Notes on mixture models:

First, the EM algorithm is an iterative procedure, and the time required for it to reach convergence – if it converges at all – depends strongly on the problem to which it is applied. The second key point is that because it is an iterative procedure, the EM algorithm requires starting values for the parameters, and algorithm performance can depend strongly on these initial values.

The **normalmixEM** procedure supports both user-supplied starting values and built-in estimation of starting values if none are supplied.

Another important point about the default starting values is that, aside from the number of components k, any unspecified initial parameter estimates are selected randomly by the **normalmixEM** procedure. This means that, even in cases where the default starting values consistently work well – again, the Old Faithful waiting time dataset seems to be such a case – the number of iterations required to obtain the final result can vary significantly from one run to the next.

Note, as a default, nromalmixEM will look for TWO component Gaussian distributions.

If you want to use the same starting value on successive runs, this can be done by setting the random number seed via the **set.seed** command before you invoke the **normalmixEM** procedure.)

The behaviour of iterative procedures like **normalmixEM** and others in the **mixtools** package can depend strongly on the starting values chosen to initialize the iteration process. In these situations we should presumably run it many times with differing starting values and see the resultant probability distribution of what is produced.

[mixtools: An R Package for Analyzing Finite Mixture Models](http://cran.r-project.org/web/packages/mixtools/vignettes/vignette.pdf" \t "_blank).

More:

Type **“??normalmixEM”** to see the relevant help page:

* **lambda** is the initial value of mixing proportions. Did they specify this or let it me default? Did the run it many times with it randomly varying (i.e. with seed changing or unset)?
* **mu is the starting (really? Just starting? Oh, well, in this help page by “mu” they mean the corresponding argument in the normalmixEM function.**
* **sigma, as an argument in the function, refers to the starting values of a vector of components distribution standard deviations.**
* **Next (assuming positional indexing, or just be explicit) is k, the number of component distributions to be tried. I don’t know why this is not contained in lambda…I suppose I could imagine why this might be sensible.**

More:

I thin that output part at the end is just the component distributions chose, their relative strengths, positions, spreads etc.

There is a mu sigma and lambda but also an **“se”** component of each of these. Hmm.

boot.se:

**boot.se** Performs Parametric Bootstrap for Standard Error Approximation

Bootstrapping I used to estimate the DISTRIBUTION OF SAMPLE STATISTIC. The process involves sampling from some approximating distribution. For example, and perhaps somewhat surprisingly, in certain situations at least, ne can supposedly go about it by sampling with replacement from the sample distribution itself. This would be inappropriate for example, if the sample were rather small for example.

It supposedly does this by producing B bootstrap samples for the parameters in the specified mixture model. It returns a list with the bootstrap samples and standard errors for the mixture of interest.

Through bootstrapping you are simply taking samples over and over again from the same group of data (your sample data) to estimate how accurate your estimates about the entire population (what really is out there in the real world) is.

MAYBE:

we are pretending that the sample we have obtained is a proxy for our population. This is nominally a reasonable thing to do, provided our sample size is reasonably large.

|  |  |  |
| --- | --- | --- |
| |  | | --- | |  | | I see, so if I understand you, then this technique assumes that the sample is an adequate model of the population, and therefore that resampling over that sample on a large enough scale will reveal something about the sampling distribution (of the statistics at issue), but only to the extent that the original sample is a good one. Now that I put it that way it seems almost obvious.. |

bootstrapping is to determine the sampling distribution of something. You can do it for a sample statistic (eg 56th percentile) or a test statistic (t), etc

Paraphrasing [Fox](http://cran.r-project.org/doc/contrib/Fox-Companion/), I would start by saying that the process of repeatedly resampling from your observed sample has been shown to mimic the process of the original sampling from the whole population. ER?!? IS THIS JUST AN EMPIRIcAL FINDING??? DO WE HAVE ANEXPLANATIONA?

The component distributions:

Normal? Er, not exponential?

Geometric mean.

Could lambda be something to do with ‘rate’ in the context of an exponential distribution? Could it be half life?

Sigma is, I suppose standard deviation.

No I don’t think we are appriximting an expoentnail with anromal lol. What is noral is the distribution of the half lives, yes? Any indvdual halve life itself is derived from cosndier the corresponding expoentnail distribution of course. So then, assuming this is true, what is lambda?