Dynamic Data Processing Using Data-Driven Formats and Informats
Jedediah J. Teres, MDRC, New York NY

ABSTRACT
PROC FORMAT is a powerful tool for reading and writing data. The PUTN, PUTC, INPUTN, and INPUTC functions use formats and informats at run time. They allow you to assign different formats and informats to values across different observations. Dynamic formats can be used instead of IF-THEN/ELSE blocks to process data. This paper describes how to use them in the context of processing school records.

INTRODUCTION
The PUT and INPUT functions can be used in conjunction with user-defined formats to replace IF-THEN/ELSE logic in SAS programs. When analyzing school records data, grade point average (GPA) is often of interest, and this technique can greatly simplify the calculation of GPA.

SAMPLE DATA
Community college transcript data will be used to illustrate the techniques described. The data contain seven variables. Each observation represents a single course taken by a student. Students are identified using a 7-digit numeric code (STUDENTID). SEMESTER indicates the semester in which each course was taken. TERM_INDEX is numeric version of SEMESTER used for sorting. CLASS shows the full course number and section ID. CREDITS shows the number credits each course is worth. GRADE shows the grade the student received in the course. For simplicity, the data are limited to the fall 2007 and spring 2008 terms, and grades are restricted to values of A, B, C, D or F.

<table>
<thead>
<tr>
<th>Obs</th>
<th>StudentID</th>
<th>Semester</th>
<th>Term_Index</th>
<th>Class</th>
<th>Credits</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0017059</td>
<td>Fall 2007</td>
<td>1</td>
<td>POLI 603 H0OL</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>BADM 610 D0DC</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>DSEM 182 A8UT</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>ECOL 210 C4KW</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>ENGL 204 P8RA</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>6</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>PSYC 101 D6TL</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>7</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>ACCT 102 B8DY</td>
<td>4</td>
<td>D</td>
</tr>
<tr>
<td>8</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>BADM 102 B9BK</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>9</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>BADM 502 C0SG</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>10</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>ECOL 102 F0DF</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>11</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>ENGL 102 Q7WV</td>
<td>4</td>
<td>D</td>
</tr>
<tr>
<td>12</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>HIST 101 F0DF</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>HPED 102 C8RC</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>14</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>SPEE 201 A4MO</td>
<td>3</td>
<td>B</td>
</tr>
</tbody>
</table>

FORMATs and INFORMATs
In SAS, informats provide instructions on how data should be read, while formats control how data are displayed. The FORMAT and INFORMAT statements are compile-time statements. They provide information to the PDV and cannot be executed conditionally.

The INPUT function can be used in conjunction with informats to create new variables. In effect, SAS re-reads the data using the specified informat and creates a new variable with the new value. The INPUT function requires a character value.
GPA is a weighted average. It is calculated by multiplying the number of credits each course is worth by the numeric equivalent of each letter grade; the resulting quality points are then summed for each student in each semester and divided by the total number of credits each student attempted.

The standard grading system in the US is that a grade of A is worth 4 points, a grade of B is worth 3 points, a grade of C is worth 2 points, a grade of D is worth 1 point, and a grade of F is worth 0 points. GPA can be calculated in different ways. The first step towards calculating GPA is calculating the quality points for each course. Using IF-THEN/ELSE logic in a DATA step, the code looks like this:

```plaintext
data transcript2 ;
  set transcript ;
  if grade eq 'A' then qpts = 4.0 * credits ;
  else if grade eq 'B' then qpts = 3.0 * credits ;
  else if grade eq 'C' then qpts = 2.0 * credits ;
  else if grade eq 'D' then qpts = 1.0 * credits ;
  else if grade eq 'F' then qpts = 0.0 * credits ;
run ;
```

The resulting data are shown below:

<table>
<thead>
<tr>
<th>Obs</th>
<th>StudentID</th>
<th>Semester</th>
<th>Term_Index</th>
<th>Class</th>
<th>Credits</th>
<th>Grade</th>
<th>qpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0017059</td>
<td>Fall 2007</td>
<td>1</td>
<td>POLI 603 H0OL</td>
<td>3</td>
<td>B</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>BADM 610 D0DC</td>
<td>3</td>
<td>A</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>DSEM 182 A8UT</td>
<td>1</td>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>ECOL 210 C4KW</td>
<td>3</td>
<td>A</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>ENGL 204 P8RA</td>
<td>3</td>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>PSYC 101 D6TL</td>
<td>3</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>ACCT 102 B8DY</td>
<td>4</td>
<td>D</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>BADM 102 B9BK</td>
<td>3</td>
<td>B</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>BADM 502 C0SG</td>
<td>3</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>ECOL 102 F0DF</td>
<td>3</td>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>ENGL 102 Q77WV</td>
<td>4</td>
<td>D</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>HIST 101 F0DF</td>
<td>3</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>HPED 102 C8RC</td>
<td>3</td>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>SPEE 201 A4MO</td>
<td>3</td>
<td>B</td>
<td>9</td>
</tr>
</tbody>
</table>

Calculating quality points using IF-THEN/ELSE logic is somewhat cumbersome. If this college starts offering +/- grades (i.e., A-, B+, C-, etc), then the programmer will have to revise the code accordingly. The input function can be used with a user-defined informat to convert the letter grades to numbers.

```plaintext
proc format ;
  invalvalue ingrade 'A' = 4.0
  'B' = 3.0
  'C' = 2.0
  'D' = 1.0
  'F' = 0 ;
run ;
```

The INGRADE informat is applied to the GRADE variable using the INPUT function, with the result assigned to the variable ALT_QPTS:

```plaintext
alt_qpts = input(grade, ingrade.) ;
```
This creates an equivalent measure using one line of code instead of five. More importantly, the code used to create ALT_QPTS does not need to be modified if the data changes. The informat will need to be updated, but that does not require revising the data step in which ALT_QPTS is calculated.

<table>
<thead>
<tr>
<th>Obs</th>
<th>StudentID</th>
<th>Semester</th>
<th>Term_Index</th>
<th>Class</th>
<th>Credits</th>
<th>Grade</th>
<th>qpts</th>
<th>alt_qpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0017059</td>
<td>Fall 2007</td>
<td>1</td>
<td>POLI 603 H0OL</td>
<td>3</td>
<td>B</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>BADM 610 D0DC</td>
<td>3</td>
<td>A</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>DSEM 182 A8UT</td>
<td>1</td>
<td>A</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>ECOL 210 C4KW</td>
<td>3</td>
<td>A</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>ENGL 204 P8RA</td>
<td>3</td>
<td>D</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>1</td>
<td>PSYC 101 D6TL</td>
<td>3</td>
<td>C</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>ACCT 102 B8DY</td>
<td>4</td>
<td>D</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>BADM 102 B9BK</td>
<td>3</td>
<td>B</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>BADM 502 C0SG</td>
<td>3</td>
<td>C</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>2</td>
<td>ECOL 102 F0DF</td>
<td>3</td>
<td>D</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>ENGL 102 Q7WV</td>
<td>4</td>
<td>D</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>HIST 101 F0DF</td>
<td>3</td>
<td>C</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>HPED 102 C8RC</td>
<td>3</td>
<td>C</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>2</td>
<td>SPEE 201 A4MO</td>
<td>3</td>
<td>B</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Converting letter grades into quality points is only one part of calculating GPA for each student. The entire DATA step used to create a semester-level GPA summary data set from the course-level data set follows. The data set TRANSCRIPT is being set by STUDENTID and SEMESTER, and has been sorted accordingly.

```
data transcript2 (keep = studentid semester tot_credits gpa) ;
do until (last.studentid) ;
  set transcript ;
  by studentid semester ;
  retain tot_credits tot_qpts ;
  if first.term_index then
    do ;
      tot_credits = 0 ;
      tot_qpts = 0 ;
    end ;
    if grade eq 'A' then qpts = 4.0 * credits ;
    else if grade eq 'B' then qpts = 3.0 * credits ;
    else if grade eq 'C' then qpts = 2.0 * credits ;
    else if grade eq 'D' then qpts = 1.0 * credits ;
    else if grade eq 'F' then qpts = 0.0 * credits ;
tot_qpts = tot_qpts + qpts ;
tot_credits = tot_credits + credits ;
  if last.semester then
    do ;
      gpa = round(tot_qpts/gpa_credits, .001) ;
      if gpa eq . then gpa = 0
      output ;
    end ;
  end ;
run ;
```
Data set TRANSCRIPT2 is shown below.

