Using Google to Solve SAS® Problems
Michael Todd, Nth Analytics, Flemington, NJ

ABSTRACT
SAS® has a long history as the language for statistical programming in the pharmaceutical industry. In the past 30+ years, most, if not all, of the problems statistical programmers face on a day-to-day basis have been solved by somebody. Usually there is a paper somewhere on the Internet about it. Using Google queries, statistical programmers can leverage the vast SAS knowledge base on the Internet, and get quick answers to a variety of problems.

INTRODUCTION
SAS has been around longer than 30 years. Most of the SAS programming problems have been worked out somewhere, by somebody. To solve a problem, it is often a matter of finding the solution online, rather than working through the problem by yourself. You can greatly enhance your programming skills by leveraging the knowledge of others.

We all get used to doing things a certain way. Many problems can be solved using code already available at our companies. However, if new situations come up, there may not be anything to copy from. If new technologies get implemented, we probably will need some help just to get them working. Training does not cover all the situations. We need an ongoing source of knowledge. With Google, there is always someone to ask for help.

In this paper, I approach the programming issues from the viewpoint of a consultant who works for many organizations in the pharmaceutical industry, including Contract Research Organizations (CROs). These companies have many different ways of doing things. Usually, they are not anxious to change. As a ‘guest in the house’, you need to do things their way, and quickly. This can require quick mastery of unfamiliar techniques.

I explore using Google to solve five very different programming problems, and explain my reasons for choosing the particular solution I did.

• Simulating the lag function in PROC SQL
• Methods of putting ‘Page x of y’ on RTF output
• Age in months calculation
• Confidence intervals for the median
• International encoding difficulties opening a dataset

All of the solutions returned by the searches are valid. It is question of which solution is most easily implemented, cleanest, and best suited to a particular organization.

GOOGLING SAS PROBLEMS: OVERVIEW
There are five main sources of information available online for solving SAS problems:

• SAS documentation: a great source of answers to SAS problems. Documentation for Base SAS, all procedures, ODS, and most other topics are available at http://support.sas.com

• SAS books are available through Google’s controversial projects to make all books available online in a searchable format. This is handy when you are looking for a solution to a single problem, and don’t need the whole book.

• Social media: SAS Institute has a strong commitment to social media. Up-to-date resources, including Twitter, Facebook, and YouTube feeds are available at http://support.sas.com/community/socialmedia/index.html.

• SAS-L. One of the original social media sites, founded in 1985, and still active. It is a listserv, which is an email server. If someone posts an email, it goes to everyone on the list, and people can respond in real time. Many expert programmers have answered questions over the years. Emails from the archives are available
from 1996 to the present, and often contain targeted solutions to SAS problems. The web address is http://listserv.uga.edu/archives/sas-l.html.

- **Conference papers** are a major source of SAS programming solutions. These papers are well-reasoned, peer-reviewed, and tend to be focused on a single topic. They often have step by step, simple solutions where one can lift the code straight out and use it.

- [http://www.lexjansen.com](http://www.lexjansen.com) often comes up on Google searches. This site has a collection of over 10,000 SAS conference papers.

**GOOGLE STORES YOUR SEARCHES**

Each time you search, Google stores what you searched and the result. It also stores the number of times you searched it. You will need an account, and to be logged in on your Google account when searching. This tends to happen automatically. If you forget how solved a problem, you can go to [https://www.google.com/history](https://www.google.com/history), log in, and search your searches. As shown in Table 1, I often search PROC SQL. I have difficulty grasping some of the syntax. In the age of Google, this is not a major problem, because I can easily find the answer.

