ABSTRACT
One of the most seldom used (and least expounded upon) of all SAS® options, #BYVAL deserves more recognition that it typically receives. Although its functionality is limited to the context of titles and footnotes, creative use of #BYVAL can contribute simple solutions to both common and complex problems. Possibly nowhere is this better demonstrated than in easing the pain associated with creating patient profile listings without resorting to data _null_.

INTRODUCTION
This paper briefly describes #BYVAL, beginning with a few basic examples, while moving towards a demonstration of it’s usage in the context of several tasks that would be considerably more difficult if not for the availability of this valuable option.

INFATUATION
Beginning with a concise understanding of the basics of any SAS® language element is always a good idea prior to implementation. Fortunately, the syntax and usage of #BYVAL are among the simplest within all of SAS® to grasp and apply (which again begs the question regarding its sparse usage!). The following example shows some of the most basic handling of this option...

```sas
options nobyline center nonumber nodate;

proc sort data=sashelp.class out=temp;
  by sex;
run;

title1 'Listing of Subject Characteristics';
title2 'Sex=#byval(sex)';

proc report nowd data=temp;
  by sex;
  column name height weight;

  define name / "Patient Name" width=30;
  define height / "Height" width=15;
  define weight / "Weight" width=15;
run;
```

Including the variable SEX in the BY statement within a procedure step makes its dynamic value available in either a title or footnote statement. It might be worth pointing out that, in contrast to the SAS® macro language, #BYVALs resolve regardless of whether single, double, or no quotes at all are used around the title or footnote within which they are situated. Notice the necessity of the NOBYLINE option to prevent the PROC step from automatically generating an additional title line (precisely, a #BYLINE) containing the variable label (or name, if the label is unavailable) followed by its current BY data value. The RTF output for the first page generated by this sample is shown below.
From here, it's a small adjustment towards accumulating additional `#BYVALs` by variable name, or even using `#BYVALn`, where `n` is an identifier taken from the order of the variable in question in the given BY statement.

```sas
proc sort data=sashelp.class out=temp;
  by sex age;
run;

title1 "Listing of Subject Characteristics";
title2 "Sex=#byval1, Age=#byval2";
proc report nowd data=temp;
  by sex age;
...;
```

To make things even more dynamic, one can specify the labels of the BY variables themselves using the `#BYVAR` option in conjunction with `#BYVAL`.

```sas
options nobyline nocenter number nodate;
proc format;
   value agegrp
     1-12 = "Elementary School"
     13-14 = "Middle School"
     other = "High School";
run;
```

In keeping with the simple and forgiving nature of the `#BYVAL` option, mixing and matching is permitted.

```sas
options nobyline nocenter number nodate;
proc format;
   value agegrp
     1-12 = "Elementary School"
     13-14 = "Middle School"
     other = "High School";
run;
```

When a BY variable has a format attached, `#BYVAL` is attentive to groupings based on the formatted values rather than the non-formatted ones.

```sas
from here, it's a small adjustment towards accumulating additional #BYVALs by variable name, or even using #BYVALn, where n is an identifier taken from the order of the variable in question in the given BY statement.

```sas
proc sort data=sashelp.class out=temp;
  by sex age;
run;

title1 'Listing of Subject Characteristics';
title2 'Sex=#byval1, Age=#byval2';
proc report nowd data=temp;
  by sex age;
...;
```

To make things even more dynamic, one can specify the labels of the BY variables themselves using the #BYVAR option in conjunction with #BYVAL.

```sas
options nobyline nocenter number nodate;
proc format;
   value agegrp
     1-12 = "Elementary School"
     13-14 = "Middle School"
     other = "High School";
run;
```

In keeping with the simple and forgiving nature of the #BYVAL option, mixing and matching is permitted.

```sas
options nobyline nocenter number nodate;
proc format;
   value agegrp
     1-12 = "Elementary School"
     13-14 = "Middle School"
     other = "High School";
run;
```

When a BY variable has a format attached, #BYVAL is attentive to groupings based on the formatted values rather than the non-formatted ones.
proc sort data=sashelp.class out=temp;
   by age;
   format age agegrp.;
run;

title1 'Listing of Subject Characteristics';
title2 '#byvar1=#byval1';

proc report nowd data=temp missing;
   by age;
   column name height weight;
   define name   / display  width=15;
   define height / display  width=15;
   define weight / display  width=15;
run;

COMPATIBILITY ISSUES
Since titles and footnotes are persistent through multiple procedure steps if not replaced, the value of #BVVALn is
dynamic across repeated PROCs. This is only desirable when intentional.

options nobyline nocenter number nodate;

proc sort data=sashelp.class out=temp;
   by sex name;
run;

title1 'Listing of Subject Characteristics';
title2 'Subject Name is #byval2';

proc print data=temp;
   by sex name;
run;

proc sort data=temp;
   by name sex;
run;

proc print data=temp;
   by name sex;
run;

On the even less convenient side of things, it's interesting to note that, at first glance, the following two statements
might be expected to yield identical titles when the second BY variable used in a procedure step has the label 'Age'.

   title2 "%sysfunc(translate(%str(#byvar2),%str(a),%str(A)))";
   title3 "%sysfunc(translate(%str(Age),%str(a),%str(A)))";

However, virtually all such macro manipulations (e.g., %upcase(#byvar(sex))) are unsuccessful when applied to #BY
options, since macro processing occurs prior to the actual value of a #BY being resolved. I mention 'virtually',
because perhaps the only macro function manipulation possible would involve shifting the BY variable reference
number itself...

   title2 "%sysfunc(translate(%str(#byvar2),%str(1),%str(2)))";

Such is most likely of little use, but might have some limited utility in a demented sort of macro loop, at best.
FOOLING AROUND

The point at which the general form of any SAS® language element is understood is where the real fun begins. Although the following exercises hardly come close to qualifying as 'pushing the limits', the reader is encouraged to experiment with the extent to which #BY options can be used. The paucity of publications on the subject should be viewed as a natural invitation towards further exploration.

SAMPLE USAGE 1
Tables split out over multiple pages by a given time point are a common fixture in Clinical Trial reporting. Occasionally, however, the requirement arises for a table or listing to include a dynamic time variable in a section of the title. One method of accomplishing this might be to subset the data and associated PROC REPORT within a macro loop repeated by visit. However, less keystrokes equals more free time, and this task is easily accomplished using #BYVAL instead.

