Calculating G-theory Quantities Using SAS

In this document I explain how to use SPSS to obtain G-theory quantities in SAS.

SAS does not have the specialized programs needed to obtain most G-theory quantities. Because of this, I created Excel spreadsheets to calculate these c using the variance component estimates obtained from SAS.

The spreadsheet examples are based on data from the tables in Chapter 10 and available in the SAS data sets "**gtheory1.sas7bdat**," **gtheory1T.sas7bdat**," **gtheory2.sas7bdat**," and "**gtheory2T.sas7bdat**" for the one-facet untransposed and transposed data and two-facet untransposed and transposed data, respectively.

Although the Excel spreadsheets show the variance components from these examples, they can easily be modified to use with other data. To do so, simply type or paste in the variance component estimates from the other data into the variance component column, and type in the number of levels for these variance components in the column labeled "levels". The other quantities will be updated automatically when these values are changed.

Similarly, although the rows in the spreadsheet are labeled as "raters," "tasks," and their interactions, this was done simply to correspond to the textbook examples. These labels can be changed to correspond to any type of facet, such as occasions or items.

Below I provide the syntax to obtain variance component estimates. These estimates can then be entered into the Excel spreadsheets to obtain the G-theory quantities. I first show the commands for a one-facet design and then the commands for a two-facet design.

I begin by providing syntax to transpose the data set.

One-facet Design

Transposing the Data

Most data sets will need to be transposed prior to obtaining the variance components. This is because data is usually entered in "wide" format, with one row for each person, as shown below and in the SAS data set "**gtheory1.sas7bdat**."

VIEWTABLE: Chap10.Gtheory1

	person	rater1	rater2	rater3
1	1	3	5	4
2	2	3	5	5
3	3	1	1	2
4	4	3	1	2
5	5	6	5	5
6	6	1	2	5
7	7	3	5	5
8	8	5	4	5
9	9	3	1	3
10	10	4	3	5
11	11	3	4	3
12	12	2	2	2

To obtain the correct variance components in SAS, the data must be transposed to create a variable called “rater” with three levels corresponding to the three raters, and a variable “score” that contains the scores for each person from each rater. The new data set will have three rows for each person, one for each rater and score, as shown below and in the SAS data set “**gtheory1T.sas7bdat”** (data in the screenshot below are cut off at the bottom):

VIEWTABLE: Chap10.Gtheory1t

	person	rater	score
1	1	1	3
2	1	2	5
3	1	3	4
4	2	1	3
5	2	2	5
6	2	3	5
7	3	1	1
8	3	2	1
9	3	3	2
10	4	1	3
11	4	2	1
12	4	3	2
13	5	1	6
14	5	2	5
15	5	3	5
16	6	1	1
17	6	2	2
18	6	3	5
19	7	1	3
20	7	2	5
21	7	3	5
22	8	1	5
23	8	2	4
24	8	3	5

The data can be transposed using **proc transpose**, but I find it easier to do this using arrays within a **data step**, as illustrated in the syntax below.

If the data are not already sorted by person, sort them first using **proc sort**.

proc sort data=gtheory1**; by** person**; run;**

In the next set of commands, I set the data set "**gtheory1**" into a new data set ("**gtheory1T**"). The original variables rater1, rater2, and rater3, are dropped. These are not needed because a new variable "score" is created that contains the three scores. The looping index "i" is also dropped.

The array "scores" contains the variables for the three rater scores shown in the untransposed data set above.

For each record, the do loop will loop through the three scores in the array "scores" and create two new variables. The variable "rater" will be equal to the value of the looping index "i" (1,2,3) and the variable "score" will be equal to the value of the corresponding element of the array (rater1, rater2, or rater3).

The subcommand **output** causes the new variables to be output to the new file "**gtheory1T**" at each iteration.

The subcommand **end** ends the do loop.

```
data gtheory1T (drop= i rater1 rater2 rater3); set gtheory1;
  array scores (3) rater1 rater2 rater3;
   do i = 1 to 3;
   rater = i;
   score = scores(i);
   output;
  end;
 run;
```

Running this syntax will create the new data set "gtheory1T" shown above, with three rows for each person corresponding to the three raters and their scores.

Obtaining Variance Components for the One-facet Data

Proc varcomp can be used to obtain variance components in SAS, based on the transposed data.

```
proc varcomp data=gtheory1T;
  class person rater ;
  model score = person|rater;
run;
```

The command syntax is the same as that for **proc glm** or **proc anova**. The **class** subcommand indicates the two independent, or classification variables, "person" and "rater." The **model** subcommand specifies that the dependent variable is "score," predicted by the variables "person" and "rater," and their interaction (a full factorial model is specified using the symbol |). Classification variables are treated as random by default.

Running this syntax will result in the output below (note that I have omitted some of the SAS output).

The Class Level Information specifies that there are 3 raters and 12 people.

Class Level Information		
Class	**Levels**	**Values**
rater	3	1 2 3
person	12	1 2 3 4 5 6 7 8 9 10 11 12

The variance components shown in the table below are the same as those in the spreadsheet for the one-facet crossed design and in Table 10.3 in the text (NOTE THAT IN PRINTINGS 1 AND 2 OF THE BOOK, THESE VALUES ARE INCORRECT IN THE TABLE. GO TO THE ERRATA SHEET ON THIS WEBSITE TO OBTAIN THE CORRECTED TABLE).

