S-PLUS function for recovering interblock, interrow-column, and intergradient information

John Barnard, Computer Center, N.Y. State Agricultural Experiment Station, Geneva, N. Y.
and
Walter T. Federer, Biometrics Unit, Cornell University, Ithaca, N. Y.

BU-1382-M January 1997

0. Abstract

The package S-PLUS is used to demonstrate the recovery of interblock, interrow-column, and intergradient information for incomplete block and lattice rectangle designed experiments. The analyses follow that first proposed by F. Yates in the late 1930's and what has become known as a mixed model analysis. The results obtained from the S-PLUS output agrees with what is obtained with SAS PROC MIXED. A discussion of S-PLUS functions used for these analyses is also presented.

Key words: Incomplete block design, lattice square design, random effects, random trends, mixed model, SAS PROC MIXED.

1. Introduction

Descriptions of GAUSS (Federer, 1995a), SAS (Federer, 1995b) and Genstat (Barnard & Federer, 1997) programs have appeared in Technical Reports of the Biometrics Unit. We presently describe an S-PLUS function for recovering information from single-treatment-factor incomplete block designs (Sections 2 & 3), and illustrate its use for recovering interblock (Section 4), interrow and intercolumn (Section 5), and interblock and intergradient (Section 6) information.

S-PLUS (MathSoft, Cambridge, MA) is an enhanced version of the S language. S was developed at AT&T Bell Laboratories and is an object-oriented language for data analysis. In general, complex operations in S are represented as functions which are called with appropriate arguments, and which return objects containing results. For an extensive introduction to S-PLUS, see Venables & Ripley (1994).

S-PLUS 3.4 Release 1 was the version used herein.

2. A function for recovering information in a single-treatment-factor design

Our function is presented below followed by a line-by-line explanation.

```
# rinfo. Recovery of information in a single-treatment-factor design.
# Arguments of the function are a model formula and a treatment factor.
# All factors in the model are considered random with the exception of the
# treatment factor. The function returns a list containing REML variance
# components, REML means & standard errors.
#
rinfo <- function(model, treat)
{
```

```
  if(missing(treat) || missing(model)) { stop('Missing argument') }
  if(class(model) != 'formula') { stop('Arg 1 not a formula') }
  if(class(treat) != 'factor') { stop('Arg 2 not a factor') }
  mf <- model.frame(model)
  for ( i in 1:length(mf) ) { is.random(mf[[i]]) <- T }
  is.random(mf[[deparse(substitute(treat))]]) <- F
  old.contrasts <- .Options$contrasts
  options(contrasts=c('contr.treatment'))
  vc.fit <- varcomp(model, method='reml',data=mf)
  options(contrasts=old.contrasts)
  vc <- vc.fit$variances
  nlevels <- length(levels(treat))
  coef <- as.vector(vc.fit$coefficients)
  mn <- c(coef[1],coef[1]+coef[2:nlevels])
  cov <- vc.fit$cov.fixed
  se.mn <- sqrt(as.vector(c(cov[1,1],
           cov[1,1]+2*cov[1,2:nlevels]+diag(cov)[2:nlevels])))
  list(vc=vc, mn=mn, se.mn=se.mn)
}
```

The function takes two arguments, a model formula and a treatment factor. The function returns a list containing variance components, REML means, and standard errors of means.

Operation of the function is as follows:

2.1. The function `rinfo` takes two arguments: a model formula and the treatment factor.

2.2. Actual arguments are checked for existence and appropriate class.

```
if(missing(treat) || missing(model)) { stop('Missing argument') }
if(class(model) != 'formula') { stop('Arg 1 not a formula') }
if(class(treat) != 'factor') { stop('Arg 2 not a factor') }
```

2.3. A new frame representing the model is constructed.

```
mf <- model.frame(model)
```

2.4. All design variables are declared as random (`is.random` is set TRUE) and then the treatment factor is declared not random (`is.random` is set FALSE).

```
for ( i in 1:length(mf) ) { is.random(mf[[i]]) <- T }
is.random(mf[[deparse(substitute(treat))]]) <- F
```

2.5. The current contrast defaults are saved and 'treatment' (corner point) contrasts established. This is done to simplify subsequent calculation of estimates and standard errors.

```
old.contrasts <- .Options$contrasts
options(contrasts=c('contr.treatment'))
```

2.6. The model is fitted using the data frame `mf`. REML estimation is specified (the S-PLUS default is MINQUE0). A varcomp object called `vc.fit` is created.

```
vc.fit <- varcomp(model, method='reml',data=mf)
```

2.7. The original contrast defaults are restored.

```
options(contrasts=old.contrasts)
```

2.8. Variance components are extracted from `vc.fit`.

```
vc <- vc.fit$variances
```

2.9. For subsequent indexing purposes, the number of levels in the treatment factor is calculated.

```
nlevels <- length(levels(treat))
```

2.10. Coefficients (effects) of the fixed part of the model are extracted from `vc.fit` and converted to a vector.

```
coef <- as.vector(vc.fit$coefficients)
```

