center for environmental statistics.

MAJORIZATION ALGORITHMS FOR PROBIT MODELS: THE R PACKAGE PROBIT

JAN DE LEEUW

ABSTRACT. The Expectation-Maximization algorithm is derived as a special case of the Majorization Method. We specialize this general derivation to both univariate and multivariate discrete normal distributions, latent variable models, and missing data imputation. The corresponding algorithms, with R code, are also given.

1. INTRODUCTION

The *majorization method* is a general approach, or family of approaches, to construct optimization methods. Some general publications about majorization are Kiers [1990]; De Leeuw [1994]; Heiser [1995]; Lange et al. [2000]; Hunter and Lange [2004]; De Leeuw and Lange [2009].

Date: Sunday 7th June, 2009 — 13h 26min — Typeset in LUCIDA BRIGHT.
Key words and phrases. Missing Data, Multivariate Normal, EM Algorithm, Majortization.

Suppose the problem is to minimize $f : \mathcal{X} \Rightarrow \mathbb{R}$ over $\mathcal{X} \subseteq \mathbb{R}^n$. A function $F : \mathcal{X} \otimes \mathcal{X} \Rightarrow \mathbb{R}$ is a *majorization function* if $f(x) \leq F(x, y)$ for all $x, y \in \mathcal{X}$ and $f(x) = F(x, x)$ for all $x \in \mathcal{X}$.

The iterative *majorization algorithm* finds the update of $x^{(k)}$ by computing

$$\mathcal{X}^{(k)} \stackrel{\Delta}{=} \mathop{\textbf{argmax}}_{x \in \mathcal{X}} F(x, x^{(k)}).$$

If $x^{(k)} \in \mathcal{X}^{(k)}$ we stop. Else we select $x^{(k+1)} \in \mathcal{X}^{(k)}$. The *sandwich inequality*

$$f(x^{(k+1)}) \leq F(x^{(k+1)}, x^{(k)}) < F(x^{(k)}, x^{(k)}) = f(x^{(k)}$$

shows that the algorithm either stops, or produces a decreasing sequence of function values. Under compactness and continuity conditions this implies convergence [Zangwill, 1969].

Of course if we are maximizing f, then we can construct a suitable minorization function and maximize that in each iterative step. To cover both minorization and majorization Lange et al. [2000] propose the name *MM algorithm*, where the first *M* stands for either majorization or minorization, and the second *M* stands for either maximation or minimization.

Majorization and minorization functions are usually derived from classical inequalities, for Taylor's Theorem, or from convexity considerations. The *Expectation-Maximization* or *EM algorithm* is a family of MM algorithms based on Jensen's Inequality, usually applied in the statistical context of computing maximum likelihood estimates [Dempster et al., 1977; McLachlan and Krishnan, 2008]. The general idea of using MM algorithms in data analysis came about by realizing that the EM algorithm, based on Jensen's Inequality, and the SMACOF method for multidimensional scaling [De Leeuw, 1977; De Leeuw and Heiser, 1977, 1980], based on the Cauchy-Schwartz Inequality, were both examples of a more general approach to algorithm construction.

1.1. **EM as MM.** Suppose that $g : X \otimes Y \Rightarrow \mathbb{R}^+$, where $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$. Define $f : X \to \mathbb{R}^+$ by

$$f(x) \triangleq \log \int_Y g(x,y)dy.$$

The problem we study in this paper is maximization of f over X.