The variance component values can be entered into the spreadsheet in the appropriate column.

MIVQUE(0) Estimates	
Variance Component	**score**
Var(rater)	0.08838
Var(person)	1.28283
Var(rater*person)	0.96717
Var(Error)	0

Two-facet Design

Transposing the Data

Commands to transpose the data for the two-facet design are slightly more complex than those for the one-facet design. The data for the values in Table 10.4 in the text are in the data set "**gtheory2.sas7bdat**" The untransposed data are shown below:

2]

olutions Window Help

	person	R1Task1	R1Task2	R1Task3	R2Task1	R2Task2	R2Task3	R3Task1	R3Task2	R3Task3
1	1	3	3	3	5	5	5	4	4	5
2	2	3	2	4	5	5	5	5	3	5
3	3	1	1	2	1	3	3	2	3	4
4	4	3	2	2	1	3	2	2	1	1
5	5	6	6	6	5	5	6	5	4	3
6	6	1	2	1	2	1	1	5	3	2
7	7	3	2	3	5	1	2	5	4	5
8	8	5	4	6	4	3	5	5	4	6
9	9	3	2	3	1	3	1	3	1	3
10	10	4	3	4	3	3	3	5	4	2
11	11	3	2	5	4	3	4	3	2	3
12	12	2	1	1	2	2	1	2	2	4

Each person has nine scores: ratings from three raters (labeled R1, R2, R3) on three tasks (Task1, Task2, and Task3). The label R1Task1 refers to the rating by Rater 1 on Task 1, and so on.

What is needed is a transposed data set with new variables for rater, task, and score. This can be obtained by running the **data step** syntax below.

As before, I set the data set "**gtheory2**" into a new data set ("**gtheory2T**"). I use the **keep** command to specify that the new variables "person," "rater," and "score" should be kept, along with the existing variable "person."

I create three arrays: one for the scores of each rater on the three tasks.

For each record, the do loop will loop through the three arrays (R1, R2, and R3) sequentially and create three new variables.

The variable "rater" is set sequentially to values of 1, 2, and 3 for the corresponding array.

The variable "task" is set to the index "i" (1,2,3) as SAS loops through the three tasks in each array.

The variable "score" is set to the value of the corresponding element of each array (e.g., R1Task(i)).

The subcommand **output** causes the new variables to be output to the new file "gtheory2T" at each iteration.

The subcommand **end** ends the do loop.

```
data gtheory2T (keep=person rater task score); set gtheory2;
array R1 (3) R1Task1 R1Task2 R1Task3;
array R2 (3) R2Task1 R2Task2 R2Task3;
array R3 (3) R3Task1 R3Task2 R3Task3;

do i = 1 to 3;
        task= i;
        rater=1;
        score=R1(i);
  output;
        rater=2;
        score = R2(i);
  output;
    rater=3;
        score=R3(i);
  output;
  end;
  run;
```

Running this syntax will yield the transposed data set **gtheory2T** shown below:

y2t]
ns Window Help

	person	task	rater	score
1	1	1	1	3
2	1	1	2	5
3	1	1	3	4
4	1	2	1	3
5	1	2	2	5
6	1	2	3	4
7	1	3	1	3
8	1	3	2	5
9	1	3	3	5
10	2	1	1	3
11	2	1	2	5
12	2	1	3	5
13	2	2	1	2
14	2	2	2	5
15	2	2	3	3
16	2	3	1	4
17	2	3	2	5
18	2	3	3	5
19	3	1	1	1
20	3	1	2	1
21	3	1	3	2
22	3	2	1	1
23	3	2	2	3
24	3	2	3	3
25	3	3	1	2
26	3	3	2	3
27	3	3	3	4
28	4	1	1	3
29	4	1	2	1
30	4	1	3	2
31	4	2	1	2
32	4	2	2	3
33	4	2	3	1
34	4	3	1	2
35	4	3	2	2
36	4	3	3	1
37	5	1	1	6
38	5	1	2	5

Obtaining the Variance Components for the Two-facet Data

To obtain the variance components, just add the “task” variable to the previous syntax for **proc varcomp**, as shown below.

```
proc varcomp data=gtheory2T;
  class person rater task ;
```

```
  model score = person|rater|task;
run;
```

Running this syntax will result in the output below (note that I have omitted some of the output):

Class Level Information		
Class	**Levels**	**Values**
person	12	1 2 3 4 5 6 7 8 9 10 11 12
rater	3	1 2 3
task	3	1 2 3

The table above shows that there are 12 people, three raters, and three tasks.

The variance components shown below are the same as those in Table 10.4 in the text and in the Excel spreadsheet on the sheet labeled “Two-facet crossed.”

MIVQUE(0) Estimates	
Variance Component	**score**
Var(person)	1.03451
Var(rater)	-0.0037879
Var(person*rater)	0.47138
Var(task)	0.06103
Var(person*task)	0.11953
Var(rater*task)	0.01221
Var(person*rater*task)	0.64057
Var(Error)	0