

Lecture 24

1. Proc SQL
2. Counting rows per subject with SQL and Transpose
3. Proc SQL to combine datasets: variety of merges
4. Proc SQL: fuzzy merge
5. Crossover designs
6. The 2 x 2 Crossover: Wine Study Example
7. Modeling fixed effects: treatment, period, carryover
8. Mixed model
9. Planning crossover designs with R: crossdes

1

Proc SQL

SQL = Standard Query Language (*Query* = question)

Developed at IBM in 1970s, based on a paper by E F Codd:
treat data structures as tables with rows and columns.

SQL designed to work with relational database, collection of tables linked by various IDs

Many commercial and some open-source versions of SQL, following ANSI standards.

Proc SQL incorporates some SQL features into SAS

2

Features of Proc SQL

- No sorting required for BY statements
- Calculate summary statistics for groups of observations and add them to the dataset
- Multiple ways to combine two datasets, including Cartesian product (all combinations of rows)
- Fuzzy merges

3

Proc SQL references

- TJ Harrington (2002) Introduction to SAS Proc SQL
- W Hu (2004) Top Ten Reasons to Use Proc SQL
- K Prairie: *The Essential Proc SQL Handbook*. 2005, SAS Institute
- SAS Help > SAS Products > Base SAS > SAS 9.2 SQL Procedure User's Guide

4

Counting rows per ID with Proc SQL

Family economic data, in long form: should be 6 years of data per family.

90% random sample: *How many annual observations for each family?*

Obs	family_ id	income	year	expenses	debt	cohort	
1	1	66483	1990	49804	no	A	
2	1	69146	1991	65634	no	A	
3	1	74643	1992	61820	no	A	1993 missing
4	1	81710	1994	85504	yes	A	
5	1	86143	1995	75640	no	A	
6	2	17510	1990	21609	yes	B	
7	2	17947	1991	12085	no	B	1992 missing
8	2	20979	1993	22985	yes	B	
9	2	21268	1994	11097	no	B	
10	2	22998	1995	21768	no	B	

5

Proc SQL;

```
create table pubh.M as
select family_id, count(income) as n_years_income
from family_economic_data
group by family_id;
quit ;
```

Create SAS permanent dataset pubh.M

composed of these 2 variables derived from family_economic_data:

- family_ID
- n_years_income, calculated as the number of nonmissing values of income

Do the calculation in groups defined by family_ID.

Ends with quit , not run.

6

Here is the result, dataset pubh.M:

	family_	n_years_
Obs	id	income
1	1	5 <i>should be 6 if complete</i>
2	2	5
3	3	6
4	4	6
5	5	5

If we want this count to be added to original data, then merge M with the original data by family_ID.

7

Or, we could keep all the original variables in the created dataset:

```
Proc SQL;
  create table M2 as
  select * , count(income) as n_years_income * = all variables
  from family_economic_data
  group by family_id;
quit;
```

From SAS log:

```
NOTE: The query requires remerging summary statistics back
with the original data.
```

8

Summary count is merged back:

Obs	family_ id	income	year	expenses	debt	cohort	n_years_ income
1	1	66483	1990	49804	no	A	5
2	1	81710	1994	85504	yes	A	5
3	1	74643	1992	61820	no	A	5
4	1	86143	1995	75640	no	A	5
5	1	69146	1991	65634	no	A	5
6	2	17510	1990	21609	yes	B	5
7	2	17947	1991	12085	no	B	5
8	2	20979	1993	22985	yes	B	5
9	2	22998	1995	21768	no	B	5
10	2	21268	1994	11097	no	B	5
11	3	75198	1994	67012	no	A	6
12	3	62964	1991	64202	yes	A	6
13	3	70957	1993	66799	no	A	6
14	3	57947	1990	54970	no	A	6
15	3	75722	1995	73025	no	A	6

Functions available in Proc SQL:

Function	Definition
AVG, MEAN	mean or average of values
COUNT, FREQ, N	number of nonmissing values
CSS	corrected sum of squares
CV	coefficient of variation (percent)
MAX	largest value
MIN	smallest value
NMISS	number of missing values
PRT	probability of a greater absolute value of Student's t
RANGE	range of values
STD	standard deviation
STDERR	standard error of the mean
SUM	sum of values
SUMWGT	sum of the WEIGHT variable values (table note 1)
T	Student's t value for testing the hypothesis that the population mean is zero
USS	uncorrected sum of squares
VAR	variance

Merging datasets in SQL: Cartesian Product

Join two datasets without conditions in Proc SQL to get all combinations of rows = *Cartesian product*.

