Generating random variables from a mixture of Normal distributions

How can I sample from a mixture distribution, and in particular a mixture of Normal distributions in `R`? For example, if I wanted to sample from:

$$0.3\times\mathcal{N}(0,1)\;+\;0.5\times\mathcal{N}(10,1)\;+\;0.2\times\mathcal{N}(3,.1)$$

how could I do that?

r random-generation mixture

edited May 28 '16 at 17:31
gung ♦
101k 33 238 494

asked Sep 24 '13 at 1:09
user30490

2 I really don't like this way of denoting a mixture. I know it's conventionally done like this but I find it misleading.The notation suggests that to sample, you need to sample all three normals and weigh the results by those coefficients which would obviously not be correct. Anyone know a better notation? – StijnDeVuyst May 28 '16 at 19:14

3 Answers

It's good practice to avoid `for` loops in `R` for performance reasons. An alternative solution which exploits the fact `rnorm` is vectorized:

```
N <- 100000

components <- sample(1:3,prob=c(0.3,0.5,0.2),size=N,replace=TRUE)
mus <- c(0,10,3)
sds <- sqrt(c(1,1,0.1))

samples <- rnorm(n=N,mean=mus[components],sd=sds[components])
```

answered Sep 24 '13 at 8:05
M. Berk
1,894 7 16

3 Alternatively, you can use the properties of the normal distribution to replace the last line by `samples <- rnorm(N)*sds[components]+mus[components]`. I find it easier to read :) – Elvis Sep 24 '13 at 8:59

Very elegant (cc @Elvis) ! – Itamar Oct 2 '13 at 7:13

In general, one of the easiest ways to sample from a mixture distribution is the following:

Algorithm Steps

1) Generate a random variable $U\sim\text{Uniform}(0,1)$

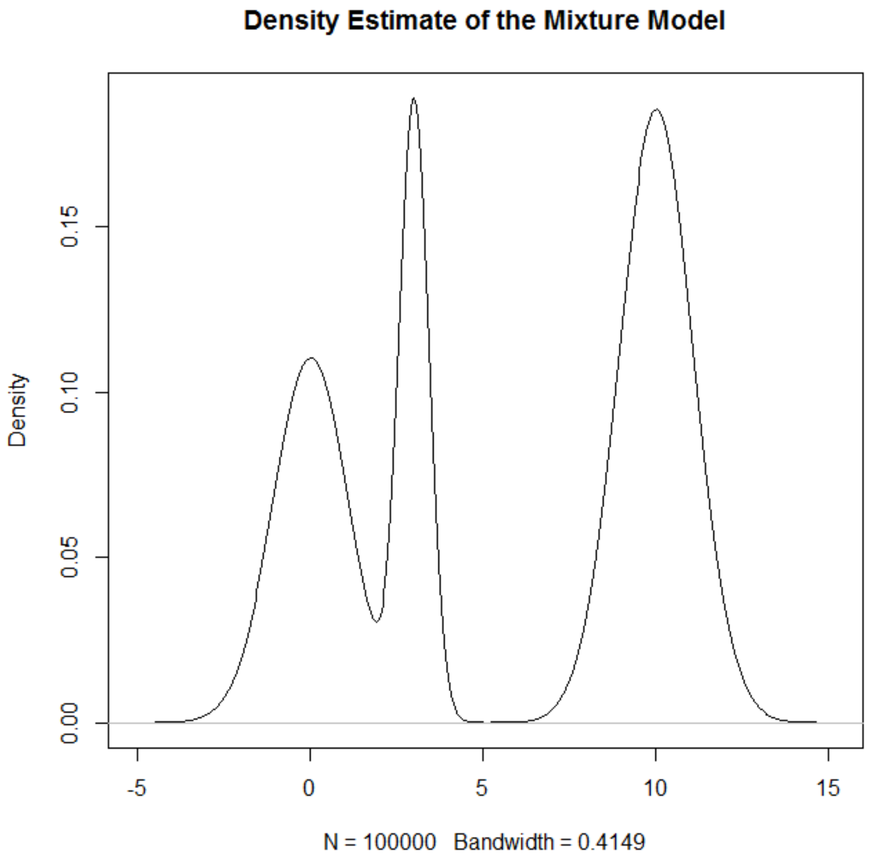
2) If $U\in\left[\sum_{i=1}^{k}p_k,\sum_{i=1}^{k+1}p_{k+1}\right)$ interval, where p_k correspond to the the probability of the k^{th} component of the mixture model, then generate from thedistribution of the k^{th} component

