Paper ST-13

Linear Logistic Test Model: Using SAS® to Simulate the Decomposition of Item Difficulty by Algorithm, Sample Size, Cognitive Component and Time to Convergence

George MacDonald, University of South Florida, Tampa, FL
Jeffrey Kromrey, University of South Florida, Tampa, FL

ABSTRACT

The Linear Logistic Test Model (LLTM) is capable of bridging cognitive processing models and psychometric models. A simulation study was undertaken to determine the utility of the algorithm methods available in PROC NLMIXED according to varying sample sizes, and number of cognitive components. The results of the simulations are interpreted and their significance and implications for LLTM applications are discussed. Guidance for the use of PROC NLMIXED for the estimation of LLTM and suggestions for future research will be highlighted.

Keywords: Linear Logistic Test Model, LLTM, PROC NLMIXED, Cognitive Diagnostic Assessment

LINEAR LOGISTIC TEST MODEL

Fischer (1973) introduced a model, called the linear logistic test model (LLTM) that is capable of bridging cognitive processing models and psychometric models. In his mathematics study, Fischer found that differentiating calculus items could be explained by eight basic cognitive operations that the examinee must implement. He postulated that item difficulty could be re-parameterized to express these operations. In that, the linear logistic test model could be used to predict the probability that the person passes a particular item from that person's ability and a cognitive processing model of item difficulty. Since Fischer built his theories on the Rasch model, LLTM is often understood to be an extension of the Rasch IRT model (Embretson & Reise, 2000). But formally speaking, the LLTM multiplicatively joins a cognitive processing matrix to Rasch's item difficulty. In that, the item difficulty parameters are multiplied by the weight of the conceptual components. The result is a combination of the information found in Rasch's a_j, Matrix, and Fischer's w_{ij} Matrix.

The Rasch model, a well-known IRT model, can be represented as:

$$\mathrm{P}(+|S_v, I_i) = \frac{exp(\theta_v - \beta_i)}{1 + exp(\theta_v - \beta_i)}$$

Where $\mathrm{P}(+|S_v, I_i)$ is the probability of examinee S_v responding to item I_i correctly;
θ_v is S_v's ability; and
β_i is I_i difficulty parameter.

When conceptual components are multiplicatively joined to the matrix structure the formulation becomes:

$$\beta_i = \sum_{j=1}^{p} w_{ij} a_j$$

Where $a_j, \mathrm{j} = 1, \ldots, \mathrm{p},$ for the basic parameters of the model, measuring the contribution of cognitive operations to the difficulty of item I_i;
w_{ij} for the given weight of a_j with respect to the difficulty of item I_i.

COGNITIVE MODELS

Since construct validity was introduced by Cronbach and Meehl (1955), this concept has guided most test research programs (Embretson & Gorin, 2001). According to Cronbach and Meehl's traditional conceptualization of construct

validity, a primary source of data for test meaning is the correlation of test scores with other measures, particularly other tests and criterion assessment (Embretson, 1995). The use of a cognitive model, with an information-processing perspective, has added a new direction to the nature of construct validity.
As an example, a researcher might specify a cognitive model for fraction items on a test by conducting cognitive think-alouds with approximately twelve grade 3-6 students who took the test. A second step might include gathering the responses and determining similarities and differences in student responses. A third step might include a team of content experts consulting to determine the concepts employed by the students as they solved the fraction items. A final step might include a group of content experts utilizing the results of the cognitive think-aloud to determine which of the concepts were present in each item on the test. In this way, the $w_{ij}a_j$ matrix would be constructed by question for each concept; 0 for absence and a 1 for presence of the cognitive concept.

The estimation of the parameters for the LLTM may be readily accomplished with SAS software, using PROC NLMIXED. However, a variety of algorithms are available in this exceptionally flexible procedure.

ALGORITHMS AVAILABLE IN PROC NLMIXED

LLTM can be coded in SAS using PROC NLMIXED. It is worth noting that clustering items within persons is considered by many to be a multi-level approach. This procedure offers the programmer a variety of algorithms for parameter estimation: the Trust Region Optimization (TRUREG) algorithm; the Newton-Raphson Optimization algorithm with Line Search (NEWRAP); the Newton-Raphson Ridge Optimization algorithm (NRRIDG); the Quasi-Newton Optimization (QUANEW) algorithm; the Double-Dogleg Optimization (DBLDOG) algorithm; the Conjugate Gradient Optimization (CONGRA) algorithm; and the Nelder-Mead Simplex Optimization (NMSIMP) algorithm. From within these optimization algorithms PROC NLMIXED makes various updates available. QUANEW offers DBFGS which performs the dual Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the Cholesky factor of the Hessian matrix; DDFP which performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix; BFGS which performs the original BFGS update of the inverse Hessian matrix; and DFP which performs the original DFP update of the inverse Hessian matrix. DBLDOG has two updates available: DBFGS performs the dual Broyden, Fletcher, Goldfarb, and Shanno update; and DDFP performs the dual Davidon, Fletcher, and Powell update. Finally, CONGRA offers PB which performs the automatic restart update method of Powell (1977) and Beale (1972); FR which performs the Fletcher-Reeves update (Fletcher 1987); PR which performs the Polak-Ribiere update (Fletcher 1987); and CD which performs a conjugate-descent update of Fletcher (1987).

