- unlist(out[,"mu1",])
mu2 <- unlist(out[,"mu2",])
```

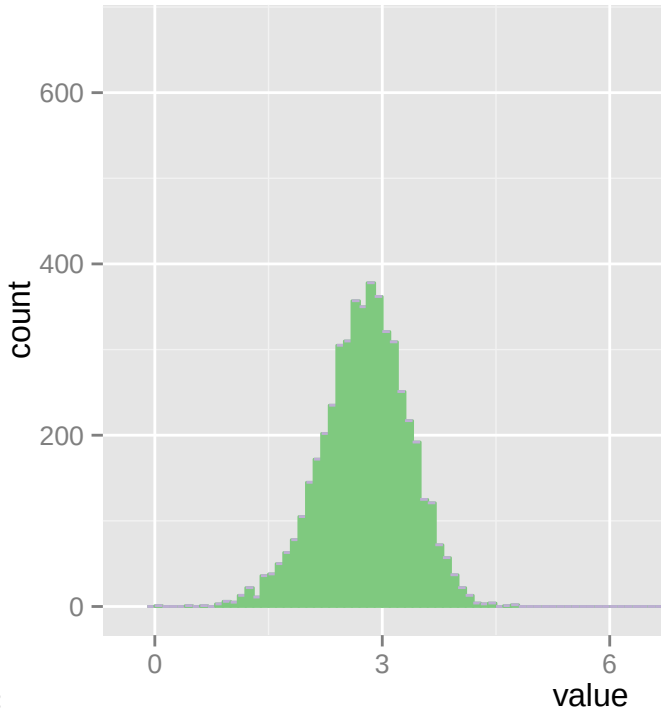
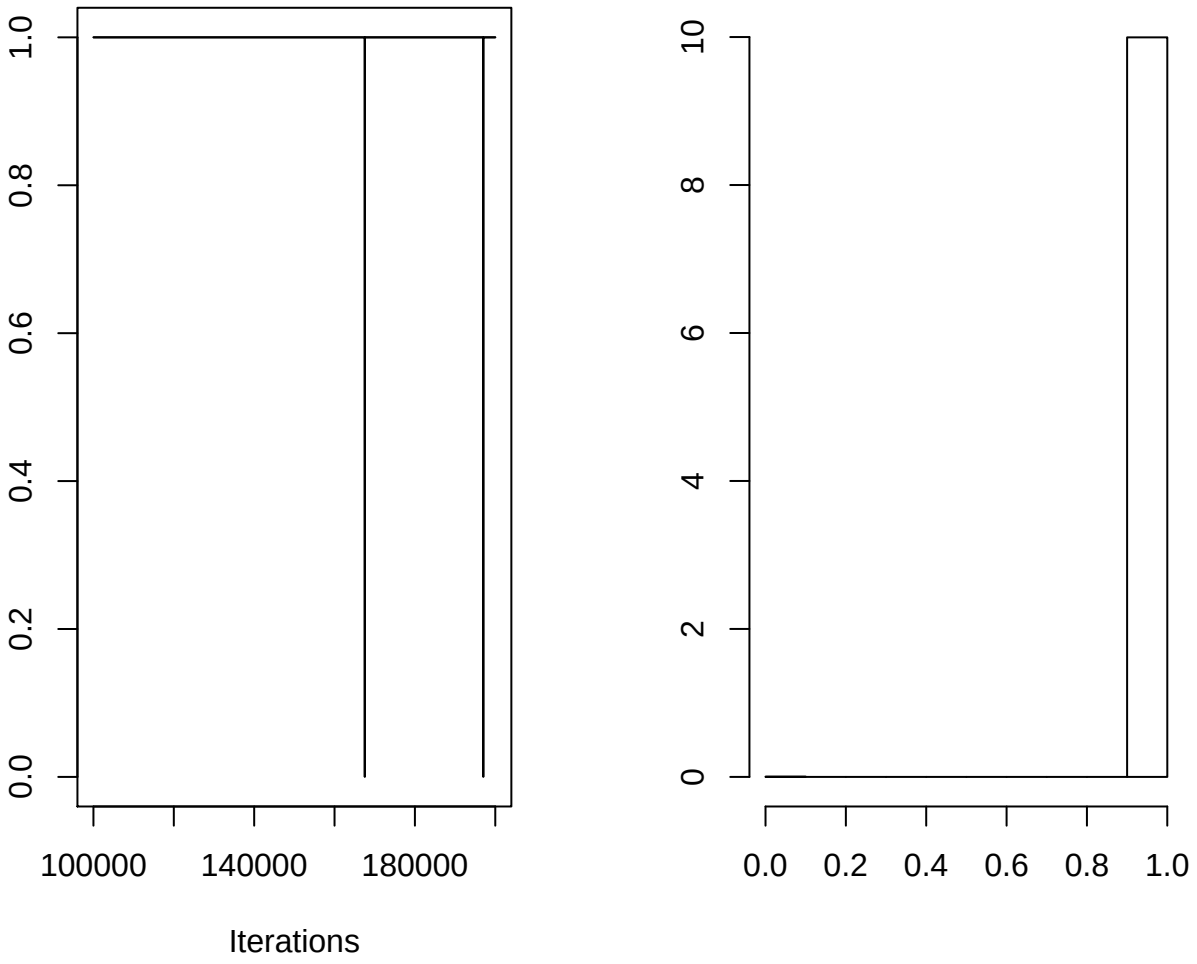


Figure 1. Density estimation for the simulated values of `mu1` and `mu2`

A second point of interest is related to the estimates of belonging to a particular group. In the model, the `z[i]` parameter is distributed Bernoulli. Thus, at each iteration, every observation is assigned a 0 or 1, indicating the observation's mean is either the sampled value for `mu1` or `mu2`. Figure 2 presents this assignment variable for the `z[1]` parameter at each iteration (left panel) along with the density estimate (right panel).

Figure 2. Traceplot and density estimate for the `z[1]` parameter

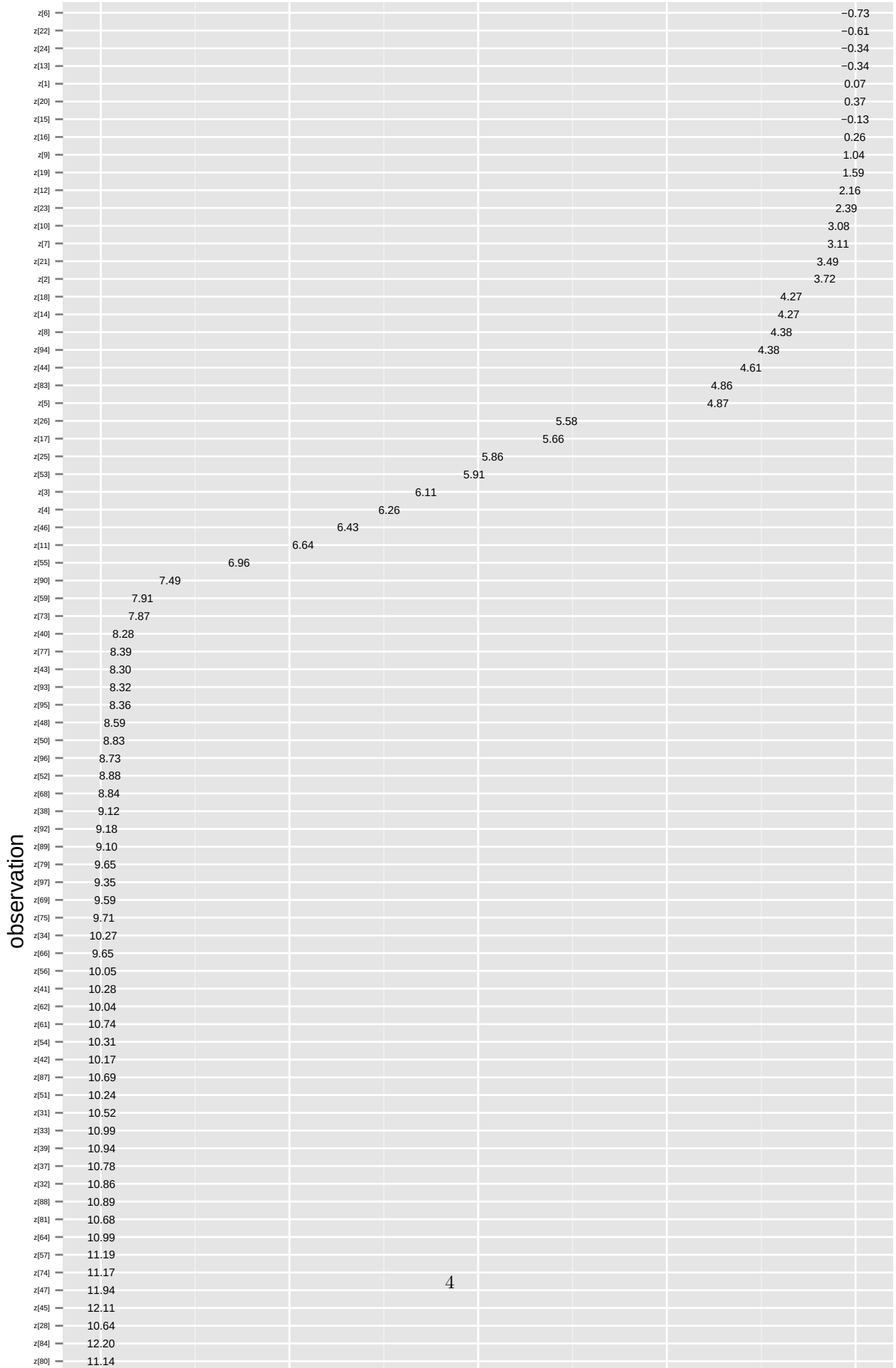
```
plot(out[,"z[1]",])
```



