Coders' Corner Panel Discussion on Problem Solving

Frank DiIorio, Paul Dorfman, Ian Whitlock
Introduction by Steve Noga

Introduction

As abstracts for the Coders' Corner section began to filter in, a veteran SESUG participant suggested that a panel discussion on problem solving might be interesting. This idea was floated among the conference and section chairs with all parties agreeing that such a panel discussion could make for an interesting session. The panel needed to be comprised of individuals with certain traits including, but not limited to outstanding knowledge of SAS®, being comfortable speaking in front of a group, desire to educate an audience without being overbearing.

One of the main reasons for presenting a topic such as 'problem solving' in a panel discussion is because there are any number of approaches and techniques to solve the problem, but an equal number of traps that a programmer can fall into. This panel of experienced SAS users have each attacked the problem in their own style. The reader, regardless of their own level of expertise with SAS, should come away with a good understanding of issues involved in writing code to solve a problem.

Problem

The full problem, location code format, and a sample of data can be found in **Attachment 1**. The problem was designed to replicate a real-world situation without making it too complex for a one-hour panel session. To solve the problem, the following tasks needed to be accomplished:

- Read and understand the specs
- Read data from an external file
- Manipulate data
- Display results

The participants were each emailed the problem and then left to their own devices to solve it. What follows are the three solutions to the problem.

Participants

Frank DiIorio, Paul Dorfman, and Ian Whitlock agreed to participate in the panel.

Frank DiIorio is author of "SAS Applications Programming: A Gentle Introduction" and (with Ken Hardy) "Quick Start to Data Analysis with SAS." He was co-chair of SESUG 94 and SESUG 96 and is past President of the SouthEast SAS Users Group.

Paul Dorfman, originally a physicist (whence Ph.D. comes), started using SAS in computational physics an undetermined period of time ago. Thereafter, used it for client-, customer-, and self-aggrandizing in engineering, telops, pharma, banking, and managed healthcare. Specializes in programming high-performance custom routines. Speaks some tongues other than SAS, not limited to C, Ukrainian, and Cobol. Awards: 'Sashole' from WACB (World Association of Cobol Bigots), 1995, 'MVS' from SAS-L, 2000.

Ian Whitlock has a Ph.D. in mathematics and comes to programming from a former teaching career. He has been active many SUGI, SESUG and NESUG conferences in addition to his local DCSUG and in house WesSug, in addition to being a frequent contributor on SAS-L.

Acknowledgments

The section chair would like to thank Frank, Paul, and Ian for taking the time and effort to participate in the panel discussion. He would also like to express his thanks to Andrew T. Kuligowski for not only suggesting the problem solving panel idea, but also for providing the sample problem as a starting point for this exercise.

Attachment 1: Sample Problem and Data

The famed oceanographer, Dr. Nigel Dripp-Drye, has obtained a data set containing water temperatures of the Gulf of Mexico, taken at various locations, from Pensacola to Key West, during Thanksgiving (Nov 25) week 1999 (Sunday through Saturday). The data is not sorted in any order. However, there may be some records for days preceding or following the week in question, these should be ignored, and since we are trying to replicate real world data files, there may be additional quirks in the file. The data is formatted as follows:

LOCATION CODE (00-99)	01-02
DATE (mm/dd/yyyy)	04-13
TIME (24 hour)	15-18
DEPTH (meters)	21-24
TEMPERATURE (see below)	26-36
COLLECTOR (last name, first)	38-50

Two things to note: First, more than one individual entered the professor's data for him. Some of them entered the temperature using a standard decimal notation (with 2 digit precision), while others entered it as a fraction, such as "27 1/4". Second, since each collector was receiving a stipend for each temperature collected, some may have used assistants to collect the data even though it was still entered into the data file under the primary collectors' name. In cases like the above, it is conceivable that the data might look like a collector was in two places at once, and because of the assistants, these records will be counted as valid. However, if an assistant got overzealous and entered more than one temperature for the same time, date, location code, and collector then only the maximum temperature taken from the maximum depth for that time, date, location code, and collector will be counted as valid.

Dr. Dripp-Drye has requested a simple report, sorted by Location Name and Collector. (A code/name translation table is included as an addendum to this text.) He is only interested in temperatures taken between 6:00pm and 8:00pm at a depth of between 10 and 15 meters - all others should be ignored. The format should resemble the following and be as pleasing on the eyes as possible since Dr. Dripp-Drye has to submit the report to his superiors.

```
                                    Sun  Mon  Tue  Wed  Thu  Fri  Sat
Location   Collector                (n=) (n=) (n=) (n=) (n=) (n=) (n=)
---------------------------         ---- ---- ---- ---- ---- ---- ----
Code (Text)
   First name Last name
        N (%)                       xx(xx.x%) ………………………..
        Mean                        xx.x ………………………………
        Min                         xx.x ………………………………
        Max                         xx.x ………………………………
```

(Only list codes and collectors that actually have valid data. For those days when a collector did not have any valid collections for a code, just leave the column stats blank. (n=) in the column headers represents the total number of valid collections for that day. N represents the total number of valid collections by the individual collector for that day and (%) is a based on N/(n=) with the sum of all (%)s in a column equaling 100.0)

Codes & Corresponding Locations

00 – PENSACOLA
01 - PANAMA CITY
08 – APALACHICOLA
09 - CEDAR KEY
13 - TARPON SPRINGS
14 – CLEARWATER
15 - ST. PETERSBURG
16 – BRADENTON
17 – SARASOTA
22 – CAPTIVA
24 – FLAMINGO
25 - KEY WEST

Sample Data (20 obs out of 1,500)

01	11/26/1999	1700	17	20.74	Wahiini
16	11/23/1999	1830	5	20	Beard, B.
08	11/27/1999	1800	2	24.64	FISH, SEYMOUR
00	11/23/1999	0700	7	23 3/4	morgan, Cap'n
14	11/26/1999	1800	10	23 13/16	Beard, B.
09	11/20/1999	1830	10	25 1/2	FISH, SEYMOUR
25	11/28/1999	1900	12	24 3/16	Beard, B.
22	11/20/1999	1830	11	20.05	Beard, B.
01	11/23/1999	1800	13	21.29	FISH, SEYMOUR
08	11/26/1999	1930	10	24 9/16	Wahiini
01	11/19/1999	1800	17	21 11/16	Beard, B.
01	11/26/1999	1930	10	21 5/16	Whostow, Jock
09	11/25/1999	1830	4	21.14	Beard, B.
01	11/21/1999	1800	0	21.68	Tide, Rip
13	11/20/1999	1830	10	24.90	Whostow, Jock
09	11/28/1999	1900	1	25 3/4	FISH, SEYMOUR
09	11/22/1999	1800	14	20.42	morgan, Cap'n
24	11/20/1999	1930	10	23 3/8	Wahiini
02	11/20/1999	1930	10	23.60	Wahiini
02	11/27/1999	1800	10	24.94	Wahiini

Deep Blue SAS: One Programmer’s Approach to Problem-Solving

Frank C. DiIorio
Advanced Integrated Manufacturing Solutions, Co.
Durham NC

My comments are organized into some of the pre-coding issues, followed by thoughts on the actual program’s techniques and organization. The last section briefly notes the impact of some changes that would probably be made in Version 2.

PRE-CODING

The first step is to look at what you have been handed, namely specs and data.