The SAS/STAT 9.2 Users Guide suggests that the second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems where the Hessian matrix is not expensive to compute. However, the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The (dual) quasi-Newton method does not need to compute second order derivatives since they are approximated. It works well for medium to moderately large optimization problems where the objective function and the gradient are much faster to compute than the Hessian. The CONGRA algorithm can be expensive in function and gradient calls, but each of the CONGRA iterations is computationally cheap. Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The double-dogleg optimization technique works well for medium to moderately large optimization problems, but generally requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques.

Given this array of optimization algorithms, questions arise about the functioning of an Item Response Theoretical Model with Cognitive Information multiplicatively added through the w matrix. Further, no algorithm for optimizing general nonlinear functions exists that always finds the global optimum. Therefore, coders may well want to know which algorithms work best under various test conditions.

THE SIMULATION STUDY

To provide some answers to these types of questions, the authors employed the algorithm methods and techniques specified above as the basis for a simulation study. The study examined the parameter estimates for $w_{ij}a_j$ given; sample sizes starting with 20 observations and doubling to 2,560 observations; a 20 item test; three, five, and seven cognitive components; and, 1085 replications per simulation. The researchers used cognitive components decomposed in an earlier study. The authors chose 1,085 replications based on table values (Robey, & Barcikowski, 1992) for the liberal criterion of $\alpha \pm \frac{1}{2}\alpha$, 1-β=.8, and ω=.01.

To help evaluate the estimated $w_{ij}a_j$ parameters, the authors computed statistical bias, the root mean square error (RMSE), estimated confidence interval coverage probabilities, and the mean confidence interval width. The bias statistic informs researchers about how close the average $w_{ij}a_j$ estimated parameter is to the original specified value for $w_{ij}a_j$; the RMSE will help to quantify the typical difference between the true and estimated value of $w_{ij}a_j$; the confidence interval coverage will indicate the degree to which the sample confidence intervals for the parameters are

accurate; and the confidence interval width will inform the researchers about the precision of the interval estimates. Finally, the time to convergence for each simulation is reported as an indicator of efficiency or cost effectiveness of the various algorithms.

SAS CODE

The SAS code used for the simulation study is described in detail below. The complete SAS code is available from the first author.

```
Line 1:     proc iml;
Line 2:     N_reps = 1085;
Line 3;     N_persons = 1280;
Line 4:     N_items = 20;
```

Comments:
Line 1 begins the LLTM Simulation by invoking PROC IML.
Line 2 specifies the number of samples that are to be generated. In this case 1,085 was chosen per Robeyand Barcikowski (1992); 1000 replications is commonly used.
Line 3 specifies the number of persons to be simulated in the sample. For this simulation we used sample sizes of 20, 40, 80, 160, 320, 640, 1280 and 2560.
Line 4 specifies the number of items found in the test. In this case there are 20 items being simulated in the study.

```
Line 5:     seed1=round(1000000*ranuni(0));
Line 6:     call randseed(seed1);
```

Comments:
Line 5 initializes the seed for the random number generator and randomly generates a seed.
Line 6 calls the seed to be randomly generated.

```
 * 20 item Guttman matrix with 5 components;
Line 7:     A = J(N_items,5,0);
Line 8            A[,1] = 1;
Line 9            c2 = {2 3 4 6 7 8 10 11 12 14 15 16 18 19 20};
Line 10           c3 = {3 4 7 8 11 12 15 16 19 20};
Line 11           c4 = {4 8 12 16 20};
Line 12           c5 = {4 5 6 12 13 15 17};

Line 13           do i = 1 to ncol(c2);
Line 14                 A[c2[1,i],2] = 1;
Line 15           end;

Line 16           do i = 1 to ncol(c3);
Line 17                 A[c3[1,i],3] = 1;
Line 18           end;

Line 19           do i = 1 to ncol(c4);
Line 20                 A[c4[1,i],4] = 1;
Line 21           end;

Line 22           do i = 1 to ncol(c5);
Line 23                 A[c5[1,i],5] = 1;
Line 24           end;

Line 25           *print 'Guttman Matrix:' A;
```

