DM03

Lab Unit Conversions: Taking Advantage of “Expert” External Resources

Sandra Minjoe, Genentech Inc, South San Francisco, CA

ABSTRACT

Analysis of lab data can be tedious. There are many different tests, and they are usually taken for each subject at several times during the course of the clinical trial. Lab results are often our largest dataset. Thus any work we do for lab data can easily get long and boring unless we can find a way to standardize or macrotize it.

This paper focuses on one of the common lab analysis needs – converting lab test results from one unit into another. I begin by describing why it is necessary to perform unit conversions. I then describe some purely SAS® - based approaches, since, as SAS programmers, this is typically how we tackle a project. Finally, I explore and recommend processes that make use of a combination of SAS programming along with non-programmer “experts” in lab data.

INTRODUCTION

Unit conversions happen all the time. When we ask for change for a dollar to put in a parking meter, that’s a unit conversion: 1 dollar bill is equivalent to 4 quarters, but when dealing with a parking meter it is more useful for us to work with quarters. In this example, we can say “4” is the conversion factor. A programmer might think of this as something like # quarters = 4 x # dollars.

In clinical studies, data is usually collected through many different investigators, often in different parts of the country or even the world. We may end up with height collected in inches by some investigators and in centimeters by others. To do any sort of summary on these numbers, even something as straightforward as mean, min, or max, we must first somehow convert everyone to the same scale or “unit”. This is the basic concept of unit conversions.

Some clinical studies choose to use the same central lab to do all of their lab testing, and thus report all the results for a specific lab test in the same unit. Conversion is not an issue here.

Often a central lab is too impractical or too expensive to justify. In these cases, we invariably have to deal with different labs reporting their test results in different units. For example, at one lab the test BUN might be reported in units of mg/dL, but at another they report in units of mmol/L. To work with this data, we need to convert into some common unit.

On the surface, this may not seem complicated. No different than, say, working with vital signs, such as height and weight, right?

Not exactly.

First of all, lab data DOES have this same conversion issue, but many times over. Just like height and weight, lab data from different countries is also often collected in different units. But even within the same country, almost every lab test has potentially several different reported units, not just one or two. And because we typically deal with many different lab tests and many different units, the different types of units (and thus the number of conversion factors) add up quickly.

Secondly, lab data has an additional scientific issue we need to work with. Sometimes the unit conversion is not the same from one lab test to another, as will be explained further a little later in this paper. This added complexity can make our programming task especially daunting.

SAS SOLUTIONS

Let’s begin by looking at a purely SAS-based approach for something very straightforward. In fact, we’ll start by examining some code that works well for non-lab data: vital signs weight. We’ll then make an attempt to expand that out for lab data and see where we run into difficulties.

Simple Example: Vital Signs Weight

Weight, similar to height mentioned above, is actually a pretty simple conversion. It generally has only two possible units, “kg” (kilograms) and “lb” (pounds). Once it has been decided which of these units to report in, a simple if-then-else statement can be used to make the necessary conversions in our data.

For example, if we want to report in kilograms, we might do something like the following:

Table 1: Simple example for Weight

```
* Convert weight in pounds into kilograms for reporting;
if (weightu = ‘lb’) then do;
     weight_rpt = weight x 0.454;
     weightu_rpt = ‘kg’;
end;

* Copy weight in kilograms for reporting;
else if (weightu = ‘kg’) then do;
     weight_rpt = weight;
     weightu_rpt = weightu;
end;

* If weight is not in pound or kilograms, check the data;
else put ‘ALERT: Check weight units for: ‘ patient=  weightu=;
```

In this way, we keep our original results and units (weight and weightu, respectively), but also create a new “reported” version (weight_rpt and weightu_rpt).

Lab Example

To expand on this for lab data unit conversions, we first need a list of all the units we’re converting from, the units we’re converting to, and the conversion factor for each pair. Once we have that, we can write code such as:

Table 2: Simple Lab Example

```
* Convert g/L into g/dL for reporting;
if (unit = ‘g/L’) then do;
     result_rpt = result x 0.10;
     unit_rpt = ‘g/dL’;
end;

... and so on for other units to convert ...

* Copy for reporting;
else (if unit = ‘g/dL’) then do;
     result_rpt = result;
     unit_rpt = unit;
end;

* If no unit conversion was made, check the data;
else put ‘ALERT: Check units for: ‘ patient=  test=  unit=;
```

Note that this could also be done using other means, such as a select statement. I’ve used basic “if-then-else” statements here so that it would be understandable by all audiences.