SPECS

The specs adequately described the background of the study, the format of the data, quirks in data format (temperature, for example), location code mappings, and the layout of the report. As is always the case with even simple projects like this, implementation revealed holes in the spec (handling of missing values, degree of parameterization, etc.). These will be discussed as we go. It’s important, though, to remember that *any* written spec, no matter how well-crafted, will have some omissions that are revealed only when coding begins.

DATA

The data source is small enough (1,500 records) to examine visually. We look both for adherence to the spec and features not covered by the spec. Notice that the time field is in 24-hour format and does not contain a colon separating hours and minutes. Notice too that the date contains slashes and location code has leading zeroes.

Since we have to manipulate the name field (changing to first-last on output from last-first on input), we should pay attention to the method used for separating names. Examination of the field reveals not only (a) people who probably know the Jimmy Buffet songbook by heart, but also (b) people with one name. Finally, we see on examination of Location that there are codes (02, 23) that were not mentioned in the spec and so will not have a descriptive label on output.

SIMPLE STATS

Visual examination is fine, but even in such a small dataset some descriptive statistics should be run. A quick DATA step will produce a bare-bones dataset. Frequency counts of values such as name, location, date/time, and temperature should be examined for inconsistent name capitalization, invalid codes, and unexpected methods of temperature data representation.

CODING TECHNIQUES

The key is to develop the program incrementally. First, read the data, do essential observation filtering and data massaging, then work on shaping the dataset for the reporting procedure(s).

The program is reproduced in **Figure 1**, which follows this text. Comments in the text match numbers in the program entered as **{number}**.

{1} Parameterize

This wasn’t in the spec, but you can’t tell me that once Dr. Drye’s superiors see such a nice report they won’t want similar output for different dates and/or times. It isn’t that hard to do, so just build it in from the start.

{2} Format-driven labeling

This is the only reasonable way to assign text identifiers to values of a variable. It can also help identify location codes that “fall through the cracks” (we can test for the formatted values mapping to “**unknown**”).

{3} Time as character

We don’t need to convert the time field to a time constant. The only way it will be used is for observation filtering, and for this purpose character representation is adequate (and a tad more efficient than reading it as numeric).

{4} Filter first, then tweak

Rather than convert temperature for every record we read and then use only some of the records, we filter the records first, then do the temperature adjustments. This efficiency consideration yields minimal processing speed improvements, but is a good habit to develop for when file sizes and/or data adjustments are more of an issue.

{5} Sort, then select

The spec requires that duplicate location – collector – date – time records be eliminated. It also requires that the max depth-temperature combination be the record selected in these situations. This means a single SORT with the NODUPKEY option *cannot* be used to eliminate duplicates because we have to use depth and temperature as sort keys.

Instead, we sort to arrange the data in the correct order, then use a DATA step to pick the first (maximum) record for a recorded time. This two-pass solution is less efficient than a single sort, but is necessary. The DATA step also allows us to easily issue a message about duplication (again, we anticipate “scope creep”).

{6} Summarize

Although they are easy to calculate in a DATA step, it makes more sense to let the procedures do the work for you. SUMMARY and MEANS could accomplish this. I use MEANS just out of habit. Notice the use of BY rather than CLASS – this takes advantage of the dataset’s sort order and saves a bit of processing time.

At this point, we have all the information we need for reporting. Now it’s a matter of choosing a reporting tool and reshaping the dataset to facilitate use of that procedure or DATA step. I chose the REPORT procedure. _NULL_ data steps require too much coding and are not responsive to formatting changes. PRINT is easy to use but limited with respect to column-spanning headers, text wrapping within columns, and similar “prettifying” aspects of report writing. Finally, I didn’t want to have to re-learn TABULATE.

{7} Transposing

Given the need for the combined N-% field it seemed to me that the most straightforward way to massage the data was to make everything that was summarized into a character field. That way, it’s easy to manipulate pieces of the field as needed – the parentheses, percent sign and the like can be inserted with concatenation operators or the SUBSTR function.

Since we had this data handling requirement it didn’t seem like the ACROSS feature of REPORT columns (in effect, a transpose) would be helpful. I used the TRANSPOSE procedure to create a dataset (SUMMTRAN) whose observations were distinct location – collectors, with variable names corresponding to dates.

As an aside, this multiple pass approach to problem solving is a hallmark of SAS programming. We read the data, passed it to a summarizing procedure, then passed *that* data to a transposing procedure. All perfectly reasonable and concise, but a difficult aspect of SAS programming to pick up if you're a novice.

{8} SQL-generated macro variables

The transposed summary dataset would be ready for some quick formatting and REPORT but not for a few things: we don't know how many distinct dates were present in the dataset; we don't know which dates were present; and we don't know the weekday of these dates or the number of measurements taken on each day (remember the column heading requirement).

Rather than postprocess the summary data and merge it with the transposed data, I used SQL to create a set of macro variables that will be used in later DATA steps and PROCs as array bounds, array constants, and in column headers in the report.

{9} Transpose dataset reshaping and formatting

Dataset SUMMTRAN is in the correct order for reporting. What we need to do now is calculate percentages across days and do some sleight of hand to insert Location-Collector rows.

We use some of the macros generated in {8} to define array bounds and elements **{9a}**. As we pass through each observation we look for the start of a location or collector **{9b}**. If so, assign text to variable ROWHEAD (the first column in the report) and output an observation with missing values for the VALUEx variables.

Finally, all that's left to do is loop through each DTyyyy_mm_dd variable and assign it to the corresponding VALUESxx variable **{9c}**. The content will depend on whether we are reading an observation contain n, mean, min, or max data. In all cases, we create a character variable (VALUEx) from a PUT, and possibly other manipulations, of a numeric variable.

{10} The REPORT procedure

Once we get this far, the work is basically done. The use of REPORT is straightforward, the only tricky part being the definition of a macro so we can have a %DO loop to define the column header text for each of the DT variables. The header scans the appropriate portion of the weekday name and daily count macro variables created in {8}. These headers could have been defined in the previous DATA step (part {9}), but seeing the values defined in-line makes for somewhat easier reading.

The title reflects the time and date constraints and is build using the user-supplied values at the top of the program. A few lines of the output are reproduced in **Figure 2,** below.

Here are some modifications to this program that are suggested from real-world experience:

- Print all names for all locations even if they don't have any data for any dates in range. Currently, if collector "x" has no data for any day at a location he/she will not be in that location's summary. It may be desirable to display collector "x" with blanks in every day. This could be done by creating a reference dataset with every *possible*, rather than every *present* combination of location-collector, then merging it with SUMMTRAN (the reference dataset could be created as a FREQ output dataset with the SPARSE option). A related refinement to the program might require display of all *days* even if no data collection occurred on a day.
- If no observations are in range, print alternate report rather than generate no LST file. This is easily achieved by using SQL to retrieve the number of observations in the dataset (dictionary table TABLES column NOBS), then conditionally executing a "good" report if the count exceeds 0 or printing a "failure" report if the observation count were 0.
- Collapse collectors across days and/or locations. That is, summarize a collector's activity for the week or the location's activity across collectors. This could be addressed in the DATA step before the REPORT procedure (item {9}). Values could be summarized across DT variables (i.e., the daily statistics) to create collector summaries. Daily summaries could be RETAINed counts and sums which would be written as a new record at end of file.
- Page breaks, "continued" marks, etc. in the report. A truly pretty report would not have a collector's information broken across pages, or at worst would have a "continued" indicator at the start of the next page. If there are enough of these and other requirements it may be faster to use a _NULL_ DATA step. An alternative which keeps using REPORT is to determine the number of lines available for printing, then as the report dataset is created, keep track of lines left on a page. Use this as a HIDDEN ORDER variable in REPORT and put it in a BREAK statement. The ROWHEADER value would have to change as well to indicate "Cont."