Suppose $x, \tilde{x} \in X$. We assume that if $x \neq \tilde{x}$ then $g(x,y) \neq g(\tilde{x},y)$ for all $y \in Y$. Now

$$f(x) - f(\tilde{x}) = \log \frac{\int_Y g(x,y)dy}{\int_Y g(\tilde{x},y)dy} = \log \frac{\int_Y g(\tilde{x},y)\frac{g(x,y)}{g(\tilde{x},y)}dy}{\int_Y g(\tilde{x},y)dy}.$$

Let

$$h(x,y) \triangleq \frac{g(x,y)}{\int_Y g(x,y)dy}.$$

Then $\int_Y h(x,y)dy = 1$ for all x and

$$f(x) - f(\tilde{x}) = \log \int_Y h(\tilde{x},y)\frac{g(x,y)}{g(\tilde{x},y)}dy.$$

Applying Jensen's Inequality to the right hand side gives

$$f(x) > f(\tilde{x}) + k(x,\tilde{x}) - k(\tilde{x},\tilde{x}),$$

where we use the abbreviation

$$k(x,\tilde{x}) \triangleq \int_Y h(\tilde{x},y) \log g(x,y)dy.$$

The function $F(x,\tilde{x}) = f(\tilde{x}) + k(x,\tilde{x}) - k(\tilde{x},\tilde{x})$ is the required minorization function.

This leads to the MM algorithm in which

$$\mathcal{X}^{(k)} \triangleq \mathop{\mathbf{argmax}}_{x \in X}\ F(x, x^{(k)}) = \mathop{\mathbf{argmax}}_{x \in X}\ k(x, x^{(k)}),$$

and $x^{(k+1)} \in \mathcal{X}^{(k)}$.

2. Probit Analysis

2.1. **The Discrete Normal.** In our first example we want to maximize

$$f(\mu,\sigma^2)=\sum_{j=1}^m n_j\log\frac{1}{\sigma}\int_{\alpha_{j-1}}^{\alpha_j}\phi(\frac{y-\mu}{\sigma})dy, \tag{1}$$

where

$$\phi(y)=\frac{1}{\sqrt{2\pi}}\exp\{-\frac{1}{2}y^2\},$$

Moreover the $-\infty=\alpha_0<\alpha_1<\cdots<\alpha_{m-1}<\alpha_m=+\infty$ are known *knots*, and the n_j are observed *frequencies*.

Alternatively, writing n for the sum of the n_j and $p_j=n_j/n$,

$$f(\mu,\sigma^2)=n\sum_{j=1}^m p_j\log\pi_j(\mu,\sigma^2),$$

where

$$\pi_j(\mu,\sigma^2)=\Phi(\frac{\alpha_j-\mu}{\sigma})-\Phi(\frac{\alpha_{j-1}-\mu}{\sigma})$$

and

$$\Phi(x)=\int_{-\infty}^x\phi(y)dy.$$

Maximizing (1) means maximum likelihood estimates of the parameters of a *discretized normal distribution*. The problem is important, because in actual data analysis so-called "continuous distributions" are always observed in a discretized form.

We now apply the theory in subsection 1.1 to each of the m terms in equation (1). In this case, for $\alpha_{j-1}<y<\alpha_j$,

$$h_j(\tilde\mu,\tilde\sigma,y)=\frac{\frac{1}{\tilde\sigma}\phi(\frac{y-\tilde\mu}{\tilde\sigma})}{\Phi(\frac{\alpha_j-\tilde\mu}{\tilde\sigma})-\Phi(\frac{\alpha_{j-1}-\tilde\mu}{\tilde\sigma})},$$

i.e. h is the doubly-truncated normal [Johnson et al., 1994, section 10.1, p. 156–162].

In the majorization algorithm we must minimize in each step

$$\ell(\mu,\sigma^2,\tilde\mu,\tilde\sigma^2)=\log\sigma^2+\frac{1}{\sigma^2}\sum_{j=1}^m p_j\int_{\alpha_{j-1}}^{\alpha_j}h_j(\tilde\mu,\tilde\sigma,y)(y-\mu)^2dy.$$

Define the conditional means and variances

$$\tilde{\mu}_j \triangleq \int_{\alpha_{j-1}}^{\alpha_j} h_j(\tilde{\mu}, \tilde{\sigma}, y)\, y\, dy,$$

$$\tilde{\sigma}_j^2 \triangleq \int_{\alpha_{j-1}}^{\alpha_j} h_j(\tilde{\mu}, \tilde{\sigma}, y)(y - \tilde{\mu}_j)^2 dy.$$

