Automating the construction and evaluation of eighteen-treatment semi-Latin squares of order six in blocks of size three

Polycarp E. Chigbu, Emmanuel I. Mba* and Eugene C. Ukaegbu

Department of Statistics, University of Nigeria, Nsukka, Nigeria

A Microsoft Excel program that has the capacity of constructing $(6\times 6)/3$ *semi-Latin squares from* $2^6.3.7^2 = 9408$ *reduced Latin squares and the seven distinct* $(6\times 6)/2$ *semi-Latin squares (SLS)of Bailey and Royle (1997) is here presented. The aim is to facilitate, through the knowledge of computer programming, the construction and statistical evaluation of the eighteen-treatment semi-Latin squares of order six and block size three. The eighteen-treatment semi-Latin squares of order six in blocks of size three are constructed by applying the method of superposition. The program also computes and displays the incidence matrices of all the constructed SLS as well as the, A-, D-, E- and MV-efficiency criteria for evaluating the usefulness of the design for experimentation. The program has a size of 20 kilobytes on hard disk space and takes about thirty minutes of computing time to construct and evaluate the semi-Latin squares. Illustrative examples were given to demonstrate the performance of the program.*

Keywords: canonical efficiency factors; incidence matrix; information matrix; simple contrasts; superposition

1 Introduction

An $(n\times n)/k$ semi-Latin square with n rows and n columns is defined as an arrangement ofnk treatments in n^2 blocks in such a way that each row-column intersection contains k treatments and each treatment occurs once in each row and once in each column: see Bailey (1992) and Soicher (2013a).Bailey (1988) listed some of the areas of applications of the semi-Latin squares such as agriculture, consumer testing, etc. as well as methods of construction of semi-Latin squares such as inflation, superposition, Trojan, etc., most of which are special cases of superposition.

Several attempts had been made to enumerate the $(n\times n)/k$ (for some values of n and k) semi-Latin squares with the aid of computer packages such as the *nauty* package by McKay (1981), GRAPE package for GAP by Soicher (1993) and *Traces* package by McKay and Piperno (2014). Each of these packages exploit the graph properties of the semi-Latin square, and acts on regular graph design, to classify the semi-Latin square design into automorphism classes and determine their isomorphisms. Chigbu and Eze (2001) automated the group-theoretic-based procedure for the construction of the $(n\times n)/k$ semi-Latin squares for n = 2, 3 and 4 and for any k. The program was designed on Qbasic programming language.

Mba et al. (2021) constructed $(6\times 6)/3$ semi-Latin squares from $2^6.3.7^2$ $(n\times n)$ reduced Latin square of order six and seven $(6\times 6)/2$ semi-Latin squaresby the method of superposition. The four popular statistical efficiency properties, the *A*-, *D*-, *E*- and *MV*-

*Corresponding Author; E-mail: emmanuel.mba@unn.edu.ng

efficiencies, of the emanating semi-Latin squares were evaluated. The efficient $(6\times6)/3$ semi-Latin squares with respect to the *A*-, *D*-, *E*- and *MV*-efficiency criteria was located. In this work, the automation of the construction procedure for the generation of the $(6\times6)/3$ semi-Latin squares and the evaluation of their *A*-, *D*-, *E*- and *MV*-efficiencies are presented. Hence, we provide a Microsoft Excel program for the construction and evaluation of these $(6\times6)/3$semi-Latin squares. The Microsoft Excel program is presented in the Appendix.

2 Combinatorial Justification for the Program

Given an $(n\times n)$ reduced Latin square of order six, Ω, and a $(6\times6)/2$semi-Latin square, Δ, also of order six,the principle of superposition is used such that $\Omega$is superimposed on Δor vice versa, to obtain a $(6\times6)/3$semi-Latin square of order six, Θ .McKay and Wanless (2005) gave the number of combinatorial compositions of Ω as $2^6.3.7^2$, which is the 9408 reduced Latin squares that exist in the literature. The superposition of the $2^6.3.7^2$ Latin squares on the seven $(6\times6)/2$semi-Latin squares give all possible $2^6.3.7^3$ $(6\times6)/3$semi-Latin squares.