COMMENTS? QUESTIONS?

Your input is always welcome. Contact the author at:

102 Westbury Drive
Chapel Hill NC 27516-9154
919.942.2028
fcd1@mindspring.com

FIGURE 1 – THE REPORT-WRITING PROGRAM

```
options nocenter pageno=1 nodate;

{1}
* Because everyone is entitled to a change of heart ;
%let start_time = 1800;
%let end_time   = 2000;
%let start_date = 21nov99;
%let end_date   = 27nov99;

filename datain "C:\SESUG00\sesug2k.dat";

{2}
proc format;
value $loc '00' = 'Pensacola'        '01' = 'Panama City'      '08' = 'Apalachicola'
           '09' = 'Cedar Key'        '13' = 'Tarpon Springs'   '14' = 'Clearwater'
           '15' = 'St. Petersburg'   '16' = 'Bradenton'        '17' = 'Sarasota'
           '22' = 'Captiva'          '24' = 'Flamingo'         '25' = 'Key West'
           other = '**unknown**'
           ;
run;
```

```
data temps;
infile datain;
input  @1 loc       $2.
       @4 date      mmddyy10.
      @15 time      $4.       {3}
      @21 depth     4.
      @26 char_temp $char11.
      @38 collector $char13.
      ;
* Adjust temp and use only for obs that we know we want to use ;
{4}
if ("&start_time."    <= time <= "&end_time." ) &
   ("&start_date."d <= date <= "&end_date."d)
   then do;

   if index(char_temp, '/') > 0 then do;
      base = input(scan(char_temp, 1, ' '), 4.);
      numer = input(scan(char_temp, 2, ' /'), 4.);
      denom = input(scan(char_temp, 3, ' /'), 4.);
      temp  = base + (numer/denom);
      end;
      else temp = char_temp;

   output;
   end;
drop base numer denom char_temp;
format date yymmdd10.  temp 6.2;
run;

{5}
* Prepare for finding highest temp per loc-collector-date/time ;
proc sort data=temps;
by loc collector date time descending depth descending temp;
run;

* Take only the highest depth-temp if duplicate loc-collector-
  date/time ;
data temps;
set temps;
by loc collector date time;
if ^(first.time & last.time) then do;
   put "Duplicate for " loc= collector= date= time=;
   end;
if first.time then output;
run;

{6}
* Get summary stats.  Take advantage of sort order (BY processing is
  faster than using CLASS). ;
proc means noprint nway data=temps;
by loc collector date;
var temp;
output out=summ(drop=_type_ _freq_) n=n mean=mean min=min max=max;

run;

{7}
* Vars in SUMMTRAN are LOC, COLLECTOR, DT[yyyy_mm_dd] ;
proc transpose data=summ out=summtran prefix=dt;
var n mean min max;
id date;
by  loc collector ;
run;

{8}
* Create macro vars used for headings in Proc Report, array bounds
  in a late DATA step. ;
proc sql noprint;
* Use for array size and macro loop control.
```

```
  This tells us the number of dates for which we have data. ;
select count(distinct date) into :n_dates
from temps;

* Use for array definition in next DATA step so we can assign/slot
  the vars in the correct order. Vars from COLUMNS are stored in
  alpha order. ;
select distinct name into :transpose_name separated by ' '
from dictionary.columns
where libname = "WORK" & memname = "SUMMTRAN" and name like "dt%";

* Use for column headers in REPORT.
  These are the prettified, readable day of the week. ;
select distinct date format=downame9. into :display_date separated by ' '
from temps
order by date;

* Use for counts per date (denominators) when computing N percents.
  This tells us how many non-missing temps we have each day.;
select n(date) into :date_counts separated by ' '
from temps
where temp is not missing
group by date;
quit;


options symbolgen;

%let n_dates = %left(&n_dates);


* Unit of observation for input dataset SUMM is loc-collector-
  date.  Collapse into one obs per statistic per loc-collector.
  We are massaging the data for minimal handling by REPORT. ;
{9}
data temps2;
set summtran;
by loc collector;
length rowheader $20 last_name first_name $13;

array ns(&n_dates) _temporary_ (&date_counts.);  * Created by SQL ; {9a}
array dt(&n_dates) &transpose_name;              * Created by SQL ;
array values(&n_dates.) $12;  * Values that will be used by REPORT ;

* Create blank lines if we are at the start of a location or
  collector. ;
{9b}
if first.loc then do;
   rowheader = put(loc, $loc.);
   output;
   end;
if first.collector then do;
   * Tweak collector name, swapping last-first names if we can. ;
   if index(collector, ',') > 0 then do;
      last_name  = scan(collector, 1, ' ,');
      first_name = scan(collector, 2, ' ,');
      collector = trim(first_name) || ' ' || last_name;
      end;
   rowheader= '   ' || collector;
   output;
   end;

do i = 1 to dim(dt);
   if _name_ = 'n' then do;
      {9c}
      if dt(i) ^= . then values(i) = put(dt(i), 3.) || " (" ||
                                     put(100*(dt(i)/ns(i)), 5.1) || '%)';
      rowheader = '     N (%)';
      end;
      else if _name_ = 'mean' then do;
```

```
           if dt(i) ^= . then values(i) = put(dt(i), 4.1);
           rowheader = '     Mean';
           end;
      else if _name_ = 'min' then do;
           if dt(i) ^= . then values(i) = put(dt(i),  4.1);
           rowheader = '     Min';
           end;
      else if _name_ = 'max' then do;
           if dt(i) ^= . then values(i) = put(dt(i),  4.1);
           rowheader = '     Max';
           end;
end;
output;

keep loc rowheader values1-values&n_dates.;
run;


%macro writerpt;
```

{10}

```
proc report headline nowindows data=temps2 split="#";
columns loc rowheader values1-values&n_dates.;
define rowheader / "Location#    Collector";
define loc / order noprint;
%do i = 1 %to &n_dates;
    define values&i / center width=12
                      "%scan(&display_date, &i.)#(n=%scan(&date_counts, &i))";
%end;
break after  loc / skip;
Title "Temperatures between time &start_time. and &end_time., &start_date. - &end_date.";
run;
%mend;
%writerpt
```

