How SAS® Uses SAS to Analyze SAS Blogs
Chris Hemedinger, SAS Institute Inc.

ABSTRACT

SAS® blogs (hosted at http://blogs.sas.com) attract millions of page views annually. With hundreds of authors, thousands of posts, and constant chatter within the blog comments, it is impossible for one person to keep track of all of the activity. In this paper, you learn how SAS technology is used to gather data and report on SAS blogs from the inside out. The beneficiaries include personnel from all over the company, including marketing, technical support, customer loyalty, and executives. The author describes the business case for tracking and reporting on the activity of blogging. You learn how SAS tools are used to access the WordPress database and how to create a "blog data mart" for reporting and analytics. The paper includes specific examples of the insight that you can gain from examining the blogs analytically, and which techniques are most useful for achieving that insight. For example, the blog transactional data are combined with social media metrics (also gathered by using SAS) to show which blog entries and authors yield the most engagement on Twitter, Facebook, and LinkedIn. In another example, we identified the growing trend of "blog comment spam" on the SAS blog properties and measured its cost to the business. These metrics helped to justify the investment in a solution. Many of the tools used are part of SAS® Foundation, including SAS/ACCESS®, the DATA step and SQL, PROC REPORT, PROC SGLOT, and more. The results are shared in static reports, automated daily email summaries, dynamic reports hosted in SAS/IntrNet®, and even a corporate dashboard hosted in SAS® Visual Analytics.

INTRODUCTION

WordPress is the world’s most popular blogging platform. There are different ways to use WordPress, including site hosting offered by WordPress.com or by using a third-party provider. At SAS, we host our own instance of the WordPress platform on-premises. The SAS run blogs.sas.com website includes the WordPress software (implemented using PHP) and a MySQL database as its back-end data store. Since we host this on SAS owned servers, we have direct access to the MySQL database.

The main purpose of the MySQL database is to serve as the operational data store for the content on the WordPress based blog. It is the system of record that tracks blog authors, posts, and comments. While the database was designed for use by the WordPress platform, at SAS we cannot resist capitalizing on this data to gain more insight into the operation and effectiveness of our blogging program.

Using WordPress data and SAS methods, gives us the following capabilities:

- Distribute daily blog activity reports to blog editors and marketers in email
- Measure engagement on blogs by adding data from reader comments and social sharing (gathered from Twitter, Facebook, and LinkedIn)
- Measure spam activity (a common affliction of blogging programs) and take effective action to end the activity
- Analyze how blog tags are used consistently (or not) to categorize posts by the hundreds of blog authors
- Create self-service reports that authors and editors can use to see historical blog activity and engagement
- Export this enriched data for use in other SAS systems, including a marketing asset management system and corporate dashboards built with SAS Visual Analytics.
HOW TO ACCESS THE WORDPRESS DATABASE WITH SAS

Because SAS is very good at accessing data from external sources, the mechanics of pulling WordPress data into SAS are simple. But there are two special challenges that we faced when beginning this exercise:

- Getting buy-in from IT
  The WordPress database is part of a production system that supports a customer-facing website. Understandably, our webmaster team and IT support require assurance that any other data access activities will not adversely affect the performance of that system.

- Preparing the WordPress data for analysis
  The WordPress database schema is highly normalized and designed for use by WordPress application code. Each blog comprises several tables that support it. With over 30 blogs hosted at blogs.sas.com, the data must be combined and transformed before we can generate useful reports.

ACCESSING THE WORDPRESS MYSQL DATABASE

SAS supports direct access to MySQL in two SAS/ACCESS products: SAS/ACCESS to ODBC and SAS/ACCESS to MySQL. At SAS, we use both methods.

We use ODBC when prototyping our reports with SAS Enterprise Guide and SAS for Windows. ODBC is an easy connection to set up from Windows. It requires the proper ODBC driver, available from the “Downloads” section of www.mysql.com. Note that the “bitness” (32-bit or 64-bit) of the ODBC driver must match the version of SAS that you plan to use. In our case, we use 64-bit SAS for Windows, and the 64-bit ODBC driver. (Remember, you can use 32-bit SAS Enterprise Guide even when using 64-bit SAS.)

Figure 1 shows an example of the database connection as configured as a “System DSN” in the Windows “Data Sources (ODBC)” facility.