3. Algebraic Basis for the Program

In order to appreciate the algebra associated with the construction of semi-Latin squares using the method of superposition, we adopt the description of the $(n\times n)/k$ semi-Latin squares given by Soicher (2013b). An $(n\times n)/k$semi-Latin square is an $(n\times n)$array, *S*, whose entries are *k*-subsets of an *nk*-set, the set of symbols for *S*, such that each symbol is in exactly one entry in each row and exactly one entry in each column of S: see Soicher (2013b). The entry in row *i* and column *j* is called the (*i*,*j*)-entry of S and is denoted by $S(i,j)$. We assume throughout that n > 1, k > 0 and note that for *k* = 1, an $(n\times n)/k$semi-Latin square is simply a Latin square of order *n*. Therefore, given two Latin or semi-Latin squares, $(n\times n)/k$ and $(n\times n)/m, m>0$, their superposition follows the description by Bailey (2000).Label all the rows and columns of the squares the same way while the letters in any one square must not appear in the other square. In the block of row *i* and column *j* put all the letters that occur in this block of any of the squares. The superimposition of these squares is a semi-Latin square of order six and blocks of size, $k+m$.

The $t\times b$incidence matrix, *N*, of Θ is obtained as the $nk\times n^2$ treatments-by-block matrix whose entry in the t^{th} row and b^{th} column is the number of times that each of*nk* treatments occurs in each of the *j* blocks. The entries in the b^{th} column/block of the incidence matrix is 1 where treatment *t* occurred and 0 where no treatment occurred. The information matrix is obtained from *N*as $C=R-NK^{-1}N'$, where $R=rI$ is a diagonal matrix with the *r* treatment replications as the diagonal entries; *I* is an identity matrix of order *n* and $K=kI$ is a diagonal matrix whose diagonal entries are the block sizes, *k*. Also, $C^*=\frac{1}{r}C$ is the normalized information matrix whose eigenvalues but one are the canonical efficiency factors, η_t, *t* = 0, 1, . . ., *v*-1. For the $(6\times6)/3$ semi-Latin squares, $v=nk=18$.

The *A*-, *D*- and *E*-efficiency criteria are derived directly from η_t as the harmonic mean of η_t, geometric mean of η_t and minimum of η_t, respectively. Hence, $A=(v-1)\left(\sum_{t=0}^{v-1}1/\eta_t\right)^{-1}$, $D=\left(\prod_{t=0}^{v-1}\eta_t\right)^{1/v-1}$ while $E=\min(\eta_t, t=0,1,\ldots,v-1)$.The *MV*-efficiency criterion is the minimum of the treatment contrasts from the Moore-Penrose inverse of *C*. Let λ_t be the t^{th} treatment effect and $\hat{\lambda}_t$ the least square estimator of λ_t. Then, the treatment contrast is $\left(\hat{\lambda}_t-\hat{\lambda}_{t'}\right)= \phi_{tt}+\phi_{t't'}-2\phi_{tt'}$, $t\neq t'$ whose variance is $Var\left(\hat{\lambda}_t-\hat{\lambda}_{t'}\right)= \phi_{tt}+\phi_{t't'}-2\phi_{tt'}$ $MV=\min\left(\hat{\lambda}_t-\hat{\lambda}_{t'}\right)$ for $t\neq t'$; where $\phi_{tt'}$ is the tt'-(th) entry of the Moore-Penrose inverse of *C*.

4 Program Description

The program was designed on Microsoft Excel 2016 which runs C^{++} programming language but it is adaptable to any version of Microsoft Excel. This Excel program has the advantage that it is a user-mode program and works well through single human interaction via the direct user interface. The program takes about half-hour of computing time to construct and evaluate the $2^6.3.7^3\,(6\times6)/3$ semi-Latin squares on University of Nigeria Nsukka Lion Grade. The size of the program is 20 Kilobyte of computer space. The program is very light and does not occupy substantial space on the computer disk.

The program is made up of 4subprograms, namely: Utilities, Main Source, Incidence Matrix and Final Result. We provide concise description of the roles of each subroutine.

Utilities: This contains the Public functions for the construction of the semi-Latin squares, incidence matrices and results. The functions for normalizing the eigenvectors are created in the utilities. It also contains the Public functions that labels and identifies each of the $(6\times6)/2$ semi-Latin squares; labels and identifies each of the 9408 (6×6) Latin squares; Labels and identifies each $(6\times6)/3$ semi-Latin square; labels and identifies each incidence matrix, information matrix and generalized inverse of the information matrix.

Main Source: This is the subprogram that accepts inputs. The inputs for this Excel program are the (6×6) Latin squares and the $(6\times6)/2$ semi-Latin squares as matrices. The construction of the $(6\times6)/3$ semi-Latin squares are done in this subroutine using the superposition functions for construction. The functions that compute the eigenvalues and the functions that compute the canonical efficiency factors are domiciled in this subroutine.

Incidence Matrices: This subroutine contains the commands and syntaxes that builds the binary incidence matrices of the semi-Latin square.

Final Results: This subroutine contains the functions which compute and displays the results of each of the four efficiency criteria, *A*-, *D*-, *E*- and *MV*-efficiencies.