FIGURE 2 –PROGRAM OUTPUT

```
Temperatures between time 1800 and 2000, 21nov99 - 27nov99                                       1

  Location,                 Sunday        Monday        Tuesday       Wednesday      Thursday       Friday        Saturday
     Collector              (n=85)        (n=92)        (n=92)         (n=79)        (n=85)         (n=74)        (n=102)
  -----------------------------------------------------------------------------------------------------------------------
  Pensacola
     B. Beard
       N (%)              1 (  1.2%)                  1 (  1.1%)    2 (  2.5%)    1 (  1.2%)
       Mean                 24.7                        24.4          22.1          21.1
       Min                  24.7                        24.4          20.5          21.1
       Max                  24.7                        24.4          23.6          21.1
     SEYMOUR FISH
       N (%)              1 (  1.2%)    2 (  2.2%)    1 (  1.1%)    1 (  1.3%)    1 (  1.2%)    1 (  1.4%)
       Mean                 20.5          22.3          23.4          24.4          22.6          23.9
       Min                  20.5          20.3          23.4          24.4          22.6          23.9
       Max                  20.5          24.3          23.4          24.4          22.6          23.9
     Rip Tide
       N (%)                            1 (  1.1%)                  1 (  1.3%)
       Mean                               20.2                        23.1
       Min                                20.2                        23.1
       Max                                20.2                        23.1
     Wahiini
       N (%)                            1 (  1.1%)                  1 (  1.3%)                                2 (  2.0%)
       Mean                               22.7                        22.7                                      21.3
       Min                                22.7                        22.7                                      21.1
       Max                                22.7                        22.7                                      21.5
     Jock Whostow
       N (%)              2 (  2.4%)                                                                          1 (  1.0%)
       Mean                 21.7                                                                                20.6
       Min                  21.5                                                                                20.6
       Max                  22.0                                                                                20.6
     Cap'n morgan
       N (%)              1 (  1.2%)
       Mean                 20.9
       Min                  20.9
       Max                  20.9

  Panama City
     B. Beard
       N (%)              3 (  3.5%)    4 (  4.3%)    3 (  3.3%)    3 (  3.8%)    3 (  3.5%)    3 (  4.1%)    4 (  3.9%)
       Mean                 23.8          21.6          21.9          22.3          21.3          21.5          22.7
       Min                  23.6          21.4          21.1          21.8          20.7          21.2          21.4
       Max                  24.2          21.8          22.8          22.7          22.5          21.7          25.4
     SEYMOUR FISH
       N (%)              2 (  2.4%)    1 (  1.1%)    2 (  2.2%)    3 (  3.8%)    1 (  1.2%)    2 (  2.7%)    3 (  2.9%)
       Mean                 21.9          22.2          23.0          22.1          20.1          21.4          24.1
       Min                  20.4          22.2          21.3          21.3          20.1          20.4          22.7
       Max                  23.4          22.2          24.8          23.0          20.1          22.4          25.3
```

Reporting from the Depths of Florida Sashole

Paul M. Dorfman
Citibank Universal Card, Jacksonville, Fl

Mainline

First, the data are scrubbed, cleaned up, and standardized, as they are being input. The fractional temperatures are separated from ones represented in decimal notation by using the bestw.d informat preceded by '??'. This way, SAS inputs decimal values uneventfully and converts fractional ones to missing without warnings. The filtered fractions are then converted to decimals by parsing the fractional strings.

Secondly, collectors' names are to be reported in reverse with respect to the first and last name. Moreover, the raw names are cased rather sporadically, while we need them title-cased. The specs call for the report sorted by the name without saying which one. Normally it would be the last name. However, one look at the raw data tells that the names are rather pseudonyms, so the collectors are more likely to be known by their nicknames, "Cap'n Morgan" or "B. Beard", say, rather than "Morgan, Cap'n" and "Beard, B". I hence decided to swap the parts of the names in the input, title case the whole thing, use it as a name ID, and order accordingly. Instead of coding title-casing from scratch, I decided to use a macro function I wrote once before:

```
%macro caps (s);
   %local l u lc uc i trn;
   %let l   = %lowcase(abcdefghijklmnopqrstuvwxyz);
   %let u   = %upcase (abcdefghijklmnopqrstuvwxyz);
   %let lc  = %substr(&l,1,1);
   %let uc  = %substr(&u,1,1);
   %let trn = tranwrd(" "||lowcase(&s)," &lc"," &uc");
   %do i=2 %to 26;
       %let lc  = %substr(&l,&i,1);
       %let uc  = %substr(&u,&i,1);
       %let trn = tranwrd(&trn," &lc"," &uc");
   %end;
   left(&trn)
%mend caps;
```

It works by assembling nested calls to function TRANWRD converting any character with a leading blank into its uppercase counterpart. At first glance, it may seem that 26 nested calls might be inefficient, but in reality, SAS handles them with aplomb.

Thirdly, there is a data selection issue. The date, time, and depth ranges can be coded as inequalities. We do not have the date range provided, but only a reference day in the middle of the working week. It is more extendible to parameterize it in the beginning of the program as a macro variable and let SAS automate the computation of the corresponding week endpoints via INTNX. The time and depth ranges could be accounted for by providing their endpoints on the top of the program in the same manner. However, I decided against this method for one reason. From the first glance cast at the task, it is clear that we will need some formats juxtaposing the location codes and their textual descriptions, so why not code the time and depth ranges in the same PROC FORMAT, too? An extra format, WKDAY, can be useful in reporting procedures, so let us throw it in as well. Now, what do we do with the observations containing codes having no valid text counterparts? Usually, they would go to an exception report broken by the original codes, so that they could be identified later on if necessary. However, since there is no demand for such a report in the professor's requirements, I follow the path of least resistance and kick such record out relying on the '*' in the LCODE format.

```
%let infile   = c:\sesug00\sesug2k.dat;
%let weekof   = "25nov1999"d ;

proc format;
  value hmrng 1800-2000 = '1' other = '0';
  value depth 10  -  15 = '1' other = '0';
  value wkday 1='Sun' 2='Mon' 3='Tue' 4='Wed'  5='Thu' 6='Fri' 7='Sat';
  value lcode 00 ='pensacola   ' 13 ='tarpon springs' 17 ='sarasota'
              01 ='panama city ' 14 ='clearwater    ' 22 ='captiva '
              08 ='apalachicola' 15 ='st. petersburg' 24 ='flamingo'
              09 ='cedar key   ' 16 ='bradenton     ' 25 ='key west'
              other ='*';
run;
```

At this point, the raw data can be read, scrubbed, manipulated as indicated, and output to the SAS file CLEAN. Since the name of the input file location can change, it is convenient to have it as an input parameter, so in the step below, only the macro reference &INFILE is used.

```
data clean (keep=loc date hhmm name depth temp);
   infile "&infile" ;
   input @ 01 lcode        02.
         @ 04 date   mmddyy10.
         @ 15 hhmm         04.
         @ 21 depth        04.
         @ 26 tempc   $char11.
         @ 38 name    $char13.
         ;
   if put(depth,depth.) = '1' and put(hhmm,hmrng.) = '1';
   if intnx('week',&weekof,0) <= date < intnx('week',&weekof,1);
   temp = input (tempc, ?? best11.);
   if temp = . then temp = round (input(scan(tempc,1),best.) +
       input(scan(tempc,2),best.)/input(scan(tempc,3),best.),.01);
   name = trim(scan(name,2,','))||' '||left(scan(name,1,','));
   name = %caps (name);
   loc  = put (lcode, lcode.);
   if loc = '*' then delete;
   loc  = %caps (loc);
run;
```

At this stage, the question is, how to comply with the ‘maximum temperature at maximum depth’ requirement? The simplest thing is to do it in two stages. First, sort by LOC NAME DATE HHMM DEPTH TEMP (at the same time, bringing LOC NAME into the correct reporting order):

```
proc sort;
   by loc name date hhmm depth temp;
run;
```

“Lazy Sashole” Approach

Second, grab the last record from each group identified by the same value of DEPTH. However, before doing that, it is time to make a decision about the reporting tool. While I was pondering about it, I heard a loud noise of TABULATE banging on the door and offering a fast and cheap service. I succumbed to the temptation:

```
data maxx (keep=loc name wd temp);
  set clean;
  by loc name date hhmm depth;
  if last.depth;
  wd = weekday(date);
run;

proc tabulate data=maxx;
  class loc name wd tab;
  var   temp;
 table loc,
        name*temp=''*
       (n*f=2. pctn<loc*name>='N(%)'*f=2.1 (mean min max)*f=5.1),
        wd='' /
        row=float rts=28 misstext='' box=_page_ printmiss;
   format wd wkday3.;
   label loc='Location: ' name='Collector';
run;
```

