

14. MISCELLANEOUS PROGRAMMING MATERIALS

REVISED FALL 1999

```

** THIS PROGRAM SHOWS HOW TO MERGE 1 RECORD ONTO **
** EVERY RECORD IN A SECOND DATASET                ** ;

```

```

PROC SUMMARY DATA=SC.CLASS NWAY;
  VAR AGE ;
  OUTPUT OUT=OUTI(KEEP=MAGE) MEAN=MAGE;
RUN;

```

NOTE: The data set WORK.OUT1 has 1 observations and 1 variables.
NOTE: The PROCEDURE SUMMARY used 0.83 seconds.

```

DATA TWO;
  SET SC.CLASS ;
  IF _N_ =1 THEN SET OUT1 ;
RUN;

```

NOTE: The data set WORK.TWO has 6 observations and 6 variables.
NOTE: The DATA statement used 0.56 seconds.

```

PPOC PRINT
TITLE 'MERGE EXAMPLE'
RUN;

```

NOTE: The PROCEDURE PRINT used 0.11 seconds.

MERGE EXAMPLE

NAME	SEX	AGE	HT	WT	MAGE
CHRISTIANSEN	M	37	71	195	25.2
HOSKING J	M	31	70	160	25.2
HELMS R	M	41	74	195	25.2
PIGGY M	F	.	48	.	25.2
FROG K	M	3	12	1	25.2
GONZO		14	25	45	25.2

```
** MERGE MEDIAN HT AND WT ONTO EVERY RECORD IN DATA SET CLASS      **
** THEN CREATE AN INDICATOR VARIABLE FOR HT WHERE:                  **
** 0= LESS THAN OR EQUAL TO THE MEDIAN, 1= GREATER THAN THE MEDIAN ** ;
```

```
PROC UNIVARIATE DATA=SC.CLASS NOPRINT;
  VAR HT WT ;
  OUTPUT OUT=OUT1 MEDIAN=MEDHT MEDWT
  MEAN = MEANHT MEANWT ;
RUN;
```

NOTE: The data set WORK.OUT1 has 1 observations and 4 variables.
NOTE: The PROCEDURE UNIVARIATE used 1.68 seconds.

```
PROC PRINT DATA=OUT1 NOOBS;
TITLE 'PRINT OUT1';
RUN;
```

NOTE: The PROCEDURE PRINT used 0.13 seconds.

```
DATA ALL;
  SET SC.CLASS(DROP=SEX AGE);
  IF _N_=1 THEN SET OUT1;

  IF .Z < HT <= MEDHT THEN HTCAT=0;
  ELSE IF HT>MEDHT THEN HTCAT=1;
RUN;
```

NOTE: The data set WORK.ALL has 6 observations and 8 variables.

NOTE: The DATA statement used 0.78 seconds.

```
PROC PRINT DATA=ALL;
  ID NAME ;
TITLE 'PRINT ALL' ;
RUN;
```

NOTE: The PROCEDURE PRINT used 0.14 seconds.

PRINT OUTI

MEANHT	MEANWT	MEDHT	MEDWT
50	119.2	59	160

PRINT ALL

NAME	HT	WT	MEANHT	MEANWT	MEDHT	MEDWT	HTCAT
CHRISTIANSEN	71	195	50	119.2	59	160	1
HOSKING J	70	160	50	119.2	59	160	1
HELMS R	74	195	50	119.2	59	160	1
PIGGY M	48	.	50	119.2	59	160	0
FROG K	12	1	50	119.2	59	160	0
GONZO	25	45	50	119.2	59	160	0

**** THIS PROGRAM SHOWS HOW TO SELECT AN EXACT SIZE RANDOM SAMPLE ** ;**

```
DATA ONE;  
  SET SC.EVENT ;
```

```
  RAND=RANUNI(O) ;  
RUN;
```

NOTE: The data set WORK.ONE has 1500 observations and 4 variables.

NOTE: The DATA statement used 6.44 seconds.

```
PROC SORT ; BY RAND;  
RUN;
```

The data set WORK.ONE has 1500 observations and 4 variables.

NOTE: The PROCEDURE SORT used 2.83 seconds.

```
  ** RANDOMLY SELECT 50 RECORDS ** ;
```

```
DATA TWO;  
  SET ONE;  
  IF N <=50;  
RUN;
```

NOTE: The data set WORK.TWO has 50 observations and 4 variables.

NOTE: The DATA statement used 0.75 seconds.

**** THIS PROGRAM SHOWS HOW TO SELECT AN EXACT SIZE RANDOM SAMPLE ** ;**

```
DATA ONE ;  
  SET SC.EVENT;
```

```
  ** N=DATASET SIZE + 1 **  
  ** K=DESIRED SAMPLE SIZE ** ;
```

```
  RETAIN K 50 N 1501 ;
```

```
  N= N - 1 ;  
  IF RANUNI(O) < K/N ;  
  K=K-1 ;  
RUN;
```

NOTE: The data set WORK.ONE has 50 observations and variables.