Table 1

<table>
<thead>
<tr>
<th>Most Frequent PROC SQL Searches</th>
<th>PROC SQL except operator</th>
<th>PROC SQL into separated by</th>
<th>SAS PROC SQL intersect</th>
<th>SAS PROC SQL begins with operator</th>
<th>PROC SQL like operator</th>
<th>SAS PROC SQL retain</th>
<th>SAS PROC SQL row difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC SQL noprint</td>
<td>PROC SQL except operator</td>
<td>PROC SQL into separated by</td>
<td>SAS PROC SQL intersect</td>
<td>SAS PROC SQL begins with operator</td>
<td>PROC SQL like operator</td>
<td>SAS PROC SQL retain</td>
<td>SAS PROC SQL row difference</td>
</tr>
<tr>
<td>SAS PROC SQL correlated subquery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROC SQL into</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAS SQL coalesce</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROC SQL insert</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAS SQL between operator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAS PROC SQL union syntax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SQL LAG FUNCTION PROBLEM**

In this section I show how to use PROC SQL to simulate the lag function. Why would I want to do this? I could easily use the DATA STEP. This type of problem comes up more than one would expect in on-site consulting. With the advent of SAS Enterprise Guide, some companies are phasing out the DATA STEP. It is possible to consult in a place where the DATA STEP is discouraged.

**GOOGLE SEARCH: SAS PROC SQL LAG**

In this case, the specific task was to use PROC SQL to simulate the lag function to find the cutoff for the 25th percentile. Figure 1 shows the results of the first search. In 2005, someone asked this question on SAS-L: “How do you do something similar to the lag function in a data step?”

**Figure 1**

![Google search results for SAS PROC SQL LAG](image)
Opening the link, as seen in Figure 2, it looks like a potentially good solution. The code is clean, and looks reasonably simple and straightforward to implement. There is a subquery that I don’t totally understand, but looks like it will probably work.

**Figure 2 (Schreier, 2005)**

Here’s an example. First use the LAG function in a DATA step.

```plaintext
data age_lag;
  set sashelp.class;
  age_lag = lag(age);
  keep name age age_lag;
run;
```

To emulate that in SQL, first add a column with serial numbers for the rows

```plaintext
data numbered;
  set sashelp.class;
  sn + 1;
run;
```

Then code a subquery to get the AGE value from the “previous” row:

```plaintext
cproc sql;
create table age_lag_sql as
  select class.age,
     (select lag.age
      from numbered as lag
      where class.sn - lag.sn = 1) as age_lag
     from numbered as class
     order by class.sn
;
quit;
```

It worked. As shown in Figure 3, I changed the variable names to match those in my programs, and was able to get the answer I wanted, the value of the 25th percentile. This is an example of a successful search.

**Figure 3**

<table>
<thead>
<tr>
<th>BASE</th>
<th>cumFrequency</th>
<th>cumPercent</th>
<th>CumPerct</th>
<th>base_lag</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>1</td>
<td>0.4002591</td>
<td>0.400259</td>
<td>46</td>
</tr>
<tr>
<td>45</td>
<td>2</td>
<td>0.8005182</td>
<td>0.800518</td>
<td>45</td>
</tr>
<tr>
<td>44</td>
<td>6</td>
<td>2.4105644</td>
<td>2.410564</td>
<td>44</td>
</tr>
<tr>
<td>51</td>
<td>12</td>
<td>4.8307068</td>
<td>4.830706</td>
<td>48</td>
</tr>
<tr>
<td>52</td>
<td>20</td>
<td>0.6403516</td>
<td>0.640351</td>
<td>52</td>
</tr>
<tr>
<td>53</td>
<td>25</td>
<td>10.600647</td>
<td>10.600647</td>
<td>52</td>
</tr>
<tr>
<td>54</td>
<td>30</td>
<td>13.201974</td>
<td>13.201974</td>
<td>53</td>
</tr>
<tr>
<td>59</td>
<td>25</td>
<td>14.112961</td>
<td>14.112961</td>
<td>54</td>
</tr>
<tr>
<td>96</td>
<td>41</td>
<td>14.919564</td>
<td>14.919564</td>
<td>55</td>
</tr>
<tr>
<td>58</td>
<td>46</td>
<td>16.545981</td>
<td>16.545981</td>
<td>57</td>
</tr>
<tr>
<td>53</td>
<td>53</td>
<td>21.370967</td>
<td>21.370967</td>
<td>58</td>
</tr>
<tr>
<td>62</td>
<td>50</td>
<td>23.870956</td>
<td>23.870956</td>
<td>58</td>
</tr>
<tr>
<td>61</td>
<td>61</td>
<td>24.545981</td>
<td>24.545981</td>
<td>60</td>
</tr>
<tr>
<td>62</td>
<td>68</td>
<td>27.419564</td>
<td>27.419564</td>
<td>61</td>
</tr>
<tr>
<td>63</td>
<td>71</td>
<td>28.629802</td>
<td>28.629802</td>
<td>62</td>
</tr>
<tr>
<td>64</td>
<td>77</td>
<td>30.408371</td>
<td>30.408371</td>
<td>63</td>
</tr>
</tbody>
</table>

