CMU MSP 36-602: SAS Macros 5: CALL EXECUTE

Howard Seltman, April 15, 2019

I. **CALL EXECUTE() statement(s) inside a DATA step**

A. **Purpose**: Construct PROC steps (or other DATA steps) from within a DATA step

B. CALL EXECUTE() is useful when you only know exactly what code needs to be run after you are inside a DATA step. This does what %LET and CALL SYMPUTX cannot: If you foolishly try to define a macro variable with %LET **inside** a DATA step, it is actually set (once) **before** the DATA step is run. That is always equivalent to putting the %LET statements before the DATA step. And, while CALL SYMPUTX() can do the equivalent of %LET based on current values of data variable while the DATA step is running, it cannot do any more complex MACRO coding.

   CALL EXECUTE can construct complex code during a DATA step, including code that depends on the values of the data in the PDV.

C. **How to use CALL EXECUTE**: The syntax is `CALL EXECUTE(someString);`

   a. The string is usually constructed by using the CAT() or CATT() function or the "||" concatenation operator to paste together some of the following into one long string

      i. constant strings surrounded by ***single*** quotes
      ii. data value currently in the PDV
      iii. macro variables (but only their values as set before the DATA step runs).

      The string usually consists of several semicolon terminated statements which together comprise one or more OPTION or TITLE statements, DATA steps, PROC steps, and/or macro statements allowed in open code (i.e., %LET and/or %PUT statements or %myMacro macro invocations). The concatenation is done dynamically while the DATA step is running, which is normally once for each line of the dataset.

   b. Alternately, the string can be an (unquoted) dataset string variable name, in which case the string itself (in the data) should be some SAS code.

   c. Aside: CAT(), CATT() and CATS() similarly allow both strings and numeric variables. Numeric variables are automatically converted to strings and leading blanks are always removed. For strings, CAT() does not trim, CATT() trims trailing blanks only, and CATS() trims leading and trailing blanks.

D. **How it works / what happens**: ***While the DATA step's implicit loop is running***, the SAS code generated from each invocation of CALL EXECUTE() during the DATA step is ***stored***. Then, ***after the DATA step ends***, the ***stored code is run*** normally.

E. **Example demonstrating the main principles:**

```
/* Setup some data sets */
DATA testData;
  LENGTH category $1.;
  INPUT category x y @@;
  DATALINES;
   A 2 5  B 5 2  A 7 9  A 12 15  B 18 20
RUN;

DATA junk; INPUT a b c @@; DATALINES;
  1 2 3  4 5 6
RUN;

/* Create a dataset of all datasets in a library */
/* using SQL "secret" dataset "DICTIONARY.TABLES" */
/* (Outside of SQL the secret dataset is called SASHELP.VTABLE) */
PROC SQL;
  CREATE TABLE tempDSnames AS
    SELECT memname, nobs
    FROM DICTIONARY.TABLES WHERE libname='WORK';
QUIT;

PROC PRINT DATA=tempDSnames; RUN;
Obs    memname       nobs
 1     JUNK            2
 2     TESTDATA        5

/* Manually code an example before trying to automate it */
TITLE "Files in WORK";
TITLE2 "2 obs. from JUNK";
PROC PRINT DATA=JUNK(OBS=2);
RUN;
...


/* First attempt at CALL EXECUTE */
DATA notnull;
  SET tempDSnames;
  nobs = MIN(20, nobs);
  LENGTH longStr $200.;
  longStr = CAT('TITLE "Files in WORK"; ',
                  'TITLE2 "', nobs, ' obs. from ', memname, '"; ',
                  'PROC PRINT DATA=', memname, '(OBS=', nobs, '); ',
                  'RUN;');
  CALL EXECUTE(longStr);
RUN;

/* Demonstrate how it worked */
PROC PRINT DATA=notnull; VAR longStr; RUN;

/* cleanup */
PROC DATASETS NOLIST;
  DELETE notnull;   /* only if debugging */
  DELETE tempDSnames;
RUN;
```

```
/* Using SASHELP.VTABLE instead of DICTIONARY.TABLES / SQL */
/* No cleanup needed */
DATA _NULL_;
  SET SASHELP.VTABLE(KEEP=libname memname nobs
    WHERE=(libname="WORK"));
  nobs = MIN(20, nobs);
  CALL EXECUTE(CAT('TITLE "Files in WORK"; ',
                   'TITLE2 "', nobs, ' obs. from ', memname, '"; ',
                   'PROC PRINT DATA=', memname, '(OBS=', nobs, '); ',
                   'RUN;'));
RUN;

/* Some Log Output: */
NOTE: There were 2 observations read from the data set WORK.TEMPDSNAMES.
NOTE: CALL EXECUTE generated line.
1   + TITLE "Files in WORK";TITLE2 "2 obs. from JUNK";
1   +          PROC PRINT DATA=JUNK(OBS=2);RUN;
NOTE: There were 2 observations read from the data set WORK.JUNK. ...
/* Alternate macro method */

/* print 'n' records from 'ds' with title 'title' */
%MACRO showData(ds, n, title);
  TITLE "&title";
  TITLE2 "&n obs. from &ds";
  PROC PRINT DATA=&ds(OBS=&n);
  TITLE;
  RUN;
%MEND showData;

/* Manual code to run the macro */
%showData(junk, 2, Files in WORK)
%showData(testdata, 5, Files in WORK)

/* Run the macro for each dataset in WORK */
DATA _NULL_;
  SET SASHELP.VTABLE(KEEP=libname memname nobs
                     WHERE=(libname="WORK"));
  nobs = MIN(20, nobs);
  CALL EXECUTE(CAT('%showData(', memname, ',', nobs,
                   ', Files in WORK)'));
RUN;
```

