

WHAT WERE WE TALKING ABOUT AT THOSE SAS® CONFERENCES, OR LET'S MAKE SOME TAG CLOUDS

Chang Y. Chung, Princeton University, Princeton, NJ
John King, Ouachita Clinical Data Services, Mount Ida, AR

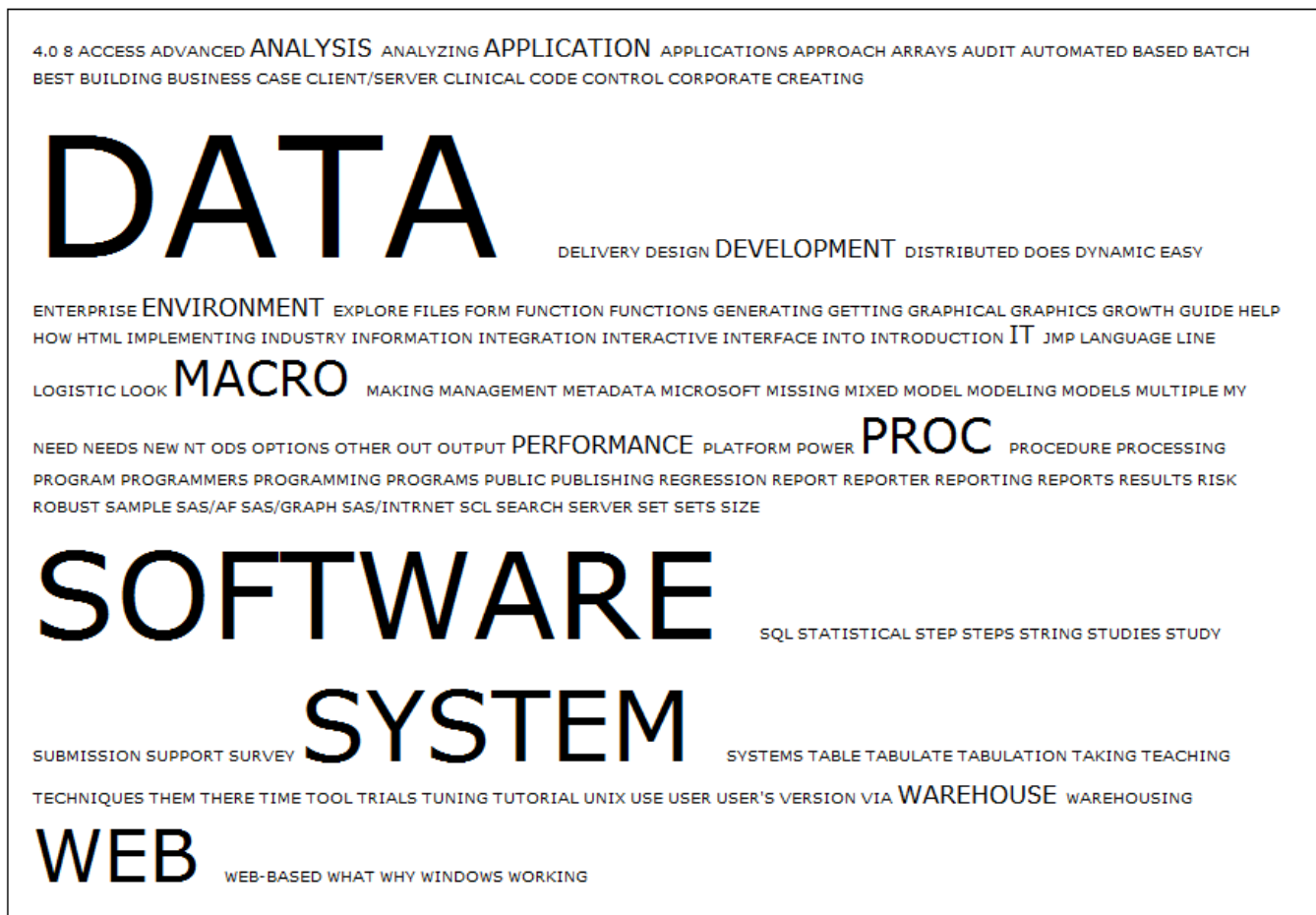
ABSTRACT

A tag cloud is a bunch of tags or words displayed nicely. More popular or important words are highlighted using different colors or font sizes. Popularized by Flickr and other Web 2.0 web sites, tag clouds can be informative and useful. This paper demonstrates making tag clouds using nothing but Base SAS®. The keywords come from the titles of all papers presented at two international SAS conferences: the 25th SAS Users Group International Conference in 2000 (SUGI 25) and the SAS Global Forum 2010.

