

# Statistical Computing with **R**

Eric Slud, Math. Dept., UMCP

October 21, 2009

## Overview of Course

This course was originally developed jointly with Benjamin Kedem and Paul Smith. It consists of modules as indicated on the Course Syllabus. These fall roughly into three main headings:

- (A). **R** (& **SAS**) language elements and functionality, including computer-science ideas;
- (B). Numerical analysis ideas and implementation of statistical algorithms, primarily in **R**; and
- (C). Data analysis and statistical applications of (A)-(B).

The object of the course is to reach a point where students have some facility in generating statistically meaningful models and outputs. Whenever possible, the use of **R** and numerical-analysis concepts is illustrated in the context of analysis of real or simulated data. The assigned homework problems will have the same flavor.

The course formerly introduced **Splus**, where now we emphasize the use of **R**. The syntax is very much the same for the two packages, but **R** costs nothing and by now has much greater capabilities. Also, in past terms **SAS** has been introduced primarily in the context of linear and generalized-linear models, to contrast its treatment of those models with the treatment in **R**. Students in this course have often had a separate and more detailed introduction to **SAS** in some other course, so in the present term we will

not present details about **SAS**, in order to leave time for interesting data-analytic topics such as Markov Chain Monte Carlo (MCMC) and multi-level modeling in **R**.

Various public datasets will be made available for illustration, homework problems and data analysis projects, as indicated on the course web-page.

The contents of these notes, not all of which are posted currently, and which will be augmented as the term progresses, are:

1. **Introduction to R**  
Unix and R preliminaries, R language basics, inputting data, lists and data-frames, factors, functions.
2. **Random Number Generation & Simulation**  
Pseudo-random number generators, shuffling, goodness of fit testing.
3. **Graphics**
4. **Simulation Speedup Methods**
5. **Numerical Maximization & Root-finding**  
(respectively for log-likelihoods and estimating equations)
6. **Commands for Subsetting**  
Manipulating Arrays and Data Frames
7. **Spline Smoothing Methods**
8. **EM Algorithm**
9. **The Bootstrap Idea**
10. **Markov Chain Monte Carlo**  
Metropolis and Gibbs Sampling Algorithms  
Convergence Diagnostics for MCMC  
Bayesian Data Analysis applications using WinBugs
11. **Multi-level Model Data Analysis**  
Linear and Generalized Linear Model Fitting and Interpretation

A few Exercises are contained in these notes, but all formal Homework assignments are posted separately in the course web-page Homework directory.

## 7 Notes on EM Algorithm

### 7.1 EM Algorithm for Multinomial & Mixture Data

**General Example 1.** Suppose that for fixed integers  $1 \leq K < C$ , cell-counts  $\mathbf{X} = (X_1, \dots, X_K)$  are observed, and cell-counts  $\mathbf{Y} = (Y_{K+1}, \dots, Y_C)$  *cannot* be observed, where

$$(X_1, \dots, X_K, Y_{K+1}, \dots, Y_C) \sim \text{Multinomial}(n, p_j(\vartheta), j = 1, \dots, C)$$

Here  $\vartheta$  is an unknown parameter of dimension  $d \leq K$ , and the functions  $p_j(\vartheta)$  which share  $\vartheta$  as a parameter are sufficiently smooth. Also denote

$$X_{K+1} = n - X_1 - \dots - X_K = Y_{K+1} + Y_{K+2} + \dots + Y_C \sum_{j=K+1}^C$$

For notational convenience, define

$$q_K(\vartheta) = 1 - p_1(\vartheta) - \dots - p_K(\vartheta)$$

In this setting, we express the conditional joint density of  $\mathbf{Y}$  given  $\mathbf{X}$  by

$$f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{X}, \vartheta) = \exp\left(\sum_{j=K+1}^C y_j \log\left(\frac{p_j(\vartheta)}{q_K(\vartheta)}\right)\right) \cdot \binom{X_{K+1}}{y_{K+1}, \dots, y_C}$$

and the *conditional log-likelihood* term can be defined omitting the multinomial-coefficient term as

$$\log L_{\mathbf{Y}|\mathbf{X}}(n_{K+1}, \dots, n_C|\mathbf{X}, \vartheta) \equiv \exp\left(\sum_{j=K+1}^C y_j \log\left(\frac{p_j(\vartheta)}{q_K(\vartheta)}\right)\right)$$

The *E-step* of the EM algorithm replaces  $E_{\vartheta_1}(\log L_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}|\mathbf{X}, \vartheta))$  by

$$X_{K+1} \sum_{j=K+1}^C \frac{p_j(\vartheta_1)}{q_K(\vartheta_1)} \log\left(\frac{p_j(\vartheta)}{q_K(\vartheta)}\right)$$

or equivalently, replaces  $Y_j$  by  $X_{K+1} \cdot p_j(\vartheta_1)/q_K(\vartheta_1)$  for  $j = K+1, \dots, C$ .

