Displays for Statistics 5303

Lecture 35

November 27, 2002

Christopher Bingham, Instructor

612-625-7023 (St. Paul)
612-625-1024 (Minneapolis)

Class Web Page
http://www.stat.umn.edu/~kb/classes/5303

Randomizing Latin Squares

For any g, there are only a finite number of g by g Latin squares. If there are M squares, full randomization involves picking one square with probability 1/M.

For example, when g = 2, M = 2:

A	B
B	A

and

B	A
A	B

For g = 3, M = 12 obtained by all orderings of the rows of

A	B	C
B	C	A
C	A	B

and

A	C	B
B	A	C
C	B	A

For g = 4, there are 576, all obtainable by permuting rows and columns of

A	B	C	D
B	A	D	C
C	D	B	A
D	C	A	B

A	B	C	D
B	C	D	A
C	D	A	B
D	A	B	C

A	B	C	D
B	D	A	C
C	A	D	B
D	C	B	A

A	B	C	D
B	A	D	C
C	D	A	B
D	C	B	A

The usual procedure is in two steps

1. Select a square at random from a list of representative squares in a table
2. Randomize the order of rows, columns and assignment of treatments to letters.

Actually you don't need quite that much randomization, but it doesn't hurt to do more than the minimum.

For larger g, you can replace step 1 by

1′. Construct a square in a systematic manner, say with A, B, C, ...Z in column 1; B, C, D, ..., Z, A in column 2; C, D, ..., Z, A, B in column 3, ..., etc. like

A	B	C	D	E	F	G	H
H	A	B	C	D	E	F	G
G	H	A	B	C	D	E	F
F	G	H	A	B	C	D	E
E	F	G	H	A	B	C	D
D	E	F	G	H	A	B	C
C	D	E	F	G	H	A	B
B	C	D	E	F	G	H	A

I entered the four 4 by 4 squares on p. 608 as matrices in MacAnova

```
Cmd> print(square1,square2,square3,square4,\
      labels:F,format:"3.0f")
square1:
    1   2   3   4      1 = A, 2 = B, 3 = C, 4 = D
    2   1   4   3
    3   4   2   1
    4   3   1   2
square2:
    1   2   3   4
    2   3   4   1
    3   4   1   2
    4   1   2   3
square3:
    1   2   3   4
    2   4   1   3
    3   1   4   2
    4   3   2   1
square4:
    1   2   3   4
    2   1   4   3
    3   4   1   2
    4   3   2   1
```

Pick one square randomly:

```
Cmd> ceiling(4*runi(1)) # random selection of square
(1)          2      Square 2
```

Now select random reordering of rows, and columns and apply them to **square2**.

```
Cmd> J_rows <- rank(runi(4)); J_rows # to reorder rows
(1)          2          4          3          1

Cmd> J_cols <- rank(runi(4)); J_cols# to reorder columns
(1)          3          4          2          1

Cmd> square <- square2[J_rows,J_cols]
```

This permutes both rows and columns.

Here is the square with rows and columns randomly permuted:

```
Cmd> square
(1,1)          4          1          3          2
(2,1)          2          3          1          4
(3,1)          1          2          4          3
(4,1)          3          4          2          1
```

Then get random assignment of treatments to letters (numbers here)

```
Cmd> J_trt <- rank(runi(4)); J_trt
(1)         3          1          4          2
```

Assign treatment 1 to B, 2 to D, 3 to A and 4 to C

```
Cmd> treat <- vector(square) # unravel square to vector

Cmd> treat1 <- rep(0,16) # new vector for treatments

Cmd> treat1[treat==1] <- 3;treat1[treat==2] <- 1

Cmd> treat1[treat==3] <- 4; treat1[treat==4] <- 2

Cmd> matrix(treat1,4) # Final square, with treatment numbers
(1,1)          2          3          4          1
(2,1)          1          4          3          2
(3,1)          3          1          2          4
(4,1)          4          2          1          3
```

To use in **anova()** you would need to turn **treat1** into a factor

```
Cmd> treat1 <- factor(treat1)
```

Unbalanced block designs

Sometimes it may be impossible to have a RCB or a LS.

This may happen because you designed a RCB or a LS, but lost some data so that there are missing values.

You still have the same model

$$y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}$$

or

$$y_{ij} = \mu + \alpha_i + \beta_j + \gamma_k + \varepsilon_{ijk}$$

Or the natural block size is too small for all g treatments.