```sas
options nobyline center nonumber nodate;

data temp;
  set sashelp.class;
  do visit=1 to 4;
    if trim(left(upcase(sex)))='M' then weight=weight+visit;
    output;
  end;
run;

proc sort data=temp;
  by visit;
run;

title1 'Listing of Subject Characteristics';
title2 'Visit=#byval(visit)';

proc report nowd data=temp;
  by visit;
  column name weight;
  define name   /  width=15;
  define weight /  width=15;
run;
```

SAMPLE USAGE 2
With the advent of ODS, page numbering is once again a tricky issue, since the page breaks from the SAS® output window no longer necessarily correspond to the page breaks that your favorite word processor application assigns. A partial solution has been to force page breaks in PROC REPORT via a BREAK statement, but this leaves the separate task of assigning the actual page number itself within the title or footnote. Fortunately, one can take advantage of the automatic break provided by a BY variable while simultaneously assigning page numbering using #BYVAL.

```sas
options nobyline center nonumber nodate;

data temp;
  set sashelp.class;
  if age in(11,12) then page=1;
  else if age in(13,14) then page=2;
  else page=3;
run;

proc sort data=temp;
  by page;
run;
```
ods rtf file="c:\wutemp\listing.rtf" style=minimal;

title1 'Listing of Subject Characteristics';
footnote1 j=r "{Page #byval(page)}~{of}~{\field{\*\fldinst { NUMPAGES }}}";

proc report nowd data=temp;
   by page;
   column name weight;
      define name /         width=15;       
      define weight /       width=15;        
run;

ods rtf close;

After all, forcing a page break in PROC REPORT already assumes some prior variable manipulation from which to assign the break, so creating a page variable for use as a #BYVAL ends up providing something the programmer would have had to create anyway, with the added bonus of passing the actual page value into the footnote in the same step. Of course, substituting some previously derived total number of pages macro variable is recommended over using RTF field instructions (e.g., NUMPAGES), which are merely shown above for brevity.

**SAMPLE USAGE 3**
Developing patient profile listings for the first time can be one of the more arduous tasks a programmer can encounter. Historically, use of a custom report driver couched in a data _null_ to provide highly customized output has been a workable, if not enjoyable, solution. This remains the case very much due to PROC REPORT offering so limited a set of tools for manipulating ‘exception’ output (i.e., text elements required to fall outside of its matrix of neatly defined titles, rows, and columns). However, #BYVAL offers an elegant solution when coupled with a simple concatenation in a proceeding data step.

options nobyline center nonumber nodate;

data temp;
   length patient_info $2000;
   set sashelp.class;
      patient_info="Patient:"||trim(left(name))||", "||"Age:"||compress(age)||
      
      
      
      ||"Gender:"||trim(left(sex));
run;

proc sort data=temp;
   by patient_info;
run;

ods rtf file="c:\wutemp\listing.rtf" style=minimal;

title1 'Listing of Subject Characteristics';
title2 ' ';
title3 ' ';
title4 j=1 "#byval1";

proc report nowd data=temp;
   by patient_info;
   column height weight;
      define height /         width=15;       
      define weight /       width=15;        
run;

ods rtf close;

The code above generates the following output...
CONCLUSION
This paper has given a brief glimpse into the little utilized option #BYVAL. Although at no time will its usage ever approach the ‘rage stage’ (a la ODS), #BYVAL can indubitably come in handy on a lonely weekend (working), and is a useful tool for the versatile programmer to keep in mind.

ACKNOWLEDGMENTS
The author would like to thank Cara, Brian, our friends in Biostatistical Services, and Tim Williams for their consistent encouragement and support.

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

Stephen Hunt
PRA International
300-730 View Street
Victoria, BC, Canada V8W 3Y7
Work Phone: 250-483-4461
Fax: 250-483-4588
Email: HuntStephen@PRAIntl.com
Web: www.prainternational.com

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.