Paper 321-2013

Wide-to-Tall: A Macro to Automatically Transpose Wide SAS® Data into Tall SAS Data

James R. Brown, Havi Global Solutions

ABSTRACT

If your SAS world involves forecasting or other date-specific data, you have probably seen column names such as forecast_19224, sales_19230, or inventory_19250. If several of these prefixes exist in a single file, the underlying SAS data file could have thousands of columns.

Analyzing this data is an exercise in scrolling, note-taking, copying and pasting.

PROC TRANSPOSE is not sophisticated enough take on this challenge. This paper presents a macro which will transform your data by automatically creating a CSV file with distinct columns for the date, each prefix variable, and any non-date-suffixed columns in your input.

The non-wizardry behind this makes use of the dictionary tables, SAS name lists (forecast_18950-forecast_19049), and colon notation (forecast_:) to eliminate the task of enumerating long lists of variable names.

INTRODUCTION

In the world of forecasting, more data is better and a lot more data is a lot better. This can lead to file structures which are designed for efficient storage and batch processing, but are not well suited for human understanding and ad hoc analysis.

This macro quickly transforms wide, non-normalized data structures into a tall, normalized form that can be easily viewed in Excel or manipulated using the SAS Procs: MEANS, SUMMARY, TABULATE and REPORT.

Our SAS data is used to forecast daily demand for more than 200 items at 14,000 retail stores. Files routinely contain more than 20,000 columns. The design uses one record per forecast item to hold all of the data for that item: historical data, forecasts, seasonality indicators, calendar information, events, and more. The structure is superb for maintaining and storing the data. It is not good for human interaction. It is difficult to investigate, manipulate and analyze.

That is why this macro exists. It transforms data into a format suitable for SAS summary processing and human understanding.

%smWideToTall

The macro was written with keyword parameters in order to provide default values. They are logically ordered to handle inputs, outputs and run-time details. Once the appropriate defaults are established, actual invocation should require very few parameters.

Input Parameters

```
inlib=work              /* input data library                                              */
indata=                 /* input data file                                                 */
indatalimit=max         /* number of records to read or max for all records                */
indatawhere=            /* where=(field="ABC") or keep=dpia_key hist_total_units: (note the colon) */
indatawhere2=           /* save complex  indatawhere strings without deleting them         */
```

The "in" parameters specify the SAS library, the data file name, the number of records to read, and how to limit the input. "Indatawhere" is optional and is used to specify anything permissible in parenthesis after a SAS data file name:

```
set &inlib..&indata (&indatawhere);
```

"indatawhere" can evolve into a complex SAS phrase, so "indatawhere2" is provided as a place to save carefully crafted "indatawhere" values. The code snippets below are taken from a live program:

```
,indatawhere=keep=dpia_key loc_key source:  hist_total_units: hist_total_price:
```

I encounter situations where the macro needs to run without "indatawhere", so "indatawhere2" is used to keep the parameter string safe until it is needed again. As a note, remember that macro parameters end at the first comma, so equal signs are valid as content. "=keep=" is not a problem.

```
,indatawhere=,
 indatawhere2=keep=dpia_key loc_key source:  hist_total_units: hist_total_price:
```

Columnwhere Parameters

```
columnwhere=          /* where=(name in:('dpia_key','loc_key',"hist_total_units")) */
columnwhere2=         /* save complex  columnwhere strings without deleting them   */
```

%smWideToTall makes use of the SAS dictionary tables to obtain column information about the data file being processed. If the input columns are being limited using "indatawhere", "columnwhere" should be used to provide similar exclusions. Similar to "indatawhere2", "columnwhere2" is provided to save complex strings without deleting them. Note the colon in the commented columnwhere syntax above.

Output Parameters

```
outlib=               /* defaults to &inlib                                        */
outdata=              /* defaults to &outprefx.&indata                             */
outdatawhere=         /* where=(columname="value")                                 */
outdatawhere2=        /* save complex  outdatawhere strings without deleting them  */
```

The "out" parameters match up with their "in" counterparts and function similarly. Defaults are provided. If "outlib" is not specified, it defaults to &inlib. If "outdata" is not specified, it defaults to &outprefx.&indata.