As a useful macro:

```
/* Macro to print data from all datasets in a library */
/* A maximum of 'nmax' records is shown from each */
%MACRO showAll(library, nmax=5);
  %LET library = %UPCASE(&library);
  DATA _NULL_;
    SET SASHELP.VTABLE(KEEP=libname memname nobs
                       WHERE=(libname="&library"));
    nobs = MIN(&nmax, nobs);
    CALL EXECUTE(CAT('TITLE "Files in ', "&library", '"; ',
                     'TITLE2 "', nobs, ' obs. from ',
                       memname, '"; ',
                     'PROC PRINT DATA=', "&library", '.',
                       memname, '(OBS=', nobs, '); ',
                     'RUN;'));
  RUN;
%MEND showAll;
```

```
%showAll(work);
%showAll(work, nmax=2);


LIBNAME here ".";
DATA here.abc;
  INPUT a@@;
  DATALINES;
  1 2 3 4 5 6 7 8
RUN;
%showAll(here);
```

II. Extended Example: Calculating Salaries By Sector

```
/* Salaries by sector example */

FILENAME secSal "&wd/data/secSal.dat";

TITLE "Salaries of Employees in Small Companies for 3 Sectors";
DATA secSal;
  INFILE secSal FIRSTOBS=2;
  LENGTH sector $1. company $2.;
  INPUT sector company age male salary;
  LABEL salary="Base salary in $1000";
RUN;

TITLE2 "First 20 Observations";
PROC PRINT DATA=secSal(OBS=20);
RUN;
```

Obs	sector	company	age	male	salary
1	A	YJ	32	1	37.22
2	A	YJ	29	1	35.89
...					
19	A	QL	31	1	34.51
20	A	QL	43	1	60.68

```
TITLE2;
PROC FREQ DATA=secSal;
  TABLES male;
  TABLES company*sector / NOPERCENT NOROW;
RUN;
```

male	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	101	46.33	101	46.33
1	117	53.67	218	100.00