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

3 rows in A × 3 rows in B = 9 combinations

Both datasets have ID and MASS variables so second will overwrite first if we don't rename them.

11

```
Proc SQL;
  create table D as
  select * from A(rename=(id=id_A mass=mass_A)),
          B(rename=(id=id_B mass=mass_B));
quit;
```

Create a new table D

using all variables from A and B,

after renaming the mass and ID variables in each dataset.

12

Output data D:

Obs	id_A	color	mass_A	id_B	mass_B	pH
1	12	orange	3650	13	11267	7.8
2	12	orange	3650	14	3568	8.2
3	12	orange	3650	15	4103	5.1
4	13	blue	3877	13	11267	7.8
5	13	blue	3877	14	3568	8.2
6	13	blue	3877	15	4103	5.1
7	15	yellow	4103	13	11267	7.8
8	15	yellow	4103	14	3568	8.2
9	15	yellow	4103	15	4103	5.1

Easy to find discrepant values of mass, missing IDs.

13

Other Merges with SQL: Intersection

No sorting is required, which can speed merging of very large datasets.

Merge observations from A and B with matching IDs, and keep mass measurements separate:

```
Proc SQL;
  create table F as
  select * from A(rename=(mass=mass_A)), B(rename=(mass=mass_B))
  where A.id = B.id ;
```

Since variable ID is in both A and B, need to identify `dataset.variable`

14

Data A			Data B		
id	color	mass	id	mass	pH
12	orange	3650	13	11267	7.8
13	blue	3877	14	3568	8.2
15	yellow	4103	15	4103	5.1

Merge where A.id = B.id

Obs	id	color	mass_A	mass_B	pH
1	13	blue	3877	11267	7.8
2	15	yellow	4103	4103	5.1

Gives the *intersection* of the two datasets: only IDs that appear in both.

Proc SQL has several other ways to combine datasets.

15

Fuzzy merge in Proc SQL

Patients in a study are supposed to have measurements of their weight and blood pressure within 7 days.

Weight data

Obs	id	date	kg
1	33	10MAR2009	78
2	33	12APR2009	81
3	33	15MAY2009	80
4	34	02FEB2009	65
5	34	05JUN2009	66
6	35	21MAY2009	71
7	35	08JUN2009	71
8	35	14AUG2009	70

Blood pressure data

Obs	patient	date	dbp	sbp
1	33	15MAR2009	72	112
2	33	10APR2009	68	105
3	33	30MAY2009	71	110
4	34	07FEB2009	55	95
5	34	15JUN2009	60	99
6	35	21MAY2009	66	110
7	35	06JUN2009	68	105

Merge on variables with different names (ID, patient)
and on observation dates within 7 days.

16

Start by merging on ID = patient

```
Proc SQL;
  create table G1 as
  select *
  from weights(rename=(date=date_wt)), BP (rename=(date=date_BP))
  where weights.id = BP.patient;
```

Use dataset option `rename` to save dates from weights and blood pressures.

