Summary
Increasingly, SAS programmers are tasked with extracting data from relational databases for analyses. One problem encountered by these programmers is that observations stored in two tables with a one-to-many relationship must be accumulated to a single SAS observation prior to analysis. This paper demonstrates and evaluates a number of methods of accomplishing the accumulation and provides a macro based on the most efficient method.

Introduction
By definition, relational databases have a built-in mechanism for linking a record from one table to several records from another table. This mechanism is called a 1-to-many relationship. When analyzing data with SAS, processing is done an observation at a time; therefore, when analyzing data from a relational database, it is necessary to convert the data relationships in the underlying table structure into a single SAS observation. This paper describes an efficient method using Base SAS to accomplish the transformation of a 1-to-many relationship to a single SAS observation, demonstrates it with an example, compares it to other methods, and provides a general SAS macro that performs the transformation.

Fundamentally, the 1-to-many transformation problem is one of reorganizing the data from the table with non-unique records so that it has one observation for each row of the table that has unique records. To facilitate the discussion, let the table with the single row matching many rows be called the parent table and any table that has many rows be denoted as a child table. Also, denote the common field between these tables as the key value.

The reorganizing consists of two steps. First, each child table must be scanned to determine the maximum number of rows in that table that correspond to a key in the parent table. Using that maximum value as a parameter for an array statement in a subsequent data step, the data from the table is read into SAS arrays. The transformation is completed after an observation is created for each key value on the child table. The final step is to merge the transformed data from the child table with the parent table.

Illustrative Example
For example, suppose you have an accounting system that stores sales transactions. The main table Transactions stores the total amount of the sale, the date of the sale, the purchaser, as well as an ID field, which is the key value. Details associated with each transaction are stored in a table named Details. Details contains one row for each type of item purchased and sales tax information. Since many detail lines can be associated with any transaction, the Details table would have a many-to-1 relationship with the table Transactions. Tables 1 and 2 show these tables.
Table 1: Transactions, the Parent Database Table

<table>
<thead>
<tr>
<th>Bill ID</th>
<th>Name</th>
<th>Date</th>
<th>Bill Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>12-May-06</td>
<td>341.23</td>
</tr>
<tr>
<td>2</td>
<td>Betty</td>
<td>22-May-06</td>
<td>417.77</td>
</tr>
</tbody>
</table>

Table 2: Details, a Child Database Table

<table>
<thead>
<tr>
<th>Bill ID</th>
<th>Item</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Item 1</td>
<td>74.99</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Item 3</td>
<td>225.00</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Item 4</td>
<td>15.23</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Sales Tax</td>
<td>26.01</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Item 1</td>
<td>74.99</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Item 2</td>
<td>120.00</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Item 4</td>
<td>15.23</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Item 5</td>
<td>84.33</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Sales Tax</td>
<td>31.84</td>
<td>1</td>
</tr>
</tbody>
</table>

After reading the data into SAS datasets, the first step in merging the data from the child table Details with the parent table Transactions is determining the maximum number of rows in Details that are associated with a particular row in Transactions. One way to accomplish this calculation is use a summarizing procedure, like PROC UNIVARIATE, twice—the first pass will determine the number of rows in the child table associated with each key value and the second pass will determine the maximum. The SAS code for this calculation is shown in Figure 1.

Figure 1: Finding the Maximum Key Matches using PROC UNIVARIATE

```sas
proc sort data=details;
    by bill_id;

proc univariate data=details noprint;
    var bill_id;
    output out=d_count n=count;
    by bill_id;

proc univariate data=d_count noprint;
    var count;
    output out=d_max max=d_max;

data _null_;  
    set d_max;
    call symput('gl_max',d_max);
run;
```

The last data step creates a macro variable named gl_max that can be used by subsequent SAS steps. Incidentally, the “run;” statement is necessary to force the compilation and execution of the data step, which assigns the macro variable its value.

This maximum value is the number of elements needed in the arrays used to store the data from the child tables. The ‘flattening’ occurs when several observations in the child dataset are reduced to a single observation that can be merged with the parent dataset. For example, in the Details table, four rows are associated with Bill ID 1. The flattened data would consist of three SAS arrays (one for item, one for price, and one for quantity), each with four elements along with the key variable bill_id. In general, there will
be as many SAS arrays as there are non-key fields in the child table and each array will have as many non-missing elements as the number of rows in the child table associated with its key value.

Figure 2 shows the data step that creates one observation for each key value with arrays for each of the other variables in the Details dataset followed by the merge with the parent dataset. Since numeric values are required in various places in the data step, the %EVAL function is used to resolve the macro variable gl_max as a number. Without this function, the SAS compiler would return an error. The data is read into the data step sorted by the key variable bill_id. Each time a new value of the key variable is encountered, the arrays are initialized. The data step uses a counter to set values of each array element during each pass through the data step. The observation is written when the last occurrence of the key value is processed.

Figure 2: Building SAS Arrays using a DATA Step

data flat_detail;
array xitem{%eval(&gl_max)} $12 xitem1-xitem{%eval(&gl_max)};
array xcost{%eval(&gl_max)} xcost1-xcost{%eval(&gl_max)};
array xqty{%eval(&gl_max)} xqty1-xqty{%eval(&gl_max)};
retain counter xitem1-xitem{%eval(&gl_max)}
       xcost1-xcost{%eval(&gl_max)}
       xqty1-xqty{%eval(&gl_max)};
set details;
by bill_id;
if (first.bill_id eq 1) then do;
   do i=1 to %eval(&gl_max);
      xitem{i}=.;
      xcost{i}=.;
      xqty{i}=.;
   end;
end;
xitem{counter}=item;
xcost{counter}=cost;
xqty{counter}=qty;
if (last.bill_id) then output;
else delete;

counter=counter+1;
keep bill_id xitem1-xitem{%eval(&gl_max)}
       xcost1-xcost{%eval(&gl_max)}
       xqty1-xqty{%eval(&gl_max)};
proc sort data=flat_detail,
   by bill_id;

proc sort data=transactions;
   by bill_id;

data transactions;
   merge flat_detail transactions;
   by bill_id;
Table 3 shows the listing of the SAS dataset after the final merge.

Table 3: Dataset Transactions, After Flattening

<table>
<thead>
<tr>
<th>Obs</th>
<th>bill_id</th>
<th>Name</th>
<th>Date</th>
<th>Bill Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Alice</td>
<td>12-May-06</td>
<td>341.23</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Betty</td>
<td>22-May-06</td>
<td>417.77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>xitem1</th>
<th>xitem2</th>
<th>xitem3</th>
<th>xitem4</th>
<th>xitem5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Item 1</td>
<td>Item 3</td>
<td>Item 4</td>
<td>Sales Tax</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Item 1</td>
<td>Item 2</td>
<td>Item 4</td>
<td>Item 5</td>
<td>Sales Tax</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>xcost1</th>
<th>xcost2</th>
<th>xcost3</th>
<th>xcost4</th>
<th>xcost5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>74.99</td>
<td>225.00</td>
<td>15.23</td>
<td>26.01</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>74.99</td>
<td>120.00</td>
<td>15.23</td>
<td>84.33</td>
<td>31.84</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>xqty1</th>
<th>xqty2</th>
<th>xqty3</th>
<th>xqty4</th>
<th>xqty5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Alternative Approaches to the Two Basic Steps**

The tasks involved in flattening a child table, (1) finding the dimension of the arrays and (2) reconstructing the child dataset using arrays, can each be done more than one way using SAS. For this paper, each task was coded using two different methods and these methods were called from a macro that took as parameters the calculation methods and the number of observations and variables in the parent and child datasets. The amount of time needed to create the final dataset was measured for a variety of combinations of the inputs to the macro and the results were analyzed in using an Analysis of Covariance technique to determine the most efficient approach.

