Creating Powerful Laboratory Listings
Bob Hull, Synteract Inc., Encinitas, CA

ABSTRACT
The summarization of laboratory data can be very exhausting. Different laboratory units, and new test names appearing each time the data is updated can cause quite a headache trying to keep the output from your programs correct. I offer suggestions that can enable your laboratory listing program to recognize these problems in updated data and appropriately display all of the data. Allowing your program to count all of the unique test/unit combinations will enable you to use macros and proc report to accurately display all of the data. Then when it is requested to have 6 columns to a page rather than five, you do not have to change all of your macro calls. Instead just identify the number of columns desired and your program will build its own macro calls.

This article is intended to give ideas on how to create a very robust laboratory listing. It outlines specific items that will be needed to accomplish that task. To implement the ideas it is assumed that you have a good knowledge of

LABORATORY LISTINGS
Lab listings are the most difficult listings to create due mainly to the manipulation required to get the data in a format acceptable for listing. Once these details are completed it is often assumed that the data will continue to follow the same format when updates arrive. When a new lab test comes in that was not dealt with before it will likely be left out of the listing. Ideally, the programs that create the data set and the listing would be able to recognize new data and create space for them appropriately.

MANUAL HANDLING OF NEW DATA
One way of handling new data is to go into the listing program and add a new column for the new data. This assumes that your data have a variable for each lab result. Your original code might look like:

```sas
Proc report data=labdata nowindows split="#";
Columns patid wbc lymph neut;
Define patid /group "Patid"
Define wbc /display "WBC#"
Define lymph /display "LYMPH#"
Define neut /display "NEUTRO#"
Run;
```

To modify this you would need to add one define statement and one variable name in the columns line. This may not seem like a lot of modification, but it has two primary problems. First, if you did not know there was a new column to add, those data would be inadvertently omitted from the listing. Secondly, it may not be so easy to add a new column in this format. Lab listings typically use the entire width of the page and require several Report procedures. To add a single column may require that all of the other components of the listing also be adjusted. Doing this every time new data arrives is difficult.

IMPROVED METHOD
A method that is better than the manual one listed above is to use an ACROSS statement in your PROC REPORT. That assumes that your data contains one variable with all of the lab test names. Then your code might look something like this:

```sas
Proc report data=labdata nowindows split="#";
Columns patid labtest, (labresu dummy);
Define patid /group "Patient"
Define labtest /across ""
Define labresu /display "Result"
Define dummy /analysis " " noprint;
Run;
```

In this instance new data will be incorporated into the listing. The listing will create an extra column for each new lab test appearing. The dummy variable is there because any use of an ACROSS statement requires an ANALYSIS variable.

The down side to using this approach is that all of the lab data can not be included in the file being used by the PROC REPORT. That would create far too many columns to fit on a single page. You then have to write code that will call the PROC REPORT several times, each time selecting a specific set of lab tests. Since the tests need to be selected manually, the program still needs to be modified. If the test needs to be included in the first part of the listing, then all of the calls to the PROC REPORT after that must be modified.

ALTERNATIVE
The alternative is to write SAS® code that will use SAS MACRO facility and PROC REPORT to determine how many listings are required and generate the SAS MACRO code itself. Using this type of programming, the arrival of new data will not require any additional programming. Your code will automatically recognize that there is a new laboratory test and create a column for it while shifting other columns around if necessary.

UNDERSTANDING YOUR DATA
This method requires the data to be stored in the same manner as was mentioned for the “Improved Method”. When new tests appear they will simply add more records to your data set. PROC REPORT will organize them alphabetically when using the ACROSS statement. However, if there are two lab tests with the same name, but are collected from different sources then only one will appear in your list. This occurs frequently with hematology white blood cells and urinalysis white blood cells. So it is important to separately list the types of lab data. This is the typical way to handle that, so doing this should not be a problem. If for some reason you need to combine all tests from all lab types together, you would have to name all of the tests differently to use this. For instance, you might name white blood cells from hematology as "WBC". Then you could just leave hematology white blood cells as "WBC". This must be done in the data, not in the PROC REPORT.

The use of the ACROSS statement means that the lab tests will be organized alphabetically from left to right. This is not always the desired result. Often times the granulocytes are grouped together while infrequently collected lab tests are pushed to the back. You will see what is necessary to accomplish this.