<table>
<thead>
<tr>
<th>Obs</th>
<th>StudentID</th>
<th>Semester</th>
<th>tot_credits</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0017059</td>
<td>Fall 2007</td>
<td>3</td>
<td>3.000</td>
</tr>
<tr>
<td>2</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>13</td>
<td>2.846</td>
</tr>
<tr>
<td>3</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>13</td>
<td>1.692</td>
</tr>
<tr>
<td>4</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>13</td>
<td>1.923</td>
</tr>
</tbody>
</table>

Using the INPUT function with a user-defined informat allows for simpler code. By replacing the IF-THEN/ELSE block with an INPUT function, PROC SQL can calculate GPA using fewer lines of code than a DATA STEP. Specifying STUDENTID and SEMESTER in the GROUP BY clause of the SELECT statement of the PROC SQL query calculates summed variables within those groups. In this example, TOT_CREDITS is calculated as the sum of CREDITS within SEMESTER for each value of STUDENTID. TERM_INDEX is used in the ORDER BY clause instead of SEMESTER because sorting by semester would order terms Fall 2007, Fall 2008, Spring 2008 instead of Fall 2007, Spring 2008, Fall 2008.

```sql
proc sql;
    create table transcript2s as
    select studentid,
           semester,
           sum(credits) as tot_credits,
           sum(input(grade, ingrade.) * credits)/calculated tot_credits as gpa
    from transcript
    group by studentid,
            semester
    order by studentid,
            term_index;
quit;
```

The data set TRANSCRIPT2S is equivalent to TRANSCRIPT2:

<table>
<thead>
<tr>
<th>Obs</th>
<th>StudentID</th>
<th>Semester</th>
<th>tot_credits</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0017059</td>
<td>Fall 2007</td>
<td>3</td>
<td>3.000</td>
</tr>
<tr>
<td>2</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>13</td>
<td>2.846</td>
</tr>
<tr>
<td>3</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>13</td>
<td>1.692</td>
</tr>
<tr>
<td>4</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>13</td>
<td>1.923</td>
</tr>
</tbody>
</table>

When analyzing GPA distributions, fractional values create too many categories for a frequency table:
Using a format to label ranges (i.e., create categories) makes the results easier to interpret.

```r
proc format;
  value gpaf
    0   <- 0.7     = 'F'
    0.7 <- 1.7     = 'D'
    1.7 <- 2.7     = 'C'
    2.7 <- 3.7     = 'B'
    3.7 - 4.0      = 'A';
run;
```

The format can be applied in the PROC FREQ, as shown below:

```r
proc freq data = transcript2g order = formatted ;
  tables gpa ;
  format gpa gpaf .;
run;
```

The resulting table is much easier to read.

<table>
<thead>
<tr>
<th>gpa</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>87</td>
<td>12.87</td>
<td>87</td>
<td>12.87</td>
</tr>
<tr>
<td>B</td>
<td>255</td>
<td>37.72</td>
<td>342</td>
<td>50.59</td>
</tr>
<tr>
<td>C</td>
<td>209</td>
<td>30.92</td>
<td>551</td>
<td>81.51</td>
</tr>
<tr>
<td>D</td>
<td>87</td>
<td>12.87</td>
<td>638</td>
<td>94.38</td>
</tr>
<tr>
<td>F</td>
<td>38</td>
<td>5.62</td>
<td>676</td>
<td>100.00</td>
</tr>
</tbody>
</table>
The PUT function can be used to create a new categorical character version of GPA as well:

```sql
proc sql;
create table transcript3s as
select studentid,
semester,
sum(credits) as tot_credits,
roundsum(input(grade, ingrade.) * credits)/calculated tot_credits as gpa,
put(calculated gpa, gpa.) as cgpa
from transcript
group by studentid,
semester
order by studentid,
semester ;
quit ;
```