Comments:

Lines 7 initializes a 20 row by 5 column matrix. The 20 rows can be thought of as items on a test, and the columns can be thought of as conceptual knowledge states (CKS). This simulation was run for 3 CKS's, 5 CKS's and 7 CKS's but only the 5 CKS model is shown.
Line 8 initializes the CKS in column 1 with a full rank of 1s.
Lines 9 to 12 are assignment statements that create row vectors contain the numbers of the items that are associated with each CKS.
Lines 13-15, 16-18, 19-21, and 24-25 are do loops which transform the zeros in CKS2 to CKS5 to 1s as defined in lines 9 through 12.
Line 25 offers a print option for the coder who wishes to check to see if their matrix is properly constructed. This line can be turned on and off as needed with an asterisk. When running the full simulation it is advised the coder turn this option off as it will fill the output window with unnecessary output.

```
Line 26     B = {.50, .75, .22, .85, .35};

Line 27     *print 'Parameters for Cognitive Components:' B;
Line 28     ccname = {"TrueB"};
Line 29     create TrueP from B [colname = ccname];
Line 30     append from B;
```

Comments:
Line 26 initializes a one column matrix filled with the five numeric values found inside the brackets. These values for the B Matrix were chosen at random between 0 and 1.
Line 27 offers the coder a print option to check to see if the matrix is properly constructed and should be turned off during the simulation.
Line 28-30 sends the parameters from the B Matrix to regular SAS where they will be employed in the calculation of bias, RMSE, and confidence interval coverage. The power and flexibility of PROC IML can be seen in the creation of both the A and B matrices.

```
Line 31     Beta = A*B;
Line 32     *print 'Item Difficulties:' Beta;
```

Comments:
Line 31 creates a Beta (Item Difficulty) vector by multiplying Matrix A by Matrix B.
Lines 32 allows the coder to check the accuracy of the Item Difficulty vector. This option should be turned off during the simulation.

```
Line 33     do replication = 1 to N_reps;
Line 34     theta = J(N_persons,1);
Line 35     call randgen(theta, 'NORMAL');
Line 36     *print 'True Sample Abilities:' theta;
```

Comments:
Lines 34 and 35 generate person abilities N(0,1) and item responses. For each replication True Sample Abilities are randomly generated.
Line 36 allows the coder to check the theta's that are generated.

```
Line 37     item_matrix = J(N_persons, N_items,0);
Line 38     ranvec = J(1, N_items,0);
Line 39           do row = 1 to N_persons;

Line 40           call randgen(ranvec, 'UNIFORM');
Line 41     do col = 1 to N_items;
Line 42           prob = (exp(theta[row,1] - beta[col,1])) /
                  (1 + (exp(theta[row,1] - beta[col,1])));
Line 43     if (prob > ranvec[1,col]) then item_matrix[row,col] = 1;
Line 44           end;
Line 45     end;
Line 46     *print item_matrix;
```

Comments
Lines 37 to 45 generate an item matrix, which in this case is a 1280 (persons) by 20 (items).
Line 41 generates a vector of uniform random numbers to compare with the probability of a correct response for each item;
Line 46 allows the coder to check the item matrix.

```
Line 47     do person = 1 to N_persons;
Line 48     score1 = item_matrix[person,]`;
Line 49     person_label = repeat(person,N_items);
Line 50     rep_label = repeat(replication,N_items);
Line 51     to_SAS = rep_label||person_label||score1||A;
Line 52     if (replication = 1 & person = 1) then do;
Line 53     cname = {"Replication" "Person" "Score1" "a1" "a2" "a3" "a4"
"a5"};
Line 54     create combine from to_SAS [colname = cname];
Line 55     append from to_SAS;
Line 56          free to_SAS;
Line 57     end;
Line 58     if (replication > 1 | person > 1) then do;
Line 59               setout combine;
Line 60                    append from to_SAS;
Line 61                    free to_SAS;
Line 62               end;
Line 63          end;
Line 64     end;
Line 65     quit;
```

Comments:
Lines 47 to 65 send the generated data to regular SAS for analysis

```
Line 66     filename junk dummy;
Line 67     proc printto print = junk;
```

Comments:
Lines 66 and 67 will turn off standard printed output from PROC NLMIXED

```
Line 68     Proc NLmixed data=combine tech=newrap;
Line 69     by replication;
Line 70     Parms b1-b5=0 sd0=1;
Line 71     beta= b1*a1+b2*a2+b3*a3+b4*a4+b5*a5;
Line 72     ex=exp(theta-beta);
Line 73     p=ex/(1+ex);
Line 74     model score1 ~ binary(p);
Line 75     Random theta ~ normal(0,sd0**2)subject=person;
Line 76     Estimate 'sd0**2' sd0**2;
Line 77     ods output Parameters = Parms ParameterEstimates = ParmEst
            AdditionalEstimates = AddEst;
```