Oehlert's eye drop example is a case in point:

Experiment is to compare eye drop brands with respect to their effect on eye irritation. There is likely to be a lot of variability between subjects and you would like to use subjects as blocks.

If there are only g = 2 brands, it's easy.

Each subject has g = 2 eyes and you can randomly assign brand A to one eye and B to the other. This gives a RCB design. If there are b subjects (blocks), each brand is replicated r = b times.

When there are g > 2 brands, you can't do a RCB since people have only two eyes.

You need some form of an **incomplete block design** for which each block has k < g different treatments.

One way you might do it, if you have b subjects and g = 3, is to

- Randomly select 2 of the three treatments for each subject
- Then allocate them randomly to the eyes.

Here's how it might be done with 100 subjects:

```
Cmd> g <- 3; k <- 2; b <- 100

Cmd> subjects <- factor(rep(run(b),rep(2,100)))

Cmd> subjects[run(10)] # blocking factor for subjects 1 - 5
(1)          1          1          2          2          3
(6)          3          4          4          5          5
```

Create treatment assignments. Start with an empty matrix with a column for each subject:

```
Cmd> trt <- matrix(rep(0,k*b),k) # 2 by 100 matrix
```

Put a randomly ordered random selection of 2 treatments in each column.

```
Cmd> for(i,1,b){
        trt[,i] <- rank(runi(3))[run(2)];;}
```

Here are the treatment assignments for the first 5 subjects (columns):

```
Cmd> trt[,run(5)] # first 5 subjects
(1,1)           3           3           2           3           1
(2,1)           2           1           3           1           3
```

Now turn **treat** into a factor.

```
Cmd> trt <- factor(vector(trt))
```

The trouble with this approach is that

- Treatments may not all be replicated the same number of times.
- Pairwise comparisons of treatments may have different standard errors.

Here I created some artificial data, $k = 2$ values for each of the $b = 100$ subjects.

```
Cmd> y <- 100 + rnorm(k*b) # create some artificial data

Cmd> tabs(y,trt,count:T) # unequal replications
(1)          69          68          63
```

Replications range from 63 to 69.

```
Cmd> anova("y=subjects + trt",fstat:T)
Model used is y=blocks + trt
WARNING: summaries are sequential
                DF          SS          MS           F     P-value
CONSTANT         1  2.0021e+06  2.0021e+06  2.5158e+06           0
subjects        99      85.958     0.86826     1.09101     0.33334
trt              2     0.65094     0.32547     0.40897     0.66546
ERROR1          98      77.992     0.79583
```

Note: **trt** is in the model after **subjects**. This must always be the case with unbalanced blocking designs. Blocking factors *must* precede treatments to assure that that SS_{trt} is a type II SS, "adjusted" for blocks.

Order doesn't matter for a RCB.

Here are the standard errors for the three pairwise contrasts.

```
Cmd> contrast(trt,vector(1,-1,0))$se
(1)     0.17371

Cmd> contrast(trt,vector(1,0,-1))$se
(1)     0.18047

Cmd> contrast(trt,vector(0,1,-1))$se
(1)      0.1818
```

They're close, but not identical.

Balanced incomplete block designs or **BIBD**s are a particular variety of incomplete blocks that don't have this problem.

Their defining property is:

- Every treatment appears the same number r times
- Each treatment appears with every other treatment the same number λ of times.

A	A	B	C	B	A	C	B	A
B	C	C	A	C	B	B	A	C

Here $g = 3$, $k = 2$, $b = 9$, $r = 6$ and $\lambda = (k-1)\times r/(g-1) = 3$

```
Cmd> b <- 9; g <- 3; k <- 2

Cmd> blk <- factor(rep(run(b),rep(k,b))) # 1,1,2,2,...,9,9

Cmd> print(matrix(blk,k),format:"3.0f")
MATRIX:
(1,1)   1   2   3   4   5   6   7   8   9
(2,1)   1   2   3   4   5   6   7   8   9

Cmd> trt <- factor(1,2, 1,3, 2,3, 3,1, 2,3, 1,2, 3,2, 2,1, 1,3)

Cmd> y <- 100 + blk + trt + rnorm(b*k) # Artificial data

Cmd> tabs(y,trt,count:T) # replications are all the same
(1)           6           6           6

Cmd> r <- 6; lambda <- r*(k-1)/(g-1); lambda
(1)           3
```

```
Cmd> anova("y=blk + trt",fstat:T)
Model used is y=blk + trt
WARNING: summaries are sequential
                DF          SS          MS           F     P-value
CONSTANT         1  2.0811e+05  2.0811e+05  2.1129e+05           0
blk              8      155.19      19.399    19.69558  0.00038481
trt              2      17.377      8.6883     8.82095    0.012218
ERROR1           7      6.8947     0.98496
```