A 10-minute deal produces the output like this:

```
----------------------------------------------------------------------------
|Location: Pensacola      | Sun | Mon | Tue |    Wed     | Thu | Fri | Sat |
|-------------------------+-----+-----+-----+------------+-----+-----+-----|
|            |N           |    2|    4|    2|            |    2|    1|    3|
|------------+------------+-----+-----+-----+------------+-----+-----+-----|
|Collector   |            |     |     |     |            |     |     |     |
|------------+------------|     |     |     |            |     |     |     |
|B. Beard    |N           |     |     |    1|            |    1|     |     |
|            |------------+-----+-----+-----+------------+-----+-----+-----|
|            |N(%)        |     |     | 50.0|            | 50.0|     |     |
|            |------------+-----+-----+-----+------------+-----+-----+-----|
|            |MEAN        |     |     | 24.4|            | 21.1|     |     |
|            |------------+-----+-----+-----+------------+-----+-----+-----|
|            |MIN         |     |     | 24.4|            | 21.1|     |     |
|            |------------+-----+-----+-----+------------+-----+-----+-----|
|            |MAX         |     |     | 24.4|            | 21.1|     |     |
|-----------+------------+-----+-----+-----+-----------+-----+-----+-----|
|Cap'n Morgan|N           |     |     |     |     |           |     |     |     |
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |N(%)        |     |     |     |     |           |     |     |     |
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |MEAN        |     |     |     |     |           |     |     |     |
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |MIN         |     |     |     |     |           |     |     |     |
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |MAX         |     |     |     |     |           |     |     |     |
|-----------+------------+-----+-----+-----+-----------+-----+-----+-----|
|Jock Whostow|N           |    2|     |     |           |     |     |    1|
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |N(%)        |100.0|     |     |           |     |     | 33.3|
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |MEAN        | 21.7|     |     |           |     |     | 20.6|
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |MIN         | 21.5|     |     |           |     |     | 20.6|
|           |------------+-----+-----+-----+-----------+-----+-----+-----|
|           |MAX         | 22.0|     |     |           |     |     | 20.6|
|-----------+------------+-----+-----+-----+-----------+-----+-----+-----|
 . . . . . . . . . . . . . . . .
```

As a bang for the programming hour, it is hard to beat. Note that the specs call not for an exact replication of the suggested layout, but rather for something resembling it, which is a rather good fit for TABULATE usage. And we are done in only 4 steps. As a side point, writing the percentages right beneath the count, just as TABULATE does, promotes better comprehension of the data presented in the report than the requested N(n%) style.

“Industrious Sashole” Approach

However, if the professor insists on the style, though, accommodating it will require custom coding. There are several ways to choose. For instance, the percentages can be computed beforehand, concatenated with the necessary characters and parentheses into a single string variable, and then fed into (altered) TABULATE. This method, though, feels rather kludgy. And then still, the TABULATE output has its own mind. It is not necessarily bad, but it would be interesting to see how much more effort DATA step reporting would require.

The weekday statistics can be obtained as a subset of SUMMARY output, or, since I am going to try DATA step anyway, they can be computed on the fly. I go with the latter because 1) I feel like it 2) it saves an extra step. Before commencing work on the final coding, the question remains, how to incorporate the summary counts at the LOC level in the reporting step? Ten different SASmen would most likely come up with ten different solutions. I compute the daily counts in the same DATA step where the duplicate temperature observations are eliminated. It is very easy to do by key-indexing the array FD(7) by the number of the current weekday, WD. The counts are output to a separate data set ALL, which thus will have exactly as many records as there are distinct valid locations. At the same time, the frequencies can be pre-formatted into the items of the array FF(7) with the parentheses and equal signs needed in the report. Then in the reporting step, it will suffice to read the next record from ALL in the beginning of each LOC by-group. As I am planning on using the ‘mass-formatting’ technique when a line containing homogeneous items is printed at once from an array, having the width of each such item assigned ahead of the time makes code more flexible and less error-prone. This is the purpose of the macro variable F.

```
%let f = 10; * Mass format length;

data maxx (keep=loc name wd temp) all (keep=f:);
  array fd(7);
  array ff(7) $ &f.;  * all*wkday freq formatted;
  do until (last.loc);
     set clean;
     by loc name date hhmm depth temp;
     if not last.depth then continue;
     wd = weekday(date);
     fd(wd) = sum(fd(wd),1);
     output maxx;
  end;
  do wd=1 to 7;
     if fd(wd) ne . then ff(wd) = '(N='||compress(put(fd(wd),best.))||')';
  end;
  output all;
run;
```

What is the condition “fd(wd) ne .” is for? Leaving the FD buckets with no hits missing is my purpose but I just happen to abhor ‘missing’ messages in the log. The intent here is to use the option MISSING=' ' to fulfil one of the reporting requirements, namely printing blanks for the cells for which the data are not available.

Now it is time for the reporting step. Its actions should be evident from the code. The step, in the interpretation below, combines reporting per se with computing the statistics on the fly and accumulating them at the categorical level of NAME. The technique of using expressions like

```
    accumvar = sum(accumvar, variable);
```

may seem unconventional. The reason for using it is that I am interested in keeping the cells with no data available populated with missing values and use MISSING option to print blanks. An expression of the type above performs the accumulation without the need to initialize the daily buckets to zeroes. The explicit DO UNTIL loops make use of the default action at the bottom of the DATA step reinitializing the buckets exactly when needed (that is, after a by-group at the correct level has been processed) and exactly with what is needed (missing), without any RETAINs (the technique learned from Ian Whitlock).

```
option missing='';
%let ls = 100;

data _null_;
    file print ls=&ls ll=ll header=hh;
    array fd(7);       * all     *N;
    array fq(7);       * loc*name*N;
    array av(7);       * loc*name*mean;
    array mi(7);       * loc*name*min;
    array ma(7);       * loc*name*max;
    array pn(7) $ &f.; * pctn<all loc*name> formatted;
    array dy(7) $ &f. ('Sun' 'Mon' 'Tue' 'Wed' 'Thu' 'Fri' 'Sat');
    retain a 01 b 16 c 18 d 24; * tabs;
    do until (last.name);
       set maxx;
       by loc name;
       if first.loc then do;
          set all; put _page_;
       end;
       fq(wd) = sum(fq(wd),   1);
       av(wd) = sum(av(wd),temp);
       mi(wd) = min(mi(wd),temp);
       ma(wd) = max(ma(wd),temp);
    end;
    if ll < 11 then put _page_;
    do wd=1 to 7;
       if nmiss(fq(wd),fd(wd),av(wd)) then continue;
       pn(wd) = put(fq(wd),2.)||'('||put(fq(wd)/fd(wd)*1e2,4.1)||'%)';
       av(wd) = av(wd)/fq(wd);
    end;
    put @b name
     // @b 'N(N%)' @c+&f  (pn(*)) ($&f.. )
      / @b 'Mean'  @d+1   (av(*)) ( &f..1)
      / @b 'Min '  @d+1   (mi(*)) ( &f..1)
      / @b 'Max '  @d+1   (ma(*)) ( &f..1) /;
    return;
    hh: put @c+3+&f (dy(*)) ($&f..) / @a &ls*'-'
          / @d+6 (ff1-ff7) (&f..) @a 'Location' @b 'Collector'
          / @a &ls*'-' / @a loc @ ;
run;
```