Comments:
Line 68 implements the PROC NLMIXED procedure using the data set combine. In this case the Newton-Raphson (newrap) technique is being used. As is pointed out in the SAS/Stat 9.2 Users Guidethese techniques are iterative and require the repeated computation of the function value (optimization criterion); the gradient vector (first-order partial derivatives); and for some techniques the approximate Hessian matrix (second-order partial derivatives). In particular: TRUREG, NEWRAP, and NRRIDG employ both first-order derivatives and second-order derivatives; QAUNEW, DBLDOG, and CONGRA uses only first-order derivatives; while NMSIMP does employs a no-derivative method.

Line 70 initializes the parameters b1 to b5, setting their initial values to 0 and initializes the standard deviation and sets its initial value to 1
Lines 71 to 75 build the DeBoeck and Wilson (2004) variant of the LLTM. It is important to note that Fischer originally specified LLTM using conditional maximum likelihood whereas DeBoeck and Wilson employed marginal maximum likelihood as available in PROC NLMIXED. One reason for this simulation study was to test the performance of marginal maximum likelihood as applied to the LLTM.
Line 71 defines Beta.
Line 75 defines theta and assigns it a normal distribution with a mean of 0 and a standard deviation of 1.
Line 76 estimates the variance.
Line 77 employs the SAS ODS output to create SAS data sets from the specified ODS tables.

```
Line 78     proc printto print = print;
```

Comments:
Line 78 turns printing back on.

```
Line 79     proc print data = Parms;
Line 80           title 'Contents of ODS Parameters File';
Line 81     proc print data = ParmEst;
Line 82           title 'Contents of ODS ParameterEstimates File';
Line 82     proc print data = AddEst;
Line 83           title 'Contents of ODS AdditionalEstimates File';
```

Comments:
Lines 79 to 83 are straight forward proc prints;

```
Line 84     data TrueP;
Line 85      set TrueP;
Line 86     if _N_ = 1 then parameter = 'b1';
Line 87      if _N_ = 2 then parameter = 'b2';
Line 88      if _N_ = 3 then parameter = 'b3';
Line 89      if _N_ = 4 then parameter = 'b4';
Line 90      if _N_ = 5 then parameter = 'b5';

Line 91     proc print data = TrueP;

Line 92     proc sort data = ParmEst;
Line 93           by parameter;

Line 94     data ParmEst2;
Line 95     merge ParmEst TrueP;
Line 96     by parameter;
Line 97      if parameter ne 'sd0';
Line 98     deviation = estimate - TrueB;
Line 99     dev_square = deviation**2;
Line 100    CICoverage = 0;
Line 101    if trueB < upper and trueB > lower then CICoverage = 1;
Line 102    CIWidth = upper - lower;

Line 103    proc means noprint data = ParmEst2;
Line 104    var deviation dev_square CICoverage CIWidth;
Line 105    by parameter;
Line 106    output out = final mean = bias rmse CICoverage CIWidth n =
n_samples;

Line 107    data final;
Line 108     set final;
```

```
Line 109     rmse = SQRT(RMSE);
Line 110    proc print data = final;
Line 111     title 'Estimates of Bias and RMSE for Parameter Estimates, CI
Coverage and Width';

Line 112    run;
```

Lines 84 to 112 provide the SAS code to combine the output files from NLMIXED with the file containing the parameter values used in the simulation. The differences between the sample estimates and the parameter values are used to compute statistical bias, RMSE, confidence interval coverage and confidence interval width. These values are printed to the output window.

SIMULATION RESULTS

Results for the estimated statistical bias are provided in Figure 1 and confidence interval coverage estimates are provided in Figure 2. Except for the smallest samples examined, the bias was near zero for each of the algorithms investigated. Similarly, the confidence interval coverage results show that coverage was near the nominal (95%) level for all algorithms and all sample sizes.