![MySQL Connector/ODBC Data Source Configuration](image)

**Figure 1. Configuring the MySQL Connector for ODBC**

With this System DSN defined on the same machine as SAS for Windows, you can use this SAS code to assign the SAS library:

```sas
libname wpblogs odbc datarsc="wpblogs";
```

We use the SAS/ACCESS to MySQL method when accessing our SAS session on UNIX. Instead of an ODBC driver, you must have the proper MySQL libraries available on your UNIX system – and you must
“tell” SAS where to find them. In our case, we set an environment variable before invoking our SAS session so that SAS can find the proper libraries:

```bash
export LD_LIBRARY_PATH=/usr/lib/mysql
```

With this directive in place, this SAS statement will define the library that we need:

```sas
libname wpblogs mysql user = blogs_ro
   pw = secret
   server = extdbase
   port = 7888
   database = wpblogs;
```

Figure 2 shows an example of the library defined in SAS Enterprise Guide, along with a small subset of the tables for some of the blogs.

---

Figure 2. The WPBLOGS Library That Shows Several Tables per Blog

GETTING BUY-IN FROM IT

Before you can use either of the above methods to access the database, you must have the database credentials. At SAS, our IT department holds the keys to our production databases. And – as is their responsibility – they do not hand out these keys to just anyone for any purpose. We found these guidelines to be useful when requesting database access:

- Include a complete business justification in the request. If there is a business owner for the system in question, make sure that the business owner endorses the activity that you want to pursue. In our case, the manager of our blogging program was keen to learn what the WordPress data could provide, so the request was fully supported by the business.

- Ask for the least amount of access needed to satisfy the problem. In our case, we needed a database account that has just **read-only** access. This eliminates the possibility that our processes could corrupt the database. Also, it’s not important for our process to run against the primary
database instance; a mirrored or replicated instance would serve us just as well. This gives IT additional flexibility when setting up our access.

- Finally, limit the time and amount of processing that you perform when accessing the database. In our case, we extract the data for our analysis just once per day, early in the morning. The data extract operation takes less than two minutes.

Following the above guidelines, we were able to obtain permission from SAS IT fairly quickly. However, because the timing of our request coincided with an ongoing project to shore up our MySQL database servers with more mirroring/replication capability, it still took several months before our credentials were granted.

**PREPARING THE WORDPRESS DATA FOR ANALYSIS**

The WordPress database contains some tables that are “WordPress-global” in nature, such as the list of registered users. The database also contains some tables that are specific to each blog, with the same schema of 13 tables repeated \( n \) times – once for each blog that is housed in the system. With over 30 blogs hosted at blogs.sas.com, this results in nearly 500 tables in the database that support the WordPress blogs.

Our daily data extract breaks down into these steps:

- Pull the records from the `wp_users` table so that we can report on who are the authors and commenters.
- Discover and prepare the list of active blogs from the `wp_blogs` table.
- Iterate through the tables for each active blog, and append/join them to make analysis and reporting easier later in the process.

The WordPress database schema is documented at wordpress.org (see References section).

**Pulling User Data**

Extracting user data is a simple step, because all of the details are contained in a single table. However, we do not need all of the fields for our reporting, so we keep only the fields that we need. For example, there is no reason to bring the `password` field into our reporting data mart (indicated by the WPMART library).

```sas
/* get a list of users and their IDs */
data wpmart.users;
set wpblogs.wp_users
(keep=ID
 user_login
 user_registered
 display_name);
run;
```

As of this writing, we have over 900 registered authors in our WordPress system, including many SAS employees, SAS users, and a few famous seasonal characters such as Santa Claus, a vampire, and Cupid.

**Pulling the Blog List**

The list of SAS blogs is not static. New blogs are frequently added to our catalog as SAS develops content for different industries and regions. Therefore, in order to prepare for the next step of iterating through each set of blog tables to fetch the records needed for reporting, we first must query the list of available blogs. Some of these details are available in the `wp_blogs` table, while others are stored in a `wp_nn_options` table of metadata for each individual blog.