Now the contrast standard errors are the same.

```
Cmd> contrast(trt,vector(1,-1,0))$se
(1)     0.66163

Cmd> contrast(trt,vector(1,0,-1))$se
(1)     0.66163

Cmd> contrast(trt,vector(0,1,-1))$se
(1)     0.66163

Cmd> varBib <- contrast(trt,vector(0,1,-1))$se^2; varBib
(1)     0.43776
```

Let's see how the $V[\overline{y}_{i\bullet} - \overline{y}_{j\bullet}]$ for this BIBD compares with $V[\overline{y}_{i\bullet} - \overline{y}_{j\bullet}] = 2\sigma^2/r$ from a RCB with r = 6 blocks (same number of replicates) and with the same σ^2.

```
Cmd> mse <- SS[4]/DF[4]; mse
      ERROR1
     0.98496

Cmd> varRcbd <- 2*mse/r; varRcbd
(1)     0.32832

Cmd> varRcbd/varBib
(1)        0.75
```

This shows the efficiency of this BIBD compared to a RCB with the same number of replicates and same σ^2 is .75 or 75%.

It can be shown that

$$E_{BIBD:RCB} = g(k-1)/((g-1)k)$$

```
Cmd> g*(k-1)/((g-1)*k)
(1)        0.75
```

Another way to put it, is that the **effective replication** is

$$E_{BIBD:RCB} \times r = (3/4)6 = 4.5:$$

```
Cmd> 2*mse/4.5
(1)     0.43776   = varBibd
```

One problem is that not all combinations of block sizes and treatment numbers may be possible without having far too many blocks.

You must have N = bk and N = rg

And $\lambda = r(k-1)/(g-1)$ must be an integer.

Analysis BIBD Example 14.2 in MacAnova. Analysis is virtually identical with a RCB design, although the "by hand" formulas are more complicated.

```
Cmd> tab14_1 <- read("","exmpl14.2")
exmpl14.2         36     3
) A data set from Oehlert (2000) \emph{A First Course in Design
) and Analysis of Experiments}, New York: W. H. Freeman.
)
) Data originally from John, P. W.~M. (1961). ``An application x
) and  balanced incomplete block design
) '' {\em Technometrics\/}~{\em 3}, 51--54.
)
) Table 14.1, p. 359
) Test of 9 different detergents.  There are three basins that
) are used simultaneously at the same rate with a different
) detergent in each basin.  Response is number of plates until
) foam disappears in a basin.
) Column 1 is session. Column 2 is treatment (kind of detergent)
) Column 3 is response (number of dishes)
) Treatments 1-4 are detergent base 1
) with (3, 2, 1, or 0) parts additive
) Treatments 5-8 are detergent base
) 2 with (3, 2, 1, or 0) parts additive
) Treatment 9 is a control.
Read from file "TP1:Stat5303:Data:OeCh14.dat"

Cmd> makecols(tab14_1,session,treatment,count)

Cmd> session <- factor(session)

Cmd> treatment <- factor(treatment)
```

The blocking factor is `session`. It must appear in the model before `treatment`.

```
Cmd> anova("count=session + treatment",fstat:T)
Model used is count=session+treatment
WARNING: summaries are sequential
                DF          SS          MS           F     P-value
CONSTANT         1       13572       13572 16469.69663  1.5466e-25
session         11      412.75      37.523    45.53320  6.0284e-10
treatment        8      1086.8      135.85   164.85393  6.8089e-14
ERROR1          16      13.185     0.82407
```

You can do pairwise multiple comparisons as for a CRD and CRB.

```
Cmd> pairwise("treatment",.05,hsd:T)
             4    -12.9
             3    -6.22
  |          2    -2.22
  | |        8   -0.222
  | |        1    0.333
    | |      7     1.67
      | |    6     3.56
        |    5     5.89
             9     10.1
```

You can check that the standard errors are the same:

```
Cmd> contrast(treatment,vector(1,-1,rep(0,7)))
component: estimate
(1)      2.5556
component: ss
(1)      9.7963
component: se
(1)      0.7412

Cmd> contrast(treatment,vector(1,0,-1,rep(0,6)))
component: estimate
(1)      6.5556
component: ss
(1)      64.463
component: se
(1)      0.7412
```