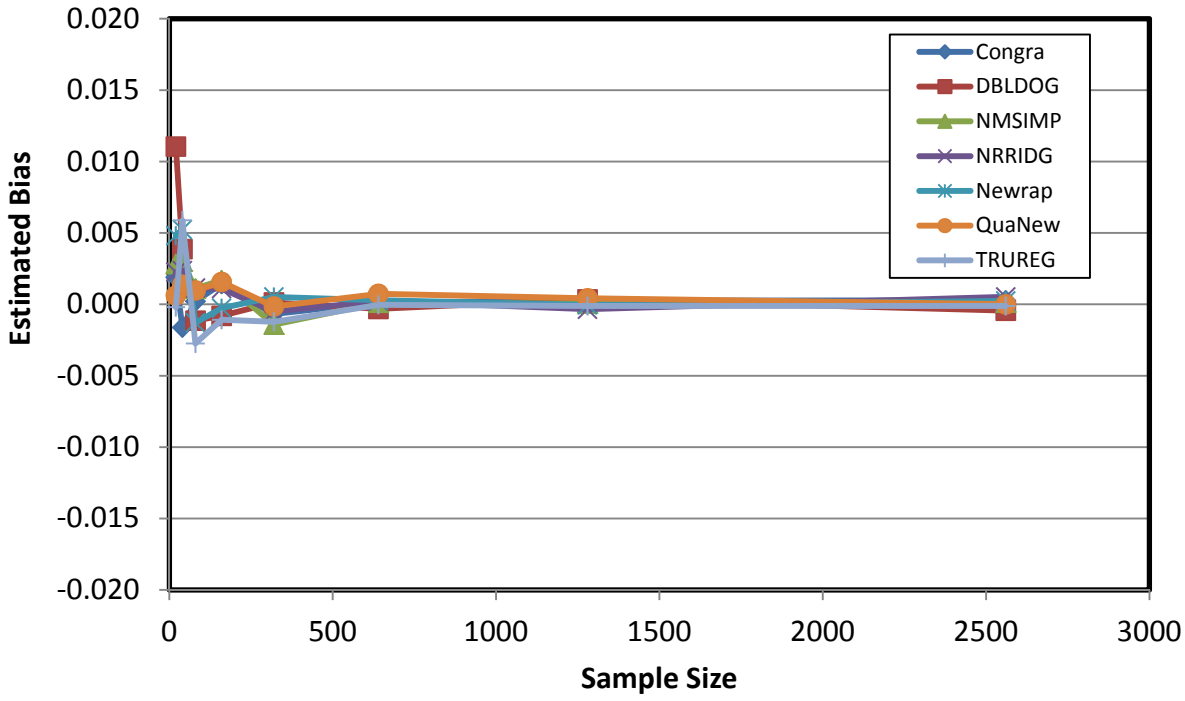

Figure 1. Estimated statistical bias in parameter estimates by sample size

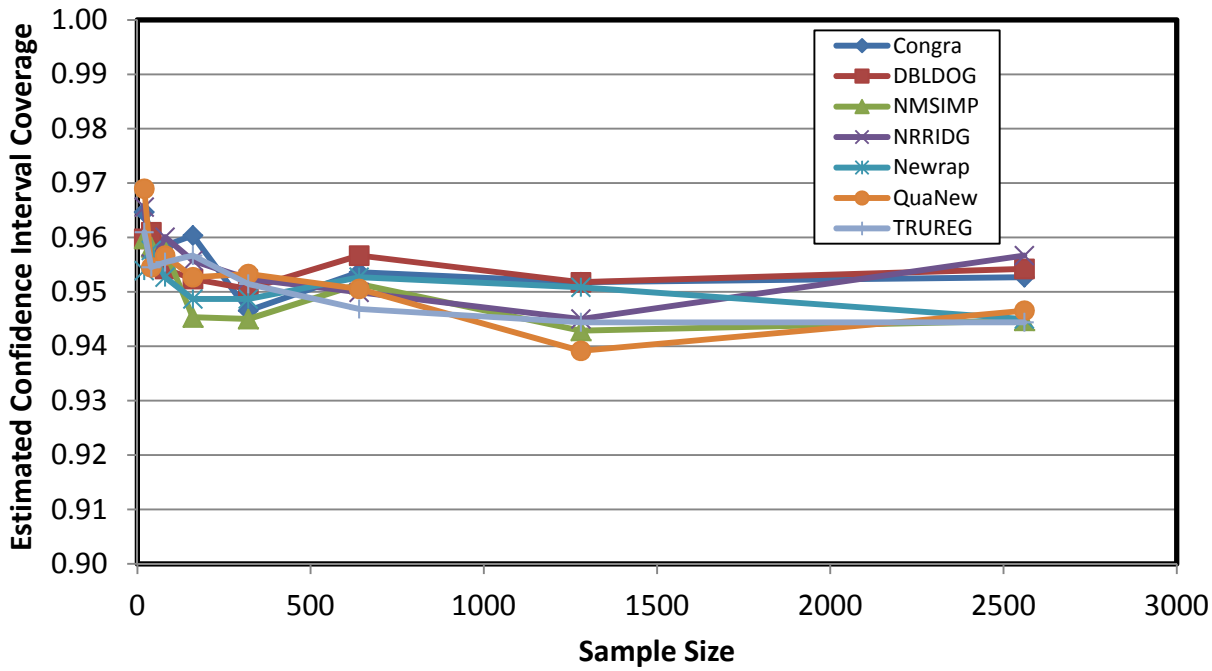

Figure 2. Estimated parameter confidence interval coverage by sample size

The estimated RMSE (Figure 3) and average confidence interval width (Figure 4) results show how sampling error decreases rapidly with increasing sample size. The differences among the algorithms are negligible until sample sizes get very large. At the largest sample sizes examined in the simulation study, the TRUEREG algorithm led to parameter estimates with slightly more sampling error than the other algorithms.