**25th percentile cutoff**

**RTF SEARCHES**

RTF is another thing I have difficulty remembering. With Google, this is not an issue. I can always search and find the answer. Over time, I have learned to do more targeted searches, particularly when I have searched the problem before. Table 2 shows the RTF topics I have searched:
The purpose of these searches is always to get the output to look exactly the way I want it. The need to do this tends to come up in the CRO industry. Clients may insist that output looks a certain way. However, I have also seen FDA reviewers make similar sorts of requests.

GOOGLE SEARCH: SAS RTF PAGE X OF Y

One thing I have searched successfully is how to put ‘Page x of y’ (for example, Page 1 of 20, Page 2 of 20, etc.) on each page of the table or listing. I searched on **SAS RTF PAGE X OF Y**. The results are in Figure 4.

There has been a lot of work done on this issue. I got five very relevant results. In Google searches, usually the first result is the best, because it has the most links to it. In this case, it was a matter of deciding which of the five solutions was best.
SOLUTION 1: MACRO-BASED

The first result was not suitable for my purposes. It is macro-based. Usually macro-based solutions are not good, because they present too many validation issues. I always look for a PROC, a function, or some really simple code.

For this problem, all I want to do is put Page x of y in my title. In Figure 5, the macro is not shown. Figure 5 just shows how to call the macro. This is complicated enough! To call the macro, you put a CALL EXECUTE inside a COMPUTE block and then pass the whole PROC REPORT in. This is clearly too complicated for what should be a fairly easy solution.

SOLUTION 2: INTERESTING, BUT NOT PRACTICAL

The solution shown in Figure 6 uses PROC TEMPLATE in a rather elegant manner. PROC TEMPLATE can apply styles to any report. Used as a global template, this method could put Page x of y on every report in the same place. The problem was no one at this particular organization used PROC TEMPLATE. Although it was the best solution, it was too much change for the organization, just to do Page x of y.

SOLUTION 3: OVERLY COMPLEX FOR THE PROBLEM

The approach shown in Figure 7 is clearly too complicated for this problem. It requires a lot of complex parameter checking before even starting to implement 'Page x of y'. Again, I am looking for ideally one line of code to solve this problem.
SOLUTION 4: TOO OLD
In Figure 8, the solution is perfectly valid. It is clean, easy to read, and requires only one line of code to implement. It is a way of embedding RTF field codes inside of SAS code. However, it mentions Word 97. That was a long time ago! I was hoping to find something a little newer.

Figure 8 (Tong, 2003)
```
"Page x of y" can be inserted into the footer using the following code:

footnote1 j=1 '{Page \field \*\fldinst PAGE \*MERGEFORMAT} { of \field \*\fldinst NUMPAGES \*MERGEFORMAT}';
```

Similarly, "Page y" can be inserted into the footer using the following code:

```
footnote1 j=1 '{ Page \field \*\fldinst NUMPAGES \*MERGEFORMAT}';
```

When the document is opened the page numbers may initially appear wrong, especially in Word 97. This is because Word has not refreshed the fields. Doing “Print preview” can solve this problem.