Then

$$\ell(\mu, \sigma^2, \tilde{\mu}, \tilde{\sigma}^2) = \log \sigma^2 + \frac{1}{\sigma^2}\{\sum_{j=1}^m p_j \tilde{\sigma}_j^2 + \sum_{j=1}^m p_j (\tilde{\mu}_j - \mu)^2\}.$$

It follows that

$$\mu^{(k+1)} = \sum_{j=1}^m p_j \tilde{\mu}_j^{(k)},$$

and

$$(\sigma^2)^{(k+1)} = \sum_{j=1}^m p_j (\tilde{\sigma}_j^2)^{(k)} + \sum_{j=1}^m p_j (\tilde{\mu}_j^{(k)} - \mu^{(k+1)})^2.$$

This can be worked out in more detail by using the formulas for the mean and the variance of the doubly-truncated normal distribution [Johnson et al., 1994, formulas 13.134 and 13.135]. Specifically

(2a)
$$\tilde{\mu}_j = \tilde{\mu} - \left[\frac{\phi(\frac{\alpha_j - \tilde{\mu}}{\tilde{\sigma}}) - \phi(\frac{\alpha_{j-1} - \tilde{\mu}}{\tilde{\sigma}})}{\Phi(\frac{\alpha_j - \tilde{\mu}}{\tilde{\sigma}}) - \Phi(\frac{\alpha_{j-1} - \tilde{\mu}}{\tilde{\sigma}})}\right] \tilde{\sigma}.$$

and

(2b)
$$\tilde{\sigma}_j^2 = \left[1 - \frac{\left(\frac{\alpha_j - \tilde{\mu}}{\tilde{\sigma}}\right)\phi(\frac{\alpha_j - \tilde{\mu}}{\tilde{\sigma}}) - \left(\frac{\alpha_{j-1} - \tilde{\mu}}{\tilde{\sigma}}\right)\phi(\frac{\alpha_{j-1} - \tilde{\mu}}{\tilde{\sigma}})}{\Phi(\frac{\alpha_j - \tilde{\mu}}{\tilde{\sigma}}) - \Phi(\frac{\alpha_{j-1} - \tilde{\mu}}{\tilde{\sigma}})} - \left\{\frac{\phi(\frac{\alpha_j - \tilde{\mu}}{\tilde{\sigma}}) - \phi(\frac{\alpha_{j-1} - \tilde{\mu}}{\tilde{\sigma}})}{\Phi(\frac{\alpha_j - \tilde{\mu}}{\tilde{\sigma}}) - \Phi(\frac{\alpha_{j-1} - \tilde{\mu}}{\tilde{\sigma}})}\right\}^2\right] \tilde{\sigma}^2$$

The R code is in Appendix A.1.

2.2. **Limiting Case.** Suppose we have n observations $y_1 < y_2 < \cdots < y_n$, without ties. Define $\epsilon < \min_{i,k}(y_i - y_{i-1})$, and let $\alpha_{i-1} = y_i - \frac{1}{2}\epsilon$ and $\alpha_i = y_i + \frac{1}{2}\epsilon$. Then each of the n open intervals (α_{i-1}, α_i) of length ϵ, contains exactly one observation. Let us now study what happens if ϵ is small.