OutCSV Parameters

```
outcsvpath=           /* path for csv file                                         */
outcsvfile=           /* csv filename                                              */
outcsvheader=yes      /* yes,no - first row should contain column names            */
outcsvlimit=max       /* number of csv records to write out or max for all records */
```

%smWideToTall provides the option to write out the transposed data to a CSV file for visualization and manipulation in Excel. The "outcsv" parameters specify the path, filename, a limit to the number of records written, and the ability to turn on or off the column headers. If "outcsvfile" is not specified, it defaults to &outprefx.&indata..csv. "outcsvheader" must be "yes" or "no" because it is passed to SAS, which does not accept "y" or "n". Your first assignment is to improve this code by allowing "y" or "n", but transform it to "yes" or "no" for SAS.

Detail Parameters

```
digitPrefx=_          /* character before the date suffix, typically null or underscore */
keyname=time_key      /* name for the key field build from suffix digits               */
outprefx=t_           /* add this prefix to the output filename, used when defaulting to &indata */
runcsv=no             /* y,n,yes,no - proc print to external file using ods csv        */
runtranspose=yes      /* y,n,yes,no - bypass the transpose logic when creating the csv */
```

The remaining parameters control operational needs of the macro. "digitPrefx" handles situations where the date portion of the column is sometimes prefixed with an underscore (forecast_19010 vs. forecast19010). "keyname" specifies the name for the key field built using the suffix digits (forecast_**001** or forecast_**19246**). "outprefx" provides the ability to add a prefix while still using the default output filename. "runcsv" controls whether or not the CSV file is produced. "runtranspose" is used to skip the transpose logic when only the CSV portion is needed to write out a previously transposed file.

Sample Invocations

```
%smWideToTall(inlib=main_i
            , indata=baseline_history_888888888
            , indatawhere=where=(loc_key=252)
            , outlib=work
            , outdata=tall_baseline_history_888888888
            , outdatawhere=where=(19100 <= time_key <= 19180)
            , runcsv=Y
            , digitPrefx=
            );

%smWideToTall(inlib=jbmainp
            , indata=forecast_dpia_&dpia_key
            , outlib=mylib
            , indatalimit=5
            , indatawhere2=where=(loc_key=252)
            , outcsvlimit=1000
            , runtranspose=N
```

```
            , runCSV=Y
             );

%smwidetotall(inlib=main_i
            , indata=baseline_history_888888888
            , indatawhere=keep=dpia_key loc_key hist_total_units:
            , outlib=work
            , outcsvpath=/ba3/users/jbrown/csv,indatalimit=5
            , outcsvlimit=500
            , outcsvheader=yes
            , columnwhere=where=(name in:('dpia_key','loc_key',"hist_total_units"))
            , digitprefx=
            , runcsv=yes
              );
```

Conclusion

%smWideToTall provides a 'blackbox' solution to transforming wide data files into a manageable form. It automates data transformations and it provides CSV files for exploratory data analysis. %smWideToTall can simplify your life, but it is still your job to understand the data and deliver your projects on time. It will not be the only tool in your toolbox, but, hopefully it will be an essential one.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please put "smWideToTall" in the email subject and contact the author at:

James R. Brown
Havi Global Solutions
305 Highland Parkway
Suite 200
Downers Grove, IL 60515
jbrown@havigs.com