NOTE: The DATA statement used 7.33 seconds.

```
** THIS PROGRAM SHOWS HOW TO SELECT A SAMPLE WHICH **  
** IS AN APPROXIMATE PROPORTION OF THE TOTAL POPULATION **;
```

```
DATA ONE ;  
SET SC.EVENT END=EOF ;  
N + 1 ;  
IF (EOF) THEN PUT 'N= ' N ;  
  
** OUTPUT 20% OF THE OBSERVATIONS: THE RANUNI FUNCTION **  
** RETURNS A PSEUDO RANDOM NUMBER THAT IS LESS THAN X **  
** ABOUT 20% OF THE TIME **;  
  
IF RANUNI(O)<=20 THEN OUTPUT;  
RUN;  
  
N= 1500
```

NOTE: The data set WORK.ONE has 282 observations and 4 variables.
NOTE: The DATA statement used 10.43 seconds.

```
** THIS PROGRAM SHOWS HOW TO SELECT A STRATIFIED EXACT **  
** SIZE RANDOM SAMPLE **;
```

```
** STEP 1: FIND # OF OBSERVATIONS IN EACH STRATA OR BY GROUP **;
```

```
PROC SUMMARY DATA=SC.EVENT NWAY ;  
  CLASS TRT ;  
  VAR CHD ;  
  OUTPUT OUT=COUNTS N=N ;  
RUN;
```

NOTE: THE DATA SET WORK.COUNTS HAS 2 OBSERVATIONS ANE 4 VARIABLES.
NOTE: THE PROCEDURE SUMMARY USED 4.44 SECONDS.

```
PROC SORT DATA=SC.EVENT OUT=EVENT;  
  BY TRT;  
RUN;
```

NOTE: THE DATA SET WORK.EVENT HAS 1500 OBSERVATIONS AND 3 VARIABLES.
NOTE: THE PROCEDURE SORT USED 0.93 SECONDS.

```
** STEP 2: SELECT 25 OBSERVATIONS FROM EACH STRATA ** ;
```

```
DATA ONE;  
  MERGE EVENT  
    COUNTS ;  
  BY TRT;
```

```
** K=DESIPED SAMPLE SIZE ** ;
```

```
  RETAIN K;  
  IF FIRST.TRT THEN K=25;  
  
  IF RANUNI(O) < K THEN DO;  
    OUTPUT ;  
    K=K-1;  
  END;  
  N=N-1;  
RUN;
```

NOTE: THE DATA SET WORK.ONE HAS 50 OBSERVATIONS AND VARIABLES.
NOTE: THE DATA STATEMENT USED 1.59 SECONDS.

STATISTICAL PROCEDURES

PROC FREQ

Tests the existence or the strength of association among variables.

- ❖ CHI-SQUARE
- ❖ MANTEL-HANSEL CHI-SQUARE
- ❖ FISHER'S EXACT TEST
- ❖ KENDALL'S TAU

PROC CORR

Computes correlation coefficients between numeric variables.

- ❖ PEARSON PRODUCT-MOMENT CORRELATION
- ❖ SPEARMAN'S RANK ORDER CORRELATION
- ❖ PEARSON PARTIAL CORRELATION
- ❖ SPEARMAN'S PARTIAL RANK ORDER CORRELATION
- ❖ CRONBACH'S COEFFICIENT ALPHA
- ❖ PROC CORR can also output both correlation and covariance matrices.

PROC T-TEST

Performs a two sample t-test for testing the hypothesis that the means of two groups of independent and normally distributed observations are equal. T-TEST also test for unequal variances.

PROC ANOVA

ANOVA is one of several SAS procedures available for analysis of variance.

- ❖ ANOVA is designed for balanced data
- ❖ if data is unbalanced use PROC GLM

PROC GLM

GLM uses the method of least squares to fit linear models. GLM can be used for many different type of analysis including:

- ❖ simple regression
- ❖ multiple regression
- ❖ analysis of variance, especially for unbalanced designs
- ❖ analysis of covariance
- ❖ weighted regression
- ❖ polynomial regression
- ❖ multiple analysis of variance(MANOVA)
- ❖ repeated measures analysis of variance

PROC REG

REG is a general purpose procedure for least square linear regression that:

- ❖ handles multiple model statements
- ❖ provides nine model selection methods
- ❖ test linear and multivariate hypotheses
- ❖ produces collinearity diagnostics
- ❖ saves estimates, predicted values, residuals, and other diagnostic statistics

PROC MIXED

PROC MIXED fits mixed linear models (i.e., fixed and random effects models). A mixed model is a generalization of the standard linear model used in the GLM procedure, the generalization being that you can analyze data generated from several sources of variation instead of just one. The inferential statistics are essentially the same, but their scope is broader. The MIXED procedure is designed for easy accessibility to a wide variety of mixed models.