You might be thinking “it’ll take a bit of work to code all the lab units, but this isn’t so bad.” Actually, though, there is more to consider.

Expanded Lab Example

As mentioned in the introduction, there is an additional complexity with lab data, where the unit conversion for one lab test is different than that for another.

For most units, conversions don’t vary based on the object we’re measuring. Converting from kilograms to pounds is done the same way regardless of whether we’re weighing a person, a chicken, or a package for shipping. But this is not the case with the unit of “moles”. Without getting too much into chemistry here, a mole is actually a count of the number of atoms in a certain amount of a substance. For example, one mole of water contains 18 grams, but a mole of hydrogen contains only 2 grams[1]. This means that if we want to convert from moles to grams, the conversion factor is different for water than it is for hydrogen. Unlike weight, we need to know what we’re measuring before we can determine the appropriate conversion factor.

The same complication applies when it comes to unit conversions of lab values that are measured in moles. We must consider not only the units of measurement, but the lab test itself. To handle this in our code, we must have conditional statements not only for the unit, but also the test name. Our code from above can be expanded to handle this as follows:

Table 3: Expanded Lab Example

```
* Convert test BUN into standard units of mmol/L;
if (test = ‘BUN’) then do;

   * Convert mg/dL into mmol/L for reporting;
   if (unit = ‘mg/dL’) then do;
      result_rpt = result x 0.357;
      unit_rpt = ‘mmol/L’;
   end;
```

... and so on for other units within this test name ...

```
   * Copy for reporting;
   else if (unit = ‘mmol/L’) then do;
      result_rpt = result;
      unit_rpt = unit;
   end;

   * If unit is not in the list, check the data;
   else put ‘ALERT: Check test and units for: ‘ patient=  test=  unit=;

end;  *** End of BUN test ***;

* Convert test CALCIUM into standard units of mmol/L;
else if test = ‘CALCIUM’ then do;

   * Convert mg/dL into mmol/L for reporting;
   if unit = ‘mg/dL’ then do;
      result_rpt = result x 0.25;
      unit_rpt = ‘mmol/L’;
   end;
```

... and so on for other units within this test name ...

```
   * Copy for reporting;
   else if (unit = ‘mmol/L’) then do;
      result_rpt = result;
      unit_rpt = unit;
   end;

   * If unit is not in the list, check the data;
   else put ‘ALERT: Check test and units for: ‘ patient=  test=  unit=;

end;  *** End of CALCIUM test ***;
```

... and so on for remaining test names ...

```
* If test name is not in the list, check the data;
else put ‘ALERT: Check test and units for: ‘ patient=  test=  unit=;
```

Now that’s a lot of code!

Again, we could use select statements, but that won’t help decrease the enormous volume by much.

Additionally, we have to think about the amount of work involved to maintain this type of code. Each time we get lab results for a new test or a new unit within a test, we would need to add more code. As you can imagine, this can be particularly unwieldy when trying to use this same standardized chunk of code across multiple studies and indications.

Consider also the difficulty in validating a program such as this. This type of checking should be done by someone with some expertise in lab unit conversions, but you’ll likely have a difficult time finding such a person who is also competent in reading SAS code!

Ways to Shorten the Code

I was faced with exactly the situation shown above in the expanded lab example: a long standard program with nested IF statements for each test and unit combination. As stated above, it was difficult to maintain across the company, and a tremendous effort to validate.