Two methods were used to find the maximum number of rows from a child table that is associated with any particular key value. The first method used two instances of the UNIVARIATE procedure run one after the other and was illustrated in the example above. The other method used a single pass through a data step to find the needed information. Both methods require that the input data set be sorted by the key value and both need to pass the calculated information to subsequent parts of the program for further processing.

Two methods were used to amalgamate fields from each set of rows with the same key value in a child table into SAS arrays. The first was used in the above example and uses first. and last. variable processing to read data into SAS arrays. This dataset was merged with the parent dataset to create the final output. The second method employed PROC TRANSPOSE to create arrays that were renamed in individual data steps. The datasets were merged with the parent dataset to create the final dataset.

**The Testing Macro and Test Results**

The macro used for testing is shown in Appendix 1. It takes as input the number of observations and number of variables in the parent dataset, the number of variables and the maximum number of observations associated with a single key value in the child dataset, the method to determine the maximum rows and the method to amalgamate the
data as well as a seed value to initialize the random number generator. The resources used during each iteration were measured using the FULLSTIMER option in SAS. These measures were read from the SASLOG and used as the responses in an Analysis of Covariance model. The fixed factors in the model were the methods and the dataset size parameters were analyzed as covariates. Both main effect and the interaction of the fixed factors were estimated.

The first step in running the simulation was to construct macro-calling statements. A SAS program was written to generate these statements; this program is shown in Appendix 2. The analysis called the macro 180 times. Each of the maximum finding and array building methods were called ninety times each and each pair of methods was present in one-quarter of the calling statements. The number of observations in the parent dataset was randomly assigned in the range of 50,000 to 200,000. The number of variables in the parent and child datasets was independently and randomly varied between five and twenty-nine and the maximum number of observations in the child dataset associated with any particular key value was randomly varied between seven and twenty. To gather the response data which was the sum of the various times and memory usage for each iteration, a SAS program was written to read the SASLOG file and create the sums and then run the covariance analysis. The analysis was performed using PROC GLM. This program is included in Appendix 3.

Based on the 180 run test, the significant regression factors were number of observations in the parent dataset, the number of variables in the child dataset, and the maximum number of observations associated with any particular key value in the child dataset for all three responses: the user time, that is time spent running SAS code; the system time, that is time spent on system calls like I/O operations; and total memory used or the sum of the memory used by all of the procedures and data steps. There was no interaction detected between the two fixed effects. Least squares means were calculated for both main effects. These means showed that the data step method was significantly quicker in a statistical sense than the dual-UNIVARIATE method based on user time for finding the maximum number of observations in the child dataset associated with a particular key value. System time and memory used were slightly higher for the data step method but neither difference was statistically significant. On all measured responses, the data step method was significantly better than the transpose method to build arrays from data in the child table. Therefore, the most efficient mechanism to flatten a child table for SAS processing is using the data step to both find the maximum number of values and to build the final data set. The results from both the full analysis and the reduced analysis using only the significant factors are given in Appendix 4.

**The Final Flattening Macro**

Figure 5 is the SAS macro that flattens and merges a child dataset with a parent dataset. It uses data step processing to accomplish both of the major tasks involved in the transformation. Besides creating the output dataset, which can be a permanent SAS dataset, the macro will output the array definitions for later use as an option.
Figure 5: The More Efficient Flattening Macro

%macro flatten(parent,child,list,sortvar,outds,file);
/* The flatten macro merges a child dataset that has a many-
to-one relationship with that parent dataset. It requires
that both datasets have a key in common. The child
dataset's observations are converted to SAS arrays before
the merge. The parameters used in the macro are:
*/
* parent  a SAS dataset with a key variable that has a
  unique value for each observation.
* child  a SAS dataset with the same key variable that
  can contain more than one observation per
  key value.
* list  a space delimited list of variables from the
  child dataset that are to be transformed into
  SAS arrays. If this parameter is left blank,
  all variables (except for the key variable) in
  the child dataset are converted to SAS arrays.
* sortvar is the key variable common to both datasets.
* outds is the output dataset. It can be a permanent
  SAS dataset.
* file is a file name for the underlying operating
  system where the array definitions are stored
  for future use.
*/
%let null=; * This assignment allows logical comparison of
null strings later in the program;

/* This section sorts the child dataset and reads it to
* determine the maximum number of observations that are
* associated with the sort variable.
*/
proc sort data=&child;
  by &sortvar;

data _null_;%
  retain _max gl_max;
  set &child end=last;
  by &sortvar;
  if (first.&sortvar eq 1) then _max=0;
  _max=_max+1;
  if (last.&sortvar eq 1 and _max gt gl_max) then gl_max=_max;
  if (last eq 1) then call symput('d_max',gl_max);
run;

/* The datasets procedure is used to extract information
* about the variables in the child dataset used to build
* the array statements.
*/
proc datasets nolist;
  contents data=&child out=&child._info noprint;

/* This section retains only the variables that should be
* transposed (based on the value of the macro variable list)
* in the information dataset created in the previous step
*/
%if &list NE &null %then
%do;
  data names;
  %let ii=1;
  %do %while(%scan(&list,&ii, ' ') ne &null);
    name="%scan(&list,&ii, ' ')"; output;
    %let ii=%eval(&ii+1);
  %end;
  proc sort data=names;
    by name;
  proc sort data=&child._info;
    by name;
  data &child._info;
    merge names(in=i1) &child._info;
    by name;
    if i1;
  %end;
/%* This dataset builds three macro variables:
*  list which is the same as previously defined. It is
*   redefined here only because if all variables are
*   to be converted into SAS arrays, it needs to be
*   defined.
*  type is a "vector" (i.e. a space delimited list of
*   character strings) of 1's and 2's that are indicators
*   for numeric and character variables, respectively
*   that are in the same position as the variables in
*   the macro variable list.
*  len is another vector that contains a zero for numeric
*   variables and the length of a character variable.
*/
data _null_;
  length list xtype len $200;
  retain list xtype len ' ';
  set &child._info end=last;
  if (upcase(name) ne "%upcase(&sortvar)") then
    do;
      list=trim(left(list)) || ' '|| trim(left(name));
      if (type eq 1) then
        do;
          xtype=trim(left(xtype))||' 1';
          len=trim(left(len)) ||' 0';
        end;
      else do;
        xtype=trim(left(xtype))||' 2';
        len=trim(left(len))  ||' '||trim(left(input(length,best12.)));
      end;
    end;
  if (last eq 1) then
    do;
      call symput('list',trim(left(list  )));
      call symput('type',trim(left(xtype)));
      call symput('len', trim(left(len  )));
    end;
run;
/* The child dataset is sorted and read into another dataset
* where SAS arrays are created for each variable listed in
* the macro variable list. FIRST. and LAST. processing is
* used along with a counter to initialize, populate, and
* output the arrays. Also, if the macro variable file is
* not null, the array definitions are written to that file
* for use later.
* After the array-creation data step, it is merged with the
* parent dataset to create the output dataset. */

proc sort data=&child;
  by &sortvar;

