Subject Profiles: (Almost) as easy as 1, 2, 3.
Greg Silva, PPD Development, Wilmington, NC

ABSTRACT
Subject profiles, CRF tabs, patient profiles: whatever you call them, they can be a daunting programming task. This paper will attempt to show that by utilizing a few simple techniques, the programmer can construct complex profiles. By defining the type of data to be presented and using a few constructs, the developer can produce profiles that are not data dependent, and can be used across protocols.

INTRODUCTION
Generating files that contain some or all of a given patient’s data is a regular task in the pharmaceutical industry. Whether it is an internal request by a physician, or a requirement from the FDA, these types of listing have to be generated.

These listings, which require most of a patient’s information to be presented, are more difficult to generate than standard listings. In this paper I will attempt to show that by splitting up the data into types, you can generate patient profiles in a nearly routine manner.

TYPES OF DATA IN PROFILES
The types of data presented in a profile, or any listing for that matter, fall into three categories: single data points, data limited to a few points, and multiple data points. By identifying and placing the data types appropriately, you can remove much of the difficulty associated with subject profiles.

The first type of data, single points, are the easiest to identify. They include such things as enrollment criteria, age, sex, and race. Most data points associated with demographic information are single, nonrecurring data.

Limited types of data can include any test that is performed a few times during the study. This may include height and weight, or even lab tests that may only be run at the start and the end of a study. The definition of limited usually has to be defined for each study.

The last category of data is what I call “unknown number of data elements”. This is typically the lab data. Although you may know from the protocol that lab tests are only to be done three times, there are usually lab retests, or extra tests run by the investigator that need to be reported.

Other examples include adverse events and concomitant medications. Both of these can occur any number of times for any patient. Building your subject profiles with assumptions on the maximum number of adverse events can be a disaster if you underestimate. Overestimating can leave a lot of blank space in your subject profiles.

With an understanding of the three types of data, you can start following the three-step process of developing subject profiles. It is not guaranteed or foolproof, but it has been a reasonable method for defining and programming subject profiles.

PART ONE: IDENTIFYING THE TYPES
The first step is to identify which variables you need for your profiles, and to define the type of each variable. Things such as demographics, labs, and adverse events are normally required. Other items, such as inclusion criteria, or medical history may not make it to the profiles.

Each variable then needs to be identified as one of the three types of data. For each identified variable, you will also need to know the label that will be associated with it, as well as the length and format. All of this information will be needed in the other two steps.

Also, you need to define which variables will be identifiers. If you have multiple pages per patient, there needs to be some unique way of identifying that patient across pages. This information should be the minimum necessary, since you do not want to take up too much space with repeated information.

Once you have identified and classified the data required for the profile, you can start laying out the sections of the report. This will give you a better feel for how much data you can place on a page.

PART TWO: PLACING SINGLE OR LIMITED DATA
Data that occur only once or a limited number of times are fairly easy to place on the subject profiles. This information should be collected into a single dataset. By merging all of the information that will be reported on once per patient, you generate the basis for the subject profiles.

Example 1 shows the top part of a subject profile. The very top section is the header information. This will be used to show patient specific information at the top of each page. This is very useful when there are multiple pages per patient.

The next section contains the demographic information. This is usually only collected once per patient. By merging all of the sources of this type of data into a single record for each patient, you can easily output it on a report.

The vital signs section shows the limited data listing. There are only four visits for this study, so there should only be four (or less for early withdrawal) lines for this section. By using arrays you can merge this type of information to get a single record per patient.

With all of the information that is limited to a known number of observations in a single record per patient, you can simply set the dataset in a DATA _NULL_ to output it to a report. The following code fragment is an example of the single item output. Note that the link to the header is done at the start of each record, since each record contains any individual patient’s information.

```
data _null_; set demog; file 'subject.txt'; link HEADER; put @5 'Age:' @20 age @23 'Years' @30 'Met inclusion criteria:' @52 inccrit ;
```

When you are presenting data that exists a limited number of times, it is usually best to use arrays, or at least subscript your variables. In the next code fragment you can see an example of how to generate

```
multiple lines from a single record in the dataset.

| put @5 visit1 |
| @10 visit1 |
| @25 templ |
| @35 hrl |
| @45 bpl |
| @60 wt1 |
| @70 ht1 / |
| @5 visit1 |
| @10 visit1 |
| @25 templ |
| @35 hrl |
| @45 bpl |
| @60 wt1 |
| @70 ht1 / |

Note that additional processing will have to be added to the program if there are less than the specified number of visits. This can be accomplished in the merging process by collapsing the values into the arrays as they are filled. Another option is to specifically code for those possibilities.

Depending on the amount of information in your demographics section, you may have to consider a page break. Since you are working with a known number of lines and variables, this is simple to accomplish.

By keeping track of the number of lines that you will output for this section, you can control when the page break occurs. It can be hardcoded, since the number of data points will not increase. However, even if you do not have to do it for this section, you will definitely run into it when you start working with an unknown number of data elements.

PART THREE: PLACING UNKNOWN NUMBER OF DATA

When you deal with an unknown number of data elements, the biggest problem in generating the subject profiles is working with the page breaks. Besides wanting to identify if a certain section of data is continuing on the next page, you usually have to worry about some sort of page footer.

By taking advantage of the LINESLEFT argument in the FILE statement, you can find out how close you are to the bottom of the page. If you check this value after each line is printed out, you can control the page breaks and provide consistent footer spacing.

The following code fragments show the parts of the program that generate output data with an unknown number of elements. The first section is straightforward: it generates the header information for the medication section.