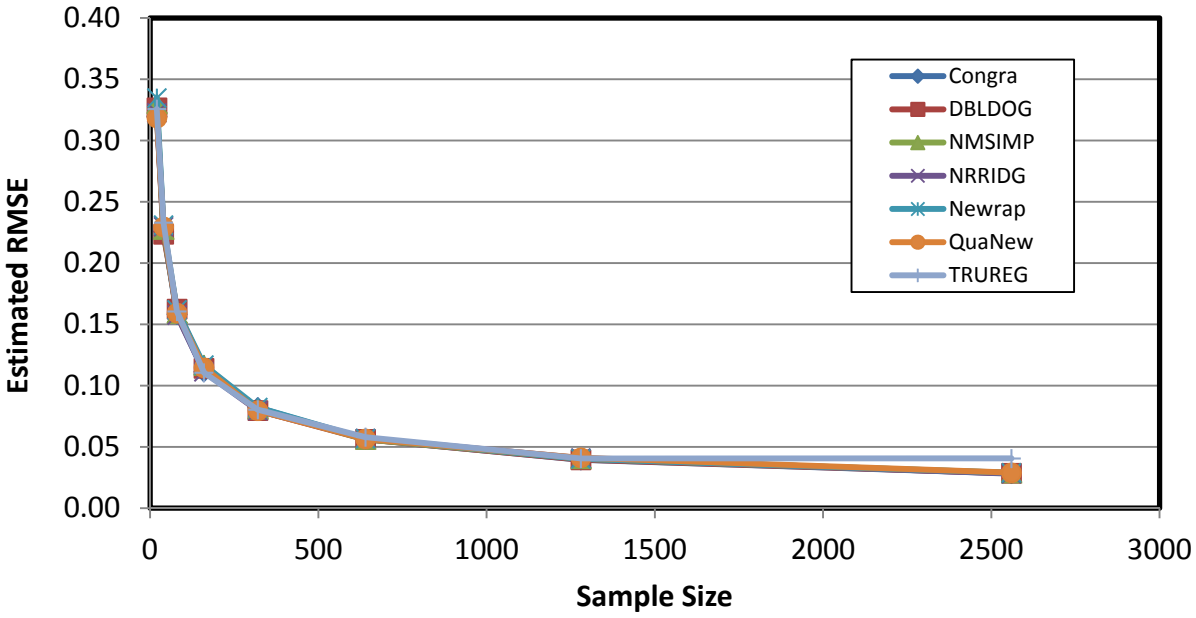

Figure 3. Estimated RMSE by sample size

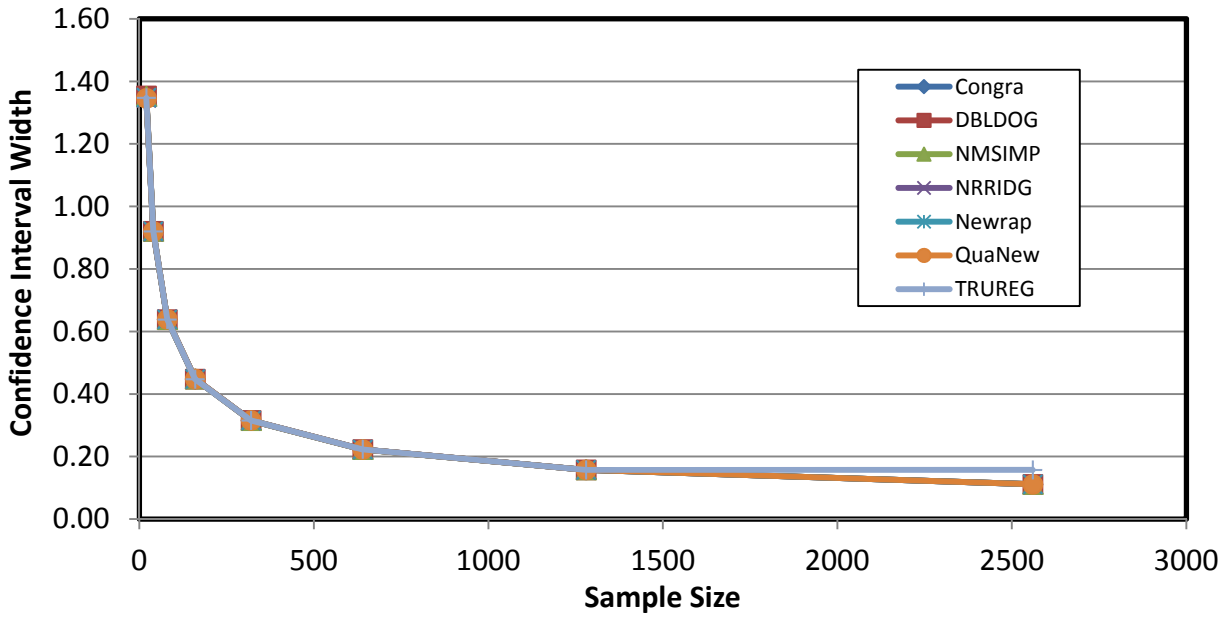

Figure 4. Estimated confidence interval width by sample size

The time to convergence for the optimization techniques which require first-order derivatives and second-order derivatives (TRUREG, NEWRAP, and NRRIDG), and the optimization techniques which require first-order derivatives only (QUANEW, DBLDOG, and CONGRA) are not computationally expensive. However, the optimization technique (NMSIMP) which does not use derivatives is computationally expensive (Figure 5).

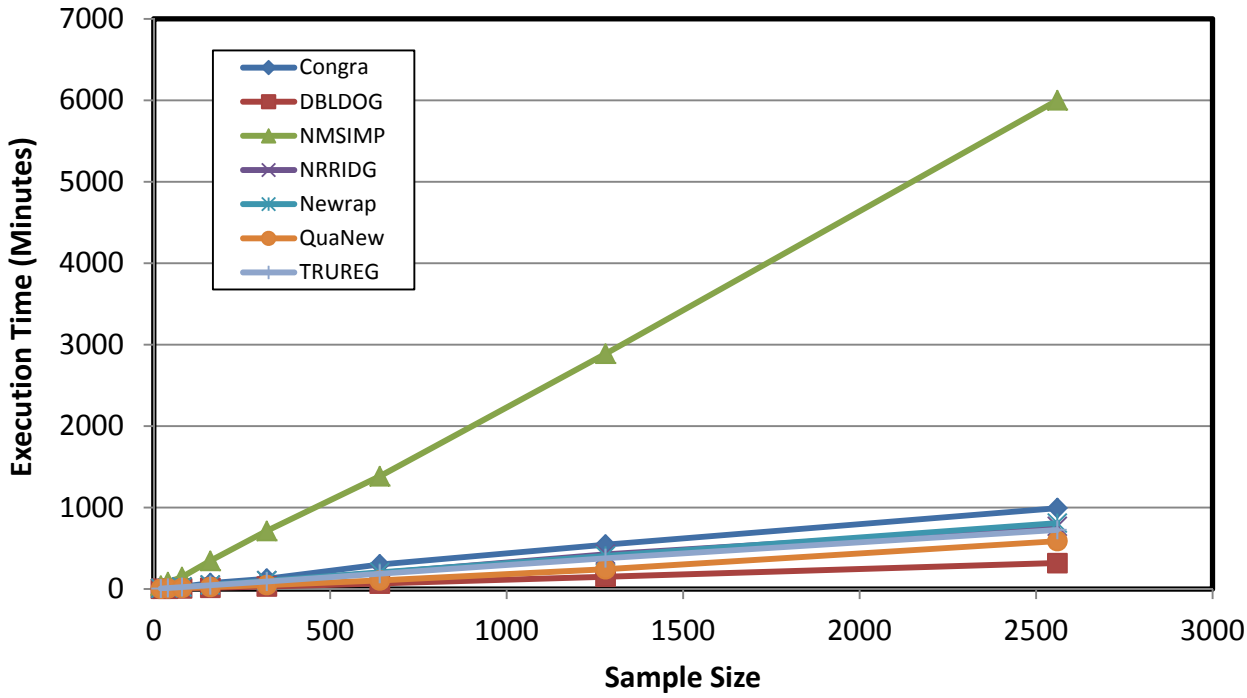

Figure 5. Standardized execution time by sample size

CONCLUSIONS AND IMPLICATIONS

The simulation provides evidence that the DeBoeck and Wilson variant of the LLTM employing marginal maximum likelihood works very, very well. The bias statistic is designed to inform the researchers about how close the $w_{ij}a_j$ estimated statistic is to the original specified value for $w_{ij}a_j$. The RMSE helps to quantify the typical difference between the true and estimated value of $w_{ij}a_j$; the CI Coverage statistic indicates the degree to which the confidence interval is accurate; the CI Width statistic informs the researcher about the precision of the interval estimate; and the time to convergence indicates the efficiency or cost effectiveness of the various algorithms.