Once we have the blog list and count, we can use the SAS macro language to iterate through the options table for each blog to assemble the additional details, such as the “friendly” blog name.
/* get general info for each blog */
data blogs;
  set wpblogs.wp_blogs
    (keep=blog_id
     path registered
     last_updated public
     /* only external visible blogs */
     where=(public=1));
run;

/* build a macro variable for each blog */
/* to iterate through in next step */
proc sql noprint;
  select blog_id into :blog_id1 - :blog_id&sysmaxlong.
  from blogs
  order by blog_id;
  %let numBlogs = &sqlobs.;
quit;

%macro pullblogdata;
%global tableName;
/* pull additional details from blog-specific tables */
data allBlogs (keep=blog_ID name);
  length blog_ID 8;
  %do index = 2 %to &numBlogs.;
    /* build the name of the expected blog options table */
    %let tableName = %sysfunc(cat(wp_,%trim(&&blog_id&index.),_options));
    set wpblogs.&tableName.
      ( where=(option_name="blogname")
        rename=(option_value=name) )
    ;
    blog_ID = &&blog_id&index.;
    output;
  %end;
run;
%mend;
%pullblogdata;
The result of this combination looks like Figure 3.
Pulling and Transforming the Details for Each Blog

Each SAS blog has content that we want to be able to report using posts, comments, tags, and view metrics. Assembling this data requires separate steps similar to the process that we used to gather the blog titles.

```
%macro pullblogposts;
%global tableName;
/* repeat for each blog in our catalog */
%do index = 2 %to &numBlogs.;
data postbuffer&index. (keep=blog_ID ID post_author post_date post_title post_name guid comment_count post_status post_type);
length blog_ID 8;
/* build the blog posts table name for this blog */
%let tableName = %sysfunc(cat(wp_,%trim(&&blog_id&index.),_posts));
blog_ID = &&blog_id&index.;
/* extract all records, but just a subset of fields */
set wpblogs.&tableName. ( keep=ID post_author post_date post_title post_name guid comment_count post_status post_type );
run;
%end;
/* append all posts tables into a single POSTS table*/
data wpmart.posts;
set postbuffer2-postbuffer&numBlogs. ;
run;

/* delete the intermediate tables */
proc datasets lib=work nolist;
delete postbuffer2-postbuffer&numBlogs ;
quit;
%mend;
%pullblogposts;
```
Figure 4 shows the result of this extraction and concatenation, after combining with additional data sources.

CREATING DAILY ACTIVITY REPORTS

One of the key objectives of this initiative was to help SAS content marketers stay informed about our blog content. With thousands of blog articles published each year, it’s difficult keep up with all of the activity simply by visiting blogs.sas.com and perusing the site. So, one of our first projects was to create a daily and weekly email summary that pushes the information to the inboxes of any SAS staffer who finds the information helpful.

Using the FILENAME EMAIL method, SAS can be automated to send reports to email recipients. We created a special 7-day summary report that contains a rolling 7-day activity stream of blog posts and comments.

Figure 5 shows an example of what our subscribers see in their inbox, either daily or weekly depending on their preferences.

Figure 5. An Email with the 7-Day Summary of Blog Activity

Upon opening the email, the reader can review the details for blog posts that have been published in the past 7 days, any blog comments that have arrived during that period, and any blog posts that are scheduled for publication in the coming days or weeks, Figure 6 shows a portion of the message.
Figure 6. An Example of the 7-Day Blog Activity Report

By pushing this via email, SAS staff can be apprised of timely content to promote to customers and keep themselves informed about what our SAS experts are currently talking about.

The complete code is too long to share here, but here is the segment that determines the email distribution list and sets up the EMAIL destination.

```sas
/* reporting on past 7 days */
%let cutoff = %sysfunc(today()) - 7;
/* determine distribution list */
/* if today is MONDAY, use WEEKLY list */
/* else use DAILY list */
%let dist = %sysfunc(ifc(%sysfunc(today(), weekday1.)=2, weekly, daily));

/* Read email recipients from text file */
filename dl "/u/sascrh/wp/distlist_&dist..csv";
data distlist;
infile dl dsd;
length email $ 50;
input email;
run;

/* populate distribution list */
proc sql noprint;
select cat("'",trim(email),"'")
into :toList separated by ' ' from distlist;
quit;

/* Set up destination fileref */
/* use this in ODS statement */
FILENAME OUTPUT EMAIL
SUBJECT = "blogs.sas.com: &postCount posts," ||
"&commentCount comments " ||
"%TRIM(%SYSFUNC(putn(&cutoff., NLDATE20.))) - " ||
"%TRIM(%SYSFUNC(today(), NLDATE20.)))"
FROM = "Chris Hemedinger <chris.hemedinger@sas.com>"
TO = (&toList)
type='text/html'
CT = 'text/html';
```
Note that FILENAME EMAIL relies on the value of EMAILSYS and other system options to control how email is sent. In our case, we use SMTP and our SAS corporate email server to deliver our messages.