Want to check all combinations of observations where ID = patient:

Result has 9 rows for ID=33, 4 for ID=34, and 6 for ID=35

17

Obs	id	date_wt	kg	patient	date_BP	dbp	sbp
1	33	10MAR2009	78	33	15MAR2009	72	112
2	33	15MAY2009	80	33	15MAR2009	72	112
3	33	12APR2009	81	33	15MAR2009	72	112
4	33	10MAR2009	78	33	10APR2009	68	105
5	33	15MAY2009	80	33	10APR2009	68	105
6	33	12APR2009	81	33	10APR2009	68	105
7	33	10MAR2009	78	33	30MAY2009	71	110
8	33	15MAY2009	80	33	30MAY2009	71	110
9	33	12APR2009	81	33	30MAY2009	71	110
10	34	02FEB2009	65	34	07FEB2009	55	95
11	34	05JUN2009	66	34	07FEB2009	55	95
12	34	02FEB2009	65	34	15JUN2009	60	99
13	34	05JUN2009	66	34	15JUN2009	60	99
14	35	21MAY2009	71	35	21MAY2009	66	110
15	35	14AUG2009	70	35	21MAY2009	66	110
16	35	08JUN2009	71	35	21MAY2009	66	110
17	35	21MAY2009	71	35	06JUN2009	68	105
18	35	14AUG2009	70	35	06JUN2009	68	105
19	35	08JUN2009	71	35	06JUN2009	68	105

18

To check whether weight dates and blood pressure dates are within 7 days,
calculate interval:

```
Proc SQL;
  create table G2 as
  select * , ABS(date_wt - date_BP) as interval
  from weights(rename=(date=date_wt)), BP (rename=(date=date_BP))
  where weights.id = BP.patient ;
```

Add comma after * for variable list

ABS is absolute value function

ABS(date_wt - date_BP) is positive number of days between dates

Note: used new names for the dates - in calculating interval

19

Obs	id	date_wt	kg	patient	date_BP	dbp	sbp	interval
1	33	10MAR2009	78	33	15MAR2009	72	112	5
2	33	15MAY2009	80	33	15MAR2009	72	112	61
3	33	12APR2009	81	33	15MAR2009	72	112	28
4	33	10MAR2009	78	33	10APR2009	68	105	31
5	33	15MAY2009	80	33	10APR2009	68	105	35
6	33	12APR2009	81	33	10APR2009	68	105	2
7	33	10MAR2009	78	33	30MAY2009	71	110	81
8	33	15MAY2009	80	33	30MAY2009	71	110	15
9	33	12APR2009	81	33	30MAY2009	71	110	48
10	34	02FEB2009	65	34	07FEB2009	55	95	5
11	34	05JUN2009	66	34	07FEB2009	55	95	118
12	34	02FEB2009	65	34	15JUN2009	60	99	133
13	34	05JUN2009	66	34	15JUN2009	60	99	10
14	35	21MAY2009	71	35	21MAY2009	66	110	0
15	35	14AUG2009	70	35	21MAY2009	66	110	85
16	35	08JUN2009	71	35	21MAY2009	66	110	18
17	35	21MAY2009	71	35	06JUN2009	68	105	16
18	35	14AUG2009	70	35	06JUN2009	68	105	69
19	35	08JUN2009	71	35	06JUN2009	68	105	2

20

Add condition that interval between dates (for same ID) must be ≤ 7 days

Proc SQL;

```
create table G3 as
select * , ABS(date_wt - date_BP) as interval
from weights(rename=(date=date_wt)), BP (rename=(date=date_BP))
where weights.id = BP.patient and (calculated interval LE 7);
```

Must refer to interval as `calculated` interval
because it is not in either data set

21

Obs	id	date_wt	kg	patient	date_BP	dbp	sbp	interval
1	33	10MAR2009	78	33	15MAR2009	72	112	5
2	33	12APR2009	81	33	10APR2009	68	105	2
3	34	02FEB2009	65	34	07FEB2009	55	95	5
4	35	21MAY2009	71	35	21MAY2009	66	110	0
5	35	08JUN2009	71	35	06JUN2009	68	105	2

Last, fix the redundant variables ID and patient

22

```

Proc SQL;
  create table G4 as
  select * , ABS(date_wt - date_BP) as interval
  from weights(rename=(date=date_wt)),
       BP (rename=(date=date_BP patient=id ))
  where weights.id = BP.id and (calculated interval LE 7);

```

Obs	id	date_wt	kg	date_BP	dbp	sbp	interval
1	33	10MAR2009	78	15MAR2009	72	112	5
2	33	12APR2009	81	10APR2009	68	105	2
3	34	02FEB2009	65	07FEB2009	55	95	5
4	35	21MAY2009	71	21MAY2009	66	110	0
5	35	08JUN2009	71	06JUN2009	68	105	2

This is a **fuzzy merge** on ID and visit dates within 7-day interval.