data &outds;
  retain counter
  %let ii=1;
  %do %while(%scan(&list,&ii,' ') ne &null);
    %do jj=1 %to %eval(&d_max);
      _&ii._&jj
    %end;
    %let ii=%eval(&ii+1);
  %end;
  %let ii=1;
  %do %while(%scan(&list,&ii,' ') ne &null);
    %let name=%scan(&list,&ii,' ');
    %let vtype=%scan(&type,&ii,' ');
    %let vlen=%scan(&len ,&ii,' ');
    %if %eval(&vtype) = 1 %then
      %do;
        array _&ii._{%eval(&d_max)}
        %do jj=1 %to %eval(&d_max);
          _&ii._&jj
        %end;
      %end;
    %else %do;
      array _&ii._{%eval(&d_max)} $%eval(&vlen)
      %do jj=1 %to %eval(&d_max);
        _&ii._&jj
      %end;
    %end;
    %let ii=%eval(&ii+1);
  %end;
  set &child;
  by &sortvar;
  %if &file NE &null %then
    %do;
      if (_n_ eq 1) then
        do;
          file "&file" mod;
          %let ii=1;
          %do %while(%scan(&list,&ii,' ') ne &null);
            %let name=%scan(&list,&ii,' ');
            %let vlen=length(trim(left("&name")));
            put @ 1 /* &name */;
            put + (16-vlen) "*/ array _&ii._{%eval(&d_max)} " @;
          %end;
        %end;
      %end;
    %end;
%do jj=1 %to %eval(&d_max);
    put " _&ii._&jj" @;
%end;
put ";";
%let ii=%eval(&ii+1);
%end;
end;

if (first.&sortvar) then do;
    counter=1;
    do ii=1 to %eval(&d_max);
        %let ii=1;
        %do %while(%scan(&list,&ii,' ') ne &null);
            _&ii._{ii}=.;
            %let ii=%eval(&ii+1);
        %end;
    end;
end;
%let ii=1;
%do %while(%scan(&list,&ii,' ') ne &null);
    _&ii._{counter}=%scan(&list,&ii,' ');
    %let ii=%eval(&ii+1);
%end;
counter=counter+1;
if (last.&sortvar) then output;
else delete;

keep &sortvar
%let ii=1;
%do %while(%scan(&list,&ii,' ') ne &null);
    %do jj=1 %to %eval(&d_max);
        _&ii..&jj
    %end;
    %let ii=%eval(&ii+1);
%end;

proc sort data=&outds;
    by &sortvar;
    proc sort data=&parent;
        by &sortvar;
        data &outds;
            merge &parent &outds;
            by &sortvar;
        %mend flatten;
Appendix 1: The SAS Macro Used to Test Different Methods

%macro doit(seed,a,b,c,d,e,f);
/* Macro doit takes seven arguments. They are:
*   seed a up to 8-digit number to seed pseudo-random
*   number functions;
*      a the number of observations in the parent dataset;
*      b the number of variables in the parent dataset;
*      c the number of variables in the child dataset;
*      d the maximum number of observations in the child
dataset that match with any one key value in
*      the parent dataset;
*      e the method used to determine the maximum; and
*      f the method used to flatten the data.
*/

/* Embed comments in the SASLOG to determine where each
* section starts.
*/
data _null_; put "DCW Iteration Begin"; put "DCW Parameters &e &f: &a &b &c &d"; run;

/* Create p, a parent dataset, with b variables and a
* observations, where a and b are macro variables.
*/
data p;
  do match=1 to %eval(&a);
    %do ii=1 %to %eval(&b);
      j%eval(&ii)=ranuni(&seed);
    %end;
    output;
  end;
end;

/* Create c, a child dataset, with the same key values as
* the parent dataset with a variable number of observations
* for each key value ensuring that at least one of the
* key values has d observations associated with it. The
* dataset will have c variables where c and d are macro
* variables.
*/
data c;
  retain maxval 0;
  do match=1 to %eval(&a);
    %do ii=1 %to %eval(&b);
      loopval=ceil((%eval(&d)*ranuni(&seed)));
      if (loopval lt %eval(&d) and maxval lt loopval) then maxval=loopval;
      if (match eq %eval(&a) and maxval lt %eval(&d)) then loopval=%eval(&d);
      do ii=1 to loopval;
        %do ii=1 %to %eval(&c);
          i%eval(&ii)=ranuni(&seed);
        %end;
        output;
      end;
    end;
  drop maxval ii loopval;
/* Sort the child dataset by the key variable and call the
* macro to find the maximum number of child observations
* associated with a key value with method e where e is a
* macro variable and is either:
*   UNI to indicate the univariate method or
*   DAT to indicate the data step method.
* and call the macro to flatten the child data set with
* method f where f is a macro variable and either
*   XX to indicate the data step method or
*   TR to indicate the transpose method.
*/
proc sort data=c;
  by match;
%
%MOM(c,match,&e);
%
%flatten(p,c, ,match,&f);
/* Create p_prime, equivalent to the original parent dataset
* from all, the output dataset from the flatten macro.
*/
data p_prime;
  set all;
  keep match
    %do ii=1 %to %eval(&b);
      j%eval(&ii)
    %end;
/* Create c_prime, equivalent to the original child dataset
* from all, the output dataset from the flatten macro.
*/
data c_prime;
  %do jj=1 %to %eval(&c);
    array _%eval(&jj)_{%eval(&d_max)} %do ii=1 %to %eval(&d_max);
      _%eval(&jj)_%eval(&ii)
    %end;
%end;
set all;
  j=1;
do while(j le %eval(&d_max));
    %do ii=1 %to %eval(&c);
      i%eval(&ii)=_%eval(&ii)_{j};
    %end;
    if (_1_{j} ne .) then output;
    j=j+1;
end;
keep match
  %do ii=1 %to %eval(&c);
    i%eval(&ii)
  %end;
/* Compare the original parent dataset to the one created
* from the output of the macro and retain the return code
* in the macro variable rc1.
proc compare base=p compare=p_prime noprint;
%let rc1=&sysinfo;

/* Compare the original child dataset to the one created
 * from the output of the macro and retain the return code
 * in the macro variable rc2.
 proc compare base=c compare=c_prime noprint;

%let rc2=&sysinfo;

/* If both return codes are zero, write that the iteration
 * was successful to the SASLOG; otherwise, note that the
 * comparison failed.
 */
data _null_;
x=&rc1;
y=&rc2;
if (x eq 0 and y eq 0) then
  do;
    put "DCW     Iteration Successful";
  end;
else do;
    put "DCW     Iteration Failed";
  end;
run;

/* Create a trivial dataset to retain in the datasets procedure
 * which implicitly deletes the rest of the datasets in the
 * WORK library. This step allows the WORK library to be
 * cleared for subsequent iterations.
 */
data x;
x=1;
proc datasets nolist;
  save x;
run;
%mend doit;

%macro MOM(ds,sortvar,engine);
/* This macro finds the maximums number of observations
 * in a sorted child dataset by one of two methods. The
 * univariate method uses two invocations of PROC UNIVARIATE
 * and the data step method uses a single pass through a
 * data step to find this maximum value. The macro variables
 * passed to this procedure are:
 *
 *     ds the dataset,
 *     sortvar the key variable, and
 *     engine the calculation method
 */
data _null_;
  put "DCW     MOM start";
run;
%global d_max;
%if %upcase(&engine) = UNI %then
  %do;
    proc univariate data=&ds noprint;
  %end;
var &sortvar;
output out=d_count n=count;
by &sortvar;

proc univariate data=d_count noprint;
var count;
output out=d_max max=gl_max;

data _null_;  
set d_max;
   call symput('d_max',gl_max);
run;
%end;
%else %do;
   data _null_;  
      retain _max gl_max;
      set &ds end=last;
by &sortvar;
      if (first.&sortvar eq 1) then _max=0;
      _max=_max+1;
      if (last.&sortvar eq 1 and _max gt gl_max) then gl_max=_max;
      if (last eq 1) then call symput('d_max',gl_max);
run;
%end;