The differences in parameter estimates obtained with the seven algorithms were negligible and there was no non-convergence in any samples with any algorithm. Bias was negligible and confidence intervals were very accurate. Confidence intervals, as expected, were very wide with small sample sizes and the RMSE was large in these conditions.

It should be noted that in the simulation world, the cognitive components in the model exactly matched the components used in data generation. In the real world, the amount of model misspecification that LLTM can tolerate and still function adequately is unknown. This is a rich area for future research. This simulation has shown that PROC NLMIXED provides accurate parameter estimates for the LLTM models. Used in combination with PROC IML, SAS provides an exceptionally powerful environment for simulation research related to these models.

REFERENCES

Baker, F. B. (1993). Sensitivity of the linear logistic test model to misspecification of the weight matrix. *Applied Psychological Measurement, 17*(3), 201-210.

Bechger, T. M., Verstralen, H. H. F. M., & Verhelst, N. D. (2002). Equivalent linear logistic test models. *Psychometrika, 67*(1), 123-136.

Bechger, T. M., Verstralen, H. H. F. M., Verhelst, N. D., & Maris, G. (2004). Equivalent lltms: A rejoinder. *Psychometrika, 69*(2), 317-318.

Cassuto, N. Y. (1997). The performance of the linear logistic test model under different testing conditions. ProQuest Information & Learning). *Dissertation Abstracts International: Section B: The Sciences and Engineering, 57* (11), 7270-7270.

Cronbach, L. J., & Meehl, P. E. (1955). Construct validity in psychological tests. *Psychological Bulletin, 5*2, 281-302.

DeBoeck, P., & Wilson, M. (2004). *Statistics for Social Science and Public Policy: Explanatory Item Response Models*. Springer-Verlag: New York, New York.

Dimitrov, D. M. (1996). Cognitive item subordinations in linear logistic test modeling. ProQuest Information & Learning). *Dissertation Abstracts International Section A: Humanities and Social Sciences, 57* (6), 2452-2452.

Diones, R. E. (1995). An integration of cognitive theory and psychometrics: Analogical reasoning. ProQuest Information & Learning). *Dissertation Abstracts International Section A: Humanities and Social Sciences, 55* (11), 3484-3484.

Embretson, S. (1984). A general latent trait model for response processes. *Psychometrika, 49*(2), 175-186.

Embretson S. E. (1995). Developments toward a cognitive design system for psychological tests. In D. Lubinski & R.V. Dawis (Eds.), *Assessing individual differences in human behavior: New concepts, methods, and findings*. Palo Alto,CA: Davies-Black Publishing.

Embretson, S. E. (1998). A cognitive design system approach to generating valid tests: Application to abstract reasoning. *Psychological Methods, 3,* 380-396.

Embretson S. E. (1994). Application of cognitive design systems to test development. In C. R. Reynolds (Ed.), *Cognitive assessment: A multidisciplinary perspective*. New York: Plenum Press.

Embretson, S. E., & Gorin, J. S. (2001). Improving construct validity with cognitive psychology principles. *Journal of Educational Measurement, 38,* 343-368.

Fischer, G. H. (2004). Remarks on 'equivalent linear logistic test models' by bechger, verstralen, and verhelst (2002). *Psychometrika, 69*(2), 305-315.

Fischer, G. H. (1973). Linear logistic test model as an instrument in educational research. *Acta Psychologica, 3*7, 359-374.

Medina-Díaz, M. (1993). Analysis of cognitive structure using the linear logistic test model and quadratic assignment. *Applied Psychological Measurement, 17*(2), 117-130.

Obiekwe, J. C. (1999). Application and validation of the linear logistic test model for item difficulty prediction in the context of mathematics problems. ProQuest Information & Learning. *Dissertation Abstracts International: Section B: The Sciences and Engineering, 60* (2), 0851-0851.

Robey, R., & Barcikowski, R. (1992). Type 1 error and the number of iterations in Monte Carol studies of robustness. *British Journal of Mathematical and Statistical Psychology, 45,* 283-288.

Tanzer, N. K., Gittler, G., & Ellis, B. B. (1995). Cross-cultural validation of item complexity in a LLTM--calibrated spatial ability test. *European Journal of Psychological Assessment, 11*(3), 170-183.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the first author at:

George MacDonald
University of South Florida
4202 East Fowler Ave., EDU 105
Tampa, FL 33620

Work Phone: (813) 974-5959
Fax: (813) 974-6126
Email: gmacdona@usf.edu