With the ability to accommodate multiple random effects of varying types, the MIXED procedure provides full accessibility to the popular mixed models methodology. A wide variety of common statistical analyses, including split-plot designs, repeated measures, random coefficients, best linear unbiased predictions, shrinkage estimators, and unequal variances, are easily specified as a mixed model. In the style of the GLM procedure, PROC MIXED fits the specified mixed linear model and produces the appropriate statistics and standard errors. PROC MIXED supports:

- ❖ Covariance structures including simple random effects, compound symmetry, unstructured, AR(1), spatial and Toeplitz
- ❖ GLM-style grammar
- ❖ correct standard errors for all specified estimable linear combinations of fixed and random effects, appropriate t-tests and F-tests
- ❖ Subject and group effects that allow blocking and heterogeneity, respectively
- ❖ REML, ML, and MIVQUE estimation methods implemented with a Newton-Raphson algorithm
- ❖ incomplete data
- ❖ appropriate F-tests in the presence of nonstandard error structures

PROC NLIN

NLIN computes least squares or weighted least squares estimates of the parameters of a nonlinear model.

PROC CATMOD

Fits linear models to functions of categorical response variables. CATMOD is the most general tool for categorical data analysis. CATMOD uses:

- ❖ weighted least squares estimation
- ❖ maximum likelihood estimation

CATMOD can be used for:

- ❖ linear modeling
- ❖ log-linear modeling
- ❖ logistic regression
- ❖ repeated measures analysis

PROC LOGISTIC

Fits linear logistic regression models for binary or ordinal response data with maximum likelihood methods

- ❖ performs stepwise regression
- ❖ provides regression diagnostics

PROC LIFETEST

Computes nonparametric estimates of the survival distribution and computes rank tests for association of the response variable with other variables

- ❖ can be used with data that is right censored

PROC PHREG

- ❖ PHREG performs regression analysis on survival data based on the COX proportional hazards model.
- ❖ COX's model is widely used in the analysis of survival data to explain the effect of explanatory variables on survival times.
- ❖ PHREG can also be used to perform conditional logistic regression for matched case-control studies.
- ❖ Time dependent covariates can be used with PHREG.

Defensive Programming Tips

I. Creating SAS Files from Raw Data(retrieval)

A. check the number of records

- is the number greater than the last retrieval
- is the number the same as in the database retrieved from(not including delete records)
- check for invalid IDs or invalid records

B. check variables and structure of the file

- is the number of variables the same as the last retrieval
- numeric variables that should have a TIME5. format must have a length ≥ 4
- SIR dates must have a length ≥ 4
- numeric variables that have, or can potentially have, decimals must have a length of 8
- print out some records
- PROC MEANS on all numeric variables, including N, NMISS MIN, & MAX
- PROC CONTENTS

II. Creating SAS Files from other SAS Files

A. be aware of the structure and contents of input files

- one record per subject or multiple records per subject
- how are input files sorted
- if a merge is being done:
 - do the by variables have the same length
 - was the by statement included
 - is there a repeat of by values

B. check the number of records input/output

- be aware of the number of records on each of the input files
- count the number of records on each input file
- count the number of records being output
- if a merge is being done count the number of matches and non-matches
- check related output to make sure that the Ns make sense
- output & print problem records

C. creating new variables

- are variables being created in the same way as in related jobs.
- how are missing values being coded
- is the appropriate length specified
- to store information as both numeric and character, use two variables
- use the PUT function for numeric-to-character conversions, and use the input function to perform character-to-numeric conversions. Avoid default type conversions.
- are assignment statements case sensitive

D. checking the file

- print out some records(10-25 is usually adequate)
- looking at the listing:
 - is the file structured as expected
 - did a merge/concatenation work as expected
 - have new variables been created correctly
 - is the number of variables on the output file the same as expected
 - are the inclusions/exclusions correct
- produce descriptive statistics for new variables (and for old variables where necessary)
 - PROC MEANS for numeric variables
 - PROC FREQ for character/discrete variables
 - PROC CONTENTS

III. Checking SAS Output

A. formatting

- are formats specified correctly
- is the format wide enough
- are upper and lower bounds for ranges specified correctly
- if you are using a picture format, make sure you understand how pictures work. they are a common source of error.
- if you use a format specified in a previous program make sure that it is still correct; it may no longer be wide enough or cover the range of the data

B. checking the numbers

- check the output carefully before turning over to the requester
- do the numbers make sense(if unsure, run a PROC FREQ or PROC MEANS)
- do numbers add up properly; for example, do the number of cardiovascular & non-cardiovascular deaths equal the number of total deaths
- are numbers on related tables consistent
- are numbers consistent with previous requests
- how are missing values being handled

C. check the appearance of the table

- do titles clearly indicate the contents of the output to the requestor ; if not, more documentation is probably necessary
- are labels meaningful and spelled correctly
- are values for discrete variables ordered correctly
- are all rows/columns included that should be

IV. General Checks

A. Check for General Errors under MVS

- did program abort before finishing
- were new data sets created as expected(was an old data set written over)

B. Check the SAS Log

- carefully examine the contents of the SAS log
- was each step executed without error
- after a data step check for:
 - is the number of observations as expected
 - missing values generated
 - character/numeric conversions
 - invalid data
 - more than 1 data set has repeat of by values
 - variable uninitialized
- look for "WARNING" and "NEVER REFERENCED" messages

C. If this job was a drop-in, is it still a drop-in