company sector

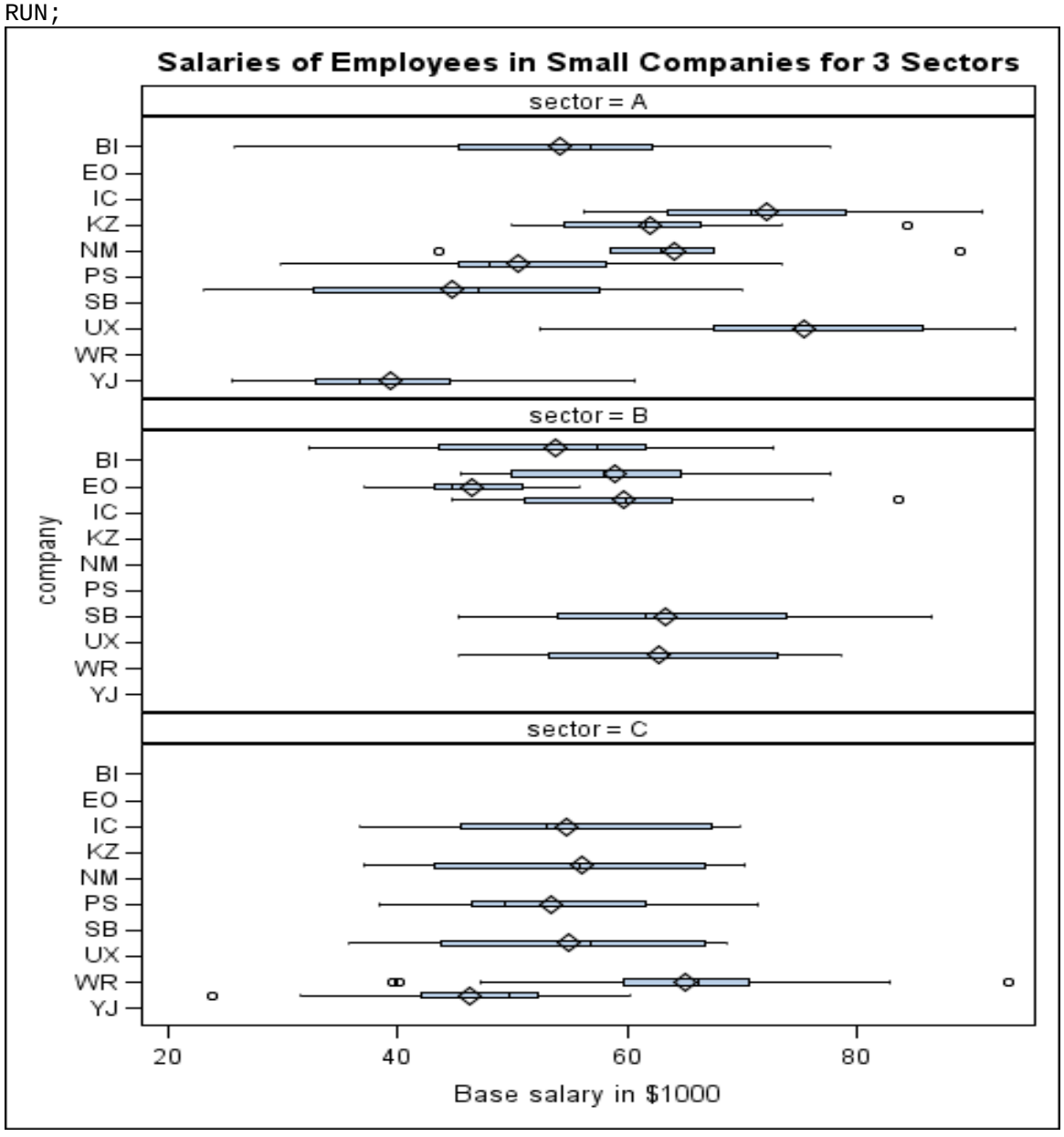
Frequency,
Col Pct

	A	B	C	Total
AQ	0 0.00	9 14.06	0 0.00	9
BI	7 8.24	0 0.00	0 0.00	7
DW	0 0.00	6 9.38	0 0.00	6
...				
XN	0 0.00	0 0.00	15 21.74	15
YJ	14 16.47	0 0.00	0 0.00	14
Total	85	64	69	218

```
PROC MEANS DATA=secSal;
  VAR age salary;
RUN;
```