2.11. Given the parameterisation (2.5), REML means are calculated as b_1, b_1+b_2,... ,b_1+b_i, where the b_i are the coefficients.

```
mn <- c(coef[1],coef[1]+coef[2:nlevels])
```

2.12. The covariance matrix for the fixed effects is extracted from `vc.fit`.

```
cov <- vc.fit$cov.fixed
```

2.13. Given the chosen parameterisation, standard errors of the means are calculated as

$$\text{se}(m_1) = \sqrt{\{\text{var}(b_1)\}}$$
$$\text{se}(m_2) = \sqrt{\{\text{var}(b_1) + 2\ \text{cov}(b_1,b_2) + \text{var}(b_2)\}}$$
$$\vdots$$
$$\text{se}(m_i) = \sqrt{\{\text{var}(b_1) + 2\ \text{cov}(b_1,b_i) + \text{var}(b_i)\}}$$

```
se.mn <- sqrt(as.vector(c(cov[1,1],
         cov[1,1]+2*cov[1,2:nlevels]+diag(cov)[2:nlevels])))
```

2.14. Results are returned as a list with three components: the variance components, REML means and standard errors.

```
list(vc=vc, mn=mn, se.mn=se.mn)
```

3. Using the function

In the following examples (Sections 4,5,6), S-PLUS code is given to analyse three incomplete block designs, the same sequence of operations being used for each.

3.1. The data is read into a 'data frame' and the data frame 'attached' so that the columns of the data frame are visible as vectors.

3.2. Where necessary, the design variables are converted to factors.

3.3. The function is called with two arguments: an appropriate model formula, and a treatment factor.

3.4. Results are printed.

4. Incomplete Block Design

The example is XI-3 from Federer (1955). The design is a triple lattice with $v = 9$ treatments in incomplete blocks of $k = 3$ treatments, and $r = 3$ replicates (complete blocks). The data set is named `fed933.dat`. The interblock analysis can be accomplished using the following S-PLUS code.

```
#sp-1  Triple lattice
#Read data
  dat <- read.table('fed933.dat', col.names=c('y', 'r', 'b', 't'))
  attach(dat)
#Convert r,b,t to factors
  r <- as.factor(r)
  b <- as.factor(b)
  t <- as.factor(t)
#Fit model & print results
  e <- rinfo(y ~ r/b+t, t)
  print('Variance components')
  print(e$vc)
  print('REML means & standard errors')
  print(cbind(e$mn,e$se.mn,deparse.level=2))
#Clean up
  rm(dat,r,b,t,e)
```

Function `rinfo` is called with the model formula `y ~ r/b+t` in which `y` is the response variate, `r` is replicate, `b` is block, `t` is treatment, and `r/b` indicates that blocks are nested within replicates.

The following output is produced:

```
[1] "Variance components"
          r  b %in% r Residuals
 0.04074147 0.2814692 0.7148254

[1] "REML means & standard errors"
         e$mn   e$se.mn
[1,] 7.210954 0.5640229
[2,] 2.544287 0.5640229
[3,] 3.935901 0.5640229
[4,] 2.951048 0.5640229
[5,] 4.828669 0.5640229
[6,] 2.911425 0.5640229
[7,] 3.235430 0.5640229
[8,] 2.210954 0.5640229
[9,] 6.837998 0.5640229
```

Estimates of variance components for replicates (`r`), blocks within replicates (`b %in% r`), and residuals are printed, followed by estimates of treatment means and their standard errors.

5. Lattice Square and Rectangle Designs with Rows and Columns

The following S-PLUS code recovers interrow and intercolumn information from the lattice square design given in Table 12.5 of Cochran and Cox (1957).

```
#sp-2  Lattice square
#Read data
  dat <- read.table('lsgr1645.dat', col.names=c('yield', 'rep', 'row',
         'col', 'grad', 'treat'))
  attach(dat)
#Convert rep,row,col,treat to factors
  rep <- as.factor(rep)
  row <- as.factor(row)
  col <- as.factor(col)
  treat <- as.factor(treat)
#Fit model & print results
  e <- rinfo(yield ~ rep/(row+col)+treat, treat)
  print('Variance components')
  print(e$vc)
  print('REML means & standard errors')
  print(cbind(e$mn,e$se.mn,deparse.level=2))
#Clean up
  rm(dat,rep,row,col,treat,e)
```

Function `rinfo` is called with the model formula `yield ~ rep/(row+col)+treat` where `yield` is the response variate, `rep` is replicate, `col` is column, `treat` is treatment, and `rep/(row+col)` indicates that rows and columns are nested within replicates.