MEASURING ENGAGEMENT WITH SOCIAL MEDIA SHARES

When you view an article on the blogs.sas.com website, a series of social-sharing widgets show how the post has been shared on social media. Figure 7 shows an example of an article title with the social-sharing widgets.

![How to split one data set into many](image)

**How to split one data set into many**

Chris Hemedinger | JANUARY 26, 2015

![Social share counts](image)

**Figure 7. Example of a Blog Post with Social Media Widgets**

The various widgets actually query the different social media services in "real time" to show the numbers to display. The queries use JavaScript code to call REST-based web services that each of the social media sites offer precisely for this purpose.

These social media counts are not in the WordPress database, but we can use SAS to gather these metrics in batch. In our daily data preparation steps, we use PROC HTTP to invoke these REST-based services to find the social share counts for each blog post URL. Each of the social APIs returns a short JSON response, which is simple to parse using SAS functions such as SCANOVER or PRXMATCH.

Here is an example program that processes a single URL to find the number of shares on LinkedIn.

```sas
%let blog = http://blogs.sas.com/content/;
%let url= sasdummy/2015/01/26/how-to-split-one-data-set-into-many/;

/* temp holding area for LinkedIn response */
filename li temp;

/* call the LinkedIn API */
proc http
  method='GET'|
  out=li;
run;

/* use RegEx to gather the "count":n value */
data liresult(keep=lishares url);
  length line $ 1000 lishares 8;
  length url $ 500;
  url = "&url.";
  infile li;
  input line;

  if _n_ = 1 then do;
    retain li_regex;
    li_regex = prxparse("/\""count\""\:\([0-9]*)/\");
  end;
```

---

9
end;

position = prxmatch(li_regex,line);

if (position ^= 0) then
do;
call prxposn(li_regex, 1, start, length);
lishares = substr(line,start,length);
put 'lishares=' lishares;
end;

run;

Here is a partial result from the SAS log:

lishares=4
NOTE: 1 record was read from the infile LI.
The minimum record length was 171.
The maximum record length was 171.

In our daily process, we use the SAS macro language to process similar queries for Facebook and Twitter as well as LinkedIn, iterating over all of the SAS blog articles. The result, as shown in Figure 8, allows us to see the most socially shared posts over time.

![Social sharing of 12-month of posts](image)

Figure 8. An Example Report of Social Sharing on SAS Blogs

Because the REST-based calls are time-consuming, we limit our queries to cover just the past 12 months of blog posts. Even with this limit, the program generates thousands of API calls, each of which requires a response from an Internet-based service. The entire process accounts for the bulk of the time spent on data preparation and reporting: usually 25 minutes of the typical 30-minute SAS job that runs each day.

MEASURING SPAM ACTIVITY

Ever since the invention of blogs, there has been spam. Spam comments contain links that “spammers” seek to add to your website in order to increase the search engine ranking of their own sites. Spam is a scourge to those of us who create and promote good content, but it’s a lucrative industry for spammers.

For blog sites, spam arrives in the form of comments on published articles. Our SAS blog authors moderate all comments before approving them, so spam rarely gets published. Moderating comments, especially when there is a high volume of spam “noise”, is a time-consuming activity.

Our SAS blogs have always been the target of spam, but in 2013 the spam activity increased significantly. Because we already have a robust SAS based blog reporting system in place, it was easy
for us to measure exactly how many more spam comments were being submitted. Figure 9 shows the spam trends in 2013, as counted by those comments that blog authors marked as spam during moderation.

Figure 9. The Spam on blogs.sas.com Before a Technology Solution Was Implemented

The graph was produced using PROC SGPLOT. As the data points show, SAS blogs were receiving more than 1000 spam comments per week in early 2013. Using these data as justification, we were able to work with our SAS IT staff to add some urgency to the adoption of a solution.

In mid-2013 SAS adopted the use of Akismet, an industry-standard service for WordPress and other content platforms. The result was dramatic – it was like turning off a spigot. The flood of spam was reduced to a trickle.

We still use our SAS process to report on spam, but its purpose has shifted to see how Akismet is performing as a spam deflector. Figure 10 shows an annotated bar chart (produced with PROC SGPLOT) that shows how much spam is blocked each day, and the small percentage that still leaks through.
CREATING DYNAMIC REPORTS WITH SAS/INTRNET

While our daily SAS job creates a useful data mart and a set of static reports, the people who use our blog reports often require dynamic views of the data. For example, blog editors want to see the metrics for just the blog titles that they monitor, while blog authors are keen to see the summaries of their complete activities across all of the blogs. Because of the high volume of data (nearly 10,000 posts written by nearly 1,000 authors over 30 different blog titles), this information is not easily surfaced in static reports.