%macro flatten(parent,child,list,sortvar,engine);
/* This macro builds that flattened child dataset and
 merges it with the parent dataset by one of two
 methods. The XX method uses a single pass through
 a datastep and the TR method used the output of 
 PROC TRANSPOSE to build SAS arrays of the variables
 in the child dataset passed in the macro variable
 list.
*/
data _null_;  
   put "DCW FLATTEN start";
run;
/* Macro variable initialization */
%global glist;
%let null=
/* The first section of the flatten macro reads the 
 contents of the child dataset into a dataset with
*/
* the same name as the child dataset appended with _info.
* This data set is used by both method to determine if
* the variables to be converted to SAS arrays are numeric
* or character. The _info dataset is merged with the list
* of variables to retain only those that should be converted.
* /
  proc datasets nolist;
  contents data=&child out=&child._info noprint;

  %if &list NE &null %then
    %do;
      data names;
      %let ii=1;
      %do %while(%scan(&list,&ii,' ') ne &null);
        name="%scan(&list,&ii,' ')"; output;
        %let ii=%eval(&ii+1);
      %end;
    %end;

    proc sort data=names;
    by name;

    proc sort data=&child._info;
    by name;

    data &child._info;
    merge names(in=i1) &child._info;
    by name;
    if i1;
  %end;

  /* This is the TRANSPOSE engine
  */
  %if %upcase(&engine) = TR %then
    %do;
      /* Build three macro variables for use later.
      * The first, list, is the list of variables to be
      * converted to SAS arrays. The second, leny, is
      * a list of indicator variables that correspond to
      * the variables in the list. If an indicator is
      * zero the variable is numeric; if it is non-zero,
      * the value of the indicator is the length of the
      * character variable in the list. Finally, chflg
      * is a flag to indicate the presence of character
      * variables in the list.
      */
      data _null_;
      length list lenty $200;
      retain list lenty ' ' charflag 0;
      set &child._info end=last;
      if (upcase(name) ne "%upcase(&sortvar)") then
        do;
          list=trim(left(list)) || ' ' || trim(left(name));
          if (type eq 1) then
            do;
              lenty=trim(left(lenty)) || ' 0';
            end;
          else do;
            lenty=trim(left(lenty)) || ' ' ||
                trim(left(input(put(length,3.),$3.)));
charflag=1;
end;
end;
if (last eq 1) then
do;
call symput( 'list',trim(left(list)));
call symput('lenty',trim(left(lenty)));
call symput('chflg',input(put(charflag,1.),$1.));
end;
run;

/*  The child dataset is transposed by the key variable
* and the output is stored in the dataset _temp. For
* each variable in the list a new dataset is created
* that reads the transposed data for that variable into
* a SAS array col1-colN (where N is the maximum number
* of observations in the child dataset associated with
* a key value) and assigns these values to the SAS
* array _I_1-I_N (where N was as before and I is the
* number of the variable in the list.
* * For example, if the list was "dog cat rat" and the
* maximum number of observations in the child dataset
* associated with a key value were 10, then
* (1) the variables dog, cat, and rat would be transposed
* by the key variable and stored in temp; and
* (2) three datasets would be created, one for each of
* the variables. The dataset associated with the
* variable cat would be the second one and would
* contain the key variable and ten SAS variables
* _2_1-_2_10.
* * Also, the array definitons for each variable and the
* variable name is written to a file named test.file for
* use later.
* * Character and numeric variables are treated differently
* and each temporary dataset is sorted for eventual merging
* with the other temporary datasets.
*/
proc transpose data=&child out=_temp;
var &list;
by &sortvar;
%let ii=1;
%do %while(%scan(&list,&ii,' ') ne &null);
%let name=%scan(&list ,&ii,' ');
%let len=%scan(&lenty,&ii,' ');
%if &len = 0 %then %let len=20;
data _xtemp%eval(&ii);
%if %scan(&lenty,&ii,' ') = 0 %then
%do;
array _&ii._{%eval(&d_max)}
%do jj=1 %to %eval(&d_max);
_&ii._&jj
%end;
;  
%if &chflg = 0 %then
%do;
array col{%eval(&d_max)} col1-col{%eval(&d_max)};
if (_n_ eq 1) then
do;
  vlen=length(trim(left("&name")));
  file "test.file" mod;
  put 1 /* &name */ @;
  put + (16-vlen) "*/ array _&ii._{%eval(&d_max)} " @;
  %do jj=1 %to %eval(&d_max);
    put " _&ii._&jj" @;
  %end;
  put ";";
end;
%end;
%else %do;
  array col{%eval(&d_max)} $%eval(&len) col1-col{%eval(&d_max)};
  if (_n_ eq 1) then
do;
    vlen=length(trim(left("&name")));
    file "test.file" mod;
    put 1 /* &name */ @;
    put + (16-vlen) "*/ array _&ii._{%eval(&d_max)} " @;
    %do jj=1 %to %eval(&d_max);
      put " _&ii._&jj" @;
    %end;
    put ";";
  end;
%end;
%end;
array _&ii._{%eval(&d_max)} $%eval(&len) col1-col{%eval(&d_max)};
%do jj=1 %to %eval(&d_max);
  _&ii._&jj
%end;
array col{%eval(&d_max)} $%eval(&len) col1-col{%eval(&d_max)};
if (_n_ eq 1) then
do;
  vlen=length(trim(left("&name")));
  file "test.file" mod;
  put 1 /* &name */ @;
  put + (16-vlen) "*/ array _&ii._{%eval(&d_max)} $%eval(&len)" @;
  %do jj=1 %to %eval(&d_max);
    put " _&ii._&jj" @;
  %end;
  put ";";
end;
%end;
set _temp;
if _name_ eq "&name";
  %do jj=1 %to %eval(&d_max);
    %if %scan(&lenty,&ii,' ') = 0 %then
do;
      _&ii._{&jj}=col{&jj};
    %end;
  %else %do;
    _&ii._{&jj}=input(put(col{&jj},$%eval(&len).),best%eval(&len).);
  %end;
%end;
keep &sortvar
%do jj=1 %to %eval(&d_max);
   &_ii._&jj
%end;
;
proc sort data=_xtemp%eval(&ii);
   by &sortvar;
%
let ii=%eval(&ii+1);
%end;

/*  This section merges the temporary datasets together
 *  to form the final output dataset.
 */
data all;
   merge &parent
      %do ii=%eval(&ii-1) %to 1 %by -1;
         _xtemp%eval(&ii)
      %end;
   by &sortvar;
/*  Delete temporary datasets from the WORK library.
 */
proc datasets nolist;
   delete
      %let ii=1;
      %do %while(%scan(&list,&ii,'') ne &null);
         _xtemp%eval(&ii)
      %let ii=%eval(&ii+1);
      %end;
   run;
%end;
/*  This is the DATA STEP engine
 */
%else %do;
   /* Build three macro variables for use later.
    * The first, list, is the list of variables to be
    * converted to SAS arrays. The second, xtype, is
    * a list of indicator variables that correspond to
    * the variables in the list. If an indicator is
    * one the variable is numeric; if it is two the
    * variable is character. The third variable, len,
    * is similar to the second and contains a zero for
    * numeric variables and the length of a character
    * variable.
    * NOTE: A similar technique as used by the TRANSPOSE
    * engine above would have reduced the number
    * of macro variables by one.
    */
data _null_
   length list xtype len $200;
   retain list xtype len ' ';
   set &child._info end=last;
   if (upcase(name) ne "%upcase(&sortvar)") then do;
      list=trim(left(list)) || ' ' ||trim(left(name));
   end;