My first reaction was to replace the code with a macro solution, passing parameters including unit and conversion information. I thought this might make the code simpler because I could just add a macro call each time I received a new test or unit. That idea developed into the following:

Table 4: Expanded Lab Example with Macro

```
data converted;
   set labdata;

   * Macro to derive new unit;
   %macro unitconv(newunit=, convfact=);
      result_rpt = result x &convfact;
      unit_rpt = &newunit;
   %mend unitconv;

   * Convert each test and unit using macro;
   if test = ‘BUN’ then do;
      if      unit = ‘mg/dL’  then %unitconv(newunit=’mmol/L’, convfact=0.357);
      else if unit = ‘mmol/L’ then %unitconv(newunit=’mmol/L’, convfact=1);
```

... and so on for other units within this test name ...

```
      * If unit is not in the list, check the data;
      else put ‘ALERT: Check test and units for: ‘ patient=  test=  unit=;
   end;

   else if test = ‘CALCIUM’ then do;
      if      unit = ‘mg/dL’  then %unitconv(newunit=’mmol/L’, convfact=0.25);
      else if unit = ‘mmol/L’ then %unitconv(newunit=’mmol/L’, convfact=1);
```

... and so on for other units within this test name ...

```
      * If unit is not in the list, check the data;
      else put ‘ALERT: Check test and units for: ‘ patient=  test=  unit=;
   end;
```

... and so on for remaining test names ...

```
   * If test name is not in the list, check the data;
   else put ‘ALERT: Check test and units for: ‘ patient=  test=  unit=;
run;
```

While it’s true that this method shortened my code a little, it wasn’t the significant savings as I’d hoped for. This is because most of the work is being done in the conditional IF statements. Additionally, the above code is still difficult to maintain, but the added complexity of the macro has made it even less clear for a non-SAS programmer to review!

LOOK-UP TABLE SOLUTION

Instead of trying to make a macro solution work here, I switched to a different route entirely: using a look-up table. This turned out to be a much better fit for my needs.

First I thought about how to design a look-up table to contain all the information I needed: test name, incoming unit, units to use in reporting, and the conversion factor. My resulting table looked something like this:

Table 5: Basic Look-Up Table

test	unit	unit_rpt	factor
BUN	mg/dL	mmol/L	0.357
BUN	mmol/L	mmol/L	1
...			
CALCIUM	mg/dL	mmol/L	0.25
CALCIUM	mmol/L	mmol/L	1
...			

With all this information now in a look-up table, I was able to pull it out of my program. The resulting code was then reduced to the following:

Table 6: Lab Example Using the Look-Up Table

```
* Bring in lab data and sort to prep for merging with lookup table;
proc sort data=libname.labdata out=labdata;
   by test unit;
run;

* Bring in LOOK-UP table of conversions;
proc sort data=libname.lookup out=lookup;
   by test unit;
run;

* Combine lab data and lookup table to convert units;
data labdata2;
   merge labdata (in=inlab)
         lookup  (in=intbl);
      by test unit;

   * Convert tests that are in both lab dataset and look-up table;
   if (inlab and intbl) then result_rpt = result * factor;

   * Warn if test or unit is missing from lookup table;
   else (if inlab and not intbl) then
      put ‘ALERT: Look-up table missing conversion for: ‘ test=  unit=;

   * Exclude records from look-up table that are not in incoming lab data;
   else if (intbl and not inlab) then to;
      put ‘ALERT: Study does not use look-up table data: ‘ test=  unit=;
      delete;
   end;
run;
```

That’s it! No more long nested IF statements to specify every single lab test and unit we might come across.

With this process all the work of maintaining plus most of the validation has been shifted to the external look-up table. All new tests and units can be incorporated simply by updating the SAS dataset containing the look-up table.

There is another benefit: because the information has been pulled out of the code, the reviewer doesn’t need to have SAS programming skills to do this work. A SAS programmer just needs print the look-up table (preferably with informative labels instead of SAS names) to have it reviewed by someone who has lab unit conversion expertise.

I developed a process like this at a previous company, and found it to work quite well there.

PROPOSED SOLUTION

The only difficulty might come when trying to maintain this look-up table across multiple studies and indications, because multiple people might need to regularly update the look-up table to add new tests and units. For small companies, this is probably not an issue. However, we can take it one step further, as described below.