5 Program Results

The program displays results in three separate sheets as follows; The first sheet contains the constructed $(6\times6)/3$semi-Latin squares, the second sheet contains the incidence matrices associated with the constructed $(6\times6)/3$semi-Latin Squares and the last

sheet contains the *A*-,*D*-, *E*- and *MV*-efficiency values evaluated for all the constructed $(6\times 6)/3$ semi-Latin squares

5.1 Construction of $(6\times 6)/3$ semi-Latin squares

The seven distinct $(6\times 6)/2$ semi-Latin squares studied by Bailey and Royle (1997) are pivotal in the construction of the $(6\times 6)/3$ semi-Latin squares. The first of the seven $(6\times 6)/2$ semi-Latin squares is presented in Figure1 below and identified as the Howell Design, Δ_1,

6	15	8	17	10	7	12	9	14	11	16	13
8	10	13	15	12	14	17	7	16	6	9	11
11	13	10	12	15	17	14	16	7	9	6	8
9	14	11	16	13	6	15	8	17	10	7	12
7	16	9	6	11	8	13	10	15	12	17	14
12	17	14	7	16	9	6	11	8	13	10	15

Figure 1: The Howell Design, Δ_1

There exists 9408 reduced (6×6) Latin square: see, for example, McKay (2019). A typical (6×6) reduced Latin square, denoted by Δ_0, is shown in Figure 2.

0	1	2	3	4	5
1	0	3	2	5	4
2	3	4	5	0	1
3	2	5	4	1	0
4	5	0	1	2	3
5	4	1	0	3	2

Figure 2: (6×6) Latin square, Δ_0

The Microsoft Excel program developed in this work superimposes the (6×6) Latin square of Figure 2 on the $(6\times 6)/2$ semi-Latin squares of Figure 1 to obtain the $(6\times 6)/3$ semi-Latin square, Δ_{01}, given in Figure 3.

The combinatorial composition of reduced Latin squares of order six was given by McKay and Wanless (2005) as $2^6.3.7^2$ which gives the 9408 reduced Latin squares. Therefore, each of the 9408 reduced (6×6) Latin square was superimposed on each of the seven $(6\times 6)/2$ semi-Latin squares of Bailey and Royle (1997) to obtained $2^6.3.7^3 = 65856$ possible $(6\times 6)/3$ semi-Latin squares.

0	6	15	1	8	17	2	10	7	3	12	9	4	14	11	5	16	13
1	8	10	0	13	15	3	12	14	2	17	7	5	16	6	4	9	11
2	11	13	3	10	12	4	15	17	5	14	16	0	7	9	1	6	8
3	9	14	2	11	16	5	13	6	4	15	8	1	17	10	0	7	12
4	7	16	5	9	6	0	11	8	1	13	10	2	15	12	3	17	14
5	12	17	4	14	7	1	16	9	0	6	11	3	8	13	2	10	15

Figure 3: The $(6\times 6)/3$ semi-Latin squares, Δ_{01}

5.2 The calculation of the incidence matrices

The incidence matrices associated with all the 65856 constructed designs are displayed in Sheet 2. The incidence matrix of the $(6\times 6)/3$ semi-Latin square displayed in Figure3 is given as Figure 4. The program written with Excel Macros was developed and used to construct the $(6\times 6)/3$ semi-Latin squares as well as the *A*, *D*, *E* and *MV*-efficiency evaluations which took about half-hour of CPU time on Lion Grade, University of Nigeria main server.

1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 4: The incidence matrix of Δ_{01}

5.3 Evaluation for the efficiency of $(6\times 6)/3$ semi-Latin squares

Sheet 3 displays of all the $65856\times 4 = 263424$ *A*-, *D*-, *E*- *and MV*-efficiency values that are calculated by the program. The efficiency values of Δ_{01} presented in Figure 3are: *A*-efficiency= 0.4909; *D*-efficiency = 0.5210; *E*-efficiency = 0.2723; *MV*-efficiency = 0.4314.