<table>
<thead>
<tr>
<th>Obs</th>
<th>StudentID</th>
<th>Semester</th>
<th>tot_credits</th>
<th>gpa</th>
<th>cgpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0017059</td>
<td>Fall 2007</td>
<td>3</td>
<td>3.000</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>0021112</td>
<td>Fall 2007</td>
<td>13</td>
<td>2.846</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>0021112</td>
<td>Spring 2008</td>
<td>13</td>
<td>1.692</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>0028600</td>
<td>Spring 2008</td>
<td>13</td>
<td>1.923</td>
<td>C</td>
</tr>
</tbody>
</table>

**USING DATA-DRIVEN INFORMATS**

A recent large-scale random assignment study conducted by MDRC evaluated the effects of learning communities on student achievement at community colleges. Learning communities are an approach to student instruction in which several courses are linked together. Instructors develop an integrated curriculum and students take the linked courses together as a cohort. The example used here focuses on one community college in particular, which linked three courses together in learning communities. Many learning communities were offered in each semester of the study, which ran from fall 2007 through fall 2009. For simplicity, only two learning communities are shown in only two semesters, fall 2007 and spring 2008.

One learning community focused on business-related courses, while another focused on health professional training courses. For example, in a “link” focused on business administration, students were expected to take specific sections of a business course, an accounting course, and a discussion seminar.

Participation rates were of interest, where full participation was defined as students enrolling in all three linked courses. Identifying students who satisfied participation criteria using course-level data proved to be challenging. This task was made easier using informats, however.

In this approach, a three-digit code was assigned to each link. For example, the code 111 is assigned to the business administration link. Only specific sections of each course were linked, and those sections were assigned a corresponding number for which the sum would total the three-digit code.

```sql
proc format;
invalue f07class 'BADM 301 L11BS' = 100 'PSYC 101 L12BS' = 10 'DSEM 182 LSM11' = 1
                                           'BIOL 101 L11H' = 200 'PSYC 302 L12H' = 20 'DSEM 181 LSM12' = 2 ;
run ;
```

The next step was to apply the informat to the class variable and sum the result.
PROC SQL;
    CREATE TABLE F07_LC_LINKS AS
    SELECT SAMPLEID, SEMESTER,
    SUM(INPUT(CLASS, F07CLASS.)) AS F07_LC_LINK
    FROM COURSE
    GROUP BY SAMPLEID, SEMESTER
    ORDER BY SAMPLEID;
QUIT;

F07_LC_LINK FREQUENCY PERCENT CUMULATIVE FREQUENCY CUMULATIVE PERCENT
. 630 95.12 643 95.12
100 1 0.15 644 95.27
110 2 0.30 646 95.56
111 7 1.04 653 96.60
222 23 3.40 676 100.00

Students who took all three courses in the business learning community will have a value of 111. Students taking all the courses in the health learning community will have a value of 222. Other non-missing sums indicate cases in which students took a subset of the three required courses. Using a format on the summed values makes the frequency table clearer.

PROC FORMAT;
    VALUE F07LINKS 111 = '1: Business'
                    222 = '2: Health';
RUN;

PROC FREQ DATA = F07_LC_LINKS;
    TABLES F07_LC_LINK / MISSING;
    FORMAT F07_LC_LINK F07LINKS.;
RUN;

F07_LC_LINK FREQUENCY PERCENT CUMULATIVE FREQUENCY CUMULATIVE PERCENT
. 643 95.12 643 95.12
100 1 0.15 644 95.27
110 2 0.30 646 95.56
111: Business 7 1.04 653 96.60
2: Health 23 3.40 676 100.00

In this example, there are two semesters of data, each with two learning communities. The linked sections of courses are dependent on the semester. That is, section numbers can repeat from one semester to the next, so course and section are not sufficient to identify linked courses. As a result, each semester has its own informat for each learning community.