SOURCE CODE

```
01    /*************************************************************************
02    *  PROGRAM:     smWideToTall.sas
03    *  SYSTEM:      jim
04    *  COMPILER     SAS
05    *  MODULE TYPE: included SAS Macro
06    *
07    *  AUTHOR:      Jim Brown
08    *  WRITTEN:     10/21/2012
09    *
10    *  FUNCTION:    transpose 'wide' data files into tall files
11    *               by reading a very wide file with many variables
12    *               suffixed with dates.  Create one record per date
13    *               and drop the suffixes
14    *
15    *  INVOCATION:
16    *
17    *       ods noresults;
18    *
19    *       %smWideToTall(inlib=jbmainp
20    *                    ,indata=forecast_dpia_&dpia_key
21    *                    ,outlib=work
22    *                    ,indatalimit=5
23    *                    ,indatawhere=where=(loc_key=252)
24    *                    ,outcsvlimit=10000);
25    *
26    *       %smWideToTall(inlib=jbmainp
27    *                    ,indata=forecast_dpia_&dpia_key
28    *                    ,outlib=work
29    *                    ,indatalimit=5
30    *                    ,indatawhere=where=(loc_key=252)
31    *                    ,outcsvlimit=10000
```

```
32    *                   ,runTranspose=N
33    *                   ,runCSV=Y
34    *                   );
35    *
36    *  NOTES:
37    *
38    *  DATE     WHO      WHAT
39    *  -------- -------- ----------------------------------------------------
40    *  10/21/12 j brown  written
41    *  03/07/13 j brown  drop numdigits, add xSequence, keyname
42    *
43    ******************************************************************/
44
45    %macro smWideToTall(inlib=work       /* lib                           */
46                       ,indata=          /* data                          */
47                       ,indatalimit=max  /* nbr input records to process  */
48                       ,indatawhere=     /* where=(abc=def)               */
49                       ,indatawhere2=    /* dummyvar to save indatawhere  */
50                       ,columnwhere=     /* where=(abc=def)               */
51                       ,columnwhere2=    /* dummyvar to save columnwhere  */
52                       ,outlib=          /* &inlib                        */
53                       ,outdata=         /* &outprefx&indata              */
54                       ,outdatawhere=    /* where=(abc=def)               */
55                       ,outdatawhere2=   /* dummyvar to save outdatawhere */
56                       ,outcsvfile=      /* &outprefx&indata.csv          */
57                       ,outcsvheader=yes /* column headers on csv file    */
58                       ,outcsvlimit=max  /* max output records for csvout */
59                       ,outcsvpath=/ba3/users/jbrown/csv
60                                         /* no trailing slash             */
61                       ,outprefx=t_      /* prefix for default filename   */
62                       ,digitPrefx=_     /* character before date suffix  */
63                       ,keyname=time_key /* name for 'digits' field       */
64                       ,runTranspose=yes /* run wide to long              */
65                       ,runcsv=no        /* ods csv to external file      */
66                       );
67                                    /************************************
68                                    simplify log messages
69                                    ************************************/
70        Data _NULL_;
71            call symputx("_Star70" ,repeat("*",70)                    ,"L");
72            call symputx("_NOTE"   ,cats("NOTE: "   ,"&SYSMACRONAME.."),"L");
73            call symputx("_NOTE2"  ,cats("NOTE2: "  ,"&SYSMACRONAME.."),"L");
74            call symputx("_WARNING",cats("WARNING: ","&SYSMACRONAME.."),"L");
75            call symputx("_ERROR"  ,cats("ERROR: "  ,"&SYSMACRONAME.."),"L");
76            run;
77                                    /************************************
78                                    Parameter Validation
79                                    ************************************/
80        %If &inlib= %Then
81            %Do;
82                %put &_STAR70;
83                %put %sysfunc(compbl(%STR(
84                     &_ERROR.105 Parameter inlib must be specified
85                     )));
86                %put &_STAR70;
87                %Return;
88            %End;
89
90        %If &indata= %Then
91            %Do;
92                %put &_STAR70;
93                %put %sysfunc(compbl(%STR(
94                     &_ERROR.110 Parameter indata must be specified
95                     )));
96                %put &_STAR70;
97                %Return;
98            %End;
99
100       %If (%Sysfunc(Exist(&inlib..&indata,data)) Eq 0) %Then
101           %Do;
102               %put &_STAR70;
103               %put %sysfunc(compbl(%STR(
104                    &_ERROR.112 data file [&inlib..&indata]
105                    Does Not Exist.
106                    )));
107               %Put &_STAR70;
108               %Return;
```

```
109          %End;
110
111      %If &indatalimit= %Then
112          %Do;
113              %put &_STAR70;
114              %put %sysfunc(compbl(%STR(
115                   &_ERROR.115 Parameter indatalimit must be specified
116                   )));
117              %put &_STAR70;
118              %Return;
119          %End;
120
121      %If &outlib= %Then
122          %Do;
123              data _NULL_;
124                  call symputx('outlib',"&inlib");
125                  run;
126              %put &_STAR70;
127              %put %sysfunc(compbl(%STR(
128                   &_WARNING.120 Parameter outlib not specified,
129                                 defaults to &outlib
130                   )));
131              %put &_STAR70;
132          %End;
133
134      %If &outdata= %Then
135          %Do;
136              data _NULL_;