Variable	Label	N	Mean	StdDev	Minimum	Maximum
age		218	35.19	7.58	14.00	56.00
salary	Base salary in $1000	218	56.39	14.83	22.98	93.69

```
PROC SGPANEL DATA=secSal;
  PANELBY sector / COLUMNS=1;
  HBOX salary / CATEGORY=company;
RUN;
```




```
TITLE2 "Full factorial + RI model";
```

```
PROC MIXED DATA=secSal COVTEST;
  CLASS sector company male;
  MODEL salary = sector|male|age / DDFM=KR;
  RANDOM INTERCEPT / SUBJECT=company;
RUN;
```

Model Information

Dependent Variable	salary
Covariance Structure	Variance Components
Subject Effect	company
Estimation Method	REML

Class Level Information

Class	Levels	Values
sector	3	A B C
company	20	AQ BI DW EO HK IC JV KZ LA NM OG PS QL SB TP UX VT WR XN YJ
male	2	0 1

Dimensions

Covariance Parameters	2
Columns in X	24
Columns in Z Per Subject	1
Subjects	20
Max Obs Per Subject	16

Number of Observations Used 218

Convergence criteria met.

Covariance Parameter Estimates

Cov Parm	Subject	Estimate	Standard Error	Z Value	Pr > Z
Intercept	company	99.0752	34.7214	2.85	0.0022
Residual		23.2080	2.3869	9.72	<.0001

AIC (smaller is better)	1367.6
BIC (smaller is better)	1369.6

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
sector	2	38.6	0.58	0.5639
male	1	190	4.29	0.0397
sector*male	2	190	0.76	0.4694
age	1	190	980.34	<.0001
age*sector	2	190	1.66	0.1938
age*male	1	190	0.64	0.4244
age*sector*male	2	190	0.78	0.4593

...

```
TITLE2 "Main effects + RI model";
PROC MIXED DATA=secSal COVTEST PLOTS=(ALL);
  CLASS sector company male;
  MODEL salary = sector male age / DDFM=KR SOLUTION;
  RANDOM INTERCEPT / SUBJECT=company;
  LSMEANS sector / ADJUST=TUKEY ADJDFE=ROW;
RUN;
```