%end;
if (type eq 1) then
  do;
    xtype=trim(left(xtype)) || ' 1';
    len=trim(left(len)) || ' 0';
  end;
else do;
  xtype=trim(left(xtype)) || ' 2';
  len=trim(left(len)) || trim(left(input(length,best12.)));
end;
end;
if (last eq 1) then
  do;
    call symput('list',trim(left(list )));
    call symput('type',trim(left(xtype )));
    call symput('len', trim(left(len )));
  end;
run;

/* The data from the child dataset is read directly into
 * SAS arrays. When the value of the sort variable changes, 
 * the arrays are initialized and the arrays are output 
 * when just before the value of the sort variable changes 
 * using "first." and "last." processing. 
 * 
 * The flattened child dataset is merged with the parent 
 * dataset to create all, the output dataset. 
 * 
 * As in the TRANSPOSE engine, the array definitions for 
 * each variable and the variable name is written to a file 
 * named test.file for use later. 
 */
proc sort data=&child;
  by &sortvar;
  data all;
    retain counter
    %let ii=1;
    %do %while(%scan(&list,&ii,' ') ne &null);
    %do jj=1 %to %eval(&d_max);
        _&ii._&jj
    %end;
    %let ii=%eval(&ii+1);
    %end;
    %let ii=1;
    %do %while(%scan(&list,&ii,' ') ne &null);
       %let name=%scan(&list,&ii,' ');
       %let vtype=%scan(&type,&ii,' ');
       %let vlen=%scan(&len ,&ii,' ');
       %if %eval(&vtype) = 1 %then
           %do;
               array _&ii._{%eval(&d_max)}
               %do jj=1 %to %eval(&d_max);
                   _&ii._&jj
               %end;
           %end;
   %else %do;
    array _&ii._{%eval(&d_max)} $%eval(&vlen)
%do jj=1 %to %eval(&d_max);
   &_ii._&jj
%end;
;
%end;
%let ii=%eval(&ii+1);
%end;
set &child;
by &sortvar;
if (first.&sortvar) then
  do;
    counter=1;
    do ii=1 to %eval(&d_max);
      %let ii=1;
      %do %while(%scan(&list,&ii,' ') ne &null);
         _&ii._{ii}=
      %end;
      %let ii=%eval(&ii+1);
  end;
end;
%let ii=1;
%do %while(%scan(&list,&ii,' ') ne &null);
   _&ii._{counter}=%scan(&list,&ii,' ');
%let ii=%eval(&ii+1);
%end;
counter=counter+1;
if (last.&sortvar) then output;
else delete;
keep &sortvar
%let ii=1;
%do %while(%scan(&list,&ii,' ') ne &null);
   %do jj=1 %to %eval(&d_max);
      _&ii._&jj
%end;
%let ii=%eval(&ii+1);
%end;
proc sort data=all;
   by &sortvar;
proc sort data=&parent;
   by &sortvar;
data all;
   merge &parent all;
   by &sortvar;
data all;
   merge &parent all;
   by &sortvar;
%end;
/* The list of variables in the array statements
   * is passed to the global macro variable glist.
   */
%let glist=&list;
data _null_;
   put "DCW     FLATTEN end";
run;
%mend flatten;
Appendix 2: Macro-call Building Program

data one;
/* This program generates macro calls of the form:
   %doit(seed,a,b,c,d,e,f);
   to repeated call the test macro.
*/
seed=331448139;
do i=1 to 180;
   if (i le 90 and mod(i,2) eq 0) then endx=',UNI,XX);';
   else if (i le 90 and mod(i,2) eq 1) then endx=',UNI,TR);';
   else if (i gt 90 and mod(i,2) eq 0) then endx=',DAT,XX);';
   else if (i gt 90 and mod(i,2) eq 1) then endx=',DAT,TR);';
random=input(put(floor(ranuni(seed)*10000000),7.),$7.);
iter=0;
do while (iter lt 50000 or iter gt 200000);
   iter=ranuni(seed)*200100;
end;
iterx=input(put(floor(iter),6.),$6.);
pvar=0;
do while (pvar lt 5 or pvar gt 30);
   pvar=ranuni(seed)*31;
end;
pvarx=input(put(floor(pvar),2.),$2.);
cvar=0;
do while (cvar lt 5 or cvar gt 30);
   cvar=ranuni(seed)*31;
end;
cvarx=input(put(floor(cvar),2.),$2.);
maxc=0;
do while (maxc lt 7 or maxc gt 21);
   maxc=ranuni(seed)*22;
end;
maxcx=input(put(floor(maxc),2.),$2.);
string='%doit('||right(random)||','
   ||right(iterx)||','
   ||right(pvarx)||','
   ||right(cvarx)||','
   ||right(maxcx)||endx;
file 'macro.calls';
put string;
end;
Appendix 3: SAS Program used to Extract Data from SASLOG and Perform Analysis