137                  work=translate("&outprefx.&indata","__",",.");
138                  call symputx('outdata',work);
139                  run;
140              %put &_STAR70;
141              %put %sysfunc(compbl(%STR(
142                   &_WARNING.125 Parameter outdata not specified,
143                                 defaults to &outdata
144                   )));
145              %put &_STAR70;
146          %End;
147
148      %If &keyName= %Then
149          %Do;
150              %put &_STAR70;
151              %put %sysfunc(compbl(%STR(
152                   &_ERROR.127 Parameter keyName must be specified
153                   )));
154              %put &_STAR70;
155              %Return;
156          %End;
157
158      %If (%SysFunc(IndexW(YES Y NO N,%UpCase(&runTranspose))) EQ 0) %Then
159          %Do;
160              %put &_STAR70;
161              %put %sysfunc(compbl(%STR(
162                   &_ERROR.130 Parameter runTranspose(&runTranspose)
163                   Must be Yes, Y, No, N
164                   )));
165              %put &_STAR70;
166              %Return;
167          %End;
168
169      %If (%SysFunc(IndexW(YES Y NO N,%UpCase(&RunCSV))) EQ 0) %Then
170          %Do;
171              %put &_STAR70;
172              %put %sysfunc(compbl(%STR(
173                   &_ERROR.135 Parameter RunCSV(&RunCSV)
174                   Must be Yes, Y, No, N
175                   )));
176              %put &_STAR70;
177              %Return;
178          %End;
179
180      %If (%SysFunc(IndexW(YES Y,%UpCase(&RunCSV))) > 0) %Then
181          %Do;
182              %If &outcsvfile= %Then
183                  %Do;
184                      data _NULL_;
185                          call symputx('outcsvfile',"&outprefx.&indata..csv");
```

```
186                         run;
187                     %put &_STAR70;
188                     %put %sysfunc(compbl(%STR(
189                          &_WARNING.145 Parameter outcsvfile not specified,
190                          defaults to &outcsvfile
191                          )));
192                     %put &_STAR70;
193                 %End;
194
195             %If (%SysFunc(IndexW(YES NO,%UpCase(&outcsvheader))) EQ 0) %Then
196                 %Do;
197                     %put &_STAR70;
198                     %put %sysfunc(compbl(%STR(
199                          &_ERROR.150 Parameter outcsvheader(&outcsvheader)
200                          Must be Yes or No
201                          when parameter RunCSV=&RunCSV
202                          )));
203                     %put &_STAR70;
204                     %Return;
205                 %End;
206
207             %If &outcsvLimit= %Then
208                 %Do;
209                     %put &_STAR70;
210                     %put %sysfunc(compbl(%STR(
211                          &_ERROR.155 Parameter outcsvLimit must be specified
212                          when parameter RunCSV=&RunCSV
213                          )));
214                     %put &_STAR70;
215                     %Return;
216                 %End;
217
218             %If "&outcsvpath"="" %Then
219                 %Do;
220                     %put &_STAR70;
221                     %put %sysfunc(compbl(%STR(
222                          &_ERROR.160 Parameter outcsvpath must be specified
223                          when parameter RunCSV=&RunCSV
224                          )));
225                     %put &_STAR70;
226                     %Return;
227                 %End;
228         %End;
229                                 /************************************
230                                 Mainline Execution
231                                 ************************************/
232
233       %If (%SysFunc(IndexW(YES Y, %UpCase(&runTranspose.))) > 0) %Then
234             %do;
235                                 /************************************
236                                 get the columns from the dictionary
237                                 never put upcase in dictionary select
238                                 die if zero rows selected
239                                 ************************************/
240             %let inlib=%upcase(&inlib);
241             %let indata=%upcase(&indata);
242
243             proc sql feedback sortmsg noprint;
244                 create table wtl_columnlist as
245                 select *
246                 from   dictionary.columns
247                 where  libname         =  "&inlib"
248                 and    memname         =  "&indata"
249                 order by varnum;
250                 quit;
251                                 /************************************
252                                 zero rows selected ?
253                                 ************************************/
254             data _NULL_;
255                 if 0 then set wtl_columnlist nobs=howmanycolumns;
256                 put "&_STAR70";
257                 put  "&_NOTE2.208 " howmanycolumns=;
258                 put "&_STAR70";
259                 if howmanycolumns=0 then
260                     do;
261                         put "&_Star70";
262                         put "&_ERROR.210 no rows selected in column select"
```

```
263                         "where libname=&inlib and memname=&indata";
264                     put "&_Star70";
265                 end;
266             stop;
267             run;
268
269         data wtl_columnlist2;
270             set wtl_columnlist (&columnwhere);
271             wtlSeq    + 1;
272                             /************************************
273                             if a digit prefix is used, search from
274                             the right side to find the last
275                             non-digit and non-prefix
276                             ************************************/
277
278             %If "&digitPrefx"^="" %Then
279                 %Do;
280                     lastalpha = findc(strip(name)