If a and b are any two functions of $\mathbb{R}$ into $\mathbb{R}^+$ that are sufficiently many times differentiable, then

$$\frac{a(\delta+\frac{1}{2}\epsilon)-a(\delta-\frac{1}{2}\epsilon)}{b(\delta+\frac{1}{2}\epsilon)-b(\delta-\frac{1}{2}\epsilon)} = \frac{a'(\delta)+\frac{1}{24}\epsilon^2 a'''(\delta)+o(\epsilon^2)}{b'(\delta)+\frac{1}{24}\epsilon^2 b'''(\delta)+o(\epsilon^2)} =$$

$$= \frac{a'(\delta)}{b'(\delta)}\left[1+\frac{1}{24}\left\{\frac{a'''(\delta)}{a'(\delta)}-\frac{b'''(\delta)}{b'(\delta)}\right\}\epsilon^2+o(\epsilon^2)\right]. \tag{3}$$

Define $y_i = \frac{y_i-\mu}{\sigma}$. Then $\frac{\alpha_i-\mu}{\sigma} = y_i + \frac{1}{2}\frac{\epsilon}{\sigma}$ and $\frac{\alpha_{i-1}-\mu}{\sigma} = y_i - \frac{1}{2}\frac{\epsilon}{\sigma}$. Now use

$$\Phi'(x) = \phi(x),$$
$$\Phi''(x) = \phi'(x) = -x\phi(x),$$
$$\Phi'''(x) = \phi''(x) = (x^2-1)\phi(x),$$
$$\Phi''''(x) = \phi'''(x) = -(x^3-3x)\phi(x),$$
$$\Phi'''''(x) = \phi''''(x) = (x^4-6x^2+3)\phi(x).$$

From (3) with $a = \phi$ and $b = \Phi$ we find

$$\mu_i - \mu = (y_i - \mu)\left\{1 - \frac{1}{12}\frac{\epsilon^2}{\sigma^2}\right\} + o(\epsilon^2),$$

and with $a = -\phi'$ and $b = \Phi$ we find

$$\sigma_i^2 = \frac{1}{12}\epsilon^2 + o(\epsilon^2).$$

The R code in the Appendix A.2 has an example with 1000 standard normals categorized in 8000 equal-length intervals between -4 and +4.

2.3. **Probit Regression.** Suppose $F = \{f_{ij}\}$ is an $n \times m$ table of frequencies. We suppose that row i is a sample from a discrete normal with mean μ_i and variance σ_i^2 and that the discretization points are the same for each row. To make this a regression problem we suppose that $\mu_i = x_i'\beta$.

The log-likelihood is $\sum_{i=1}^n f_{i\bullet} \sum_{j=1}^m p_{ij} \log \pi_{ij}$, where the $f_{i\bullet}$ are the row marginals, $p_{ij} = f_{ij}/f_{i\bullet}$, and

$$\pi_{ij} = \frac{1}{\sigma_i}\int_{\alpha_{j-1}}^{\alpha_j} \phi(\frac{y - x_i'\beta}{\sigma_i})dy = \Phi(\frac{\alpha_j - x_i'\beta}{\sigma_i}) - \Phi(\frac{\alpha_{j-1} - x_i'\beta}{\sigma_i}).$$

Exactly as before we define

$$h_{ij}(\tilde\beta, \tilde\sigma, y) = \frac{\frac{1}{\tilde\sigma_i}\phi(\frac{y - x_i'\tilde\beta}{\tilde\sigma_i})}{\Phi(\frac{\alpha_j - x_i'\tilde\beta}{\tilde\sigma_i}) - \Phi(\frac{\alpha_{j-1} - x_i'\tilde\beta}{\tilde\sigma_i})}$$

as well as

$$\tilde\mu_{ij} \triangleq \int_{\alpha_{j-1}}^{\alpha_j} h_{ij}(\tilde\beta, \tilde\sigma, y)y\,dy,$$
$$\tilde\sigma_{ij}^2 \triangleq \int_{\alpha_{j-1}}^{\alpha_j} h_{ij}(\tilde\beta, \tilde\sigma, y)(y - \tilde\mu_{ij})^2 dy.$$