(FF1-FF7) are coded as an explicit list instead of FF(*) because the array FF(7) is not declared in the step: Since nothing relies on the subscripted FF values, the array is unnecessary. Here is a sample of the kind of report this step prints:

```
                              Sun       Mon       Tue       Wed       Thu       Fri       Sat
----------------------------------------------------------------------------------------------------
Location       Collector     (N=2)     (N=4)     (N=2)               (N=2)     (N=1)     (N=3)
----------------------------------------------------------------------------------------------------
Pensacola      B. Beard

               N(N%)                            1(50.0%)            1(50.0%)
               Mean                               24.4                21.1
               Min                                24.4                21.1
               Max                                24.4                21.1

               Jock Whostow

               N(N%)        2( 100%)                                                    1(33.3%)
               Mean           21.7                                                        20.6
               Min            21.5                                                        20.6
               Max            22.0                                                        20.6

               Rip Tide

               N(N%)                  1(25.0%)
               Mean                     20.2
               Min                      20.2
```

```
Max                      20.2

Seymour Fish

N(N%)                  2(50.0%)  1(50.0%)            1(50.0%)  1( 100%)
Mean                     22.3      23.4                22.6      23.9
Min                      20.3      23.4                22.6      23.9
Max                      24.3      23.4                22.6      23.9

Wahiini

N(N%)                  1(25.0%)                                          2(66.7%)
Mean                     22.7                                              21.3
Min                      22.7                                              21.1
Max                      22.7                                              21.5
```

Note that in this report, Cap'n Morgan is absent from the Pensacola group: At this location, he has collected nothing. In TABULATE, the empty cells are included because of PRINTMISS, which I had to use because I wanted to print all the daily columns regardless of them being empty or not, and could not have one without the other. Thus, at least in this respect, DATA step is more flexible than TABULATE, to say nothing of the report appearance being somewhat closer to the layout in the specs. However, there is a price for everything! It took mere 10 minutes to code TABULATE, and well over an hour to write the DATA step and format the printout satisfactorily, plus some more time to cross-check the figures by comparing it with the output produced by TABULATE and quick-and-dirty SUMMARies I ran against CLEAN to get an initial feel of the data.

The Penalty for not Reading the Specs Carefully

Just when I thought I was done, it struck me that I might have produced something quite different from what is actually requested. Rereading the specs, I realized that nothing there tells to produce percentages relative to the cumulative counts at the location level! It was rather my interpretation, since without the ability to communicate with the professor directly (even though both of us live in Florida) I assumed that he would be rather interested in knowing how relatively successful was an individual in thermodiving at a particular location. If my assumption were wrong, how difficult would it be to fix the report? One might be given a whole business day to produce a report like that, but when it is on the desk before the boss and something is wrong with it, it has to be redone in an hour!

Luckily, it is not difficult. If we stick with TABULATE, the cumulatives at the LOC level have to go, and the denominator definition has to be slightly altered. Besides, the professor would like to see the daily totals at the beginning of the report. These can be produces by a separate TABLE statement. The altered TABLE statement(s) will now look like

```
table wd=''*n=''*f=8. / rts=28;
table loc ,
      name*temp=''*
     (n*f=2. pctn<loc*name>='N(%)'*f=2.1 (mean min max)*f=5.1),
    wd='' / row=float rts=28 misstext='' box=_page_ printmiss;
```

The rest of the proc remains intact. Now at the beginning of the report, it prints the summary header

```
----------------------------------------------------------------
|  Sun   |  Mon   |  Tue   |  Wed   |  Thu   |  Fri   |  Sat   |
|--------+--------+--------+--------+--------+--------+--------|
|      60|      57|      65|      56|      61|      53|      69|
----------------------------------------------------------------
```

On the subsequent pages, the location totals disappear, and the percentages change accordingly:

```
----------------------------------------------------------------------------
|Location: Pensacola       | Sun | Mon | Tue |    Wed    | Thu | Fri | Sat |
|--------------------------+-----+-----+-----+-----------+-----+-----+-----|
|Collector   |             |     |     |     |           |     |     |     |
|------------+-------------|     |     |     |           |     |     |     |
|B. Beard    |N            |     |     |    1|           |    1|     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |N(%)         |     |     |  1.5|           |  1.6|     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MEAN         |     |     | 24.4|           | 21.1|     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MIN          |     |     | 24.4|           | 21.1|     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MAX          |     |     | 24.4|           | 21.1|     |     |
|------------+-------------+-----+-----+-----+-----------+-----+-----+-----|
|Cap'n Morgan|N            |     |     |     |           |     |     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |N(%)         |     |     |     |           |     |     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MEAN         |     |     |     |           |     |     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MIN          |     |     |     |           |     |     |     |
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MAX          |     |     |     |           |     |     |     |
|------------+-------------+-----+-----+-----+-----------+-----+-----+-----|
|Jock Whostow|N            |    2|     |     |           |     |     |    1|
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |N(%)         |  3.3|     |     |           |     |     |  1.4|
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MEAN         | 21.7|     |     |           |     |     | 20.6|
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MIN          | 21.5|     |     |           |     |     | 20.6|
|            |-------------+-----+-----+-----+-----------+-----+-----+-----|
|            |MAX          | 22.0|     |     |           |     |     | 20.6|
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

So, with the TABULATE, it does not take much – the proc is deservedly known for its chameleon-like abilities. However, it is not difficult with the DATA step, either, even though changes have to be made in two places. First, the unduplicating step producing MAXX and ALL should accumulate at the grand total level instead of location level, and thus output a single observation containing total daily counts (and their formatted siblings) across all collectors and locations. All the changes to this step are confined to its inner loop. LAST.LOC gets replaced with LAST.EOF, and the corresponding END= option has to be added to the SET statement. Here is how the altered loop looks like (with changes in boldface):

```
   do until (EOF);
      set clean END=EOF;
      by loc name date hhmm depth temp;
      if not last.depth then continue;
      wd = weekday(date);
      fd(wd) = sum(fd(wd),1);
      output maxx;
   end;
```

Secondly, in the reporting step, ALL has to be read just once before anything happens, so now the step acquires the form (again, the changes are shown in boldface):