6 Conclusion

In this work, a Microsoft Excel program was presented for the construction of the eighteen-treatment semi-Latin squares of order six arranged in blocks of size three. The program utilizes the $2^6.3.7^2$ Latin squares of order six and the 7 $(6\times6)/2$ semi-Latin square in Bailey and Royle (1997) to obtain the $2^6.3.7^3$ $(6\times6)/3$ semi-Latin squares using the method of superposition. By treating the squares as incomplete-block designs and converting the squares to incidence matrices with entries of 0's and 1's, the program computes the popular *A*-, *D*-, *E*- and *MV*-efficiency criteria for judging the usefulness of the semi-Latin squares for experimentation. The *A*-, *D*-, *E*- and *MV*-efficiency values of any $(6\times6)/3$ semi-Latin square constructed and evaluated by this programshould be compared with the corresponding efficiency values of the hypothetical (pseudo) $(6\times6)/3$ Trojan square given byBailey (1992). Cheng and Bailey (1991) proved that the Trojan square of an $(n\times n)/k$ semi-Latin square of order *n* is efficient with respect to the *A*-, *D*- and *E*-efficiency criteria. Bailey (1992) conjectured that an $(n\times n)/k$ pseudo Trojan square is also *A*-, *D*- and *E*-optimal and included the *MV*-efficiency which was computed as the minimum simple efficiency measure: see also Soicher (2013a).

The Excel program presented herein occupies only about 20 Kilobyte of disk space and requires about half-hour of computing time to construct all the $2^6.3.7^3$ semi-Latin squares, transform them to incidence, concurrence and information matrices and compute their *A*-, *D*-, *E*-and *MV*-efficiency measures, as demonstrated through the illustrative example.

References

Bailey, R.A. (1988). Semi-Latin Squares, *Journal of Statistical Planning and Inference*, 18, 299 – 312.

Bailey, R.A. (2000). Semi-Latin Squares: Constructions. Available at http://www.maths.qmul.ac.uk/~rab/slsconst.html.

Bailey, R.A.(1992), Efficient Semi-Latin Squares, *Statistica Sinica*, 2, 413–437.

Bailey, R.A. and Royle, G. (1997). Optimal Semi-Latin Squares with Side Six and Block Size Two, *Proceedings of the Royal Society*, 453 (1964), 1904 – 1914.

Cheng, C.-S. and Bailey, R.A.(1991). Optimality of some Two-Associate-Class Partially Balanced Incomplete-Block Designs, *Annals of Statistics*, 19, 1667–1671.

Chigbu, P.E. and Eze, B.C. (2001). Automating the Group-Theoretic-Based Construction Procedure for the $(n \times n)/k$ Semi-Latin Square, *Utilitas Mathematica*, 60, 107 – 123.

Mba, E.I., Chigbu, P.E. and Ukaegbu, E.C. (2021).An Efficient Semi-Latin Square with Side Six and Block Size Three, *Preprint*.

McKay, B.D. (1981). Practical Graph Isomorphism, *Congressus Numerantium*, 30 (1981), 45 – 87.

McKay, B.D. and Piperno, A. (2014). Practical Graph Isomorphism II, *Journal of Symbolic Computation*, 60, 94 – 112.

McKay, B.D. and Wanless, I.M. (2005). On the Number of Latin Squares, *Annals of Combinatorics*, 9, 335 – 344.

Soicher, L.H. (1993). GRAPE: A System for Computing with Graphs and Groups, In: Larry Finkelstein and William M. Kantor(Editors), *Groups and Computation*, Vol. 11 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 287–291, American Mathematical Society.

Soicher, L.H. (2013a). Optimal and Efficient Semi-Latin Squares, *Journal of Statistical Planning and Inference*, Vol. 143, 573 – 582.

Soicher, L.H. (2013b). Designs, Groups and Computing, In: Detinko, Flanery and O'Brien (Editors), *Probabilistic Group Theory, Combinatorics and Computing: Lecture Notes from Fifth de Brun Workshop*, Springer-Verlag, London.