Recall that the PUT and INPUT functions specify formats and informats at compile time and cannot be run conditionally. PUTN, PUTC, INPUTN, and INPUTC are similar functions to PUT and INPUT that specify formats and informats at run time. Because they are specified at run time, they can operate conditionally on the value of another variable.

In order to take advantage of the run time specifications, intermediary formats must be specified. These are formats that return values of formats for different values of semester, as shown below.
proc format;
   value $ termsc 'Fall 2007' = 'f07class.'
          'Spring 2008' = 's08class.';

   value $ lclinx 'Fall 2007' = 'f07links.'
          'Spring 2008' = 's08links.';
run ;

In this case, when the $termsc. format is applied to the variable semester, different informat names are returned based on the value of semester. When semester is 'Fall 2007,' then the $termsc. format will point to the informat f07class. The syntax for this example is:

inputn(class, put(semester, $termsc.))

The INPUTN function is used because the f07class informat is numeric. In this example, for an observation with semester equal to 'Fall 2007,' then

put(semester, $termsc.)

resolves to

f07class.

so the function call can be thought of as being equivalent to

input(class, f07class.) ;

The following code creates links for both semesters and both types of learning communities. A PUTN function is used to assign the semester-specific Learning Communities link formats for each semester.

proc sql;
   create table all_links as
   select   studentid, semester,
            putn(sum(inputn(class, put(semester,$termsc.))),
                put(semester,$lclinx.)) as lc_link
   from      transcript
   group by  studentid, semester
   order by  studentid;
quit ;

<table>
<thead>
<tr>
<th>Semester</th>
<th>lc_link</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 2007</td>
<td>.</td>
<td>266</td>
<td>39.35</td>
<td>266</td>
<td>39.35</td>
</tr>
<tr>
<td>Fall 2007</td>
<td>100</td>
<td>1</td>
<td>0.15</td>
<td>267</td>
<td>39.50</td>
</tr>
<tr>
<td>Fall 2007</td>
<td>110</td>
<td>2</td>
<td>0.30</td>
<td>269</td>
<td>39.79</td>
</tr>
<tr>
<td>Fall 2007</td>
<td>1: Business</td>
<td>7</td>
<td>1.04</td>
<td>276</td>
<td>40.83</td>
</tr>
<tr>
<td>Fall 2007</td>
<td>2: Health</td>
<td>23</td>
<td>3.40</td>
<td>299</td>
<td>44.23</td>
</tr>
<tr>
<td>Spring 2008</td>
<td>.</td>
<td>351</td>
<td>51.92</td>
<td>650</td>
<td>96.15</td>
</tr>
<tr>
<td>Spring 2008</td>
<td>11</td>
<td>2</td>
<td>0.30</td>
<td>652</td>
<td>96.45</td>
</tr>
<tr>
<td>Spring 2008</td>
<td>1: Business</td>
<td>11</td>
<td>1.63</td>
<td>663</td>
<td>98.08</td>
</tr>
<tr>
<td>Spring 2008</td>
<td>2: Health</td>
<td>13</td>
<td>1.92</td>
<td>676</td>
<td>100.00</td>
</tr>
</tbody>
</table>
It should be noted that using formats and informats specified at run time can be slower than using formats and informats specified at compile time.

CONCLUSION
The INPUTN, INPUTC, PUTN, and PUTC functions allow informats and formats to be specified dynamically at run time. This can be in turn be leveraged as a powerful data processing tool. This approach requires a fair amount of planning and set up of informats and formats.
The advantages of this approach are that it reduces the amount of code necessary for complex data processing. This is accomplished largely by moving a lot of the IF-THEN/ELSE logic out of SAS coding and into informats and formats, which can be maintained by less experienced users. Most updates to the coding can be implemented through revising formats and informats rather than modifying the SAS code itself.

REFERENCES

ACKNOWLEDGMENTS
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

CONTACT INFORMATION
Jedediah Teres
MDRC
16 East 34th St
New York, NY 10016
(212) 340-8807
jedediah.teres@mdrc.org