281                                     ,'0123456789&digitPrefx'
282                                     ,"v",0-length(name));
283                     if  0 < lastalpha < length(name) then
284                         do;
285                             datepart = substr(name  ,lastalpha+1);
286                             namepart = substr(name,1,lastalpha-1);
287                         end;
288                     else
289                         do;
290                             datepart = "";
291                             namepart = name;
292                         end;
293                 %end;
294                             /************************************
295                             if a digit prefix is not used, search
296                             from the right side to find the last
297                             non-digit
298                             ************************************/
299             %else
300                 %Do;
301                     lastalpha = findc(strip(name)
302                                     ,'0123456789'
303                                     ,"v",0-length(name));
304
305                     if  0 < lastalpha < length(name) then
306                         do;
307                             datepart = substr(name   ,lastalpha+1);
308                             namepart = substr(name, 1,lastalpha  );
309                         end;
310                     else
311                         do;
312                             datepart = "";
313                             namepart = name;
314                         end;
315                 %end;
316                             /************************************
317                             if datepart is not numeric, there is
318                             unexpected variable name which must
319                             be dropped, or the code must be changedsearch
320                             ************************************/
321             if  lengthn(datepart)> 0
322             and verify(strip(datepart),'0123456789') > 0 then
323                 do;
324                     xlengthn=lengthn(datepart);
325                     put "&_STAR70";
326                     put "&_ERROR.214 Unexpected variable name structure:"
327                         xlengthn= name= namepart= datepart=;
328                     put "&_STAR70";
329                     delete;
330                 end;
331             run;
332                             /************************************
333                             get the maximum length for each
334                             namepart
335                             ************************************/
336
337         proc sql feedback sortmsg noprint;
338             create table wtl_columnlist3 as
339             select namepart
```

```
                    , type
                    , max(length) as maxlength
                    , case
                         when upcase(type)='CHAR' then "$" || put(max(length),z5.)
                                                  else       put(max(length),z5.)
                     end as xlength format=$8.
                from   wtl_columnlist2

                group by namepart
                       , type
                order by namepart
                       , type
                ;
                                /************************************
                                build the length statement except for
                                literal 'length'
                                ************************************/
            %let zLength=;
            proc sql feedback sortmsg noprint;
                select    catx(" ",namepart, xlength)
                into      :zLength separated by  " "
                from      wtl_columnlist3
                order by  catx(" ",namepart, xlength);
                quit;

            %put &_STAR70;
            %put &_NOTE2.212 zLength:&zLength;
            %put &_STAR70;
                                /************************************
                                fields without number suffixes are keys
                                ************************************/
            %let _keylist=;
            proc sql feedback sortmsg noprint;
                select   name
                       , varnum
                into     :_keylist separated by " "
                       , :_varnum  separated by " "
                from     wtl_columnlist2
                where    datepart=""
                order by varnum;
                quit;
                                /************************************
                                no keys is an error
                                ************************************/

            %If &_keylist= %Then
                %Do;
                    %put &_STAR70;
                    %put %sysfunc(compbl(%STR(
                         &_ERROR.220 No keylist values selected
                         )));
                    %put &_STAR70;
                    %return;
                %End;
            %else
                %Do;
                    %put &_STAR70;
                    %put %sysfunc(compbl(%STR(
                         &_NOTE2.222 keylist values:&_keylist
                         )));
                    %put &_STAR70;
                %End;
                                /************************************
                                fields with number suffixes are names
                                ************************************/
            %let _namelist=;
            proc sql feedback sortmsg noprint;
                select   namepart
                       , min(varnum) as varnum
                into     :_namelist separated by " "
                       , :_varnum   separated by " "
                from     wtl_columnlist2
                where    datepart ^=" "
                group by namepart
                order by varnum;
                quit;