Then

$$\ell(\beta, \sigma^2, \tilde\beta, \tilde\sigma^2) = \\ = \sum_{i=1}^n f_{i\bullet}\left\{\log \sigma_i^2 + \frac{1}{\sigma_i^2}\{\sum_{j=1}^m p_{ij}\tilde\sigma_{ij}^2 + \sum_{j=1}^m p_{ij}(\tilde\mu_{ij} - x_i'\beta)^2\}\right\}.$$

Thus a majorization step involves solving the equations

$$(\sigma_i^2)^{(k+1)} = \sum_{j=1}^m p_{ij}(\sigma_{ij}^2)^{(k)} + \sum_{j=1}^m p_{ij}(\mu_{ij}^{(k)} - x_i'\beta^{(k+1)})^2,$$

and

$$\beta^{(k+1)} = \mathop{\mathbf{argmin}}_{\beta} \sum_{i=1}^n \sum_{j=1}^m \frac{f_{ij}}{(\sigma_i^2)^{(k+1)}}(\mu_{ij}^{(k)} - x_i'\beta)^2.$$

In the special case in which we assume that all σ_i^2 are equal, we can solve these equations directly, but in the general case a simple block relaxation algorithm is needed. The R code is in Appendix A.3.

2.3.1. *Variable Knots.* There is a further elaboration incorporated in most probit regression programs. It treats the α_j as unknowns and computes them along with the σ and β. We treat this as a separate optimization problem, not using MM, but the method of scoring.

In order to improve the α_j for given σ and β we write the loss function in the form

$$f(\alpha) = \sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}\log\left\{\Phi(\frac{\alpha_j-\mu_i}{\sigma_i}) - \Phi(\frac{\alpha_{j-1}-\mu_i}{\sigma_i})\right\}.$$

Remember that $\alpha_0 = -\infty$ and $\alpha_m = +\infty$, so only $\alpha_1, \cdots, \alpha_{m-1}$ are variable. Define

$$\delta_{ij} \triangleq \frac{f_{ij}}{\pi_{ij}} - \frac{f_{ij+1}}{\pi_{ij+1}},$$
$$\phi_{ij} \triangleq \frac{1}{\sigma_i}\phi(\frac{\alpha_j-\mu_i}{\sigma_i})$$

We find

$$\frac{\partial f}{\partial\alpha_j} = \sum_{i=1}^{n}\phi_{ij}\delta_{ij}.$$

For scoring we need the expected value of the cross product of the partials. Thus

$$\mathbf{E}\left(\frac{\partial f}{\partial\alpha_j}\frac{\partial f}{\partial\alpha_\ell}\right) = \sum_{i=1}^{n}\phi_{ij}\phi_{i\ell}\mathbf{E}(\delta_{ij}\delta_{i\ell}).$$

Now

$$\mathbf{E}(\delta_{ij}\delta_{i\ell}) = f_{i\bullet}\begin{cases}\frac{1}{\pi_j}+\frac{1}{\pi_{j+1}} & \text{if } j=\ell,\\ -\frac{1}{\pi_{\max(j,\ell)}} & \text{if } |j-\ell|=1,\\ 0 & \text{otherwise.}\end{cases}$$

This is enough to do the actual computations. Again the R code is in Appendix A.3.

References

J. De Leeuw. Correctness of Kruskal's Algorithms for Monotone Regression with Ties. *Psychometrika*, 42:141–144, 1977.

J. De Leeuw. Block Relaxation Methods in Statistics. In H.H. Bock, W. Lenski, and M.M. Richter, editors, *Information Systems and Data Analysis*, Berlin, 1994. Springer Verlag.

J. De Leeuw and W. J. Heiser. Multidimensional Scaling with Restrictions on the Configuration. In P.R. Krishnaiah, editor, *Multivariate Analysis, Volume V*, pages 501–522, Amsterdam, The Netherlands, 1980. North Holland Publishing Company.

J. De Leeuw and W.J. Heiser. Convergence of Correction Matrix Algorithms for Multidimensional Scaling. In J.C. Lingoes, editor, *Geometric Representations of Relational Data*, chapter 32. Mathesis Press, Ann Arbor, Michigan, 1977.