A PICTURE IS WORTH A THOUSAND WORDS

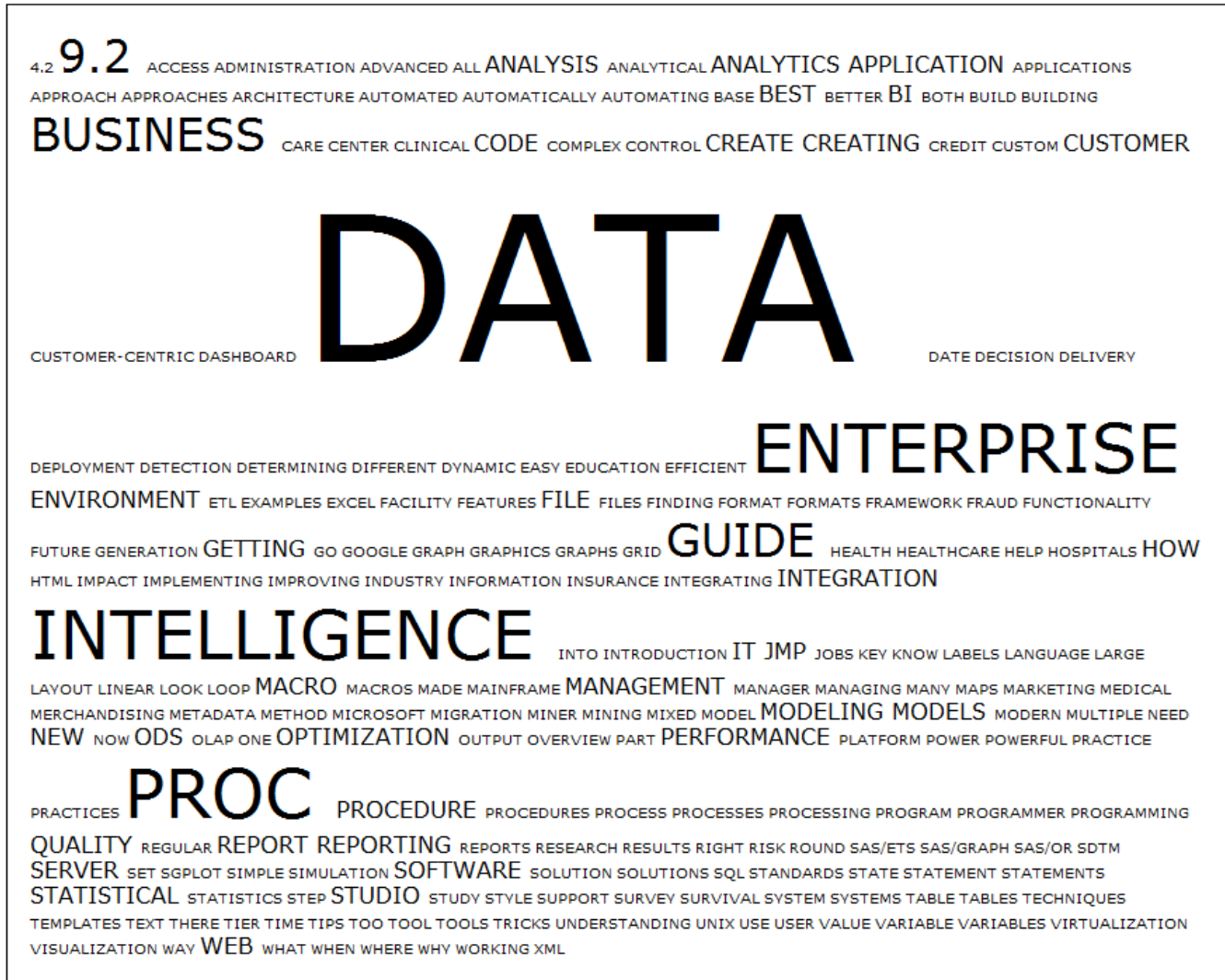
Then are two pictures worth two thousand words? A deep question! In any case, here they are. First from the 25th SAS Users Group International Conference (SUGI 25) held in Indianapolis, IN, April 9-12, 2000.

Figure 1 Popular Title Keywords at SUGI 25



And here is another one for the recent SAS Global Forum 2010 held in Seattle, WA, April 11-14, 2010.

Figure 2 Popular Title Keywords at SGF 2010



We have been talking about DATA for sure, but it seems that our conversation has not always been about the same things.

The full Base SAS source code that produced Figure 2 above is found in Appendix A. In the rest of this paper, we explain the SAS code section by section. The code is written and submitted in an interactive Display Manager (DM) session on locally installed Windows (32bit) SAS 9.2 (TS1M0).