The average for `z[i]` across the entire simulation represents the probability that an observation belongs to one or the other components. Observations with a value that is close to one mean have sharper assignment probabilities – that is, an average value that is close to either 0 or 1.

Figure 2 plots these average probabilities, where the numeric labels are the `x[i]` values. Values that are close to the means are highly likely to have been drawn from one distribution or the other. The model has difficulty assigning probabilities to those few observations that fall at roughly the midpoint between 3 and 10.

Figure 2. Each observation's probability of belonging to the `mu1` distribution


observation
z[6] −0.73
z[22] −0.61
z[24] −0.34
z[13] −0.34
z[1] 0.07
z[20] 0.37
z[15] −0.13
z[16] 0.26
z[9] 1.04
z[19] 1.59
z[12] 2.16
z[23] 2.39
z[10] 3.08
z[7] 3.11
z[21] 3.49
z[2] 3.72
z[18] 4.27
z[14] 4.27
z[8] 4.38
z[94] 4.38
z[44] 4.61
z[83] 4.86
z[5] 4.87
z[26] 5.58
z[17] 5.66
z[25] 5.86
z[53] 5.91
z[3] 6.11
z[4] 6.26
z[46] 6.43
z[11] 6.64
z[55] 6.96
z[90] 7.49
z[59] 7.91
z[73] 7.87
z[40] 8.28
z[77] 8.39
z[43] 8.30
z[93] 8.32
z[95] 8.36
z[48] 8.59
z[50] 8.83
z[96] 8.73
z[52] 8.88
z[68] 8.84
z[38] 9.12
z[92] 9.18
z[89] 9.10
z[79] 9.65
z[97] 9.35
z[69] 9.59
z[75] 9.71
z[34] 10.27
z[66] 9.65
z[56] 10.05
z[41] 10.28
z[62] 10.04
z[61] 10.74
z[54] 10.31
z[42] 10.17
z[87] 10.69
z[51] 10.24
z[31] 10.52
z[33] 10.99
z[39] 10.94
z[37] 10.78
z[32] 10.86
z[88] 10.89
z[81] 10.68
z[64] 10.99
z[57] 11.19
z[74] 11.17
z[47] 11.94
z[45] 12.11
z[28] 10.64
z[84] 12.20
z[80] 11.14

What if there are three distributions?

In the simulation, we knew `x` was a mixture of two random normal variables. What happens when we attempt to use the two-component normal model with data that is more accurately characterized by a three-component mixture? Let's find out by simulating some new data and re-running.