J. De Leeuw and K. Lange. Sharp Quadratic Majorization in One Dimension. *Computational Statistics and Data Analysis*, 53:2471–2484, 2009.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM algorithm (with Discussion). *Journal of the Royal Statistical Society Series B*, 39:1–38, 1977.

W.J. Heiser. Convergent Computing by Iterative Majorization: Theory and Applications in Multidimensional Data Analysis. In W.J. Krzanowski, editor, *Recent Advantages in Descriptive Multivariate Analysis*, pages 157–189. Clarendon Press, Oxford, 1995.

D.R. Hunter and K. Lange. A Tutorial on MM Algorithms. *American Statistician*, 58(30–37), 2004.

N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions, Volume I.* Wiley, second edition, 1994.

H. Kiers. Majorization as a Tool for Optimizing a Class of Matrix Functions. *Psychometrika*, 55:417–428, 1990.

K. Lange, D.R. Hunter, and I. Yang. Optimization Transfer Using Surrogate Objective Functions. *Journal of Computational and Graphical Statistics*, 9:1–20, 2000.

G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, second edition, 2008.

W. I. Zangwill. *Nonlinear Programming: a Unified Approach*. Prentice-Hall, Englewood-Cliffs, N.J., 1969.

Appendix A. Code

A.1. Discrete Normal Fitting: pDiscrete.R.

```
source("pUtilities.R")

pDiscrete<-function(a,f,eps=1e-10,itmax=100,verbose=TRUE) {
n<-sum(f); p<-f/n; r<-length(f); itel<-1; k<-length(a)
lw<-c(-Inf,a); up<-c(a,Inf); fmax<-sum(xlogx(p))
z<-qbNorm(cumsum(p))
mv<-qr.solve(cbind(1,a),butLast(z))
m<--mv[1]/mv[2]; v<-(1/mv[2])^2
pp<-diff(pnorm(addInf(a),m,sqrt(v)))
fold<-2*(fmax-sum(p*log(pp)))
repeat {
    sm<-sapply(1:r,function(i) msTruncate(m,v,lw[i],up[i]))
    mm<-unlist(sm[1,]); ms<-unlist(sm[2,])
    m<-sum(p*mm); v<-sum(p*ms)+sum(p*(mm-m)^2)
    pp<-diff(pnorm(a,m,sqrt(v)))
    fnew<-2*(fmax-sum(p*log(pp)))
    if (verbose) cat("Iteration: ",formatC(itel,width=3, format="d"),
        " fold: ",formatC(n*fold,digits=8,width=12,format="f"),
        " fnew: ",formatC(n*fnew,digits=8,width=12,format="f"),
        " mean: ",formatC(m,digits=8,width=12,format="f"),
        " vari: ",formatC(v,digits=8,width=12,format="f"),
        "\n")
    if (((fold-fnew) < eps) || (itel == itmax)) break()
    itel<-itel+1; fold<-fnew
    }
return(list(m=m,v=v,f=n*fnew,drf=r-3,p=1-pchisq(n*fnew,r-3)))
}

msTruncate<-function(m,v,a,b)  {
if (!(a<b)) stop("smallest truncation point first")
s<-sqrt(v); aa<-(a-m)/s; bb<-(b-m)/s
da<-dnorm(aa); db<-dnorm(bb)
pa<-pnorm(aa); pb<-pnorm(bb)
r1<-(db-da)/(pb-pa)
if (is.finite(a)) ada<-aa*da else ada<-0
if (is.finite(b)) bdb<-bb*db else bdb<-0
r2<-(bdb-ada)/(pb-pa)
mm<-m-s*r1
vv<-v*abs((1-r2-(r1^2)))
```

```
return(list(m=mm,v=vv))
}
```