SOLUTION 5: WINNER
Figure 9 shows the best solution, given the particular organization that will use it. Although technically this requires an extra line of code to define the ODS ESCAPE-CHAR, this is as easy as it gets. Placing the ODS ESCAPECHAR in front of the automatic PAGEOF variable (available causes Page x of y to print on every page of the report. It is update-to-date, simple to read, simple to use, and very reliable.

Figure 9 (Mason, 2007)
```
SAMPLE CODE FOR RTF
ods escapechar = '\';
title 'This document will have page x of y';
   j=r 'Page \{pageof}';
ods rtf file='c:\test.rtf';
proc print data=sashelp.prdsale;
   run;
ods rtf close;
```

AGE CALCULATION SEARCH
Most pharmaceutical companies and CROs have a standard macro to compute age. This is the standard formula:

```
ageyear=floor((vdate - birthdt + 1) /365.25 );
```

Age in months in required for pediatric studies (infants < 2 years of age). We can easily extend the standard formula to return age in months by multiplying by 12.

```
agemos=floor((vdate - birthdt +1)*12 /365.25);
```

Examining this code, it is easy to see that it is an approximation. The formula for years adjusts for leap years by dividing by 365.25 instead of 365. The formula for months further approximates that by multiplying by 12, assuming all months have an equal number of days. For validation, it would be interesting to try a different method.

GOOGLE SEARCH: SAS AGE IN MONTHS
This is another case where a lot of work has been done. As shown in Figure 10, there are several solutions to choose from.
In this case, the most authoritative answer would be from the SAS knowledge base on http://support.sas.com: 

CALCULATING AGE WITH ONLY ONE LINE OF CODE

That's what I want. This article presents the standard solution under 'what doesn't work'. What they mean is this is an approximation, and does not take into account calendar months. For this, you either need some complicated programming, or the INTCHK function. INTCHK gives you the interval between any two dates, in days, weeks, months, etc. Here we use it for months

Figure 11 (SAS Institute, 2004)

```
intck('month',birth,somedate)
```

returns the number of times the first day of a month is passed between birth and somedate. An enhancement is needed to alter this into the number of times the same day of the starting month is passed. This simply consists of subtracting one month if the day number of somedate is earlier than the day number of birth. Although one could program this concept using a separate if-then statement, it can be calculated more concisely as a logical expression returning a 0 or 1 value. The 0 or 1 is then subtracted from the result of intck, as in the following example:

```
intck('month',birth,somedate) - (day(somedate) < day(birth))
```

This now gives exactly the correct number of months for any pair of dates.

A one-line solution

Converting months to years, we get:

```
age = floor((intck('month',birth,somedate) - (day(somedate) < day(birth))) / 12);
```

Substituting the variable names for my particular dataset, I used the following methods on a dataset with 3077 records from an integrated database of pediatric studies.

Production Method:  
```
floor((VDATE - BIRTHDT +1)*12 /365.25)
```

Validation Method:  
```
intck('month',BIRTHDT, VDATE) - (day(VDATE) < day(BIRTHDT))
```
Comparing the results, 68 of 3077 age calculations (2%) did not match. If we look at the results in detail in Figure 12, subject 10194, born on December 29, would reach his 3rd month birthday on March 29. The standard method computes the age as 2 months old. For such young infants, this is a big discrepancy.