Covariance Parameter Estimates

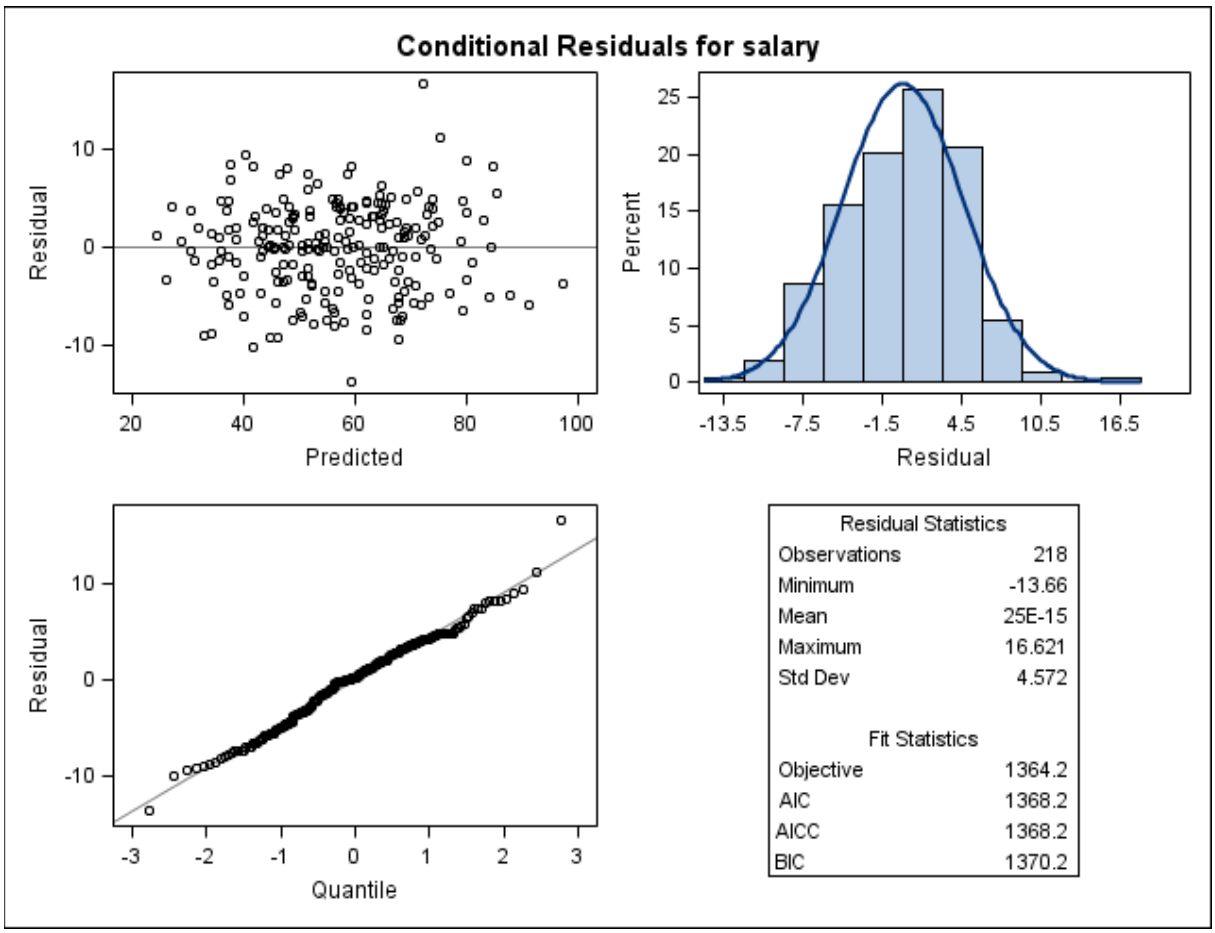
Cov Parm	Subject	Estimate	Standard Error	Z Value	Pr > Z
Intercept	company	98.0481	34.3650	2.85	0.0022
Residual		23.0968	2.3328	9.90	<.0001

Solution for Fixed Effects

Effect	sector	male	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept			7.4692	4.3695	22.2	1.71	0.1013
sector	A		0.5260	5.4087	17	0.10	0.9237
sector	B		1.0640	5.7819	17	0.18	0.8562
sector	C		0	.	.	.	.
male		0	-4.4373	0.6670	196	-6.65	<.0001
male		1	0	.	.	.	.
age			1.4438	0.04490	197	32.16	<.0001

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
sector	2	17	0.02	0.9832
male	1	196	44.25	<.0001
age	1	197	1034.09	<.0001