Appendix: The Excel Program

```
Option Explicit                                    (Utilities subroutine)

Dim lCalcSave As Long
Dim bScreenUpdate As Boolean

Public RCount As Long
Public RFrom As Long

Public Const LSheet As String = "LatinSquares"
Public Const MSheet As String = "IncidenceMatrix"
Public Const RSheet As String = "Results"

Sub SwitchOff(bSwitchOff As Boolean)
   Dim ws As Worksheet

   With Application
      If bSwitchOff Then
         ' OFF
         lCalcSave = .Calculation
         bScreenUpdate = .ScreenUpdating
         .Calculation = xlCalculationManual
         .ScreenUpdating = False
         .EnableAnimations = False

         ' switch off display pagebreaks for all worksheets

         For Each ws In ActiveWorkbook.Worksheets
            ws.DisplayPageBreaks = False
         Next ws
      Else
         ' ON
         If .Calculation <> lCalcSave And lCalcSave <> 0 Then .Calculation = lCalcSave
         .ScreenUpdating = bScreenUpdate
         .EnableAnimations = True
      End If
   End With
End Sub

Function SheetExists(sheetToFind As String) As Boolean
   SheetExists = False
   Dim sheet As Worksheet
   For Each sheet In Worksheets
      If sheetToFind = sheet.Name Then
         SheetExists = True
         Exit Function
      End If
   Next sheet
End Function
```

```
Function Eye(ByVal n As Integer, ByVal m As Integer)
   Dim i, j, k As Integer
   Dim X As Variant
   X = NewVariant(n, n)
   k = n

   For i = 1 To k
      For j = 1 To k
         If (i = j) Then
            X(i, j) = m
         Else
            X(i, j) = 0
         End If
      Next j
   Next i
   Eye = X
End Function

Function MSubtract(svM As Variant, sfM As Variant) As Variant
   Dim i, j, m, n, X, y As Integer
   m = LBound(svM, 1)
   n = LBound(svM, 2)
   X = UBound(svM, 1) - m + 1
   y = UBound(svM, 2) - n + 1
   Dim XY As Variant
   XY = NewVariant(X, y)

   For i = m To X
      For j = n To y
         XY(i, j) = sfM(i, j) - svM(i, j)
      Next j
   Next i
   MSubtract = XY
End Function

Function MAdd(svM As Variant, sfM As Variant)
   Dim i, j, m, n, X, y As Integer
   X = UBound(svM, 1) - LBound(svM, 1) + 1
   y = UBound(svM, 2) - LBound(svM, 2) + 1
   Dim XY As Variant
   XY = NewVariant(X, y)

   For i = 1 To X
      For j = 1 To y
         XY(i, j) = sfM(i, j) + svM(i, j)
      Next j
   Next i
   MAdd = XY
End Function
```

```
Function MxVMult(v As Double, m As Variant, Optional dm As Integer = 2) As Variant
   Dim i, j, X, a, b, y As Integer
   a = LBound(m, 1)
   X = UBound(m, 1) - a + 1
   If dm = 2 Then
      b = LBound(m, 2)
      y = UBound(m, 2) - b + 1
      For i = a To X
         For j = b To y
            m(i, j) = v * m(i, j)
         Next j
      Next i
   Else
      For i = a To X
         m(i) = v * m(i)
      Next i
   End If

   MxVMult = m
End Function

Function Ones(X As Integer, y As Integer)
   Dim i, j, m, n As Integer
   Dim XY As Variant
   XY = NewVariant(X, y)
   For i = 1 To X
      For j = 1 To y
         XY(i, j) = 1
      Next j
   Next i
   Ones = XY
End Function

Function MEigen(ByRef m As Variant) As Variant

   '************************************************************************
   '** Function computes the eigenvalues and eigenvectors for a real      **
   '** symmetric positive definite matrix using the "JK Method".  The     **
   '** first column of the return matrix contains the eigenvalues and     **
   '** the rest of the p+1 columns contain the eigenvectors.              **
   '** See:                                                               **
   '** KAISER,H.F. (1972) "THE JK METHOD: A PROCEDURE FOR FINDING THE     **
   '** EIGENVALUES OF A REAL SYMMETRIC MATRIX", The Computer Journal,     **
   '** VOL.15, 271-273.                                                   **
   '************************************************************************

   Dim a() As Variant, Ematrix() As Double
   Dim i As Long, j As Long, k As Long, iter As Long, P As Long