**Figure 12**

<table>
<thead>
<tr>
<th>Subject Number</th>
<th>Date of Birth</th>
<th>Visit Date</th>
<th>Age (Months) New Method</th>
<th>Age (Months) Old Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>101048</td>
<td>26APR2004</td>
<td>27NOV2006</td>
<td>✔ 30</td>
<td>✔ 31</td>
</tr>
<tr>
<td>101053</td>
<td>23JUN2004</td>
<td>22NOV2006</td>
<td>✔ 16</td>
<td>✔ 17</td>
</tr>
<tr>
<td>101054</td>
<td>03JAN2006</td>
<td>09FEB2007</td>
<td>✔ 12</td>
<td>✔ 13</td>
</tr>
<tr>
<td>101069</td>
<td>10DEC2004</td>
<td>10MAR2005</td>
<td>✔ 15</td>
<td>✔ 14</td>
</tr>
<tr>
<td>101094</td>
<td>29DEC2005</td>
<td>29MAR2006</td>
<td>✔ 3</td>
<td>✔ 2</td>
</tr>
<tr>
<td>101136</td>
<td>13APR2006</td>
<td>1JAN2007</td>
<td>✔ 6</td>
<td>✔ 9</td>
</tr>
<tr>
<td>101140</td>
<td>14APR2006</td>
<td>12JAN2006</td>
<td>✔ 8</td>
<td>✔ 9</td>
</tr>
<tr>
<td>101140</td>
<td>14APR2006</td>
<td>13JAN2006</td>
<td>✔ 8</td>
<td>✔ 9</td>
</tr>
<tr>
<td>101143</td>
<td>01APR2006</td>
<td>31JUL2007</td>
<td>✔ 15</td>
<td>✔ 16</td>
</tr>
<tr>
<td>101197</td>
<td>02MAR2005</td>
<td>31JAN2007</td>
<td>✔ 20</td>
<td>✔ 21</td>
</tr>
<tr>
<td>101222</td>
<td>06FEB2006</td>
<td>06SEP2006</td>
<td>✔ 7</td>
<td>✔ 6</td>
</tr>
<tr>
<td>101233</td>
<td>19FEB2005</td>
<td>19SEP2006</td>
<td>✔ 19</td>
<td>✔ 18</td>
</tr>
<tr>
<td>101236</td>
<td>03NOV2005</td>
<td>03APR2007</td>
<td>✔ 17</td>
<td>✔ 16</td>
</tr>
<tr>
<td>101238</td>
<td>15AUG2005</td>
<td>14DEC2006</td>
<td>✔ 15</td>
<td>✔ 16</td>
</tr>
<tr>
<td>300001</td>
<td>28DEC2004</td>
<td>27JAN2006</td>
<td>✔ 12</td>
<td>✔ 13</td>
</tr>
<tr>
<td>300001</td>
<td>20DEC2004</td>
<td>28MAR2005</td>
<td>✔ 15</td>
<td>✔ 14</td>
</tr>
<tr>
<td>300002</td>
<td>19AUG2003</td>
<td>19SEP2006</td>
<td>✔ 36</td>
<td>✔ 37</td>
</tr>
<tr>
<td>300063</td>
<td>07AUG2004</td>
<td>06FEB2006</td>
<td>✔ 17</td>
<td>✔ 18</td>
</tr>
<tr>
<td>300120</td>
<td>31OCT2005</td>
<td>31JUL2007</td>
<td>✔ 21</td>
<td>✔ 20</td>
</tr>
</tbody>
</table>

**CONFIDENCE INTERVAL FOR A MEDIAN**

Another unusual request I received as a consultant was to compute the confidence interval for the median, as part of the descriptive statistics summary. I have been a statistician or programmer in the pharmaceutical industry since 1981, and I was asked to provide this exactly once. Fortunately, I had Google to rely on. Otherwise, I had no idea how to do this.

**GOOGLE SEARCH 1: SAS CONFIDENCE INTERVAL MEDIAN**

Once again, a surprising amount had been published on this topic. Although I was completely unfamiliar with the problem, several people worked on it. The search results are shown in Figure 13:

**Figure 13**
The title of the first result looked interesting: CONFIDENCE INTERVALS IN THE ANALYSIS AND REPORTING OF CLINICAL TRIALS. Additionally, it was NESUG paper, and therefore likely to contain SAS code I could use. Based on a quick read, however, the article did not look promising. Figure 14 summarizes my initial reaction:

**Figure 14 (Guangbin, 2003)**

I should have read a little further. Two pages later, the code for the solution was there:

```sas
ods output quantiles;
proc univariate cipctldf ...
```