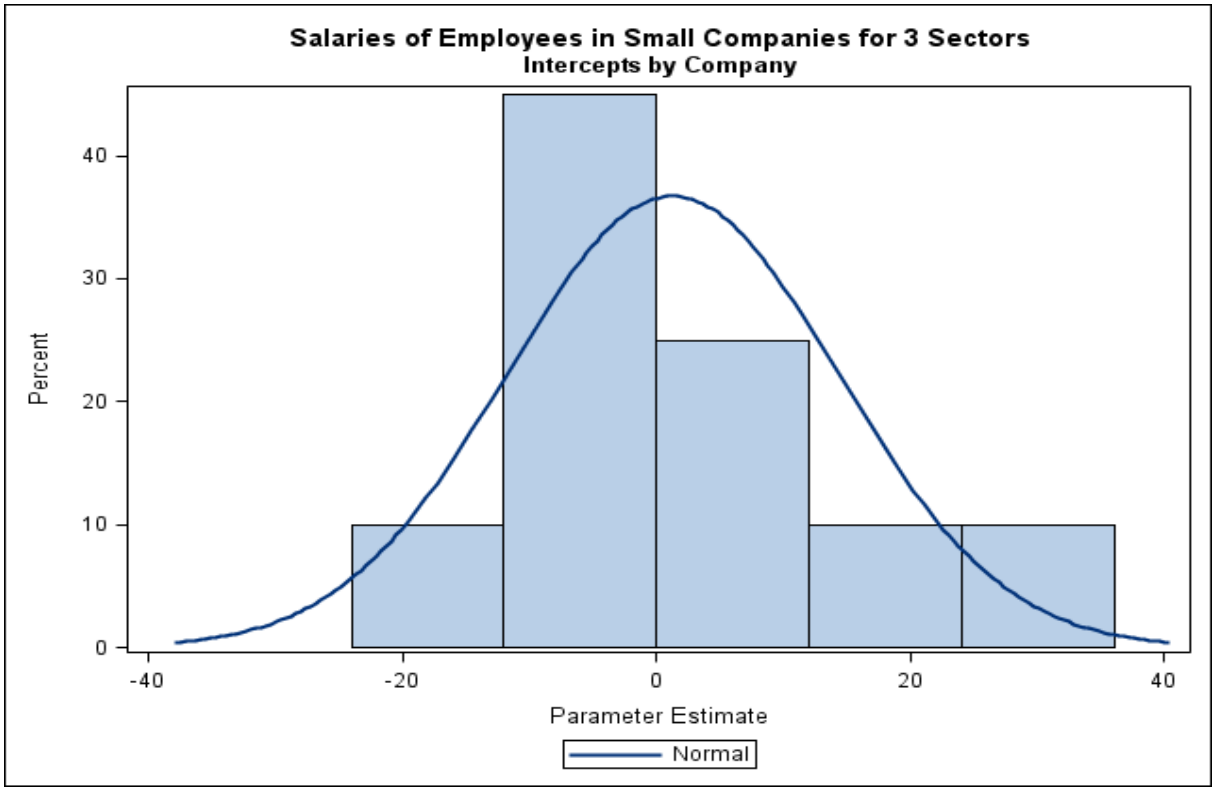
▸ **Now think about how to check the distribution of parameter estimates by company.**

```
PROC SORT DATA=secSal;
  BY company;
RUN;
/* "Center" age */
DATA secSal; SET secSal; ageC35 = age - 35; RUN;

TITLE2 "Checking individual betas";
PROC REG DATA=secSal;
  BY company;
  MODEL salary = age male;
  ODS OUTPUT PARAMETERESTIMATES=tempBetas;
RUN;

PROC PRINT DATA=tempBetas;
RUN;
```

Obs	company	Model	Dependent	Variable	DF	Estimate	StdErr	tValue	Probt
1	AQ	MODEL1	salary	Intercept	1	-1.76	11.58	-0.15	0.8840
2	AQ	MODEL1	salary	age	1	1.44	0.38	3.84	0.0086
3	AQ	MODEL1	salary	male	1	6.73	5.32	1.27	0.2526
4	BI	MODEL1	salary	Intercept	1	-0.50	5.79	-0.09	0.9354
5	BI	MODEL1	salary	age	1	1.40	0.16	8.65	0.0010
6	BI	MODEL1	salary	male	1	5.17	3.19	1.62	0.1805

...

```
TITLE2 'Intercepts by Company';
PROC SGPLOT DATA=tempBetas(WHERE=(variable='Intercept'));
  HISTOGRAM estimate;
  DENSITY estimate;
RUN;
```