```
data _null_;
    file print ls=&ls ll=ll header=hh;
    array fd(7);       * all     *N;
    array fq(7);       * loc*name*N;
    array av(7);       * loc*name*mean;
    array mi(7);       * loc*name*min;
    array ma(7);       * loc*name*max;
    array pn(7) $ &f.; * pctn<all loc*name>=f*&f;
    array dy(7) $ &f. ('Sun' 'Mon' 'Tue' 'Wed' 'Thu' 'Fri' 'Sat');
    retain a 01 b 16 c 18 d 24; * tabs;
    IF _N_ = 1 THEN SET ALL;
    do until (last.name);
       set maxx;
       by loc name;
       IF FIRST.LOC OR LL < 11 THEN PUT _PAGE_;
       fq(wd) = sum(fq(wd),   1);
       av(wd) = sum(av(wd),temp);
       mi(wd) = min(mi(wd),temp);
       ma(wd) = max(ma(wd),temp);
    end;
    do wd=1 to 7;
       if nmiss(fq(wd),fd(wd),av(wd)) then continue;
       pn(wd) = put(fq(wd),2.)||'('||put(fq(wd)/fd(wd)*1e2,4.1)||'%)';
       av(wd) = av(wd)/fq(wd);
    end;
    put @b name
     // @b 'N(N%)' @c+&f  (pn(*)) ($&f.. )
      / @b 'Mean'  @d+1   (av(*)) ( &f..1)
      / @b 'Min '  @d+1   (mi(*)) ( &f..1)
      / @b 'Max '  @d+1   (ma(*)) ( &f..1) /;
    return;
    hh: put @c+3+&f (dy(*)) ($&f..) / @a &ls*'-'
          / @d+6 (ff1-ff7) (&f..) @a 'Location' @b 'Collector'
          / @a &ls*'-' / @a loc @ ;
run;
```

So, it actually takes surprisingly few alterations to make the transition from percentages at the location level to ones at the level of daily totals across all locations. The amended DATA step results in the picture similar to the one before. Now the totals displayed at the beginning of each location are the ones for all locations, and the percentages are consequently much smaller, but of course the statistics at the location*name crossings remain intact:

```
Sun       Mon       Tue       Wed       Thu       Fri       Sat
----------------------------------------------------------------------------------------------------
Location       Collector      (N=60)     (N=57)     (N=65)     (N=56)     (N=61)     (N=53)     (N=69)
```

```
----------------------------------------------------------------------------------------------------
Pensacola     B. Beard

              N(N%)                           1( 1.5%)            1( 1.6%)
              Mean                              24.4                21.1
              Min                               24.4                21.1
              Max                               24.4                21.1

              Jock Whostow

              N(N%)        2( 3.3%)                                                    1( 1.4%)
              Mean           21.7                                                        20.6
              Min            21.5                                                        20.6
              Max            22.0                                                        20.6

              Rip Tide

              N(N%)                 1( 1.8%)
              Mean                    20.2
              Min                     20.2
              Max                     20.2

              Seymour Fish

              N(N%)                 2( 3.5%)  1( 1.5%)            1( 1.6%)  1( 1.9%)
              Mean                    22.3      23.4                22.6      23.9
              Min                     20.3      23.4                22.6      23.9
              Max                     24.3      23.4                22.6      23.9
```

Conclusion

The most difficult thing in SAS reporting is not programming per se, but deciding what to choose from the roster of programming, analytic, and reporting tools the SAS System offers. Just during the course of this exercise, I was split between SUMMARY, TABULATE, FREQ, and DATA step, TRANSPOSE, and arrays, etc., before finally settling on something concrete. Luckily, for me, it was easy to disregard the REPORT procedure, since I am not sufficiently good at it. I do realize, though, that in this situation, the proc may be just the ticket for an expert REPORT programmer.

Author Contact Information

Paul M. Dorfman
10023 Belle Rive Blvd. 817
Jacksonville, FL 32256
(904) 564-1931 (h)
(904) 954-8533 (o)
sashole@mediaone.net
paul.dorfman@citicorp.com

Coders' Corner Panel Problem

Ian Whitlock, Westat, Rockville, MD

Analysis of the Problem

At first glance the problem looks like a simple two-part problem - read the data, report the statistics. Let's analyze the problem.

What are the problems with the data?

1. Records out of scope (wrong date, wrong time, wrong depth)
2. Multiple readings (at same location and time by the same person)
3. Fractions and decimals used in depth
4. Name - order (first last) and missing parts

What are the problems with the report? It looks like a PROC TABUATE except that

1. Header labels include data values
2. Cell contains two numbers for the line N (%)
3. Order of report

The data problems (3) and (4) are easily handled in the step that reads the file. Problem (2) cannot be solved in the reading step, hence it is reasonable to postpone all the subsetting to a subsequent step. This means one can get a better picture of name problems by looking at the complete file. In fact, there were few problems with names. Beard abbreviated his first name and Whahinii didn't give it. A call to Dr. Dripp-Drye ironed out these problems.

The header problem, can be answered with a few macro variables, but the report problem (2) is the big question. I decided to use TABULATE for a debugging report, but not use it for the main report. So what's left? PROC SUMMARY can get the statistics, but how should they be reported. The CLASSDATA= option would fill in with the missing categories. These can easily be generated from a cartesian product of sets of distinct elements using PROC SQL. The new TYPES statement is used to make the code more efficient since now only types 8 and 15 are generated. The data out of PROC SUMMARY could be massaged to make a DATA _NULL_ report, a PROC REPORT report, or possibly a PROC PRINT report. Having abandoned PROC TABULATE, I could afford to leave the decision until after massaging the data. A double PROC TRANSPOSE should rearrange the statistics for reporting - the first reduces the multiple statistic variables to one column and the second redistributes them over days as required by report structure.

What about the N (%) line? Well it looks like this has to be a character variable, so lets make all the statistics into character variables (no problem for PROC TRANSPOSE). At this point I decided to aim for a PROC PRINT. Now what about order? A nasty twist. The report should be Name (first then last) but I wanted them alphabetized by last name. The statistic names also caused a problem; they wouldn't sort nicely. I had to abandon PROC PRINT because and variables used in ordering a BY statement must appear in the report.

Normally I would go for a DATA _NULL_ report and did start to code it. But, I really wanted to include an HTML version and didn't want to get into a mess between HTML and DATA _NULL_. PROC REPORT looks like my answer. You don't have to report all variables used and you can get some of the flexibility of a true DATA _NULL_ report. The big problem here was how to get the location and collector on different lines. After a few false tries with options like FLOW, it became clear that it would be best to use a COMPUTE block to write the location and another compute block to write the name. The weakness here is getting the output of the LINE command where you want it. I added long blank strings with an unprintable character, "00"x on the end. Now when PROC REPORT centers the line the printed portion is on the left.

Well we should be ready to consider the code.

The Code

```
/* Report.sas - read file, massage, get stats,
                massage, and report
      input:  Dripp-Drye Water temps - SESUG.dat
      output: report (print and html)
      author: IW  24jun2000
*/
%let apppath = work ;  /* work or home */
/* ----------------------------------------- */
%let home =
  c:\my documents\ian\sas\sastalk\sesug00\panel;
%let work =
  h:\my documents\sas\sastalk\sesug00\panel ;

libname sesug "&&&apppath" ;
filename main "&&&apppath\sesug2k.dat" ;

title
  "Dripp-Drye Water Temperatures in Western "
  "Florida Week of 21Nov1999" ;
```

```
/* read in data and fix record level problems */
data sesug.main ( keep = loc datetm depth temp
fname lname name ) ;
   infile main truncover ;
   input     loc    $char2.
         +1  dt     mmddyy10.
         +1  hr     2.
             min    2.
         @21 depth  3.
             ctemp  $char10.
         @38 name   $char13.
   ;

   /* get correct week */
   if "21nov1999"d <= dt <= "27nov1999"d ;
   datetm = dhms ( dt , hr , min , 0 ) ;

   /* fix temperature */
   if index ( ctemp , "." ) then
      temp = input ( ctemp , best10.) ;
   else
   if index ( ctemp , "/" ) = 0 then
      temp = input ( ctemp , best10.) ;
   else
   do ;
      temp = input (scan(ctemp, 1), best10.)
          + (input(scan(ctemp, 2), best10.)
              / (input(scan(ctemp, 3), best10.))
            ) ;
   end ;