options nodate nonumber;
title 'Full Factorial Model with All Covariates';
title2 'Data from 180 Executions of the Test Macro';
filename data 'sim3_180.log';
data one;
/* This data step reads in the log (beginning after the listing
 * of the program) and searches for lines beginning with DCW
 * to find the beginning and ending of various steps. It also
 * searches for timing and memory usage statistics embedded
 * in the SASLOG. Here are some lines from the SASLOG that
 * was analyzed showing examples of the types of lines that
 * are read and processed by this program:
 *  581        %doit(5521840, 93539, 5, 8, 7,UNI,TR);
 *  DCW     Iteration Begin
 *  DCW     Parameters UNI TR: 93539 5 8 7
 *  NOTE: DATA statement used (Total process time):
 *        real time           0.01 seconds
 *        user cpu time       0.01 seconds
 *        system cpu time     0.00 seconds
 *        Memory              135k
 *  NOTE: The data set WORK.P has 93539 observations and 6 variables.
 *  NOTE: DATA statement used (Total process time):
 *        real time           0.19 seconds
 *        user cpu time       0.12 seconds
 *        system cpu time     0.01 seconds
 *        Memory              152k
 */
retain max_meth arr_meth key_vals par_vars ch_vars ch_mxky
   rsum usum ssum msum
   rtsum utsum stsum mtsum
   m_real m_user m_sys m_mem
   f_real f_user f_sys f_mem;
infile data firstobs=618;
input;
if (_infile_ ne '') then
   do;
      if (substr(_infile_,1,3) eq 'DCW') then
         do;
            select (scan(_infile_,2,' '));
            when ('MOM')
               do;
                  if (scan(_infile_,3,' ') eq 'start') then
                     do;
                        rsum=0;
                        usum=0;
                        ssum=0;
                        msum=0;
                        end;
                     else do;
                        ...
m_real=rsum;
m_user=usum;
m_sys=ssum;
m_mem=msum;
end;
end;
when ('FLATTEN')
do;
  if (scan(_infile_,3, ' ') eq 'start') then
do;
    rsum=0;
    usum=0;
    ssum=0;
    msum=0;
  end;
  else do;
    f_real=rsum;
    f_user=usum;
    f_sys=ssum;
    f_mem=msum;
  end;
end;
when ('Parameters')
do;
  max_meth=scan(_infile_,3,' ');  
  arr_meth=scan(_infile_,4,' :');
  key_vals=input(put(scan(_infile_,5,' '),$3.),3.);
  par_vars=input(put(scan(_infile_,6,' '),$3.),3.);
  ch_vars=input(put(scan(_infile_,7,' '),$3.),3.);
  ch_mxky=input(put(scan(_infile_,8,' '),$3.),3.);
end;
when ('Iteration')
do;
  if (scan(_infile_,3, ' ') eq 'Sucessful') then
do;
    t_real=rtsum;
    t_user=utsum;
    t_sys=stsum;
    t_mem=mtsum;
    output;
  end;
else if (scan(_infile_,3, ' ') eq 'Begin') then
do;
    rtsum=0;
    utsum=0;
    stsum=0;
    mtsum=0;
  end;
else
  do;
    file print;
    put "ERRORs encountered during processiong";
  end;
end;
otherwise
  do;
    file print;
    put "Unexpected number of DCW comments";
  end;
end;
end; delete;
end;
if(length(trim(left(_infile_))) ge 16) then
  testval=substr(trim(left(_infile_)),1,16);
else delete;
if (testval in
  ('real time' 'user cpu time' 'system cpu time' 'Memory')) then
do;
  if (testval eq 'Memory') then
do;
    str=put(scan(_infile_,2,' '),$7.);
    if (substr(str,length(str),1) eq 'b') then mult=1;
    if (substr(str,length(str),1) eq 'k') then mult=1024;
    if (substr(str,length(str),1) eq 'm') then mult=1024*1024;
    strip=substr(str,1,length(str)-1);
    msum=msum +(input(put(strip,$7.),7.)*mult);
    mtsum=mtsum+(input(put(strip,$7.),7.)*mult);
  end;
  else if (substr(_infile_,length(_infile_)-6,7) eq 'seconds') then
do;
    if (testval eq 'real time') then
do;
      rsum=rsum +input(put(scan(_infile_,3,' '),$7.),7.2);
      rtsum=rtsum+input(put(scan(_infile_,3,' '),$7.),7.2);
    end;
    else if (testval eq 'user cpu time') then
do;
      usum=usum +input(put(scan(_infile_,4,' '),$7.),7.2);
      utsum=utsum+input(put(scan(_infile_,4,' '),$7.),7.2);
    end;
    else if (testval eq 'system cpu time') then
do;
      ssum=ssum +input(put(scan(_infile_,4,' '),$7.),7.2);
      stsum=stsum+input(put(scan(_infile_,4,' '),$7.),7.2);
    end;
  end;
else do;
  if (testval eq 'real time') then
do;
    rsum=rsum +input(put(scan(_infile_,3,' '),$12.),stimer12.);
    rtsum=rtsum+input(put(scan(_infile_,3,' '),$12.),stimer12.);
  end;
  else if (testval eq 'user cpu time') then
do;
    usum=usum +input(put(scan(_infile_,4,' '),$12.),stimer12.);
    utsum=utsum+input(put(scan(_infile_,4,' '),$12.),stimer12.);
  end;
  else if (testval eq 'system cpu time') then
do;
    ssum=ssum +input(put(scan(_infile_,4,' '),$12.),stimer12.);
    stsum=stsum+input(put(scan(_infile_,4,' '),$12.),stimer12.);
  end;
end;
end; delete;
end;
keep max_meth arr_meth key_vals par_vars ch_vars ch_mxky
  m_real  m_user  m_sys  m_mem
f_real f_user f_sys f_mem
  t_real t_user t_sys t_mem;

proc glm data=one;
  class max_meth arr_meth;
  model t_user t_sys t_mem=
     max_meth arr_meth key_vals par_vars ch_vars ch_mxky
     max_meth*arr_meth
     max_meth*key_vals max_meth*par_vars max_meth*ch_vars max_meth*ch_mxky
     arr_meth*key_vals arr_meth*par_vars arr_meth*ch_vars arr_meth*ch_mxky/ss3;
run;

  title 'Main-Effects Model with Significant Covariates';
  title2 'Data from 180 Executions of the Test Macro';
proc glm data=one;
  class max_meth arr_meth;
  model t_user t_sys t_mem =
     max_meth arr_meth key_vals ch_vars ch_mxky
     max_meth*key_vals max_meth*ch_vars max_meth*ch_mxky
     arr_meth*key_vals arr_meth*ch_vars arr_meth*ch_mxky/ss3;
  lsmeans max_meth arr_meth/pdiff adjust=dunnett;
run;
Appendix 4: SAS Listing of Analysis of Covariance Comparing the Different Methods

Full Factorial Model with All Covariates
Data from 180 Executions of the Test Macro

The GLM Procedure

Class Level Information

<table>
<thead>
<tr>
<th>Class</th>
<th>Levels</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>2</td>
<td>DAT UNI</td>
</tr>
<tr>
<td>arr_meth</td>
<td>2</td>
<td>TR XX</td>
</tr>
</tbody>
</table>

Number of Observations Read 180
Number of Observations Used 180
Full Factorial Model with All Covariates
Data from 180 Executions of the Test Macro

The GLM Procedure

Dependent Variable: t_user

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>15</td>
<td>137241.8657</td>
<td>9149.4577</td>
<td>43.88</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>164</td>
<td>34192.1595</td>
<td>208.4888</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>179</td>
<td>171434.0252</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Square          Coeff Var       Root MSE     t_user Mean
0.800552          42.17497        14.43914     34.23628

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>1</td>
<td>676.11871</td>
<td>676.11871</td>
<td>3.24</td>
<td>0.0736</td>
</tr>
<tr>
<td>arr_meth</td>
<td>1</td>
<td>15.26668</td>
<td>15.26668</td>
<td>0.07</td>
<td>0.7870</td>
</tr>
<tr>
<td>key_vals</td>
<td>1</td>
<td>13133.86407</td>
<td>13133.86407</td>
<td>63.00</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>par_vars</td>
<td>1</td>
<td>99.36143</td>
<td>99.36143</td>
<td>0.48</td>
<td>0.4910</td>
</tr>
<tr>
<td>ch_vars</td>
<td>1</td>
<td>44851.41112</td>
<td>44851.41112</td>
<td>215.13</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky</td>
<td>1</td>
<td>6672.77528</td>
<td>6672.77528</td>
<td>32.01</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>max_meth*arr_meth</td>
<td>1</td>
<td>0.19200</td>
<td>0.19200</td>
<td>0.00</td>
<td>0.9758</td>
</tr>
<tr>
<td>key_vals*max_meth</td>
<td>1</td>
<td>40.11870</td>
<td>40.11870</td>
<td>0.19</td>
<td>0.6615</td>
</tr>
<tr>
<td>par_vars*max_meth</td>
<td>1</td>
<td>60.71574</td>
<td>60.71574</td>
<td>0.29</td>
<td>0.5902</td>
</tr>
<tr>
<td>ch_vars*max_meth</td>
<td>1</td>
<td>240.71536</td>
<td>240.71536</td>
<td>1.15</td>
<td>0.2842</td>
</tr>
<tr>
<td>ch_mxky*max_meth</td>
<td>1</td>
<td>245.38664</td>
<td>245.38664</td>
<td>1.18</td>
<td>0.2796</td>
</tr>
<tr>
<td>key_vals*arr_meth</td>
<td>1</td>
<td>2345.31726</td>
<td>2345.31726</td>
<td>11.25</td>
<td>0.0010</td>
</tr>
<tr>
<td>par_vars*arr_meth</td>
<td>1</td>
<td>39.43881</td>
<td>39.43881</td>
<td>0.19</td>
<td>0.6642</td>
</tr>
<tr>
<td>ch_vars*arr_meth</td>
<td>1</td>
<td>16880.20022</td>
<td>16880.20022</td>
<td>80.96</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky*arr_meth</td>
<td>1</td>
<td>35.27489</td>
<td>35.27489</td>
<td>0.17</td>
<td>0.6814</td>
</tr>
</tbody>
</table>
**Full Factorial Model with All Covariates**  
Data from 180 Executions of the Test Macro