   Dim den As Double, hold As Double, Sin_ As Double, num As Double
```

```
Dim Sin2 As Double, Cos2 As Double, Cos_ As Double, Test As Double
Dim Tan2 As Double, Cot2 As Double, tmp As Double
Const eps As Double = 1E-16

On Error GoTo EndProc

a = m
P = UBound(a, 1)
ReDim Ematrix(1 To P, 1 To P + 1)

For iter = 1 To 15

   'Orthogonalize pairs of columns in upper off diag
   For j = 1 To P - 1
      For k = j + 1 To P

         den = 0#
         num = 0#
         'Perform single plane rotation
         For i = 1 To P
            num = num + 2 * a(i, j) * a(i, k)   ': numerator eq. 11
            den = den + (a(i, j) + a(i, k)) * _
               (a(i, j) - a(i, k))           ': denominator eq. 11
         Next i

         'Skip rotation if aij is zero and correct ordering
         If Abs(num) < eps And den >= 0 Then Exit For

         'Perform Rotation
         If Abs(num) <= Abs(den) Then
            Tan2 = Abs(num) / Abs(den)         ': eq. 11
            Cos2 = 1 / Sqr(1 + Tan2 * Tan2)    ': eq. 12
            Sin2 = Tan2 * Cos2                 ': eq. 13
         Else
            Cot2 = Abs(den) / Abs(num)         ': eq. 16
            Sin2 = 1 / Sqr(1 + Cot2 * Cot2)    ': eq. 17
            Cos2 = Cot2 * Sin2                 ': eq. 18
         End If

         Cos_ = Sqr((1 + Cos2) / 2)            ': eq. 14/19
         Sin_ = Sin2 / (2 * Cos_)              ': eq. 15/20

         If den < 0 Then
            tmp = Cos_
            Cos_ = Sin_                        ': table 21
            Sin_ = tmp
         End If

         Sin_ = Sgn(num) * Sin_                ': sign table 21
```

```
            'Rotate
            For i = 1 To P
               tmp = a(i, j)
               a(i, j) = tmp * Cos_ + a(i, k) * Sin_
               a(i, k) = -tmp * Sin_ + a(i, k) * Cos_
            Next i

         Next k
      Next j

      'Test for convergence
      Test = Application.SumSq(a)
      If Abs(Test - hold) < eps And iter > 5 Then Exit For
      hold = Test
   Next iter

   'If iter = 16 Then MsgBox "JK Iteration has not converged."

   'Compute eigenvalues/eigenvectors
   For j = 1 To P
      'Compute eigenvalues
      For k = 1 To P
         Ematrix(j, 1) = Ematrix(j, 1) + a(k, j) ^ 2
      Next k
      Ematrix(j, 1) = Sqr(Ematrix(j, 1))

      'Normalize eigenvectors
      For i = 1 To P
         If Ematrix(j, 1) <= 0 Then
            Ematrix(i, j + 1) = 0
         Else
            Ematrix(i, j + 1) = a(i, j) / Ematrix(j, 1)
         End If
      Next i
   Next j

   MEigen = Ematrix

   Exit Function

EndProc:
   MsgBox prompt:="Error in function MEigen!" & vbCr & vbCr & _
      "Error: " & Err.Description & ".", Buttons:=48, _
      Title:="Run time error!"
End Function
```

```
Function MCef(ByVal r As Integer, m As Variant) As Variant
   MCef = MxVMult((1 / r), m, 1)
End Function

Function MMinValues(ByVal m As Variant, ByVal t As Integer) As Variant
   Dim i, j, k As Integer
   Dim X(1 To 17) As Variant
   For i = 2 To t
      k = i - 1
      X(k) = 1000
      For j = 1 To t - 1
         If (m(j, i) < X(k)) Then
            X(k) = m(j, i)
         End If
      Next j
   Next i
   MMinValues = X
End Function

Function MMaxValues(m As Variant, t As Integer) As Double
   Dim i, j, X, y As Integer, P, Z As Double
   Z = 0
   For i = 1 To t - 1
      If (m(i) > Z) Then
         Z = m(i)
      End If
   Next i
   MMaxValues = Z
End Function

Function MProd(m As Variant, t As Integer) As Double
   Dim i As Integer, P As Double
   P = 1
   For i = 1 To t - 1
      P = P * m(i)
   Next i
   MProd = P
End Function

Function NewVariant(ByVal X As Integer, ByVal y As Integer) As Variant
   Dim Z As Variant
   If X = 18 And y = 18 Then
      ReDim Z(1 To 18, 1 To 18) As Variant
   ElseIf X = 18 And y = 36 Then
      ReDim Z(1 To 18, 1 To 36) As Variant
   ElseIf X = 36 And y = 18 Then
      ReDim Z(1 To 36, 1 To 18) As Variant
   ElseIf X = 36 And y = 36 Then
```

```
      ReDim Z(1 To 36, 1 To 36) As Variant
   End If
   NewVariant = Z
End Function