Very difficult in SAS data step.

23

From K Prairie: *The Essential Proc SQL Handbook*, page 8:

Fuzzy logic can be applied to pattern-matching criteria in an SQL-query. SAS functions such as SCAN and CONTAINS allow us to parse a [character] string for the inclusion of various characters. The LIKE operator can be used with wildcard symbols to restrict character string matches to a particular position within a column value. For criteria based on a range, the BETWEEN operator can be used to set the bounds.

PROC SQL may also include the SAS SOUNDEX function in the WHERE clause of a query. This function will match column values that sound similar to the given value.

24

Crossover Designs

In a **crossover design**, subjects get all the treatments in sequence: they *cross over* from one treatment to another.

Advantages: treatments are compared within the same person, which can greatly reduce error variance:

variability between subjects is eliminated from comparison

“Each subject serves as their own control.”

25

Disadvantages:

repeated measurements from each subject means correlated observations

complicated designs are more sensitive to errors in treatment assignment and to missing values.

Carryover: effect of treatment *A* may persist during next treatment *B*

Jones and Kenward (2003) *Design and Analysis of Cross-Over Trials, 2nd Edition*.

26

2 x 2 Crossover Example: Wine Study

Trial to study effects of moderate consumption of wine in adults with diabetes
(*Metabolism Clinical and Experimental*, 2008; 57: 241–245.)

Subjects: 17 adults with type 2 diabetes

Treatments: no alcohol for a month (A = abstinence treatment);
one glass of wine with dinner each evening for a month (W = wine treatment).

Outcome: insulin (fasting serum insulin), measured at the end of each month.

9 participants randomly assigned to sequence AW, 8 to WA.

27

Time interval for treatment is **period**.

2 periods and 2 treatments give only 2 sequences: AW and WA.

Assign equal numbers to each sequence at random.

Compare demographics and baseline values between sequence groups.

If no problems, summary is treatment means \pm SE

28

Data from the first 6 participants:

Obs	Subject	month	Wine	sequence	Insulin
1	CK	1	0	AW	6
2	CK	2	1	AW	6
3	DH	2	0	WA	10
4	DH	1	1	WA	8
5	DP	1	0	AW	24
6	DP	2	1	AW	16
7	DS	2	0	WA	20
8	DS	1	1	WA	16
9	DS2	2	0	WA	14
10	DS2	1	1	WA	16
11	EM	1	0	AW	14
12	EM	2	1	AW	16