The GLM Procedure

Dependent Variable: t_sys

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>15</td>
<td>28909.56074</td>
<td>1927.30405</td>
<td>39.64</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>164</td>
<td>7973.85552</td>
<td>48.62107</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>179</td>
<td>36883.41626</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Square     Coeff Var      Root MSE  t_sys Mean  
0.783809      35.45493      6.972881      19.66689

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>1</td>
<td>2.21659</td>
<td>2.21659</td>
<td>0.05</td>
<td>0.8312</td>
</tr>
<tr>
<td>arr_meth</td>
<td>1</td>
<td>212.91993</td>
<td>212.91993</td>
<td>4.38</td>
<td>0.0379</td>
</tr>
<tr>
<td>key_vals</td>
<td>1</td>
<td>3891.06621</td>
<td>3891.06621</td>
<td>80.03</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>par_vars</td>
<td>1</td>
<td>74.52270</td>
<td>74.52270</td>
<td>1.53</td>
<td>0.2175</td>
</tr>
<tr>
<td>ch_vars</td>
<td>1</td>
<td>15732.30225</td>
<td>15732.30225</td>
<td>323.57</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky</td>
<td>1</td>
<td>5105.56893</td>
<td>5105.56893</td>
<td>105.01</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>max_meth*arr_meth</td>
<td>1</td>
<td>28.00195</td>
<td>28.00195</td>
<td>0.58</td>
<td>0.4490</td>
</tr>
<tr>
<td>key_vals*max_meth</td>
<td>1</td>
<td>74.06709</td>
<td>74.06709</td>
<td>1.52</td>
<td>0.2189</td>
</tr>
<tr>
<td>par-vars*max_meth</td>
<td>1</td>
<td>1.77457</td>
<td>1.77457</td>
<td>0.04</td>
<td>0.8487</td>
</tr>
<tr>
<td>ch_vars*max_meth</td>
<td>1</td>
<td>4.93483</td>
<td>4.93483</td>
<td>0.10</td>
<td>0.7504</td>
</tr>
<tr>
<td>ch_mxky*max_meth</td>
<td>1</td>
<td>28.10449</td>
<td>28.10449</td>
<td>0.58</td>
<td>0.4482</td>
</tr>
<tr>
<td>key_vals*arr_meth</td>
<td>1</td>
<td>14.84984</td>
<td>14.84984</td>
<td>0.31</td>
<td>0.5813</td>
</tr>
<tr>
<td>par-vars*arr_meth</td>
<td>1</td>
<td>47.23280</td>
<td>47.23280</td>
<td>0.97</td>
<td>0.3258</td>
</tr>
<tr>
<td>ch_vars*arr_meth</td>
<td>1</td>
<td>198.81831</td>
<td>198.81831</td>
<td>4.09</td>
<td>0.0448</td>
</tr>
<tr>
<td>ch_mxky*arr_meth</td>
<td>1</td>
<td>335.86004</td>
<td>335.86004</td>
<td>6.91</td>
<td>0.0094</td>
</tr>
</tbody>
</table>
Full Factorial Model with All Covariates
Data from 180 Executions of the Test Macro

The GLM Procedure

Dependent Variable: t_mem

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>15</td>
<td>4.7721122E18</td>
<td>3.1814081E17</td>
<td>65.53</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>164</td>
<td>7.961825E17</td>
<td>4.8547713E15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>179</td>
<td>5.5682947E18</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Square       Coeff Var    Root MSE  t_mem Mean
0.857015       26.88455     69676189   259168142

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>1</td>
<td>6.9536539E15</td>
<td>6.9536539E15</td>
<td>1.43</td>
<td>0.2331</td>
</tr>
<tr>
<td>arr_meth</td>
<td>1</td>
<td>5.0124365E16</td>
<td>5.0124365E16</td>
<td>10.32</td>
<td>0.0016</td>
</tr>
<tr>
<td>key_vals</td>
<td>1</td>
<td>2.7504584E17</td>
<td>2.7504584E17</td>
<td>56.65</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>par-vars</td>
<td>1</td>
<td>9.1426047E14</td>
<td>9.1426047E14</td>
<td>0.19</td>
<td>0.6649</td>
</tr>
<tr>
<td>ch_vars</td>
<td>1</td>
<td>8.0854396E17</td>
<td>8.0854396E17</td>
<td>166.55</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky</td>
<td>1</td>
<td>1.6616959E17</td>
<td>1.6616959E17</td>
<td>34.23</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>max_meth*arr_meth</td>
<td>1</td>
<td>4.8821007E14</td>
<td>4.8821007E14</td>
<td>0.10</td>
<td>0.7516</td>
</tr>
<tr>
<td>key_vals*max_meth</td>
<td>1</td>
<td>3.2262778E15</td>
<td>3.2262778E15</td>
<td>0.66</td>
<td>0.4161</td>
</tr>
<tr>
<td>par-vars*max_meth</td>
<td>1</td>
<td>2.1362922E15</td>
<td>2.1362922E15</td>
<td>0.44</td>
<td>0.5080</td>
</tr>
<tr>
<td>ch_vars*max_meth</td>
<td>1</td>
<td>7.2147714E15</td>
<td>7.2147714E15</td>
<td>1.49</td>
<td>0.2246</td>
</tr>
<tr>
<td>ch_mxky*max_meth</td>
<td>1</td>
<td>2.8207838E15</td>
<td>2.8207838E15</td>
<td>0.58</td>
<td>0.4470</td>
</tr>
<tr>
<td>key_vals*arr_meth</td>
<td>1</td>
<td>1.5658037E17</td>
<td>1.5658037E17</td>
<td>32.25</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>par-vars*arr_meth</td>
<td>1</td>
<td>3.5874789E15</td>
<td>3.5874789E15</td>
<td>0.74</td>
<td>0.3912</td>
</tr>
<tr>
<td>ch_vars*arr_meth</td>
<td>1</td>
<td>7.5005448E17</td>
<td>7.5005448E17</td>
<td>154.50</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky*arr_meth</td>
<td>1</td>
<td>1.1086047E17</td>
<td>1.1086047E17</td>
<td>22.84</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>
Main-Effects Model with Significant Covariates  
Data from 180 Executions of the Test Macro

The GLM Procedure

Class Level Information

<table>
<thead>
<tr>
<th>Class</th>
<th>Levels</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>2</td>
<td>DAT UNI</td>
</tr>
<tr>
<td>arr_meth</td>
<td>2</td>
<td>TR XX</td>
</tr>
</tbody>
</table>

Number of Observations Read  180
Number of Observations Used  180
Main-Effects Model with Significant Covariates
Data from 180 Executions of the Test Macro

The GLM Procedure

Dependent Variable: t_user

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>11</td>
<td>137034.0443</td>
<td>12457.6404</td>
<td>60.84</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>168</td>
<td>34399.9810</td>
<td>204.7618</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>179</td>
<td>171434.0252</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Square: 0.799340
Coeff Var: 41.79631
Root MSE: 14.30950
Mean: 34.23628