NUMBERING LAB TESTS
Since the whole goal of the program is to handle any data that is given, you can never refer to specific lab tests within your program. Each unique lab test must be assigned an integer value. The numbers representing lab tests should be sequential integers with no gaps (You will see why gaps would cause problems later). When summarizing lab data, it is incorrect to place data in the same column that has different units. That takes away the relevance of the column. Thus what makes each lab test unique is the lab type (Hematology, Blood Chemistry, Urinalysis etc.), the lab test, and the units of the test. Each unique lab type, lab test, and unit should have an integer value associated with it. This could be accomplished using a PROC SORT and a DATA STEP as it might appear below:
Now the essential pieces of the program are in place for you to be able to write SAS/MACRO code that will call the Report procedure the appropriate number of times. This requires one large macro to be written that will surround all of the SAS code. Inside of this macro should be a PROC REPORT. Then you have to write SAS/MACRO code that will call the macro that contains the PROC REPORT until all of the lab tests have been called. The SAS/MACRO code to do this can be written so that you can identify that you desire any number of lab tests to appear across the top of your page. Then by changing a single number the listing will adjust all of the Report procedures.

This is why your unique lab numbers must be sequential integers. The SAS/MACRO code you write will cycle through the lab tests four at a time (or whatever number you specify) sequentially until it reaches the last lab test available. Depending on how you handle lab types, your final SAS/MACRO call might look something like this one that references the macro “printit” made earlier in the program:

```sas
%do q=1 %to &ntypes;
  %let j=&&&&t&&type&q;
  %let f=&&&&t&&type&q;
  %do m=1 %to &&&&t&&type&q;
    %printit(&&f&m);
  %end;
%end;
```

This macro call creates separate listings based on the type of the lab test. The variable “’j’ will be the total number of lab types. The variable ‘q’ is the part number of the listing and ‘t’ is the total number of parts. This enables a nice title like “Part 1 of 7” to be generated automatically. You can see that the variable ‘m’ will just cycle through all the parts, as it references the same variable ‘f’ references. The ’y’ variable inside of the ‘printit’ call was created earlier and contains the list of unique lab numbers that will go into this particular part of the listing.

**CHANGING THE ORDER**

The changing of the order of the lab tests from something other than alphabetical requires an additional variable in the data. SAS will automatically sort the tests alphabetically as discussed earlier. To avoid this you must have a variable that identifies how the lab test should be sorted. You could insert something like this into your program that creates the raw data:

```sas
If labtest="WBC" then order=1;
Else if labtest="BASOPHILS" then order=2;
Else order=3;
```

Then tell your macro to put the variable order before labtest when assigning unique numbers to each lab test. This method will allow you to put specific tests together and have the others where order is not as important be sorted alphabetically.

**ADDITIONAL POSSIBILITIES**

The listing as described above will accomplish the task described in the abstract. However, it is likely that you will want your listing to do more than just that. You may incorporate the idea of lining up the decimal place into your listing. By placing the lab result into a character variable you can determine where the decimal is move the spaces around so that when your listing is displayed all of your decimals are lined up.

You can increase the functionality of your column headings as well. Putting the laboratory normal ranges in the heading is something that is very useful at times. Since the same laboratory test may have several different normal ranges, you will have to take the lowest low normal range and the highest high normal range for each lab test.

**DRAWBACKS**

Clearly, the only draw back to such a robust, powerful, all encompassing lab listing program is that it is difficult to program. Putting all of the pieces together to create your lab
listing will require that you carefully examine the macro variables you create.

CONCLUSION
Dealing with lab data is one of the most difficult aspects of pharmaceutical programming. Data are constantly updated and you can never wait until the data are final to write your SAS programs. You must have robust programs that can handle new data. Thoroughly examining lab data for new lab tests and adjusting programs every time new data arrives is a difficult task. Taking the time to write an all encompassing lab listing program that can be used for any study is well worth the effort!

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

Bob Hull
Synteract, Inc.
187 Calle Magdalena
CA 92024
Work Phone: 760 634 2133
Fax: 760 634 2170
Email: rhull@synteract.com