To confirm that the definition of log-likelihood and conditional log-likelihood terms as above, without multinomial coefficients, is legitimate, we observe

that the property needed in the proof of log-likelihood improvement for EM iterations holds, that is,

$$E_{\vartheta} \left( \log L_{\mathbf{Y}|\mathbf{X}}(n_{C+1}, \dots, n_K | \mathbf{X}, \vartheta) - \log L_{\mathbf{Y}|\mathbf{X}}(n_{C+1}, \dots, n_K | \mathbf{X}, \vartheta_1) \right) \geq 0$$

or equivalently, for all  $\vartheta, \vartheta_1$ ,

$$\sum_{j=C+1}^K \frac{p_j(\vartheta)}{q(\vartheta)} \log \frac{p_j(\vartheta) q(\vartheta_1)}{q(\vartheta) p_j(\vartheta_1)} \geq 0$$

But this is a standard, discrete version of the famous ‘Information Inequality’ proved more generally in the form  $\int f(x) \log(f(x)/g(x)) d\nu(x) \geq 0$  for probability densities with respect to a measure  $\nu$ , using Jensen’s Inequality.

With the conditional likelihood for  $\mathbf{Y}$  given  $\mathbf{X}$  defined as above, we have seen that the *E-step* of the EM algorithm replaces this conditional log-likelihood, when the current parameter-iterate is  $\vartheta_1$ , by

$$E_{\vartheta_1} \left( \log L_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}|\mathbf{X}, \vartheta) \mid \mathbf{X} \right) = n^* \sum_{j=C+1}^K \frac{p_j(\vartheta_1)}{q(\vartheta_1)} \log \left( \frac{p_j(\vartheta)}{q(\vartheta)} \right)$$

This expression is also equal to  $\log L_{\mathbf{Y}|\mathbf{X}}(n_{C+1}^*, \dots, n_K^* | \mathbf{X}, \vartheta)$ , where  $n_j^* = E_{\vartheta_1}(n_j^* | \mathbf{X}) = n^* \cdot p_j(\vartheta_1)/q(\vartheta_1)$  for  $j = C + 1, \dots, K$ .

Thus we have the following comparison between maximization approaches. First, the complete-data likelihood to maximize, if  $\mathbf{Y}$  could also be observed, would be

$$\sum_{j=1}^K X_j \log p_j(\vartheta) + X_{K+1} \log q_K(\vartheta) + \sum_{j=K+1}^C Y_j \log \frac{p_j(\vartheta)}{q_K(\vartheta)}$$

while the crude marginal-observed-data likelihood to maximize is

$$\sum_{j=1}^K X_j \log p_j(\vartheta) + X_{K+1} \log q_K(\vartheta)$$

On the other hand, the *M-step* of the EM algorithm, after replacement of the unobservable  $Y_j$  values in the complete-data likelihood by their *E-step* imputed values, is

$$\sum_{j=1}^K X_j \log p_j(\vartheta) + X_{K+1} \log q_K(\vartheta) + X_{K+1} \sum_{j=K+1}^C \frac{p_j(\vartheta_1)}{q_K(\vartheta_1)} \log \frac{p_j(\vartheta)}{q_K(\vartheta)}$$

$$= \sum_{j=1}^K X_j \log p_j(\vartheta) + X_{K+1} \sum_{j=K+1}^C \frac{p_j(\vartheta_1)}{q_K(\vartheta_1)} \log p_j(\vartheta)$$

Note that the M-step involves a step of maximizing the complete-data likelihood using imputed data for the  $Y_j$ 's, which will be very easy in some problems.

*A key aspect of the usefulness of the EM algorithm in multinomial missing data problems is that no sums of terms  $p_j(\vartheta)$  appear inside the logarithms arising in the maximization-step. Especially in so-called log-linear contingency-table models with some missing cell-counts, where the  $p_j(\vartheta)$  have some multiplicative structure, this is very useful !*

#### SPECIAL EXAMPLE FROM THE ORIGINAL EM PAPER

This example fits into the structure of the general multinomial example, with scalar unknown parameter  $\vartheta = \pi$ ,  $K = 3$ ,  $C = 5$ , and

$$p_1(\pi) = p_2(\pi) = \frac{1 - \pi}{4}, \quad p_3(\pi) = p_4(\pi) = \frac{\pi}{4}, \quad p_5(\pi) = \frac{1}{2}$$

The cell-counts given as data in Dempster, Laird & Rubin (1978) are:

$(X_1, X_2, X_3, X_4) = (18, 20, 34, 125)$ . Appealing to the formulas above, we find that the complete-data M-step involves maximizing  $\sum_{j=1}^3 X_j \log p_j(\pi) + \sum_{j=4}^5 Y_j \log p_j(\pi)$ . In this particular problem, we are equivalently maximizing  $(X_3 + Y_4) \log(\pi/4) + (X_1 + X_2) \log((1 - \pi)/4)$ , which leads to

$$\hat{\pi} = (X_3 + Y_4)/(n - Y_5)$$

Substituting the E-step imputed values for the  $Y_j$  gives the EM iteration explicitly, starting from initial guess  $\pi_1$ , as:

$$\begin{aligned} \pi_2 &= \left( X_3 + X_4 \cdot \frac{\pi_1/4}{1/2 + \pi_1/4} \right) / \left( n - X_4 \cdot \frac{1/2}{1/2 + \pi_1/4} \right) \\ &= \frac{34 + 125 \cdot \frac{\pi_1}{2 + \pi_1}}{197 - 125 \cdot \frac{2}{2 + \pi_1}} = \frac{68 + 159 \pi_1}{144 + 197 \pi_1} \end{aligned}$$

In this little example, EM iterates the mapping  $h(\pi) \equiv (68 + 159\pi)/(144 + 197\pi)$  to find the fixed-point. (The unique fixed-point  $\pi = 0.6268$  solves  $h(\pi) = \pi$ , which is a quadratic equation.) The Quasi-Newton optimization of the marginal likelihood is messier but, using a modern computer, quicker and more reliable.

```

> optimize(function(x) 38*log(1-x)+34*log(x)+125*log(x+2),
  c(.01,.99), max=T)$max
[1] 0.6268036
> h = function(x) (159 * x + 68)/(197 * x + 144)
  x = .5; for (i in 1:6) {x = h(x); cat(round(x,5)," \n")}
0.60825
0.62432
0.62649
0.62678
0.62682    ### converged to 5 places after 5 iterations

```

**General Example 2.** Consider ‘mixture’ data  $X_i$  which are *iid* continuously distributed *rv*’s with density

$$f_X(x) = pe^{-x} + \lambda(1-p)e^{-\lambda x}, \quad x > 0$$

where  $\vartheta = (p, \lambda) \in (0, 1) \times [0, \infty)$  is the unknown parameter. These r.v.’s are of *mixture* type because they have the same density as random variables

$$X_i = \epsilon_i U_i + (1 - \epsilon_i) V_i \quad U_i \sim \text{Expon}(1), \quad V_i \sim \text{Expon}(\lambda)$$

where  $\epsilon_i \sim \text{Binom}(1, p)$  is independent of  $(U_i, V_i)$ . The marginal density for the observed variables is  $f_X$ , but the problem would be much simpler to analyze with the ‘complete’ data  $(X_i, \epsilon_i)$ ,  $i = 1, \dots, n$ . Now the *E-step* of the EM algorithm based on observing only  $\mathbf{X} = (X_i, i = 1, \dots, n)$  consists of calculating

$$E_{\vartheta_1}(\epsilon | X) = \frac{p_1 e^{-X}}{p_1 e^{-X} + \lambda_1 (1 - p_1) e^{-\lambda_1 X}} = \epsilon^*(X, \vartheta_1) = \epsilon^*$$

and then substituting to obtain

$$\begin{aligned}
E_{\vartheta_1} \log p_{\epsilon|X}(\epsilon | X, \vartheta) &= \epsilon^* \log \left( \frac{p e^{-X}}{p e^{-X} + \lambda (1 - p) e^{-\lambda X}} \right) \\
&+ (1 - \epsilon^*) \log \left( \frac{\lambda (1 - p) e^{-\lambda X}}{p e^{-X} + \lambda (1 - p) e^{-\lambda X}} \right)
\end{aligned}$$

As a result, starting from initial guess  $\vartheta_1 = (\lambda_1, p_1)$ , the *M-step* of the EM algorithm is to maximize the ‘complete-data log-likelihood’ for the data