SAS/IntrNet is a set of tools that allow you to “web-enable” your SAS programs so that they can be run from a browser. Although it does not represent the latest in SAS technology, SAS/IntrNet is a workhorse of an application that still serves its purpose. At SAS, we have several internal instances of SAS/IntrNet that we use to build dynamic web-based reports and mini applications. Any SAS employee can request folder space on the SAS/IntrNet program repository and thus create reports for the entire company.

For our blog-related reports, we use the SAS/IntrNet Application Dispatcher to execute SAS programs with parameters that are determined as the end-user clicks. The result is a report that contains just the
records that the user wants to see, with data sliced by blog author, blog title, and other attributes. Figure 11 shows two example reports: one with a summary of blog authors and the other with a summary of blog titles. If the user clicks on a hyperlink in the first column, new reports are generated to show more details about the specific posts created by the selected author or within the selected blog.

Figure 11. Two Dynamic SAS/IntrNet Reports: One for Authors and One for Blogs

CREATING A CROSS-BLOG TAG REPORT

Blog tags represent a layer of metadata that makes it easy to search for and group-related blog topics. Many blog sites, including blogs.sas.com, feature "tag clouds" that show the most popular tags that are used across the blog articles (example in Figure 12).

Figure 12. Tag Cloud from blogs.sas.com. “Analytics” Is a Popular Tag, Which Is Not a Surprise!

The tables that contain Tags, also known as “terms” within the WordPress database, represent the database’s most complex table relations. Because each tag can apply to more than one article, there is a one-to-many relationship between tag and post. Further, each tag fits into a taxonomy – that is represented in yet another table. If you want to generate a report that shows all blog posts that reference a particular tag and also include a few details about the post for context, you need to execute a six-way join across the tables that contain the various pieces. Here is a snippet of that PROC SQL code that finds all blog posts that mention a given tag:

```sql
/* with PROC SQL ... */
/* The six-way join that combines */
/* all of the fields needed to generate our report */
FROM WPBLOGS.TERMS t1,
WPBLOGS.TERM_RELATIONSHIPS t2,
WPBLOGS.TERM_TAXONOMY t3,
WPBLOGS.POSTS t4,
WPBLOGS.BLOGLIST t5,
WPBLOGS.POSTVIEWS t6
WHERE (lower(t1.name) CONTAINS lower("&tag.") and
  t4.post_status in ('publish', 'future') and
  t1.blog_ID = t2.blog_ID AND
  t1.blog_ID = t3.blog_ID AND
  t1.term_id = t3.term_id AND
  t3.term_taxonomy_id = t2.term_taxonomy_id AND
  t1.blog_ID = t4.blog_id AND
  t2.object_id = t4.ID AND
  t1.blog_ID = t5.blog_id and
  t6.blog_id=t4.blog_id and t6.post_id = t4.id)

Because our content marketers are not likely to run this SAS program themselves each time they want to
search for content, we built a simple web page to drive a SAS/IntrNet report. Figure 13 shows the form
that we host on the SAS intranet, accessible to all SAS employees.

![Figure 13. Simple Front End for the Blog Tags Report](image)

Here is a sample of the HTML code behind this simple form. SAS/IntrNet users might recognize the
mechanism of specifying the Application Broker and standard directives such as the SAS program to run.