Public Function GetIdentifier(ByVal n As Long) As String
   Dim letr As String
   Select Case n
      Case 1
         letr = "A"
      Case 2
         letr = "B"
      Case 3
         letr = "C"
      Case 4
         letr = "D"
      Case 5
         letr = "E"
      Case 6
         letr = "F"
      Case 7
         letr = "G"
      Case Else
         letr = "Unknown"
   End Select
   GetIdentifier = letr
End Function

Option Explicit                                    (Main Source subroutine)

Sub StartUp()
   Dim f As frmCoverage
   Set f = New frmCoverage
   f.Show
End Sub

Sub FillBlanks()
   Dim template As Range
   Dim i, j, k, m, n, RC, RF As Long

   RF = ((RFrom - 1) * 7) + 9
   RC = ((RCount - 1) * 7) + RF

   Utilities.SwitchOff True

   Dim sWS As Worksheet

   If Utilities.SheetExists(LSheet) Then
      Set sWS = Worksheets(LSheet)
```

```
      sWS.Activate
   Else
      MsgBox "The source worksheet does not exist in this workbook or you have not renamed it to " &
LSheet & ". Please do so and try again.", vbExclamation, "Missing worksheet"
      End
   End If

   Set template = Range("K2", "ER7")
   template.Copy

   For i = RF To RC Step 7
      Cells(i, 11).Select
      ActiveSheet.Paste
      For j = 2 To 7
         m = ((j - 2) * 3) + 11
         For n = 0 To 120 Step 20
 Range(Replace(Cells(i, m + n).Address, "$", ""), Replace(Cells(i + 5, m + n).Address, "$", "")).Value2 = _
            Range(Replace(Cells(i, j).Address, "$", ""), Replace(Cells(i + 5, j).Address, "$", "")).Value2
         Next n
      Next j
   Next i
   IncidenceMatrix.GenerateMetrix
End Sub
Sub GenerateMetrix()                                    (Incidence matrix subroutine)
   Dim RC, RF, RFIM, sCol, sRow, col, row, imsCol, imsRow, imc, imr, hCol As Long, clr As String
   Dim i, j, k, m, n, X, y, Z As Long
   Dim fsCol, ftCol As String

   RF = ((RFrom - 1) * 7) + 9
   RC = ((RCount - 1) * 7) + RF
   RFIM = ((RFrom - 1) * 21) + 5

   fsCol = "AY"
   ftCol = "C"

   sCol = 11
   sRow = RF
   row = sRow
   imsCol = 4
   imsRow = RFIM
   imr = imsRow

   If Not Utilities.SheetExists(MSheet) Then
      Sheets.Add After:=Sheets(Sheets.Count)
      ActiveSheet.Name = MSheet
   End If

   If Not Utilities.SheetExists(RSheet) Then
```

```
    Sheets.Add After:=Sheets(Sheets.Count)
    ActiveSheet.Name = RSheet
  End If

  Utilities.SwitchOff True ' turn off features that slow down the execution

  Dim cls As String
  Dim sWS As Worksheet
  Dim tWS As Worksheet
  Dim rWS As Worksheet
  If Utilities.SheetExists(LSheet) Then
    Set sWS = Worksheets(LSheet)
    sWS.Activate
  Else
    MsgBox "The source worksheet does not exist in this workbook or you have not renamed it to " &
LSheet & ". Please do so and try again.", vbExclamation, "Missing worksheet"
    End
  End If

  Set tWS = Worksheets(MSheet)
  Set rWS = Worksheets(RSheet)

  For i = RF To RC Step 7
    hCol = 3
    imc = imsCol
    col = sCol
    row = i

    For X = 1 To 7
      fsCol = Replace(Cells(imr - 1, hCol).Address, "$", "")
      ftCol = Replace(Cells(imr + 17, hCol).Address, "$", "")
      With tWS.Range(fsCol).Interior
        .Color = RGB(255, 0, 0)
        .ThemeColor = xlThemeColorAccent1
        .ColorIndex = 2
      End With
      tWS.Cells(imr - 1, hCol).Value = "SQR: " & Replace(Cells(i, col + (20 * (X - 1))).Address, "$", "")

      With tWS.Range(fsCol, ftCol)
        .Font.Bold = True
      End With
      ftCol = Replace(Cells(imr - 1, hCol + 36).Address, "$", "")
      With tWS.Range(fsCol, ftCol)
        .Font.Bold = True
      End With

      For y = 0 To 17
        tWS.Cells(imr + y, hCol).Value = y
```

```
        Next y
        For y = 1 To 36
          hCol = hCol + 1
          tWS.Cells((imr - 1), hCol).Value = y
        Next y
        hCol = hCol + 3
      Next X
      For j = 0 To 5 'Rows count for each blocl (Square)
        col = sCol
        imc = imsCol + (6 * j)
        For k = 0 To 120 Step 20 'Iterating the columns
          For m = 0 To 5
            cls = ""