▸ Now think about how to not include gender when either the Male or Female count is low.

```
/** Method 1: complex embedded CALL EXECUTE()'s **/
PROC SORT DATA=secSal;  BY company;  RUN;

/* Empty dataset to hold results (STOP makes row count be zero) */
DATA Parameters; LENGTH company $2. variable $9. estimate 8; STOP; RUN;

/* Count M and F by company and compute appropriate regression */
/* Merge regression results into 'Parameters' */
%LET minCount = 4;
DATA _NULL_;
  SET secSal;
  BY company;
  RETAIN m f;

  /* Reset counts for each new company */
  IF first.company THEN DO;
    m=0; f=0;
  END;

  /* Count males and females per company */
  IF male=1 THEN m=m+1;
  ELSE f=f+1;

  /* Perform regressions for each company */
  IF last.company THEN DO;
    IF m<&minCount OR f<&minCount THEN DO;
      CALL EXECUTE(CAT('TITLE "Company=', company, '  M/F=',
                       m, '/', f, '"; ',
                      'PROC REG DATA=secSal(WHERE=(company="',
                       company, '")); ',
                      '  MODEL salary = ageC35;',
                      '  ODS OUTPUT parameterEstimates=tempBetas;',
                      'RUN;'));
    END;
    ELSE DO;  /* else if sufficient males and females */
      CALL EXECUTE(CAT('TITLE "Company=', company, '  M/F=',
                       m, '/', f, '"; ',
                      'PROC REG DATA=secSal(WHERE=(company="',
                       company, '")); ',
                      '  MODEL salary = ageC35 male;',
                      '  ODS OUTPUT parameterEstimates=tempBetas;',
                      'RUN;'));
    END; /* end few vs. enough males and females */

    /* Merge this company's results into all results */
    CALL EXECUTE(CAT('TITLE; ',
                     'DATA Parameters; ',
                     '  SET Parameters tempBetas(KEEP=VARIABLE ',
                                                 'ESTIMATE);',
                     '  IF company="" THEN company="', company, '";',
                     'RUN;'));
  END; /* end last.company */
RUN;  /* end DATA _NULL_ */

/* cleanup */
PROC DATASETS NOLIST;  DELETE tempBetas; RUN;
```

```
/* Make single pdf file of all three plots */
ODS PDF FILE="ParDistsOverCompany.pdf";
TITLE 'Intercepts by Company';
PROC SGPLOT DATA=Parameters(WHERE=(variable='Intercept'));
  HISTOGRAM estimate;
  DENSITY estimate;
RUN;
TITLE 'Age Slopes by Company';
PROC SGPLOT DATA=Parameters(WHERE=(variable='age'));
  HISTOGRAM estimate;
  DENSITY estimate;
RUN;
TITLE "Male " color=red "Intercept " color=black
      "Differences by Company (#M>=&minCount %STR(&) #F>=&minCount)";
PROC SGPLOT DATA=Parameters(WHERE=(variable='male'));
  HISTOGRAM estimate;
  DENSITY estimate;
RUN;
ODS PDF CLOSE;


/***************************/
/* Method 2: Macro Approach */
/***************************/
/* This macro regresses 'salary' on 'age+male' or 'age'
   for a particular 'company', including 'male' only if
   there are enough (>'cutoff') people of each gender.
   The number of males and females are included in the
   input to this macro ('nM' and 'nF').  The consequence
   of this macro is to add lines to the 'parameters' data
   set.
*/

%MACRO regAndMerge (company, nM, nF, cutoff=4);
  TITLE "Company=&company  M/F=&nM/&nF";
  %IF &nM >= &cutoff AND &nF >= &cutoff %THEN %DO;
    PROC REG DATA=secSal(WHERE=(company="&company"));
      MODEL salary = age male;
      ODS OUTPUT parameterEstimates=tempBetas;