```
put @4 'Medications Taken' / ;
put @4 'Medication Name' @20 'Dose' @26 'Unit' @35 'Start Date' @60 'Stop Date' ;
```

The next section uses the POINT argument of the SET statement to read in another dataset. The variables read in with the second SET statement need to be renamed prior to the SET to avoid problems with having two variables with the same name. The RETAIN statement is used to hold the pointer variable.

Note that the cdone flag is retained outside of the SET. This is necessary since you are using the cdone flag in the DO loop, which occurs before the SET statement.

```
The cdone flag, as well as the pointer variable are defined before the SET to initialize the flag and the pointer. The POINT= argument for the SET needs to have the variable set before it is used.

retain cdone 0 ;
cpoint = 1 ;
cdone = 0 ;
do until (cdone) ;
 set crx point = cpoint;
 retain cpoint 1 ;
```

Once the second dataset is opened, you need to get to the data for the patient in the first dataset. By comparing the key variables from the first and second dataset, you can get to the observations with the correct patient information.

At this point you need to start checking for how many lines are left on the page. If you do not have enough for a line of data and the footer information, then you need to put some message on the page and link to the footer section.

After the footer information has been output, a page break and the header should then be put out. At this point the header information for the section of the subject profile should be output to clarify the information provided. The following is a code fragment containing all of this logic.

```
if (inv2=cinv2 and subj=csubj) then do ;
if (lineslft = (&footsz + 1)) then do;
 put @10 '(Continued)';
 link foot ;
 put @4 'Medication Name'
 @20 'Dose'
 @26 'Unit'
 @35 'Start Date'
 @60 'Stop Date' ;
end ;
```

Whether all of that work to generate the footer and header needs to be done, or there is enough space on the current page, the patient’s data needs to be output at this point. Remember that the check for the number of lines left on the page should be done before anything is output.

```
put @4 medname
 @20 meddose
 @26 medunit
 @35 medstart
 @60 medstop ;
end ; * inv2=cinv2... ;
```

Since you have to control the processing of the second SET, you will need to increment the pointer. This will continue for all of the records until you get to the end of file. However, by using the POINT argument in the SET statement, you will have to tell the SET when to stop.

There are a number of methods you can use to simulate the END= argument in the SET statement. Perhaps the simplest method is to get the number of observations in the dataset, store it in a macro, and check the pointer against it. The following code fragment illustrates this method.