```html
<form id="tagform" action="http://sww.sas.com/sww-bin/broker94"
  onsubmit="return validateForm()" method="post">
  <fieldset>
    <legend>Tag to search for</legend>
    <input type="hidden" name="_service" value="appdev94en">
    <input type="hidden" name="_program" value="blogtest.blog_tagged.sas">
    <b>Search term:</b> <input id="tag" type="text" name="tag">
    <input type="submit" value="Run Report">
    <br>
    <input type="checkbox" name="dest" value="CSV">Output to CSV
  </fieldset>
</form>
```

The value of the tag field is passed to the blog_tagged.sas program as a SAS macro variable. The
program runs the query and returns the result as an HTML table (populated using PROC REPORT).
Figure 14 shows an example result from a recent search for “admin”. Note how the “admin” partial text is
used to match with the “SAS Administrators” tag. The program also accepts an optional “CSV”
parameter. If selected, the result is streamed as a CSV file that the user downloads via the browser.
OTHER SYSTEMS THAT CONNECT TO SAS BLOGS DATA

Blogs at SAS make up just one part of a large marketing and support ecosystem, all of which is tracked and measured using SAS tools. The processes described in this paper produce an enriched data set of blog activity and social engagement; the data files are then used as inputs into other systems.

EXPORTING DATA FOR USE IN SAS VISUAL ANALYTICS

It should come as no surprise that SAS uses a SAS Visual Analytics dashboard to help track the performance of many SAS website assets, including the blogs site. SAS Visual Analytics, backed by the SAS LASR™ Analytic Server, can accept data sources in several formats. For greatest flexibility, our process creates a single denormalized table that is exported in CSV format. (See Figure 15) This CSV file can then be copied to a designated autoload location, where the SAS LASR Analytic Server can load it into SAS Visual Analytics immediately.

INTEGRATION WITH ASSET MANAGEMENT SYSTEMS

As part of its marketing operations, SAS maintains a digital asset library that links to whitepapers, videos, and other online marketing materials that can be used within marketing campaigns. Blog articles are among those assets. A successful digital asset library thrives on metadata, which can be used to find relevant content quickly. In addition to the blog title and subject area, we can use the article tags to supplement the metadata in the system.

To accomplish this, we needed to navigate the complicated table relationships of WordPress terms. This time there is a two-step process. First, use PROC SQL to gather the list of tags used on all articles (which can have a one-to-many relationship). Then, use a DATA step to output just one record per article, each with a comma-separated list of tags. The SAS team that maintains the digital asset library can then easily import these records and link the metadata with the asset. Here is the segment of code that performs this work:
/* assemble complete list of tags and their ids */
proc sql;
  create table work.posttags as
  select t1.blog_ID,
         t2.object_id AS post_ID,
         t3.term_id,
         t1.name AS term
  from WPMART.TERMS t1,
       WPMART.TERM_RELATIONSHIPS t2,
       WPMART.TERM_TAXONOMY t3
  where (t1.blog_ID = t2.blog_ID
         and t1.blog_ID = t3.blog_ID
         and t1.term_id = t3.term_id
         and t3.term_taxonomy_id = t2.term_taxonomy_id) and
         t3.taxonomy = 'post_tag'
  /* Sort by blog_ID and object_ID so that      */
  /* FIRST-dot and LAST-dot processing can work */
  /* in the next DATA step                      */
  order by t1.blog_ID,
           t2.object_id;
QUIT;

/* coalesce the tags into comma-sep list */
/* and output just one record per post */
data tagslist (keep=blog_id post_id tags);
  set posttags;
  length tags $ 500;
  by blog_id post_id;
  retain tags;
  if first.post_id then
    tags="";
  tags = catx (',', tags, term);
  if last.post_id then
    output;
run;

WHAT MORE CAN WE DO?
Even while we consider our blog reporting system to be fairly rich already, there is still the potential to do much more using SAS tools. Some ideas include:

- Use SAS Text Analytics to categorize blog content, and perhaps even apply sentiment analysis to the blog comments provided by readers
- Analyze the transactional activity in WordPress. In addition to published posts, it is possible to see the revision history of posts before they were published. It might be useful to analyze the WordPress activity of our most effective authors to inform our own best practices.
- Combine the operational WordPress data with the website analytics about site traffic and visitors. These could help to identify the best performing content, as well as help us understand what content appeals to which customers.

CONCLUSION
By using SAS to analyze our WordPress blog data, we are able to squeeze more insights from our blogging activity and use our blog content in new ways. The project began (as many do) as a grassroots
“skunkworks” activity, but over time the data has proved to be essential to the blogging program and to the SAS overall content marketing activity.

REFERENCES


ACKNOWLEDGMENTS

A special thanks to Alison Bolen of SAS for her early work in establishing the SAS blogging program, and especially for her support and encouragement toward creating these SAS based analyses. The blogging program has grown tremendously, and these reports help us to quantify the growth and the positive impact to our organization. Also, thank you to Brandy Mann, our current WordPress guru, for her continued advocacy of our blogging program and metrics.

CONTACT INFORMATION

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

Chris Hemedinger
SAS Institute Inc.
chris.hemedinger@sas.com
http://blogs.sas.com/sasdummy

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 trademarks of their respective companies.