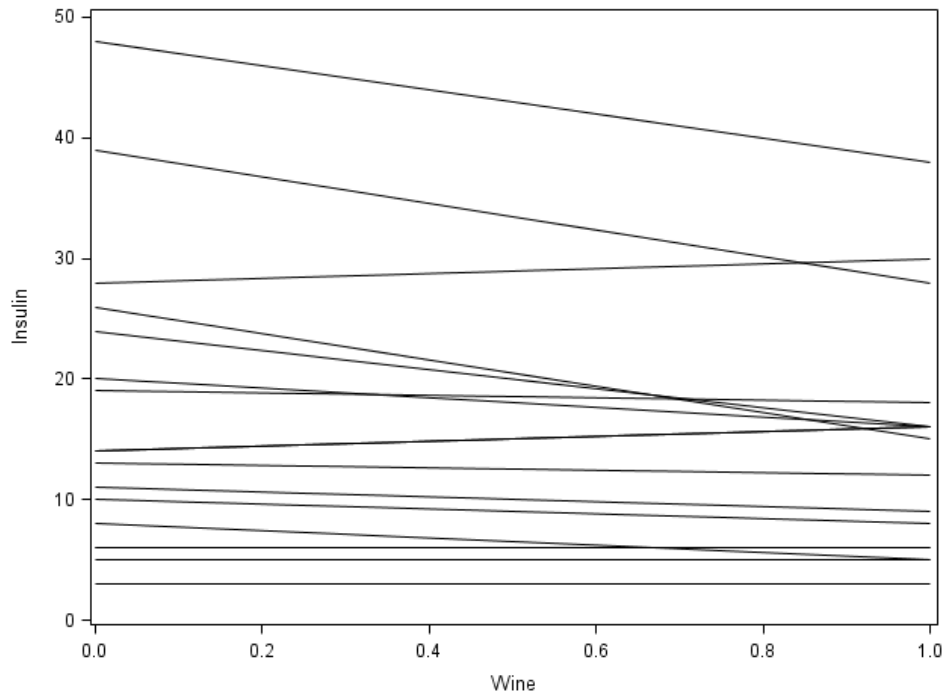
29

Make longitudinal plots by treatment and by time

```
Proc SGplot noautolegend data=ph6470.wine; suppress legend
  series x=wine y=insulin /
  group=subject LINEATTRS= (pattern=1 color="black");
```

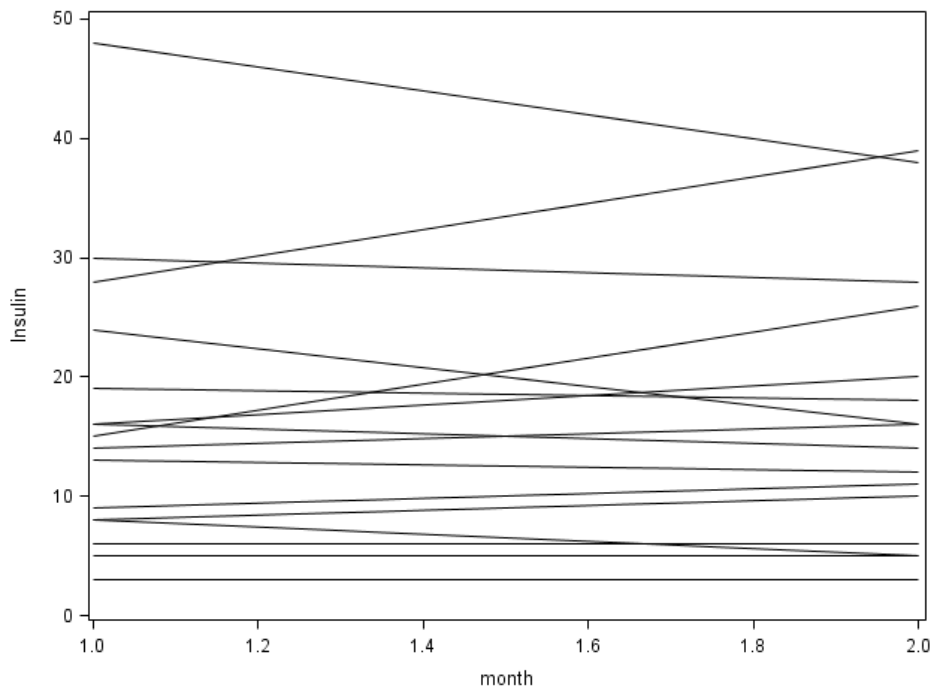
```
Proc SGplot noautolegend data=ph6470.wine;
  series x=month y=insulin /
  group=subject LINEATTRS= (pattern=1 color="black");
```

30



What do we hope to see here?

31



What do we hope to see here?

32

Demographic table

Demographic table reports baseline characteristics (age, gender, weight, etc.) of study subjects.

Usually compare treatment groups at baseline, to show randomization worked.

Here, everyone gets all treatments, so everyone is in all the treatment groups.

Report and compare sequence groups:

Random assignment to the two sequences, AW and WA, *should* result in similar characteristics.

33

Model for mean fixed effects

Fixed effects in crossover designs:

- t_j is the effect of treatment j
- p_i is the effect of period i (which is month i in this study)
- c_j is the **carryover effect** from treatment j : any persisting effects during period 2 from treatment j in period 1

Mean fixed effects:

Sequence	Period 1	Period 2
AW	$t_A + p_1$	$t_W + p_2 + c_A$
WA	$t_W + p_1$	$t_A + p_2 + c_W$

34

Why not a paired t -test?