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>1</td>
<td>658.84280</td>
<td>658.84280</td>
<td>3.22</td>
<td>0.0746</td>
</tr>
<tr>
<td>arr_meth</td>
<td>1</td>
<td>82.30888</td>
<td>82.30888</td>
<td>0.40</td>
<td>0.5269</td>
</tr>
<tr>
<td>key_vals</td>
<td>1</td>
<td>13371.16696</td>
<td>13371.16696</td>
<td>65.30</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_vars</td>
<td>1</td>
<td>45771.54754</td>
<td>45771.54754</td>
<td>223.54</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky</td>
<td>1</td>
<td>6714.28607</td>
<td>6714.28607</td>
<td>32.79</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>key_vals*max_meth</td>
<td>1</td>
<td>43.72616</td>
<td>43.72616</td>
<td>0.21</td>
<td>0.6446</td>
</tr>
<tr>
<td>ch_vars*max_meth</td>
<td>1</td>
<td>234.16703</td>
<td>234.16703</td>
<td>1.14</td>
<td>0.2864</td>
</tr>
<tr>
<td>ch_mxky*max_meth</td>
<td>1</td>
<td>249.56779</td>
<td>249.56779</td>
<td>1.22</td>
<td>0.2712</td>
</tr>
<tr>
<td>key_vals*arr_meth</td>
<td>1</td>
<td>2477.76301</td>
<td>2477.76301</td>
<td>12.10</td>
<td>0.0006</td>
</tr>
<tr>
<td>ch_vars*arr_meth</td>
<td>1</td>
<td>17390.96645</td>
<td>17390.96645</td>
<td>84.93</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch mxky*arr_meth</td>
<td>1</td>
<td>32.52336</td>
<td>32.52336</td>
<td>0.16</td>
<td>0.6907</td>
</tr>
</tbody>
</table>
Main-Effects Model with Significant Covariates
Data from 180 Executions of the Test Macro

The GLM Procedure

Dependent Variable: t_sys

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>11</td>
<td>28740.38072</td>
<td>2612.76188</td>
<td>53.90</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>168</td>
<td>8143.03554</td>
<td>48.47045</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>179</td>
<td>36883.41626</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Square: 0.779222
Coef Var: 35.39997
Root MSE: 6.962072
Mean t_sys: 19.66689

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>1</td>
<td>7.72101</td>
<td>7.72101</td>
<td>0.16</td>
<td>0.6903</td>
</tr>
<tr>
<td>arr_meth</td>
<td>1</td>
<td>159.47852</td>
<td>159.47852</td>
<td>3.29</td>
<td>0.0715</td>
</tr>
<tr>
<td>key_vals</td>
<td>1</td>
<td>3950.46283</td>
<td>3950.46283</td>
<td>81.50</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_vars</td>
<td>1</td>
<td>16066.17140</td>
<td>16066.17140</td>
<td>331.46</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky</td>
<td>1</td>
<td>5199.33286</td>
<td>5199.33286</td>
<td>107.27</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>key_vals*max_meth</td>
<td>1</td>
<td>64.74241</td>
<td>64.74241</td>
<td>1.34</td>
<td>0.2494</td>
</tr>
<tr>
<td>ch_vars*max_meth</td>
<td>1</td>
<td>27.74312</td>
<td>27.74312</td>
<td>0.57</td>
<td>0.4504</td>
</tr>
<tr>
<td>key_vals*arr_meth</td>
<td>1</td>
<td>25.63517</td>
<td>25.63517</td>
<td>0.53</td>
<td>0.4681</td>
</tr>
<tr>
<td>ch_vars*arr_meth</td>
<td>1</td>
<td>326.36519</td>
<td>326.36519</td>
<td>4.88</td>
<td>0.0286</td>
</tr>
<tr>
<td>ch_mxky*arr_meth</td>
<td>1</td>
<td>328.34318</td>
<td>328.34318</td>
<td>6.77</td>
<td>0.0101</td>
</tr>
</tbody>
</table>
Main-Effects Model with Significant Covariates
Data from 180 Executions of the Test Macro

The GLM Procedure

Dependent Variable: t_mem

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>11</td>
<td>4.7648975E18</td>
<td>4.331725E17</td>
<td>90.58</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>168</td>
<td>8.0339722E17</td>
<td>4.7821263E15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>179</td>
<td>5.5682947E18</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Square    Coeff Var      Root MSE    t_mem Mean
0.855719    26.68265      69152920     259168142

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Type III SS</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_meth</td>
<td>1</td>
<td>4.9705382E15</td>
<td>4.9705382E15</td>
<td>1.04</td>
<td>0.3094</td>
</tr>
<tr>
<td>key_vals</td>
<td>1</td>
<td>2.7415961E17</td>
<td>2.7415961E17</td>
<td>57.33</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch-vars</td>
<td>1</td>
<td>8.2877423E17</td>
<td>8.2877423E17</td>
<td>173.31</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky</td>
<td>1</td>
<td>1.6601917E17</td>
<td>1.6601917E17</td>
<td>34.72</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>key_vals*max_meth</td>
<td>1</td>
<td>3.6707232E15</td>
<td>3.6707232E15</td>
<td>0.77</td>
<td>0.3822</td>
</tr>
<tr>
<td>ch_vars*max_meth</td>
<td>1</td>
<td>6.8497688E15</td>
<td>6.8497688E15</td>
<td>1.43</td>
<td>0.2331</td>
</tr>
<tr>
<td>ch_mxky*max_meth</td>
<td>1</td>
<td>3.2372004E15</td>
<td>3.2372004E15</td>
<td>0.68</td>
<td>0.4118</td>
</tr>
<tr>
<td>key_vals*arr_meth</td>
<td>1</td>
<td>1.6308226E17</td>
<td>1.6308226E17</td>
<td>34.10</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_vars*arr_meth</td>
<td>1</td>
<td>7.7178776E17</td>
<td>7.7178776E17</td>
<td>161.39</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ch_mxky*arr_meth</td>
<td>1</td>
<td>1.1074141E17</td>
<td>1.1074141E17</td>
<td>23.16</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>
Main-Effects Model with Significant Covariates  
Data from 180 Executions of the Test Macro

The GLM Procedure  
Least Squares Means  
Adjustment for Multiple Comparisons: Dunnett-Hsu

| max_meth | H0:LSMean1= | t_user | LSMean2 | Pr > |t| |
|----------|-------------|--------|---------|------|---|
| DAT      | LSMEAN      | 31.9502459 | 0.0460 |
| UNI      | 36.3438631  |        |        |      |

| max_meth | H0:LSMean1= | t_sys | LSMEAN | Pr > |t| |
|----------|-------------|-------|--------|------|---|
| DAT      | 19.8166354  | 0.8170 |
| UNI      | 19.5701606  |       |        |      |

| max_meth | H0:LSMean1= | t_mem | LSMEAN | Pr > |t| |
|----------|-------------|-------|--------|------|---|
| DAT      | 256071310   | 0.9663 |
| UNI      | 255624310   |       |        |      |
Main-Effects Model with Significant Covariates
Data from 180 Executions of the Test Macro

The GLM Procedure
Least Squares Means
Adjustment for Multiple Comparisons: Dunnett-Hsu

| arr_meth | t_user       | LSMEAN   | Pr > |t| |
|----------|--------------|----------|------|----------------|
| TR       | 48.0237036   | <.0001   |      |               |
| XX       | 20.2704054   |          |      |               |

| arr_meth | t_sys        | LSMEAN   | Pr > |t| |
|----------|--------------|----------|------|----------------|
| TR       | 21.4903518   | 0.0008   |      |               |
| XX       | 17.8964442   |          |      |               |

| arr_meth | t_mem        | LSMEAN   | Pr > |t| |
|----------|--------------|----------|------|----------------|
| TR       | 354807421    | <.0001   |      |               |
| XX       | 156888199    |          |      |               |