WEB SCRAPING

The titles of the conference papers are “harvested” from Lex Jansen’s wonderful SAS conference paper site (<http://lexjansen.com/>). In Lex’s web pages, each paper title is written in a single line of HTML with distinct id properties, which makes extracting the title easy.¹

¹ We considered two alternative data sources. The SAS Conference Proceedings in Adobe Portable Document Format (PDF) are readily available. But in general, these PDF files are for printing and not for further processing. SAS Institute user support site also has conference proceeding pages. Unfortunately, these are coded with a dirty and irregular HTML, which makes web scraping very difficult, if not impossible.

The first data step in Appendix A generates a data set, TITLES, by reading Lex's web page using FILENAME URL. The ENCODING= option is necessary because the site is encoded in UTF-8. The ENCODING= option does a reasonable job translating the characters to my local encoding.

A title appears in a line that has an id attribute "sgf2010." (including the dot) but not "hilite". Once we know that a given input line contains a title, we extract it with powerful Perl regular expression functions.² Once we have all the titles, we let go of the file handle.

INFORMAT FOR TABLE LOOKUP

Not all the words in the title are equally worthy. Some words like "A" and "AN" convey little information. The word, "SAS", appears so many times that if we allow it, the frequency distribution becomes severely skewed. To check and drop these "ignore" words, we need a convenient way of looking them upon a list. SAS offers many different ways to implement table lookups. When you have a moderate amount of items in a lookup table (say less than three thousand), using (IN)FORMAT is a fast and easy solution. We write a numeric informat, IGNORED, which generates 1 if an input word is one of the ignore words or 0 otherwise.

FROM TITLE INTO KEYWORDS

The next DATA step, WORDS, takes titles and splits each word out into its own observation. This is done using the powerful SCAN function. It allows custom delimiters, which makes it easy for us to handle punctuation marks like colons (:) and commas (.). When we first wrote the data step, the SCAN function was called in two different places in the code, making it difficult to maintain. Later, we re-factored them into a single link destination. Notice that we also do UPCASE'ing to normalize words.

Each word in the title is checked to see if it is among the ignore words. This is done by calling the INPUT function with our IGNORED informat. If it returns 1 then we don't output the word.

COUNTING AND CATEGORIZING

Once we have words data set, then we use PROC FREQ to count the words. Since there are so many words with very low frequencies, we arbitrarily decide to process only those words with three or more occurrences. This is easily done by applying the WHERE= DATASET OPTIONS to the OUT= dataset in the TABLES statement.

The row frequencies may range from three to some maximum number. To color and size the keywords, however, we need to have a limited number of frequency levels. We prepared ten levels, so we need to scale the raw frequencies. Using the coding technique known as "Double DoW³", it is simple to find out the maximum frequency and use that number to scale the raw frequencies as shown in the DATA step sizes.

FINALLY, HTML OUTPUT

We are now ready to write out an html file with a tag cloud. The DATA _NULL_ step reads in the HTML5 source in-lined under the CARDS4 statement. It is a simple template of a sort with two placeholders (#pageTitle# and #spans#). The DATA step read in this in-lined source line-by-line and outputs to an html file. If the input line has a placeholder, however, the placeholder is replaced with the real content before the line is output. The first placeholder (#pageTitle#) is replaced with a title that describes web page. It is used in the HTML header as well as in the body. The second placeholder (#spans#) however is replaced with keywords wrapped with a pair of

² We also extract the paper id as well.

³ Another way of doing the same thing is to "interleave a dataset itself." See Schreier (2003) for explanation.

open and close HTML SPAN tags. The opening SPAN tag has an appropriate CSS style class attribute for styling. The TRANSTRN function makes placeholder replacing straightforward.

CHECK

Voilà! A tag cloud is made. You can see it in any modern browser. In the SAS Display Manager (DM) interactive environment, you can launch your default browser by using the X statement.

SUMMARY

This paper shows how to create a simple tag cloud using nothing but Base SAS. SAS's powerful language features like Perl Regular Expressions and PROCs make it easy to implement the end-to-end process: from scraping web pages, to handling complicated strings, to generating fun and informative graphics like tag clouds.

REFERENCES

Schreier, Howard (2003). "Interleaving a Dataset with Itself: How and Why" Paper CC002 in the *Proceedings of North East SAS Users Group Conference* (NESUG 16). <<http://www.howles.com/saspapers/cc002.pdf>>.