Each person gets both treatments. How about basing analysis on within-person differences between treatments?

Paired t -test uses the differences

$$d_k = (\text{response to W}) - (\text{response to A})$$

for each subject k .

Test statistic is $\bar{d}/SE(\bar{d})$.

35

Assume each subject gets mean outcome:

Sequence	Number	Period 1	Period 2	<i>Mean Difference W – A</i>
AW	n_{AW}	$t_A + p_1$	$t_W + p_2 + c_A$	$t_W - t_A + p_2 - p_1 + c_A$
WA	n_{WA}	$t_W + p_1$	$t_A + p_2 + c_W$	$t_W - t_A - p_2 + p_1 - c_W$

Want \bar{d} to estimate $t_W - t_A$

Summing the last column for all subjects and dividing to get \bar{d} :

$$t_W - t_A + \frac{n_{AW}(p_2 - p_1) - n_{WA}(p_2 - p_1) + n_{AW}c_A - n_{WA}c_W}{n_{AW} + n_{WA}}$$

What happens to \bar{d} if $n_{AW} \neq n_{WA}$ or $c_A \neq c_W$?

36

If the design is badly unbalanced or if 2 carryover effects differ, then \bar{d} is biased and the paired t -test is not valid.

How do you prevent unbalanced design?

Check carryover effects before comparing treatment effects.

37

Longitudinal model for crossover

Model correlation within subjects with a random intercept.

mixed-effects model for the response from subject k in sequence i at month j :

$$y_{ijk} = (\beta_0 + b_k) + t_j + p_i + c_{i'} + \varepsilon_{ijk} ,$$

- β_0 = overall (mean) intercept,
- b_k = random intercept for subject k , with $\text{Normal}(0, \sigma_b^2)$ distribution,
- t_j = effect of the treatment j (**wine**),
- p_i = effect of period i (**month**),
- $c_{i'}$ = carryover effect of the treatment in the preceding period (**sequence**),
- and the errors ε_{ijk} are independent $\text{Normal}(0, \sigma_\varepsilon^2)$, and independent of random effects $\{b_k\}$.

38

```

Proc Mixed data=wine;
  class sequence subject month wine;
  model insulin = wine month sequence
    / solution ddfm=kenwardroger; recommended for crossover
  random intercept / subject = subject v vcorr ;
  lsmeans wine / diff;

```

wine is treatment effect

month is period effect

sequence is carryover effect

39

We get a 2×2 correlation matrix for the two responses from a subject:

Estimated V Correlation
 Matrix for Subject CK

Row	Col1	Col2
1	1.0000	0.9187
2	0.9187	1.0000

Within-subject correlation estimate $r = .92$

Crossover design really improved efficiency.

40

The Mixed Procedure

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
sequence	1	15	0.35	0.5614
month	1	15	0.17	0.6885
wine	1	15	6.20	0.0250

F-test for `sequence` is test for $c_A = c_W$ (equal carryover effects).

F-test for `month` is test for $p_1 = p_2$ (equal period effects).

F-test for `wine` is test for $t_A = t_W$, comparing treatments adjusted for carryover and period effects.

In the 2×2 crossover design, sequence is confounded with both treatment×period interaction and carryover. So only one test to check all three.

41

Main comparison between treatments

`lsmeans wine / diff ;`

Least Squares Means

Effect	Wine	Estimate	Standard Error	DF	t Value	Pr > t
Wine	0	17.4028	2.7812	15	6.26	<.0001
Wine	1	14.6111	2.7812	15	5.25	<.0001

Differences of Least Squares Means

Effect	Wine	Wine	Estimate	Standard Error	DF	t Value	Pr > t
Wine	0	1	2.7917	1.1213	15	2.49	0.0250

The wine increased insulin by 2.8 ± 1 on average.

