**Using an example design from an iTRAQ experiment to illustrate the Fisher’s scoring algorithm**

**Experimental design**

This experiment consists of a completely randomised design with 8 animals and 2 treatments for the first phase, and a 4-by-4 iTRAQ experiment for the second phase.

The following table shows the allocation of disease status (**Con**trol and **Dis**eased) to runs and tags in the iTRAQ experiment. Since each disease status occurs exactly twice in every run and tag, the disease status is orthogonal to both runs and tags.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run | Tag | | | |
| 114 | 115 | 116 | 117 |
| 1 | Con | Con | Dis | Dis |
| 2 | Dis | Dis | Con | Con |
| 3 | Dis | Dis | Con | Con |
| 4 | Con | Con | Dis | Dis |

The following table shows the allocation of animals (1 to 8) to runs and tags. For this design, Runs 1 and 2 contain Animals 1 to 4; Runs 3 and 4 contain Animals 5 to 8, hence animals are not orthogonal to runs. Similarly, Tags 114 and 116 contain Animals 1, 3, 5 and 7 and Tags 115 and 117 contain Animals 2, 4, 6, and 8, hence the animals are also not orthogonal to tags.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run | Tag | | | |
| 114 | 115 | 116 | 117 |
| 1 | 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 1 | 2 |
| 3 | 5 | 6 | 7 | 8 |
| 4 | 7 | 8 | 5 | 6 |

**Linear model**

Let denote the abundance of a nominal protein in the proteomic sample from animal with disease status labelled with iTRAQ tag assayed in run . The linear model for the above design can then be written as

where µ denotes the overall mean abundance of the nominal protein, τi and γj denote the fixed effects of disease status *i* and tag *j*, respectively; Rk, Al and εijkl denote the random effects of run *k*, animal *l* and measurement error, respectively. These random effects are assumed to be mutually uncorrelated and normally distributed with mean zero and variances , and .

**ANOVA table**

The following table shows the theoretical ANOVA, with the expected mean square (EMS) corresponding to the above design. DF denotes the degrees of freedom. The th mean square (MS), denoted by (i = 1,…, 4), is the estimate of the th pure error EMS, denoted by (i = 1,…, 4), where here pure error is used to refer to EMS which contain only those variance components associated with random effects. Define

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Source of variation** | **DF** | **MS** | **Pure error of EMS** | **EMS** |
| Between Run |  |  |  |  |
| Between Animal | 1 |  |  |  |
| Residual | 2 |  |  |  |
| Within Run |  |  |  |  |
| Between Animal |  |  |  |  |
| Disease status | 1 |  |  |  |
| Tag | 1 |  |  |  |
| Residual | 4 |  |  |  |
| Within Animal |  |  |  |  |
| Tag | 2 |  |  |  |
| Residual | 4 |  |  |  |

**Fisher’s scoring algorithm**

Here, we attempt to illustrate the estimation of the variance components in using Fisher’s scoring algorithm, an iterative procedure which can be used to solve maximum likelihood equations. The algorithm stops when the difference between the variance component estimates from two consecutive iterations is less than 1e-7.

The formula for Fisher’s scoring algorithm can be written as

,

where and are vector of variance component estimates at the *t*th and (*t+*1)th iterations, , is the inverse of the Fisher information matrix and is the score function. Therefore, in order to use the Fisher scoring algorithm, we need to define the score function and Fisher information matrix.

**Constructing the score function and Fisher information matrix**

The are assumed to have a chi-square distribution, i.e.

where is the DF of the corresponding to . The log-likelihood function, L, of the can then be shown to be

.

It follows that the score function is a 4 × 1 vector with th element given by

.

The second partial derivative of the log-likelihood function is given by the 4 × 4 matrix

where the *i*th diagonal element is given by

,

which has expectation

.

The off-diagonal elements of matrix are all zero, i.e.

It now follows that the Fisher information matrix, defined as the expectation of the negative of the second derivative of the log-likelihood function, is given by

.

**Transformation from to**

The score function and Fisher information matrix, as defined above, are functions of the EMS, , which cannot be applied in the Fisher’s scoring algorithm to estimate Hence, we need to transform the score function and the Fisher information matrix so that both are functions of (i.e. the vector of VCs) rather than from with respect to . This change of variables be achieved by applying the chain rule, i.e.

where

.

By the product rule, it follows

multiplying both the score function and Fisher information matrix by . This technique is also known as *change of variables* or *chain rule*.

For this experiment, the transformation can be achieved by using a 4×3 matrix, where the columns correspond to the elements of and the rows correspond the elements of This matrix, denoted by G matrix, can be written as

.

As result of constructing the G matrix, we can recognise that the G matrix contains the coefficients of the variance components of EMS of the theoretical ANOVA table.

**Summary of steps in estimating**

The estimation of consists of three steps. The first step is to transform the estimation of to by using the G matrix as shown

.

The second step is to use the from above, from the experimental data and from the experimental design to construct the score function with respect to. This score function is then transform to with respect to by pre-multiplying the G matrix and shown to be

.

The Fisher information matrix with respect to is constructed using and. The Fisher information matrix is then transformed to with respect to by pre-and post-multiplying the G matrix and shown to be

.

The last step is estimating the by applying the Fisher’s scoring algorithm, i.e.

.

These three steps are then iterated until the differences between the variance component estimates in of two consecutive iterations is less than 1e-7. The at the last iteration is the variance component estimates of the Fisher’s scoring algorithm.

**Pseudo code of simulation and Fisher scoring algorithm for estimating the variance components**

For this case, 10000 simulated datasets are generated. The variance component estimates are obtained for each simulated data. The means of the 10000 sets of variance components estimates are then computed.

reml.VC = matrix(0, nrow=1, ncol = 3) # matrix used to store the variance component estimates from each simulated data set

Repeat 10000 times{

#Simulate a single dataset based on the linear model.

VC.base = variance component of the measure error.

VC.animal = variance component of the animal effects.

VC.run = variance component of the run effects.

Simulated dataset = N(0, VC.base) + N(0, VC. animal) + N(0, VC.run)

#Construct the theoretical ANOVA table based on the experimental design.

G = a matrix consists of coefficients of the variance components obtained from the theoretical ANOVA table.

DF = vector of degrees of freedom of the corresponding mean square based on the experimental design

#Perform ANOVA on the simulated data.

MS = vector of mean squares from ANOVA based the simulated dataset.

EMS = vector of expected mean squares compute by pre-multiplying the current variance component estimates by the G matrix.

newV = c(VC.base , VC.animal, VC.run) # Vector of current variance component estimates. Initialise VCs to their true values, i.e. values used to simulate the dataset

oldV = c(0, 0, 0) # Vector of previous variance component estimates. Initialise all VCs to zero

counter <- 1 # Initialise counter

#the convergence tolerance is the differences between the current variance component estimates and the previous variance component estimates. This differences should be less than 1e-7

while((newV – oldV) >1e-7){

oldV = NewV

EMS = G × oldV

score function =

information matrix =

newV = oldV + (information matrix)-1 × (score function)

if ( counter > 1000 or information matrix is invertible)

stop the iteration of the while loop and start a brand new simulation dataset

counter = counter +1

} #end of while((newV – oldV) >1e-7)

reml.VC = rbind(reml.VC, newV) #store the estimates into a matrix

} #end of repeat 10000 times

apply(reml.VC, 2, mean) #each variance components estimates, i.e. , are then obtained from the means of the variance components estimates from the 10000 simulated datasets