A.2. Discrete Normal Examples: pDiscExamp.R.

```
set.seed(12345)
fnorm<-as.vector(table(round(rnorm(1000))))
anorm<-c(-Inf,-2.5,-1.5,-.5,.5,1.5,2.5,Inf)

asmall<-c(-Inf,-1,1,Inf)
fsmall<-c(2,7,1)

fquetelet<-c(28620,11580,13990,14410,11410,8780,5530,3190,2490)
aquetelet<-c(-Inf,1.570,1.597,1.624,1.651,1.678,1.705,1.752,1.759,Inf)

set.seed(12345)
x<-rnorm(1000)
acont<-c(-Inf,seq(-4,4,by=.001),Inf)
fcont<-rep(0,length(acont)-1)
tab<-table(sapply(x,function(z) which.max(z<acont)-1))
fcont[as.integer(names(tab))]<-as.vector(tab)
```

A.3. Probit Regression Fitting: pReg.R.

```
source("pUtilities.R")

pRegres<-function(f,x,a=NULL,eps=1e-6,ops=1e-6,itmax=100,jtmax=1,verouter
     =TRUE,verinner=!verouter,sigeq=FALSE) {
nn<-rowSums(f); p<-f/nn; n<-nrow(f); m<-ncol(f)
itel<-1; k<-length(a)
lw<-a[-k]; up<-a[-1]; fmax<-sum(nn*xlogx(p)); mm<-ms<-f
z<-qbNorm(apply(p,1,cumsum))
mv<-qr.solve(cbind(1,up[-(k-1)]),z[-(k-1),])
mn<--mv[1,]/mv[2,]; vr<-(1/mv[2,])^2
bb<-qr.solve(x,mn); mn<-drop(x%*%bb)
ps<-pnorm(outer(1/sqrt(vr),a,"*")-mn/sqrt(vr))
pp<-t(apply(ps,1,diff))
fold<-2*(fmax-sum(f*log(pp)))
repeat {
    for (i in 1:n) for (j in 1:m) {
        mv<-msTruncate(mn[i],vr[i],lw[j],up[j])
        mm[i,j]<-mv[[1]]; ms[i,j]<-mv[[2]]
        }
```

```
    jtel<-1; finn<-fold
    repeat {
        vr<-rowSums(p*ms)+rowSums(p*(mm-mn)^2)
        bb<-qr.solve(x,rowSums((p*mm)/vr)); mn<-drop(x%*%bb)
        ps<-pnorm(outer(1/sqrt(vr),a,"*")-mn/sqrt(vr))
        pp<-t(apply(ps,1,diff))
        fmid<-2*(fmax-sum(f*log(pp)))
        if (verinner) cat("Iteration: ",paste(formatC(itel,width=3,
            format="d"),letters[jtel],sep=""),
           " finn: ",formatC(finn,digits=8,width=12,format="f"),
           " fmid: ",formatC(fmid,digits=8,width=12,format="f"),
           "\n")
        if (((finn-fmid) < ops) || (jtel == jtmax)) break()
        jtel<-jtel+1; finn<-fmid
        }
    fnew<-fmid
    if (verouter) cat("Iteration: ",formatC(itel,width=3, format="d"),
        " fold: ",formatC(fold,digits=8,width=12,format="f"),
        " fnew: ",formatC(fnew,digits=8,width=12,format="f"),
        "\n")
    if (((fold-fnew) < eps) || (itel == itmax)) break()
    itel<-itel+1; fold<-fnew
    }
return(list(m=mn,v=vr,f=n*fnew,drf=r-3,p=1-pchisq(n*fnew,r-3)))
}

adjustA<-function(f,mn,vr,ktmax=100,pps=1e-6,veradj=TRUE) {
ss<-sqrt(vr); nn<-rowSums(f); n<-nrow(f); m<-ncol(f)
a<-c(-Inf,apply(mn+ss*qnorm(colCums(f/nn))[,-m],2,mean),Inf)
fmax<-sum(nn*xlogx(f/nn)); fold<-Inf; ktel<-1
repeat{
    ps<-pnorm(outer(1/sqrt(vr),a,"*")-mn/ss)
    pp<-colDiff(ps)
    fnew<-2*(fmax-sum(nn*xlogx(pp)))
    aa<-dropFirst(dropLast(a))
    mm<-length(aa)
    ds<-dnorm(outer(1/sqrt(vr),aa,"*")-mn/ss)/ss
    dt<--colDiff(f/pp)
    g<-colSums(ds*dt)
    geps<-max(abs(g))
    if (veradj) cat("Iteration: ",formatC(ktel,width=3, format="d"),
        " grad: ",formatC(geps,digits=8,width=12,format="f"),
        " fold: ",formatC(fold,digits=8,width=12,format="f"),
```

```
        " fnew: ",formatC(fnew,digits=8,width=12,format="f"),
        "\n")
    if ((geps < pps) || (ktel == ktmax)) break()
    h<-matrix(0,mm,mm)
    for (i in 1:n) {
        pn<-pp[i,]; di<-ds[i,]
        v<-diag(1/dropLast(pn)+1/dropFirst(pn))
        w<--1/dropLast(dropFirst(pn))
        upDiag(v)<-w; lwDiag(v)<-w
        h<-h+nn[i]*outer(di,di)*v
        }
    a<-c(-Inf,aa+solve(h,g),Inf)
    fold<-fnew; ktel<-ktel+1
    }
}

pRegInitial<-function(f,x,a=NULL) {
nn<-rowSums(f); p<-f/nn; n<-nrow(f); m<-ncol(f)
zz<-qbNorm(colCums(p)[,-m])
if (!is.null(a)) {
        a<-dropInf(a)
        mv<-qr.solve(cbind(1,a),t(zz))
        mn<--mv[1,]/mv[2,]; vr<-(1/mv[2,])^2
        }
if (is.null(a)) {
        ms<-apply(zz,1,mean)
        zm<-t(apply(zz,1,function(x) x-mean(x)))
        rz<-rankOne(zm); sg<-sign(sum(rz$left))
        ss<-sg/rz$left; a<-sg*rz$right
        mn<--ms*ss; vr<-ss^2
        }
bb<-qr.solve(x,mn); mn<-drop(x%*%bb)
return(m=mn,v=vr,b=bb,a=a)
}
```