42

Planning a crossover study

Balance for first-order carryover effects. Check that

- Each treatment is given the same number of times
- Each treatment is given *first* the same number of times, *second* the same number of times, etc.
- For any two treatments, T_j directly follows T_k the same number of times T_k directly follows T_j

Randomization for a crossover study:

Select a set of treatment sequences that satisfy these criteria.

Randomly assign equal numbers of participants to each sequence.

43

2 treatments: AB or BA

3 treatments: $3 \times 2 \times 1 = 6$ different sequences

4 treatments: $4 \times 3 \times 2 \times 1 = 24$ different sequences

5 treatments: 120 sequences

Assigning equal numbers to *every* sequence usually requires a large sample.

Choose a subset of sequences that satisfy the design criteria. Common approaches:

- mutually orthogonal Latin squares
- Williams design: use the minimum number of sequences to give balance, with each treatment following all others.

44

A **latin square** is an $n \times n$ table filled with n treatments so that each treatment appears exactly once in each row and exactly once in each column. Here is a 4×4 latin square:

	[,1]	[,2]	[,3]	[,4]
[1,]	1	2	3	4
[2,]	2	1	4	3
[3,]	3	4	1	2
[4,]	4	3	2	1

Which treatments does 2 precede and follow? Balanced for carryover?

Mutually orthogonal latin squares: if any two of them are superimposed, the resulting array will contain each ordered pair of treatments (i, j) exactly once.

45

R package `crossdes` gives Williams designs or mutually orthogonal latin squares.

```
get.plan {crossdes} R Documentation
```

Menu-Driven Construction of Carryover Balanced Experimental Designs

Description

This menu based function constructs and randomizes simple experimental designs for repeated measurements with one or two block variables. It is assumed that each subject is assigned to each treatment at most once. A maximum number of subjects in the study is also requested. Five possible construction methods available. These construction methods and the characteristics of the resulting designs are described in Wakeling and MacFie (1995). See also Jones and Kenward (1989), Ch. 5, for a discussion of these designs. The function is demonstrated in more detail in Sailer (2005).

Usage

```
get.plan(trt, k = trt, maxsub = 1000, random = TRUE)
```

Arguments

`trt` An integer > 1 , giving the number of treatments.

`k` An integer in $\{2, \dots, trt\}$ giving the number of periods.

`maxsub` The maximum number of subjects available.

`random` Logical flag. If TRUE, the design is randomized after construction.

R is free open-source statistical software that runs on Macs, PCs, Linux, Unix.

<http://www.R-project.org> Install R, then install `crossdes`

46

Suppose 4 treatments, 4 periods.

```
> get.plan (trt=4,k=4, random=F )
```

Possible constructions and minimum numbers of subjects:

	1	2	3	
Method:	all.combin	williams	des.MOLS	
Number:	24	4	12	<i>block size</i>

Please choose one of the following constructions

1: all.combin

2: williams

3: des.MOLS

4: Exit

Selection: 2

des.MOLS selected. How many 'replicates' do you wish (1 - 1)?

Selection: 1

47

We can recruit 12 subjects:

Rows represent subjects, columns represent periods.

	[,1]	[,2]	[,3]	[,4]
[1,]	1	2	3	4
[2,]	2	1	4	3
[3,]	3	4	1	2
[4,]	4	3	2	1
[5,]	1	3	4	2
[6,]	2	4	3	1
[7,]	3	1	2	4
[8,]	4	2	1	3
[9,]	1	4	2	3
[10,]	2	3	1	4
[11,]	3	2	4	1
[12,]	4	1	3	2

Mutually orthogonal latin squares, use 12 different sequences—one for each subject. One missing value or dropped subject unbalances the design.

48

However, the Williams design uses only 4 sequences:

Rows represent subjects, columns represent periods.

	[,1]	[,2]	[,3]	[,4]
[1,]	1	2	4	3
[2,]	2	3	1	4
[3,]	3	4	2	1
[4,]	4	1	3	2

Which treatments does 2 precede and follow?

Randomly assign 3 subjects to each sequence. This is more robust to drop-outs or missing values.