```
x1 <- rnorm(20, 3, 2)
x2 <- rnorm(60, 10, 2)
x3 <- rnorm(20, 20, 4)

x <- c(x1,x2,x3)
data <- list(x=x,n=length(x))
```

```
model <- jags.model(textConnection(modelstring), data=data, n.adapt=1e5)
# Run samples using coda package sampler
out <- coda.samples(model=model, init=init, variable.names=c("mu1", "mu2", "z"), n.iter=1e5, thin=20)
# Extract elements from list: our estimates of the two means
mu1 <- unlist(out[,"mu1",])
mu2 <- unlist(out[,"mu2",])
```

As is clear in Figure 3, the separation is no longer evident. The reason is illustrated in Figure 4, which shows that low data points are pushed into a single component. The solution to this problem is to allow for the three components explicitly in the statistical model.

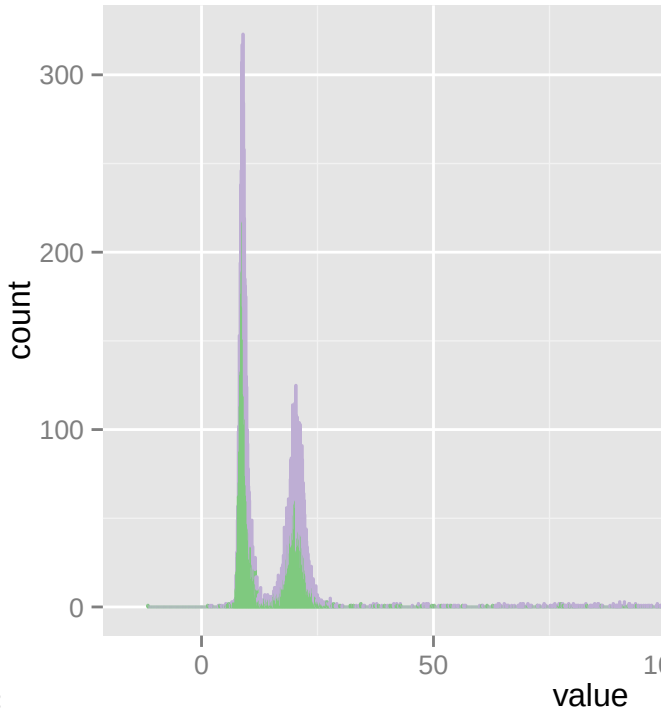
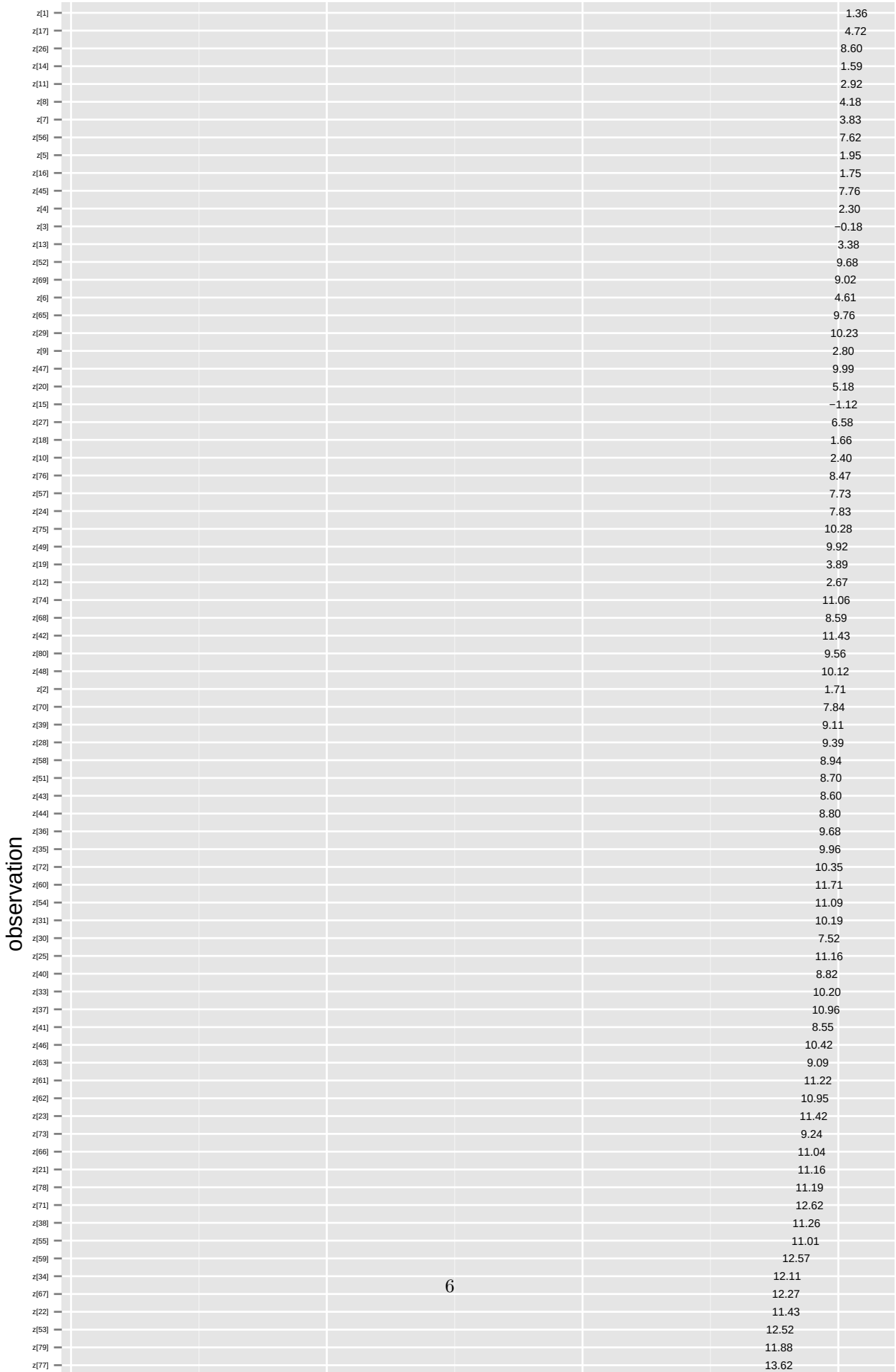


Figure 3. Density estimation for the simulated values of `mu1` and `mu2`