3) Repeat steps 1) and 2) until you have the desired amount of samples from the mixture distribution

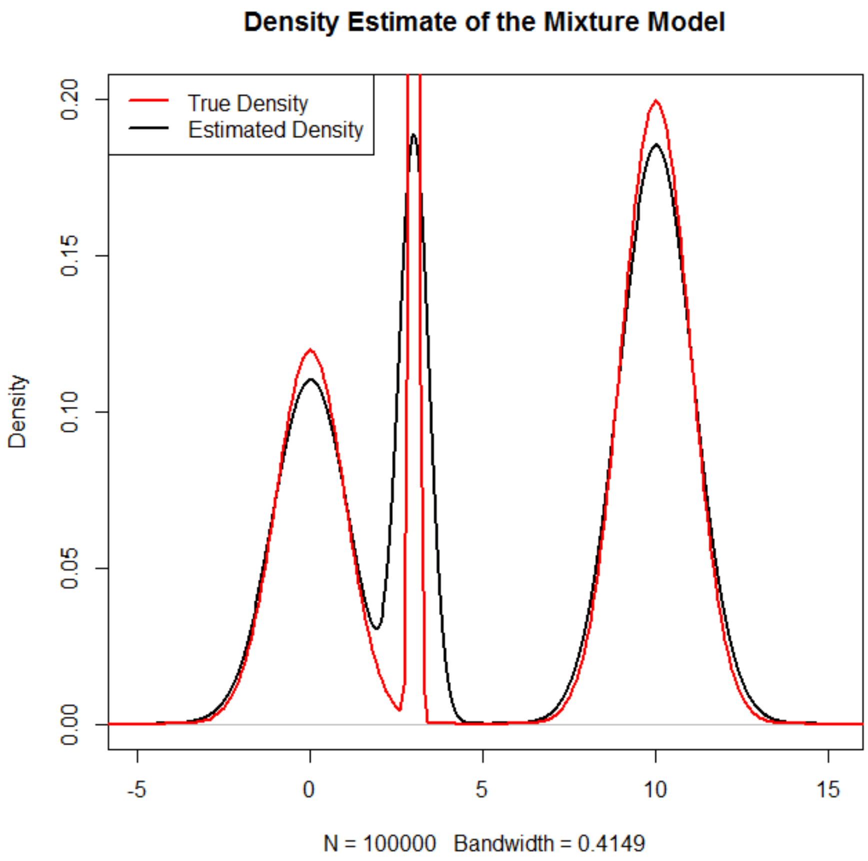
Now using the general algorithm given above, you could sample from your example mixture of normals by using the following `R` code:

```
#Variable to store the samples from the mixture distribution
rand.samples = rep(NA,N)

#Sampling from the mixture
for(i in 1:N){
    if(U[i]<.3){
        rand.samples[i] = rnorm(1,0,1)
    }else if(U[i]<.8){
        rand.samples[i] = rnorm(1,10,1)
    }else{
        rand.samples[i] = rnorm(1,3,.1)
    }
}

#Density plot of the random samples
plot(density(rand.samples),main="Density Estimate of the Mixture Model")

#Plotting the true density as a sanity check
x = seq(-20,20,.1)
truth = .3*dnorm(x,0,1) + .5*dnorm(x,10,1) + .2*dnorm(x,3,.1)
plot(density(rand.samples),main="Density Estimate of the Mixture
Model",ylim=c(0,.2),lwd=2)
lines(x,truth,col="red",lwd=2)

legend("topleft",c("True Density","Estimated
Density"),col=c("red","black"),lwd=2)
```

Which generates:



and as a sanity check:

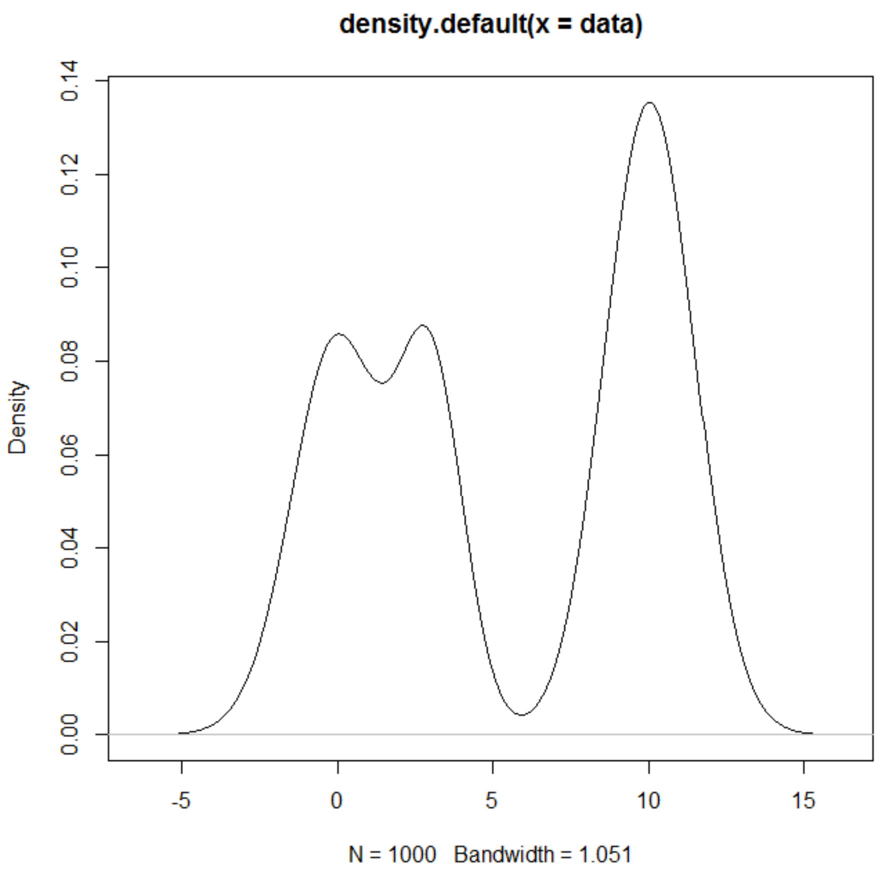


answered Sep 24 '13 at 2:16
user25658

Hi! Thanks so much! This answer helped me greatly. I am using this in a research project. I wish to quote a reference for the above. Can you please suggest a research article citation. – Abhishek Bhatia Jun 29 '15 at 11:24

Conceptually, you are just picking one distribution (from k possibilities) with some probability, and then generating pseudo-random variates from that distribution. In `R`, this would be (e.g.):

```
set.seed(8)               # this makes the example reproducible
N     = 1000              # this is how many data you want
probs = c(.3,.8)          # these are *cumulative* probabilities; since they
                          #   necessarily sum to 1, the last would be redundant
dists = runif(N)          # here I'm generating random variates from a uniform
                          #   to select the relevant distribution

# this is where the actual data are generated, it's just some if->then
#   statements, followed by the normal distributions you were interested in
data = vector(length=N)
for(i in 1:N){
  if(dists[i]<probs[1]){
    data[i] = rnorm(1, mean=0, sd=1)
  } else if(dists[i]<probs[2]){
    data[i] = rnorm(1, mean=10, sd=1)
  } else {
    data[i] = rnorm(1, mean=3, sd=.1)
  }
}

# here are a couple of ways of looking at the results
summary(data)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# -3.2820  0.8443  3.1910  5.5350 10.0700 13.1600

plot(density(data))
```



edited May 28 '16 at 17:27

answered Sep 24 '13 at 1:19

gung ♦
101k 33 238 494

Nice answer, you beat me to posting :P – user25658 Sep 24 '13 at 1:23

1 Thanks for the tip, @BabakP. I'm not sure what it was. It was something in the `ifelse()` statement, but I'll have to figure it out later. I replaced that code w/ a loop. – gung ♦ Sep 24 '13 at 2:32

6 (cc @BabakP) These are both good answers and are obviously correct (+1s). Just an `R` programming trick: you can also use the `findInterval()` and `cumsum()` commands to simplify the code and, more importantly, make it easier to generalize to a different number of dimensions. For example, for an input vector of means μ (`mu`) and variances σ^2 (`s`), and mixture probabilities (`p`), a simple function to generate n samples from this mixture would be `mix <- function(n,mu,s,p) { ii <- findInterval(runif(n),cumsum(p))+1; x <- rnorm(n,mean=mu[ii],sd=sqrt(s[ii])); return(x); }` – Macro Sep 24 '13 at 3:19

1 @Macro, very true and very nice code! I have not seen the `findInterval()` command before, however, I like to write code on here as simplistically as I can because I want it to be a tool for understanding rather than efficiency. – user25658 Sep 24 '13 at 3:22

1 I said these were good answers. My purpose was not to criticize you but to offer an approach that easily generalizes to more than three dimensions by only changing a single argument, not any code. It's not clear to me why what you've written is any more transparent than what I've written but I surely don't want to argue about that. Cheers. – Macro Sep 24 '13 at 3:54