FOLLOW-UP: MORE TARGETED SEARCH

Instead of reading on, however, I tried another search. It turns out that I needed to include the words PROC UNIVARIATE. In other words, I had to guess that if PROC UNIVARIATE computed the confidence interval for the mean, perhaps it also had an option to do so for the median. These results are shown in Figure 16:
The first result, a link to the PROC UNIVARIATE Version 9.2 documentation, is particularly useful. It has all of the options for computing both normal-approximation and distribution-free confidence intervals, as well as a sample program. There is a wealth of easy-to-use information here, a portion of which is shown in Figure 17:

Figure 17 (SAS Institute, 2010 [2])

INTERNATIONAL ENCODING METHODS

Companies often receive datasets created in different countries in using different encoding methods due to language differences. Encoding establishes the environment to process SAS syntax and to read and write SAS data sets. Encoding issues present difficult problems. First, without the right encoding, you cannot open the dataset. Secondly, these problems tend to be random. Therefore, the likelihood that someone has solved your particular problem is unfortunately low.

In this next example, I describe using a Google search to eventually open a data from a Chinese affiliate. Figure 18 shows the error message we got in trying to read the dataset:
The Chinese-speaking programmers contacted the group that created the dataset, but still could not resolve the issue. I then tried to solve the problem. Unfortunately I know very little about this area of SAS.

**GOOGLE THE ERROR MESSAGE**

When you know nothing about the problem, sometimes Googling the error message itself works. I tried searching **SAS SOME CHARACTER DATA WAS LOST DURING TRANSCODING**. Unfortunately, the search returned a lot of technical manuals. These were aimed at IT administrators dealing with global deployments, and not at all helpful to me.

**BROADER FOLLOW-UP SEARCH**

By this point, I was running out options. I next tried searching **SAS CHINESE ENCODING**. This returned some useful information; however, none of it worked.

Figure 19 (SAS Institute, 2003)
Suffice it to say after a lot of trial and error, I stumbled upon the solution. For some reason, using ASCIIANY as the encoding option opened the dataset.

Figure 20 (SAS Institute, 2010)

Why did ASCIIANY work, and the other options did not? The documentation in Figure 21 provides some of the answer, but honestly, I don’t know, and I really don’t care. I am glad to be done with this problem. It is unlikely to come up again in this particular form. Other encoding problems may well require different solutions. But this example is instructive in that shows how to use Google to solve weird SAS problems, even if you know very little about the issue.

Figure 21 (SAS Institute, 2010)

```
1   options nocenter;
2   libname mylib 'C:\test\drug\china';
3   NOTE: Libref IN was successfully assigned as follows:
4     Engine: V9
5     Physical Name: C:\test\drug\china
6   data eg;
7   set in.eg(encoding='asciiany');
8   run;
9   NOTE: Format VISF was not found or could not be loaded.
10  NOTE: Format ECORSLF was not found or could not be loaded.
11  NOTE: Format YNF was not found or could not be loaded.
12  NOTE: There were 1509 observations read from the data set IN.EG.
13  NOTE: The data set WORK.EG has 1509 observations and 27 variables.

ASCIIANY
ASCIIANY normally occurs when SAS detects that the session encoding and data set encoding are different. ASCIIANY enables you to create a data set that SAS will not transcode if the SAS session that accesses the data set has a session that encoding value of ASCII. If you transfer the data set to a machine that uses EBCDIC encoding, transcoding occurs.
```
CONCLUSIONS

In summary, you can leverage the experience of others by using Google searches to solve SAS problems. As the examples in this paper show, you can solve new problems you and really expand your skills. Using Google searches effectively requires some knowledge of SAS, and the ability to recognize the solution once you have found it. In general, I look for solutions from a reliable source (particularly support.sas.com), that are clear, concise, easy to implement, and do not require a lot of follow-up validation. These tend to be solutions based on PROCs or functions.

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.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

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

Michael Todd Work Phone: 908.672.5649
Nth Analytics Fax: 253.595.7413
12 Crimson King Trail Email: info@nthanalytics.com
Flemington, NJ 08822 Web: www.nthanalytics.com