$(X_i, \epsilon^*(X_i, \vartheta_1), i = 1, \dots, n)$ , which is given simply in terms of

$$m^* = \sum_{i=1}^n \epsilon^*(X_i, \vartheta_1) \quad , \quad \bar{U} = (m^*)^{-1} \sum_{i=1}^n X_i \epsilon^*(X_i, \vartheta_1)$$

and

$$\bar{V} = (n - m^*)^{-1} \sum_{i=1}^n X_i (1 - \epsilon^*(X_i, \vartheta_1))$$

as

$$m^* (\log p - \bar{U}) + (n - m^*) (\log(\lambda(1 - p)) - \lambda \bar{V})$$

Thus the *M-step* is given in closed form by maximizing the last expression in  $(\lambda, p)$  to obtain

$$p_2 = m^*/n \quad , \quad \lambda_2 = 1/\bar{V}$$

In summary, the entire EM iteration-step in this example, starting from initial guess  $\vartheta_1 = (\lambda_1, p_1)$ , is given in closed form by:

$$p_2 = n^{-1} \sum_{i=1}^n \frac{p_1 e^{-X_i}}{p_1 e^{-X_i} + \lambda_1 (1 - p_1) e^{-\lambda_1 X_i}}$$

$$1/\lambda_2 = \frac{1}{n(1 - p_2)} \sum_{i=1}^n \frac{(1 - p_1) \lambda_1 X_i e^{-\lambda_1 X_i}}{p_1 e^{-X_i} + \lambda_1 (1 - p_1) e^{-\lambda_1 X_i}}$$

We code this, and evaluate the results in a little simulated dataset, as follows.

```
> EMiter
function(thet, Xvec)  {
## On input, thet is the vector consisting of old values of
##   p, lambda in General Example 2 of Notes, and Xvec is
##   the observed data vector. The output is the new theta.
  frac = 1/(1 + (1/thet[1] - 1) * thet[2] * exp((
    1 - thet[2]) * Xvec))
  pnew = mean(frac)
  lamnew = (1 - pnew)/mean(Xvec * (1 - frac))
  list(thet = c(pnew, lamnew), logL = sum(log(pnew * exp(
    - Xvec) + (1 - pnew) * lamnew * exp(- lamnew * Xvec))))
}
```

```

> epsv = rbinom(10000, 1, .6)
  Xv = rexp(10000)/exp(.3*(1-epsv))
> round(c(mean(epsv), .4/exp(.3)+.6, mean(Xv)),5)
[1] 0.59870 0.89633 0.89548

> theta = c(.5,1.5)
## Initial log-likelihood
  sum(log(.5 * exp( - Xv) + .5*1.5*exp(-1.5*Xv))) ## = -8922.3
## Log-likelihood at true values:
> sum(log(.6 * exp( - Xv) + .4*exp(.3-exp(.3)*Xv)))## -8894.8

> unlist(EMiter(theta,Xv))  ## values after one EM iteration
      thet1      thet2      logL
  0.5072665    1.4103608 -8905.1229932

> for(i in 1:100) {
  tmpitr = EMiter(theta,Xv)
  theta = tmpitr$thet
  if(i %% 10 ==0) cat(round(unlist(tmpitr),5),"\n") }
0.51622 1.27855 -8894.966
0.51684 1.27756 -8894.964
0.51738 1.27793 -8894.962
0.51792 1.27832 -8894.961
0.51847 1.27871 -8894.96
0.51901 1.27909 -8894.959
0.51955 1.27948 -8894.958
0.52008 1.27987 -8894.956
0.52062 1.28026 -8894.955
0.52116 1.28064 -8894.954

> for(i in 1:100) theta = EMiter(theta,Xv)$thet
  unlist(EMiter(theta,Xv))
      thet1      thet2      logL
  0.5264933    1.2845611 -8894.94227

> for(i in 1:100) theta = EMiter(theta,Xv)$thet

```

```

unlist(EMiter(theta,Xv))
      thet1      thet2      logL
0.5316379      1.2884347 -8894.93122

## Convergence is painfully slow !!!

> nlm(b=c(0.5, 1.5), function(x) - sum(log(x[1] * exp(-Xv) +
      (1 - x[1]) * x[2] * exp(-x[2] * Xv))), lower
      = c(0.01, 0.1), upper = c(0.99, 10),
      control=list(trace=1))[c(1,2,4)]
...
[prints estimate and objective at each iteration, ending with]
13:      8894.8192: 0.618188  1.37204
$par
[1] 0.6181876 1.3720355

$objective
[1] 8894.82

$message
[1] "relative convergence (4)"
## Final code of 0 for successful convergence

```

Note the very slow convergence of the EM algorithm implemented and tested here. The maximized *logLik* must be larger than  $-8894.83$ , since that is the value at the true parameters ( $p = .6$ ,  $\lambda = e^{-3}$ ), but from the not-too-awful starting values  $p_1 = .5$ ,  $\lambda = 1.5$ , it took more than 300 EM iterations to get there! As can be seen from the final converged maximization via **nlminb**, the final maximized *logLik* is  $-8894.82$ .

Many of the interesting and computationally challenging applications of EM arise in so-called *random effect models* where unobserved random variables (often, unobserved random errors at some intermediate level of aggregation like “cluster”) must be integrated out to find log-likelihood. We discuss random-effect linear and nonlinear/generalized-linear regression models in the next segment of the course.