External Resources

SAS programmers typically have to find conversion factors each time we get a new lab unit. We might need to research them (on-line or via booklet), but many companies have some sort of internal resource personnel. This may be someone from the medical or nursing profession, such as a clinical scientist, medical monitor, clinical trials manager, etc.

I work with a group of lab data managers who understand the lab data and make unit conversions. They are our external resource. But it traditionally has been difficult to get them involved in checking to make sure our conversions were done correctly, as they are not SAS programmers themselves.

My proposed solution takes advantage of these lab unit conversion experts, even though they are not fluent in reading SAS code.

SAS Programmers + Non-SAS Experts

One distinct advantage to a look-up table is that it can be stored as something other than a SAS dataset. This allows non-SAS programmers the capability of accessing and modifying it, not just reading it from a printout.

I wanted to get our external experts, in my case the lab data managers, more involved in maintaining the conversion factors. My original goal was to give them access (possibly using Microsoft Excel™), to a look-up table file similar to Table 5 shown above. We SAS programmers would initially create something like this for the lab data managers as a first draft, convert it to and Excel spreadsheet, and the lab data managers could then update it with additional records as needed. By storing this file in a central location, where the lab data managers could write to it and we SAS programmers could only read it, we then could simply bring in the most recent version (converting back to a SAS dataset) anytime we needed it.

Thus the only step additional to the Look-Up Table Solution mentioned above is the concept of exporting and importing the table, or converting from and to a SAS dataset. This can be done with SAS in several different ways, including the DATA STEP, PROC IMPORT, or the Import and Export Wizard tools[2] in SAS Display Manager.

The most significant advantage of this approach is that it puts the responsibility of the unit conversions on the group who actually understands it, rather than on us SAS programmers. No longer would we SAS programmers need to be concerned about writing and validating each specific unit conversion in our programs.

Of course we'd still need to validate that our program correctly brings in the data, converts it to a SAS dataset, and uses the information on it, but this is a much smaller task than dealing with all the IF subsetting.

Actual Process

We are in the process of taking this one step even further. Our lab data managers work with Oracle Clinical™, which includes tables for lab unit conversions. We are developing a new procedure to extract these tables directly from Oracle Clinical into a SAS dataset. The lab data managers are happy with it because they are already familiar with Oracle Clinical and won't have to learn a new system. This has another advantage in that it is a controlled system and thus compliant with FDA Part 11. However, because it requires we get data from one server in one form and bring it over to another server in a different form in this validated system, it has ended up a larger and longer process to set up than the Excel file solution.

Once this project is up and running, maintenance and validation of lab unit conversions will be dramatically simplified for SAS programmers, yet not too much work for the lab data managers.

CONCLUSIONS

Look-up tables can make a SAS programmer's life easier. They are flexible: they can be easily modified and validated by external experts. They are portable: they can be used for all lab data in all studies. And they require less, not more, code to maintain.

By adding in a simple step to convert between some other file type (such as an Excel spreadsheet) and a SAS dataset, you can also take advantage of external lab unit experts (who are not SAS programmers) to maintain, validate, and maybe even create your look up table. This method allows SAS programmers to focus on what they do best (writing code), while leaving the actual unit conversions work to the experts.

Consider using a look-up table for your lab unit conversions and for any other situation where you have many IF conditions nested in your code.

REFERENCES

1. For more information about the mole and how it is used in chemistry, the website http://antoine.frostburg.edu/chem/senese/101/moles/index.shtml is a nice reference.

2. If you don't typically do much importing and exporting of data, I highly recommend using the Import and Export Wizard to generate code that you can save and use again later. Search SAS OnlineDoc® for 'wizard' to read more about this feature.

ACKNOWLEGEMENTS

I'd like to thank Karen Petrie, Senior Clinical Data Coordinator and one of our "internal expert" lab data managers, for suggesting the use of Oracle Clinical and volunteering her group to do all the lab unit conversion data entry work and maintenance.

I'm also grateful to Genentech for allowing and encouraging us to move ahead with projects such as this.

CONTACT INFORMATION

Sandra Minjoe
Genentech, Inc
1 DNA Way
South San Francisco, CA 94080
Phone: (650) 225-4733
Fax: (650) 225-2630
Email: sminjoe@gene.com