A.4. Utilities: pUtilities.R.

```

xlogx<-function(x) ifelse(x==0,0,x*log(x))

qbNorm<-function(x){
z<-qnorm(x)
z[which(x==0)]<--5; z[which(x==1)]<-5
```

```
return(z)
}

rankOne<-function(z) {
sz<-svd(z,nu=1,nv=1)
return(list(left=drop(sz$u),right=drop((sz$v)*(sz$d[1]))))
}

"upDiag<-"<-function(x,p=1,value) {
n<-nrow(x); m<-ncol(x)
if (p > m-1) return(x)
q<-min(n,m-p)-1
x[(p*n+1)+(0:q)*(n+1)]<-value
return(x)
}

"lwDiag<-"<-function(x,p=1,value) {
n<-nrow(x); m<-ncol(x)
if (p > n-1) return(x)
q<-min(m,n-p)-1
x[(p+1)+(0:q)*(n+1)]<-value
return(x)
}

colDiff<-function(x) t(apply(x,1,diff))

colCums<-function(x) t(apply(x,1,cumsum))

dropLast<-function(x,p=1) x[1:(length(x)-p)]

dropFirst<-function(x,p=1) x[-(1:p)]

dropInf<-function(x) x[is.finite(x)]

addInf<-function(x) c(-Inf,x,Inf)
```

Department of Statistics, University of California, Los Angeles, CA 90095-1554

E-mail address, Jan de Leeuw: `deleeuw@stat.ucla.edu`

URL, Jan de Leeuw: `http://gifi.stat.ucla.edu`