Figure 4. Each observation's probability of belonging to the `mu1` distribution


observation
z[1] 1.36
z[17] 4.72
z[26] 8.60
z[14] 1.59
z[11] 2.92
z[8] 4.18
z[7] 3.83
z[56] 7.62
z[5] 1.95
z[16] 1.75
z[45] 7.76
z[4] 2.30
z[3] −0.18
z[13] 3.38
z[52] 9.68
z[69] 9.02
z[6] 4.61
z[65] 9.76
z[29] 10.23
z[9] 2.80
z[47] 9.99
z[20] 5.18
z[15] −1.12
z[27] 6.58
z[18] 1.66
z[10] 2.40
z[76] 8.47
z[57] 7.73
z[24] 7.83
z[75] 10.28
z[49] 9.92
z[19] 3.89
z[12] 2.67
z[74] 11.06
z[68] 8.59
z[42] 11.43
z[80] 9.56
z[48] 10.12
z[2] 1.71
z[70] 7.84
z[39] 9.11
z[28] 9.39
z[58] 8.94
z[51] 8.70
z[43] 8.60
z[44] 8.80
z[36] 9.68
z[35] 9.96
z[72] 10.35
z[60] 11.71
z[54] 11.09
z[31] 10.19
z[30] 7.52
z[25] 11.16
z[40] 8.82
z[33] 10.20
z[37] 10.96
z[41] 8.55
z[46] 10.42
z[63] 9.09
z[61] 11.22
z[62] 10.95
z[23] 11.42
z[73] 9.24
z[66] 11.04
z[21] 11.16
z[78] 11.19
z[71] 12.62
z[38] 11.26
z[55] 11.01
z[59] 12.57
z[34] 12.11
z[67] 12.27
z[22] 11.43
z[53] 12.52
z[79] 11.88
z[77] 13.62