                                /************************************
```

```
417                                 no names is an error
418                                 ************************************/
419
420             %If &_namelist= %Then
421                 %Do;
422                     %put &_STAR70;
423                     %put %sysfunc(compbl(%STR(
424                          &_ERROR.230 No namelist values selected
425                          )));
426                     %put &_STAR70;
427                     %return;
428                 %End;
429             %else
430                 %Do;
431                     %put &_STAR70;
432                     %put %sysfunc(compbl(%STR(
433                          &_NOTE2.232 namelist values:&_namelist
434                          )));
435                     %put &_STAR70;
436                 %End;
437                                 /************************************
438                                 get the date range
439                                 ************************************/
440             %let _datelo=;
441             %let _datehi=;
442             proc sql feedback sortmsg noprint;
443                select  min(datepart)
444                       ,max(datepart)
445                into   :_datelo
446                      ,:_datehi
447                from    wtl_columnlist2
448                where   datepart ^=" ";
449                quit;
450                                 /************************************
451                                 no dates is an error
452                                 ************************************/
453
454             %If &_datelo&_datehi= %Then
455                 %Do;
456                     %put &_STAR70;
457                     %put %sysfunc(compbl(%STR(
458                          &_ERROR.240 No date values selected
459                          )));
460                     %put &_STAR70;
461                     %return;
462                 %End;
463             %else
464                 %Do;
465                     %put &_STAR70;
466                     %put %sysfunc(compbl(%STR(
467                          &_NOTE.242 date range: ==>&_datelo<== ==>&_datehi<==
468                          )));
469                     %put &_STAR70;
470                 %End;
471                                 /************************************
472                                 save current option values before
473                                 resetting them
474                                 ************************************/
475             data _NULL_;
476                 call symputx("_OptObs",getoption("obs","keyword"));
477                 run;
478
479             options obs=&indatalimit;
480
481                                 /************************************
482                                 transpose wide to long.  each name
483                                 gets a separate record with all of the
484                                 keys.
485                                 ************************************/
486
487             data &outlib..&outdata
488                 (keep= xSequence &keyname &_keylist &_namelist
489                 &outdatawhere
490                 );
491                 retain xSequence 0 &keyname &_keylist &_namelist;
492                 length &zLength.;
493                 %let syscc=0;
```

```
494
495                 set &inlib..&indata
496                     (
497                     &indatawhere
498                     );
499
500                 %if &syscc > 4 %then
501                     %do;
502                         %put &_STAR70;
503                         %put %sysfunc(compbl(%STR(
504                              &_ERROR.260 set &inlib..&indata failed,
505                                          syscc=&syscc
506                              )));
507                         %put &_STAR70;
508                         %return;
509                     %end;
510                                 /************************************
511                                 loop thru each of the names
512                                 assume all names have the same range
513                                 ************************************/
514                 xSequence+1;
515
516                 %do _xdate=&_datelo %to &_datehi %by 1;
517                     &keyname    = &_xdate;
518                     %let _xn    = 1;
519                     %let _xname = %scan(&_namelist,&_xn);
520
521                     %do %while("&_xname"^="");
522
523                         &_xname     = &&_xname.&digitPrefx.&_xdate;
524                         %let _xn    = %eval(&_xn+1);
525                         %let _xname = %scan(&_namelist,&_xn);
526                     %end;
527
528                     output;
529                 %end;
530                 run;
531                                 /************************************
532                                 restore original option values
533                                 ************************************/
534             options &_OptObs;
535         %end;
536                                 /************************************
537                                 send it to csv file
538                                 ************************************/
539
540     %If (%SysFunc(IndexW(YES Y, %UpCase(&RunCSV.))) > 0) %Then
541         %do;
542             ods listing close;
543             ods csv file="&outcsvpath/&outcsvfile"
544                 options(table_headers="&outcsvheader");
545                 proc print data=&outlib..&outdata(obs=&outcsvlimit)  noobs;
546                 run;
547             ods csv close;
548             ods listing;
549         %end;
550
551 %mend smWideToTall;
552
553 /*********************************************************************
554 * end smWideToTall
555 *********************************************************************/
```