The following output is produced:

```
[1] "Variance components"
          rep row %in% rep col %in% rep Residuals
 3.035689e-08     10.91943     3.170059  23.85978

[1] "REML means & standard errors"
            e$mn  e$se.mn
 [1,]  6.089939 2.566344
 [2,] 13.730803 2.566344
 [3,]  8.493211 2.566344
 [4,] 11.343945 2.566344
 [5,]  9.562314 2.566344
 [6,]  7.296142 2.566344
 [7,]  7.298382 2.566344
 [8,]  9.413093 2.566344
 [9,] 10.114016 2.566344
[10,] 15.183967 2.566344
[11,] 17.904368 2.566344
[12,] 12.846232 2.566344
[13,] 11.032131 2.566344
[14,] 14.249753 2.566344
[15,]  9.294736 2.566344
[16,] 10.626969 2.566344
```

6. Incomplete Block and Lattice Rectangle Designs with Differential Gradients in Blocks or in Rows

Following Federer (1996), we note that in certain experimental situations gradients may occur within the incomplete block or within the row (or column) of the design. When this occurs the analyses in Sections 4 and 5 are inappropriate. Since the gradients or trends may vary from block to block or from row to row and these occur at random, an analysis recovering both interblock

(interrow) and intergradient information is required in order to efficiently analyse the results from the experiment.

S-PLUS code to recover interrow and intergradient information is as follows:

```
#sp-3  Lattice square with gradients
#Read data
  dat <- read.table('lsgr1645.dat', col.names=c('yield', 'rep', 'row',
         'col', 'grad', 'treat'))
  attach(dat)
#Convert rep,row,treat to factors
  rep <- as.factor(rep)
  row <- as.factor(row)
  treat <- as.factor(treat)
#Fit model & print results
  e <- rinfo(yield ~ rep/row/grad+treat, treat)
  print('Variance components')
  print(e$vc)
  print('REML means & standard errors')
  print(cbind(e$mn,e$se.mn,deparse.level=2))
#Clean up
  rm(dat,rep,row,treat,e)
```

Function `rinfo` is called with the model formula `yield ~ rep/row/grad+treat` where `yield` is the response variate, `rep` is replicate, `row` is row, `grad` is gradient, `treat` is treatment, and `rep/row/grad` indicates that rows are nested within replicates, and gradients are nested within rows.

Results are:

```
[1] "Variance components"
          rep row %in% rep grad %in% (rep/row) Residuals
 6.359006e-10     11.25767             1.39749  18.77049

[1] "REML means & standard errors"
           e$mn  e$se.mn
 [1,]  5.503888 2.480640
 [2,] 15.047591 2.477027
 [3,]  8.848429 2.572556
 [4,] 12.054901 2.395097
 [5,]  9.397238 2.322052
 [6,]  8.268893 2.481599
 [7,]  6.181779 2.397966
 [8,]  9.377469 2.396196
 [9,]  9.163896 2.321542
[10,] 15.427754 2.680954
[11,] 17.686838 2.572005
[12,] 13.382055 2.398006
[13,] 10.712818 2.478733
[14,] 13.199300 2.321296
[15,] 10.205082 2.395422
[16,] 10.022068 2.395611
```

7. Comments

An S-PLUS function for recovering several types of error information in single-treatment-factor

designs has been described. It will be noted that the numerical results agree with those previously obtained using SAS (Federer, 1995b).

Extension to incomplete block designs with factorial treatment structure would require a little more programming, but involve no new concepts.

The original version of our function included calculation of standard errors of the variance components. Such calculation simply requires taking the square roots of the diagonal of the covariance matrix of the variance component estimates, and adds just one line of S-PLUS code to the function,

```
se.vc <- sqrt(diag(summary(vc.fit)$cov.ran))
```

is inserted before

```
nlevels <- length(levels(treat))
```

In addition, `se.vc` is put into the return list

```
list(vc=vc, se.vc=se.vc, mn=mn, se.mn=se.mn)
```

However, the covariance matrix, `cov.ran`, produced by `summary` was found to be grossly inaccurate. Discussion with a MathSoft consultant determined that (what we consider to be) an unacceptable finite difference method was used to calculate the associated Hessian. The matter has been recorded as a bug and will be addressed in a future release of S-PLUS.

8. References

Barnard, J. & Federer, W.T. (1997) Genstat and SAS Programs for Recovering Interblock, Interrow-column, and Intergradient Information. BU-134?-M in the Technical Report Series of the Biometrics Unit, Cornell University, Ithaca, NY, January.

Cochran, W. G. & Cox, G.M. (1957) *Experimental Designs*, 2nd edition. John Wiley & Sons, Inc., New York.

Federer, W. T. (1955) *Experimental Design: Theory and Application.* Macmillan Co., New York (Republished by Oxford and IBH Publishing Co., New Delhi, in 1967 and 1974).

Federer, W. T. (1995a) GAUSS and recovery of interblock, intercolumn, and intergradient information. BU-1271-M in the Technical Report Series of the Biometrics Unit, Cornell University, Ithaca, NY, June.

Federer, W. T. (1995b) SAS for ANOVAs and recovery of interblock, intercolumn, and intergradient information. BU-1293-M in the Technical Report Series of the Biometrics Unit, Cornell University, Ithaca, NY, June.

Federer, W. T. (1996) Recovery of interblock, intergradient, and intervariety information in incomplete block and lattice rectangle designed experiments. BU-1315-MA in the Biometrics Unit Technical Report Series, Cornell University, Ithaca, NY, June.

Venables, W.N. & Ripley, B.D. (1994) *Modern Applied Statistics with S-PLUS.* Springer-Verlag, New York.