            For Z = 0 To 2
              cls = cls & "[" & Cells(row, col).Value & "]"
              col = col + 1
            Next Z

            For n = 0 To 17
              If InStr(1, cls, "[" & n & "]", 1) > 0 Then
                tWS.Cells(imr + n, imc + m).Value = 1
              Else
                tWS.Cells(imr + n, imc + m).Value = 0
              End If
            Next n
          Next m

          imc = imc + 39
          col = col + 2
        Next k
        row = row + 1
      Next j

      imr = imr + 21
    Next i
    Utilities.SwitchOff False ' turn these features back on

    FinalResults.Finaliz ' tWS, rWS

    MsgBox "We are done generating incidence metrix", vbOKOnly + vbInformation, "Done"
    End
End Sub
'Option Explicit                                    (Final Results subroutine)

Private RN As Long, CN As Integer

Sub Finaliz()
    Dim i, j, k, ez, RC, RF As Long
```

```
    Dim sWS, tSW As Worksheet

    RF = ((RFrom - 1) * 21) + 5
    RC = ((RCount - 1) * 21) + RF
    RN = RFrom

    Utilities.SwitchOff True

    Set sWS = Worksheets(MSheet)
    Set tWS = Worksheets(RSheet)

    For i = RF To RC Step 21
       ez = i
       CN = 0
       For j = 4 To 238 Step 39
          CN = CN + 1
          GenerateFML i, j, sWS, tWS
       Next j
       RN = RN + 1
    Next i
    Utilities.SwitchOff False ' turn these features back on
End Sub

Sub GenerateFML(ByVal startRow As Long, ByVal startCol As Long, ByVal sWS As Worksheet, ByVal
tWS As Worksheet)
    Dim i, j As Integer ', n, r, k, b, t
    Dim X, Xp, XXp As Variant ', Rc, Kc, Cc, detC, GC, detGC, GCInv, EIG, CEF, nzEIG
    Dim qr, qc As Long

    qr = (((startRow - 5) / 21)) + 4
    qc = (((startCol - 4) / 39) * 5) + 2

    Const n As Integer = 6
    Const r As Integer = n
    Const k As Integer = 3
    Const b As Integer = n * n
    Const t As Integer = n * k

    On Error Resume Next

    X = sWS.Range(Replace(Cells(startRow, startCol).Address, "$", ""), Replace(Cells(startRow + 17,
startCol + 35).Address, "$", "")).Value
    Xp = Application.WorksheetFunction.Transpose(X)
    XXp = Application.WorksheetFunction.MMult(X, Xp)
    RC = Utilities.Eye(t, r)
    Kc = Utilities.Eye(b, k)
    Zc = WorksheetFunction.MMult(WorksheetFunction.MMult(X, WorksheetFunction.MInverse(Kc)),
Xp)
```

```
Cc = Utilities.MSubtract(Zc, RC)
detC = WorksheetFunction.MDeterm(Cc)
GC = MAdd(Cc, Ones(t, t))
detGC = WorksheetFunction.MDeterm(GC)
GCInv = WorksheetFunction.MInverse(GC)
eigvv = MEigen(Cc)
Dim eig(1 To 18) As Variant
For i = 1 To 18
   eig(i) = eigvv(i, 1)
Next i
cef = MCef(r, eig)
nzEig = eig(2)
sum = 0
For i = 1 To t - 1
   sum = sum + (k / ((r * k) - eig(i)))
Next i
EFF = (t - 1) / (r * sum)
l = r * (k - 1) / (t - 1)
EF = (l * t) / (r * k)
sum1 = 0
For i = 1 To t - 1
   sum1 = (sum1 + 1 / cef(i))
Next i
f = MxVMult((1 / r), Cc)
P = MxVMult((1 / t), Ones(t, t))
GG = MAdd(f, P)
MM = MSubtract(P, WorksheetFunction.MInverse(GG))

Dim tt(1 To 17, 1 To 18) As Variant, sum2 As Double
For i = 1 To t - 1
   For j = 1 To t
      If j < i + 1 Then
         tt(i, j) = 0
      Else
         ff = (MM(i, i) + MM(j, j) - MM(i, j) - MM(j, i))
         tt(i, j) = 2 / ff
      End If
   Next j
Next i

mve = MMinValues(tt, t)
A_Efficiency = (t - 1) * (1 / sum1)
sum2 = MProd(cef, t)
D_efficiency = (sum2) ^ (1 / (t - 1))
E_efficiency = cef(t - 1)
MV_efficiency = MMaxValues(mve, t)
```

```
If (qr = 4) Then
      tWS.Cells(qr - 1, qc).Value = "A Efficiency"
      tWS.Cells(qr - 1, qc + 1).Value = "D Efficiency"
      tWS.Cells(qr - 1, qc + 2).Value = "E Efficiency"
      tWS.Cells(qr - 1, qc + 3).Value = "MV Efficiency"

      If (CN = 1) Then tWS.Cells(qr - 1, qc - 1).Value = "S/N"
      tWS.Cells(qr - 2, qc + 2).Value = Utilities.GetIdentifier(CN)
   End If

   If (CN = 1) Then tWS.Cells(qr, qc - 1).Value = RN
   tWS.Cells(qr, qc).Value = A_Efficiency
   tWS.Cells(qr, qc + 1).Value = D_efficiency
   tWS.Cells(qr, qc + 2).Value = E_efficiency
   tWS.Cells(qr, qc + 3).Value = MV_efficiency

End Sub
```