    RUN;
  %END;
  %ELSE %DO;
    PROC REG DATA=secSal(WHERE=(company="&company"));
      MODEL salary = age;
      ODS OUTPUT parameterEstimates=tempBetas;
    RUN;
  %END;
  DATA Parameters;
    SET Parameters tempBetas(KEEP=VARIABLE ESTIMATE);
   /* Below happens when we move from 'parameters' to 'tempBetas' */
    IF company="" THEN company="&company";
  RUN;
%MEND regAndMerge;
```

```
/* Manually test the macro with one company */
OPTIONS SYMBOLGEN MPRINT MLOGIC;
%regAndMerge(AQ, 5, 5, cutoff=4)
OPTIONS NOSYMBOLGEN NOMPRINT NOMLOGIC;

/* Run all companies (secSal must be sorted by company) */

/* Create an empty version of the dataset that holds all results */
DATA Parameters; LENGTH company $2. variable $9. estimate 8; STOP; RUN;

DATA _NULL_;
  SET secSal;
  BY company;
  RETAIN m f;
  /* Reset counts for each new company */
  IF first.company THEN DO;
    m=0;
    f=0;
  END;

  /* Count males and females per company */
  IF male=1 THEN m=m+1;
  ELSE f=f+1;

  /* Perform regressions for each company */
  IF last.company THEN
    CALL EXECUTE(CAT('%regAndMerge(', company, ', ', m, ', ',
                                      f, ', cutoff=4)'));
RUN; /* end DATA _NULL_ */

/* cleanup */
PROC DATASETS NOLIST;  DELETE tempBetas;  RUN;



/* An even better approach, using SQL (no SORT needed): */
PROC SQL;
  CREATE TABLE tempCompaniesMF AS
    SELECT company, sum(male) AS nM, sum(1-male) AS nF
      FROM secSal GROUP BY company;
QUIT;

DATA _NULL_;
  SET tempCompaniesMF;
  CALL EXECUTE(CAT('%regAndMerge(', company, ', ', nM, ', ',
                                   nF, ', cutoff=4)'));
RUN; /* end DATA _NULL_ */
PROC DATASETS NOLIST; DELETE tempCompanies; RUN;
```

```
/* Full macro approach */
%MACRO doMixed(cutoff=4);
  %LOCAL i;
  /* Count unique companies */
  PROC SORT DATA=secSal(KEEP=company) OUT=secSalND NODUPKEYS;
    BY company;
  RUN;
  PROC SQL NOPRINT;
    SELECT COUNT(company) INTO :nCompanies FROM secSalND;
  QUIT;
  PROC DATASETS; DELETE secSalND; RUN;
  %LET nCompanies = %TRIM(&nCompanies);

  /* Get per-company info */
  PROC SQL NOPRINT;
      SELECT company, sum(male) AS nM, sum(1-male) AS nF
        INTO :companies1 - :companies&nCompanies,
             :males1 - :males&nCompanies,
             :females1 - :females&nCompanies
      FROM secSal GROUP BY company;
  QUIT;

  /* Clear dataset */
  %PUT Note: Creating empty PARAMETERS dataset.;
  %PUT (Ignore three 'Variable xxx is uninitialized' statements);
  DATA Parameters;
    LENGTH company $2. variable $9. estimate 8;
    STOP;
  RUN;

  /* Run all companies */
  %DO i = 1 %TO &nCompanies;
    TITLE "Company=&&companies&i  M/F=&&males&i/&&females&i";
    %IF &&males&i >= &cutoff AND &&females&i >= &cutoff %THEN %DO;
      PROC REG DATA=secSal(WHERE=(company="&&companies&i"));
        MODEL salary = ageC35 male;
        ODS OUTPUT parameterEstimates=tempBetas;
      RUN;
    %END;
    %ELSE %DO;
      PROC REG DATA=secSal(WHERE=(company="&&companies&i"));
        MODEL salary = ageC35;
        ODS OUTPUT parameterEstimates=tempBetas;
      RUN;
    %END;
    DATA Parameters;
      SET Parameters tempBetas(KEEP=variable estimate);
      /* Below happens when we move from 'parameters' to 'tempBetas' */
      IF company="" THEN company="&&companies&i";
    RUN;
  %END;

  /* Cleanup last temporary dataset */
  PROC DATASETS NOLIST; DELETE tempBetas; QUIT;
%MEND doMixed;

%doMixed(cutoff=3)
```