ACKNOWLEDGEMENTS

Many thanks to Lex Jansen for providing the comprehensive SAS Conference Paper contents and links page through his web site <http://lexjansen.com>. Without his effort, scraping paper titles would have been much harder and more time-consuming. Thanks also to Howard Schreier and Dawn Koffman for taking time to read earlier versions of this paper and for their thoughtful comments and suggestions!

CONTACT INFORMATION

Chang Y. Chung
Princeton University
#216 Wallace Hall
Princeton, NJ 08540
cchung@princeton.edu
<http://changchung.com/>

John King
Ouachita Clinical Data Services, Inc.
24 Loblolly Circle
Mount Ida, AR 71957

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.

APPENDIX A

```
/*-- making a tag cloud of keywords in the SAS Global Forum 2010 paper titles
   by chang y chung and John King on 2010-04-30 -----*/

x cd "c:\tagCloud";

/*-- read sgf2010 page from lex^s web site -----*/
%let amp = %nrstr(&);
%let param = s=sugi&amp.c=sugi&amp.x=;
%let lex = http://lexjansen.com/cgi-bin/xsl_transform.php?&param;

filename sgf2010 url "&lex.sgf2010" encoding="utf-8";
data titles;
  infile sgf2010 lrecl=10000 trunccover;
  input line $10000.;

  sgf = prxmatch("|sgf2010\.|", line);
  hilite = prxmatch("|hilite|", line);
  if sgf and not hilite;

  id = substr(substr(line,sgf), 9, 8);
  title = prxchange("s|.*>([^\>]+)</a><br />.*$|$", 1, line);
  keep id title;
run;
filename sgf2010 clear;

/*-- for table lookup -----*/
proc format;
  invalue ignored
    "SAS", "THE", "AND", "TO", "A", "IN", "USING",
    "FOR", "OF", "WITH", "YOUR", "AN", "ON",
    "BY", "YOU", "FROM", "IS", "AT", "OR", "DO", "GET",
    "NOT", "MORE", "AS", "I", "BE", "US", "VERY", "WE",
    "YES", "YET", "YOU'RE", "THAT", "SO", "WHAT'S",
    "WORKSHOP", "OUR", "IT'S", ".", "IT'S", "-",
    "THAT'S", "THEIR", "SUCH", "CAN", "BETWEEN",
    "WHICH", "I'VE", "ITSELF" = 1
  other = 0;
run;

/*-- split titles into words skipping ignore words -----*/
data words;
  set titles;
  length word $20;
  i = 1;
  link getWord;
  do while (not missing(word));
    if not input(word, ignored.) then output;
    i + 1;
    link getWord;
  end;
  keep id word;
  return;
getWord:
  word = upcase(scan(title, i, " :,(?)@™-!=""", "r"));
  return;
run;

/*-- get frequencies and scale -----*/
proc freq data=words;
  tables word/out=freqs(keep=word count where=(count>=3));
run;

%let maxSize = 10;
data sizes;
  retain maxCount 0;
  do until (endl);
    set freqs(keep=count) end=endl;
```

```

    maxCount = max(count, maxCount);
end;
do until (end2);
    set freqs end=end2;
    size = ceil(&maxSize * count / maxCount);
    output;
end;
run;

/*-- generate a web page with the tag cloud -----*/
data _null_;

    infile cards;
    file "csgf2010.htm";

    input;

    if index(_infile_, "#spans#") then do until (end);
        set sizes end=end;
        length span $200;
        span = transtrn(transtrn(
            "<span class='s#size#'>#word# </span>",
            "#size#", trim(put(size,best.-1))),
            "#word#", trim(word));
        len = length(span);
        put span $varying. len @@;
    end;

    else if index(_infile_, "#pageTitle#") then do;
        length line $200;
        line = transtrn(_infile_, "#pageTitle#",
            "What We Talked Most About in sgf2010");
        len = length(line);
        put line $varying. len;
    end;

    else do;
        put _infile_;
    end;
cards4;
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>#pageTitle#</title>
    <style>
        body { margin: 2em; font-family: verdana; }
        h1 { font-weight: normal; font-size: 1.5em; }
        .cloud { padding: 1em; border: solid 1px #000000 }
        .s1 { font-size:0.5em; }
        .s2 { font-size:1em; }
        .s3 { font-size:2em; }
        .s4 { font-size:3em; }
        .s5 { font-size:4em; }
        .s6 { font-size:5em; }
        .s7 { font-size:6em; }
        .s8 { font-size:7em; }
        .s9 { font-size:8em; }
        .s10 { font-size:9em; }
    </style>
</head>
<body>
    <h1>#pageTitle#</h1>
    <div class="cloud">
        #spans#
    </div>
</body>
</html>
;;;
run;

/*-- check -----*/
x csgf2010.htm;

```