```
cpoint=cpoint + 1 ;
```
if (cpoint > &MEDnobs) then cdone = 1;
end;
end;

However, if you are processing a large dataset, you should be looking for a more efficient way to accomplish this. One way would be to keep track of the last record that met the condition you were checking, and store that observation number in the pointer variable. That way you would start the next patient's print out on the record with the next patient's data.

Using that method does require a little more checking. Since there could be cases where there is no concomitant medication data for a given patient, then the check for the end of that patient's data would never occur. For that reason a fail-safe check for the end of the dataset should always be put in place.

A final check can be done at the end of a section to see if there is enough room to start the next section of data. If there are 6 lines left and the section header takes 2 lines and the footer takes 4 lines, then there is no reason not to start the next section on a new page. By putting out blank lines, you can guarantee that the spacing of the footer will be consistent across the different sections of the subject profile.

* If not enough space, start a new page. *
if (lineslft < (&footsz + 6)) then do;
  blanks = lineslft - &footsz;
  do i = 1 to blanks;
    put;
  end;
  link foot;
end;

By repeating this method for each section of data with an unknown number of types, you can avoid any problem with sections of data breaking at unknown points. The only additional step you would need is to add logic to break when the last section for a patient is complete. The following code will accomplish this

    link foot;

This assumes that your foot section also has some mechanism for adding blank lines until the footer section is printed. If not, then the same logic mentioned previously could be used.

So, with the header and footer sections defined, and an understanding of the types of data elements that are needed for your subject profiles, you can generate a consistent, well-defined subject profile regardless of the number of large data sections that each patient has.

**CONCLUSION**

If you identify your data and lay it out before programming begins on a subject profile, you can quickly define a guide for programming it. This identification process should always be your first step.

By collecting the constant part of the subject profile from single or known numbers of data elements and storing them in a record per patient format, you can easily develop the demographic section of your subject profile. This single record per patient will also be used to drive the rest of the subject profile.

Taking advantage of the multiple SETs allowed in the datastep will let you develop multi-page sections for concomitant medications and adverse events. These sections will take into account headers and footers to produce a nicely formatted report.

Putting all of these together, you can generate a complete subject profile. A side benefit is that the tools can be used in the future to make the next subject profile a little less painful.

**TRADEMARKS**

SAS is a registered trademark of SAS Institute, Inc. in the USA and other countries.

**CONTACT INFORMATION**

Your comments and questions are valued and encouraged. Contact the author at:

Greg Silva  
PPD Development  
3151 South 17th Street  
Wilmington, NC 28412  
(910) 772-6986  
greg.silva@wilm.ppd.com
COMPANY XXX STUDY A-BCD
SUBJECT PROFILE

Center Number: 125  Subject Number: 231  Subject Initials: ITH  Randomization Code: A

Age: 12 Years  Met inclusion Criteria: Yes
Sex: Male  Date of First Dose: 06/17/1998
Race: Hispanic  Date of Last Dose: 08/21/1998
Date of Birth: 03/13/1986

VITAL SIGNS

<table>
<thead>
<tr>
<th>Visit</th>
<th>Visit Date</th>
<th>Temp (F)</th>
<th>Heart Rate (bpm)</th>
<th>Blood Pressure (mmHg)</th>
<th>Weight (lb)</th>
<th>Height (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>06/17/1998</td>
<td>97.2</td>
<td>70</td>
<td>112/72</td>
<td>123</td>
<td>56</td>
</tr>
<tr>
<td>2</td>
<td>06/29/1998</td>
<td>97.7</td>
<td>74</td>
<td>110/73</td>
<td>131</td>
<td>56</td>
</tr>
<tr>
<td>3</td>
<td>08/01/1998</td>
<td>98.2</td>
<td>80</td>
<td>109/72</td>
<td>145</td>
<td>56</td>
</tr>
<tr>
<td>3</td>
<td>08/21/1998</td>
<td>97.2</td>
<td>83</td>
<td>106/70</td>
<td>143</td>
<td>57</td>
</tr>
</tbody>
</table>

Example 1 – Single or known number of occurrences per patient.

Laboratory Results

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Units</th>
<th>Visit 1</th>
<th>Visit 2</th>
<th>Visit 3</th>
<th>Visit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT (SGPT)</td>
<td>IU/L</td>
<td>126</td>
<td>103</td>
<td>112</td>
<td>42</td>
</tr>
<tr>
<td>AST (SGOT)</td>
<td>IU/L</td>
<td>65</td>
<td>48</td>
<td>53</td>
<td>37</td>
</tr>
<tr>
<td>BASOPHILS, ABS</td>
<td>X10E3/UL</td>
<td>0.02</td>
<td>0.05</td>
<td>0.03</td>
<td>0.06</td>
</tr>
<tr>
<td>BICARBONATE</td>
<td>MEQ/L</td>
<td>22</td>
<td>22</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>BUN</td>
<td>MG/DL</td>
<td>15</td>
<td>13</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>CALCIUM</td>
<td>MG/DL</td>
<td>9.7</td>
<td>16</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>CHLORIDE</td>
<td>MEQ/L</td>
<td>113</td>
<td>103</td>
<td>104</td>
<td>112</td>
</tr>
<tr>
<td>CREATININE</td>
<td>MG/DL</td>
<td>0.6</td>
<td>0.6</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td>HEMOGLOBIN</td>
<td>G/DL</td>
<td>47.2</td>
<td>41.6</td>
<td>43.7</td>
<td>40.6</td>
</tr>
<tr>
<td>HEMATOCRIT</td>
<td>%</td>
<td>14.1</td>
<td>14.4</td>
<td>14.3</td>
<td>14.1</td>
</tr>
<tr>
<td>HDL</td>
<td>MG/DL</td>
<td>376</td>
<td>347</td>
<td>354</td>
<td>342</td>
</tr>
<tr>
<td>PLATELET COUNT</td>
<td>X10E3/UL</td>
<td>3.8</td>
<td>3.6</td>
<td>3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>SODIUM</td>
<td>MEQ/L</td>
<td>141</td>
<td>141</td>
<td>153</td>
<td>146</td>
</tr>
</tbody>
</table>

Medications Taken

<table>
<thead>
<tr>
<th>Medication Name</th>
<th>Dose</th>
<th>Unit</th>
<th>Start Date</th>
<th>Stop Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTIVITAMIN</td>
<td>2</td>
<td>TABS</td>
<td>22SEP1998</td>
<td></td>
</tr>
<tr>
<td>NYQUIL</td>
<td>1</td>
<td>TSP</td>
<td>29SEP1998</td>
<td>29SEP1998</td>
</tr>
<tr>
<td>KEPZOL</td>
<td>2000</td>
<td>MG</td>
<td>03MAR1999</td>
<td>14MAR1999</td>
</tr>
<tr>
<td>NIZORAL CREAM 2%</td>
<td>1</td>
<td>APP</td>
<td>11MAR1999</td>
<td>21MAR1999</td>
</tr>
<tr>
<td>TYLENOL SINUS</td>
<td>1</td>
<td>TAB</td>
<td>15APR1999</td>
<td>15APR1999</td>
</tr>
<tr>
<td>CYCRIN</td>
<td>2</td>
<td>TAB</td>
<td>01JUN1999</td>
<td>24JUL1999</td>
</tr>
</tbody>
</table>

Example 2 – Unknown number of occurrences per patient.