   /* fix up names */
   lname = lowcase(scan ( name , 1 )) ;
   substr(lname,1,1)= upcase(substr(lname,1,1));
   fname = lowcase(scan ( name , 2 )) ;
   substr(fname,1,1)= upcase(substr(fname,1,1));
   if lname = "Beard" then fname = "Bob";
   else
   if lname = "Wahiini" then fname = "Frank" ;
   name = trim(fname) || " " || lname ;

run ;

/* sort and subset to max temp
   for datetm loc fname lname
   where time between 6pm and 8pm
   where depth 10 and 15 meters.
*/
proc sort data = sesug.main out = main ;
   by datetm loc lname fname descending temp ;
run ;
```

```
data wmain ;
   set main ;
   where loc not in ( "02", "22", "23" ) ;
   by datetm loc lname fname descending temp ;
   if first.fname ;
   if "18:00"t <= timepart(datetm) <= "20:00"t ;
   if 10 <= depth <= 15 ;
   day = weekday(datepart(datetm)) ;
run ;

title2 "Debugging Report" ;
proc tabulate data = wmain ;
   class loc fname lname day ;
   var temp ;
   table (all loc*fname*lname)
         * temp=" " * ( n mean min max )
       , day / rts = 50 ;
run ;
title2 ;

/* --------------------------------------------
   get stats
       generate all possible class combinations
       and summarize
   --------------------------------------------
*/
proc sql ;
   create table shell as
     select day, loc, lname, name
     from (select distinct day from wmain)
       , (select distinct loc from wmain)
       , (select distinct lname , name
             from wmain)
    ;
quit ;

proc summary data = wmain classdata = shell ;
   class day loc lname name ;
   types day day*loc*lname*name ;
   var temp ;
   output out=summary
          n=N mean=Mean min=Min max=Max
   ;
run ;

/* --------------------------------------------
   massage summary from report
   --------------------------------------------
*/
proc format ;
   value $loc
      "00" = "00-Pensacola"
      "01" = "01-Panama City"
      "08" = "08-Apalachicola"
      "09" = "09-Cedar Key"
      "13" = "13-Tarpon Springs"
      "14" = "14-Clearwater"
      "15" = "15-St. Petersberg"
      "16" = "16-Badenton"
      "17" = "17-Sarasota"
      "24" = "24-Captiva"
      "25" = "25-Key West"
   ;
   value $stat
      "apctn" = "N(%)"
      "bmean" = "Mean"
      "cmin"  = "Min"
      "dmax"  = "Max"
   ;
run ;

/* combine summary information to create %
   and massage data */
data chrsum ( keep = day loc lname name
                     apctn bmean cmin dmax ) ;
   length name $17 ;
   merge
      summary ( where = ( _type_ = 8 )
                keep = day n _type_
                rename = ( n = daytot ) )
      summary ( where = ( _type_ = 15 ) )
   ;
   by day ;
   /* create macro variables for report labels*/
   if first.day then
      call symput ( "d" || put(day,1.) ,
                    trim(left(put(daytot,3.))));

   /* indent name for paper report */
   name = "    " || name ;

   /* convert stats to character to combine
      n and % names chosen to sort correctly
  */
   if n > 0 then
   do ;
      pctn=100 * n / daytot ;
      apctn =
           put(n,2.)||"("||put(pctn,3.1)||"%)" ;
      bmean = put(mean,4.1) ;
      cmin = put(min,4.1) ;
      dmax = put(max,4.1) ;
   end ;
run ;

/* reduce all stats to one column */
proc transpose data = chrsum out = t1  ;
   by day loc lname name ;
   var apctn bmean cmin dmax ;
run ;

/* remove day from sort
   to combine stats under day columns
*/
proc sort data = t1 ;
   by loc lname name _name_ ;
run ;

proc transpose data = t1 out = report prefix =
day;
   by loc lname name _name_ ;
   var col1 ;
run ;

/* --------------------------------------------
   write report to output window and html
   --------------------------------------------
*/
ods html body="&&&apppath\report.htm" ;

proc report data = report nowd ;
   column   loc lname name _name_ day1-day7 ;
   define loc    / group noprint ;
   define lname  / group noprint width = 1 ;
   define name   / group  ;
   define _name_ / "Location/  Name"
                   format=$stat.;
   define day1   / display "Sun/(n=&d1)" ;
   define day2   / display "Mon/(n=&d2)" ;
   define day3   / display "Tue/(n=&d3)" ;
   define day4   / display "Wed/(n=&d4)" ;
   define day5   / display "Thr/(n=&d5)" ;
   define day6   / display "Fri/(n=&d6)" ;
   define day7   / display "Sat/(n=&d7)" ;

   break after lname / skip ;

   compute before loc ;
      x =
       put(loc, $loc20.)||repeat(" ",69)||"00"x;
      line " " ;
      line  x $char91. ;
   endcomp ;

   compute before name ;
     x =
    put(name, $char17)||repeat(" ", 69)||"00"x ;
     line  x $char91. ;
   endcomp ;
run ;

ods html close ;
```

Conclusion

One should spend as much time analyzing the problem before coding as possible. One should plan the code for flexibility, postponing decisions that can be delayed.

Approximately half the code was need to fix the problems and produce the basic information. The remaining half was used to get a report in the precise form required. Perhaps more time should have been spent investigating a TABULATE solution.

The author may be contacted by mail at

Ian Whitlock
Westat
1650 Research Boulevard
Rockville, MD 20850

or by e-mail

whitloi1@westat.com

Corrections

The key format $LOC. was poorly executed. The location 22 was omitted and its name applied to 24. This had two disastrous consequences. Data for location 22 were deleted and the report for location 24 was mislabeled.

The second biggest mistake due to a hasty reading of the problem specification was to take the maximum temperature for multiple readings instead of the maximum temperature at the maximum depth.

One could question the basic INPUT statement on the basis of the specs, however the data given support the INPUT statement. Of course, future data might not be readable and still be with in specification.

No attempt was made to make the program parameter driven. On the other hand, any of the most probable changes covered by parameters are easily made and parameterization would not be difficult.

Report

```
     Dripp-Drye Water Temperatures in West Florida Week of 21Nov1999                 1
                                                           20:03 Friday, July 21, 2000

          Location  Sun       Mon       Tue       Wed       Thr       Fri       Sat
           Name     (n=51)    (n=43)    (n=53)    (n=49)    (n=48)    (n=40)    (n=53)

   00-Pensacola
      Bob Beard
          N(%)                           1(1.9%)             1(2.1%)
          Mean                          24.4                21.1
          Min                           24.4                21.1
          Max                           24.4                21.1

      Seymour Fish
          N(%)                 2(4.7%)   1(1.9%)             1(2.1%)   1(2.5%)
          Mean                22.3      23.4                22.6      23.9
          Min                 20.3      23.4                22.6      23.9
          Max                 24.3      23.4                22.6      23.9

      Cap'n Morgan
          N(%)
          Mean
          Min
          Max

      Rip Tide
          N(%)                 1(2.3%)
          Mean                20.2
          Min                 20.2
          Max                 20.2

      Frank Wahiini
          N(%)                 1(2.3%)                                           2(3.8%)
          Mean                22.7                                              21.3
          Min                 22.7                                              21.1
          Max                 22.7                                              21.5

      Jock Whostow
          N(%)       2(3.9%)                                                     1(1.9%)
          Mean      21.7                                                        20.6
          Min       21.5                                                        20.6
          Max       22.0                                                        20.6
```