SESUG Paper 155-2018

You're Doing It Wrong! Volume 001

Shane Rosanbalm, Rho, Inc.

ABSTRACT

You might think that you're a good programmer. But you're not. It's not just that you're doing it differently than I would do it. It's that you're actually doing it in a way that is unquestionably, incontrovertibly wrong!

But, take heart. I am here to set you on the righteous path. Listen to me, and you will be adored by your coworkers, accepted by SUG section chairs, and solicited by recruiters.

The focus of volume 001 will be on the appropriate naming of macro parameters.

INTRODUCTION

Stop me if you've heard this one before.

- Comedian: Do you know how flexible SAS® is?
- Audience: How flexible is it!?
- Comedian: SAS is soooo flexible, if you ask a dozen different programmers to solve a simple problem, you'll get a dozen different solutions.

Flexibility is great, but there's also something to be said for consistency. For instance, SAS procedures are remarkably consistent in the naming of the options and statements. You use DATA to name the input dataset, VAR to indicate a list of variables, WEIGHT to indicate the variable containing weights to apply to observations, etc.

It is this author's opinion that a similar consistency should be strived for in the naming of parameters in SAS macros. When coming up with names for macro parameters, look to SAS to see whether or not a precedent exists. If it does, use it!

CLEAR PRECEDENT

The papers from PharmaSUG 2017 and SESUG 2017 were searched for SAS macros. The parameter names were collected into a text file and reviewed for consistency. As one might guess, there wasn't a lot of consistency to be found.

INPUT DATASETS

Paper authors managed to come up with 11 different ways to name the single input dataset in their macros in 2017.

- Positional: data, data_, dataset, dsn
- Keyword: data=, datain=, dataset=, ds=, dsn=, indat=, inds=

Many SUG authors over the years have lobbied against positional parameters, so we won't re-litigate that here.

As you can see from the above list, there's a lot of variety. It's not that using DATAIN= or DATASET= are inherently evil. The issue is that there's a convention in place within the language. If you use this convention in your macros then it's easy for your users to remember what keyword to use with your macro. If you choose to go away from DATA= then you make things harder on your users. They either have to remember unconventional parameter names by rote or they have to look up the unconventional parameter names in the documentation. And who likes reading documentation!?

Make it easy on your target audience. When your macro accepts a single input dataset, use DATA= as the parameter name.

OUTPUT DATASETS

Paper authors only managed to come up with 4 different ways to name the single output dataset in their macros in 2017.

- Keyword: outdat=, outds=, out=, dataout=

Again, there's nothing inherently evil about DATAOUT=. But there's a clear convention within SAS to use OUT= when naming output datasets. Make life easier on your users. When your macro creates an output dataset, have the user specify the name with OUT=.

OTHER COMMON OPTION NAMES

SAS has many other options which are common across procedures. Here are examples of unconventional parameter names that people sometimes use in place of the established precedent.

Precedent	Wild Wild West
VAR=	VARS=, VARLIST=, VAR1|VAR2|VAR3, etc.
ID=	IDVAR=, IDVARS=, IDLIST=, etc.
BY=	BYVAR=, BYVARS=, BYLIST=, BYVARLIST=, BYVAR1|BYVAR2|BYVAR3, etc.
GROUP=	GROUPVAR=, GROUPVARS=, GROUPLIST=, GROUPVARLIST=, etc.
X=	XVAR=
Y=	YVAR=

The consistent sin here is to needlessly suffix the established naming convention. Variations include tacking on -VAR, -VARS, -LIST, -VARLIST, -VARn, etc. These suffixes are wasted typing at best. Make life easier on your users. If a convention exists, use it!

SOME AMBIGUITY

MULTIPLE INPUT DATASETS

This case is more interesting. What if your macro requires that you specify multiple input datasets, and all of the datasets are equally important? This is the case in PROC COMPARE where two coequal input datasets are needed: BASE and COMPARE.

One 2017 macro needed 3 input datasets: a subject-level dataset, an adverse events dataset, and a randomization dataset. The author's choice was to use the suffix -DS to name the datasets, resulting in SUBJLVLDS=, ADAEDS=, RANDDS=. These are reasonable parameter names. Personally I would have used the hidden assumption of CDISC-like input datasets to guide the naming. Here are two possibilities:

- Use the CDISC dataset names alone: ADSL=, ADAE=, RD=
- Use conventional DATA as a prefix: DATAADSL=, DATAADAE=, DATARD=

Again, the author's way is not wrong. But the root names of SUBJLVL, ADAE, and RAND lack consistency and don't seem to have originated from a holistic view of the macro.

FOLDER/FILE PATHS

There are a couple of SAS conventions to choose from here. Sometimes they use PATH= and other times they use FILE=. And then there are variations on these themes: OUTFILE= (PROC EXPORT), DATAFILE= (PROC IMPORT), GPATH= (ODS LISTING), etc.

It seems natural that you would have as your default to use PATH= when pointing to folders and FILE= when pointing to files. The use of variants such as OUTFILE=, DATAFILE=, GPATH=, etc. only seem warranted when they map directly to that syntax within your macro (e.g., use OUTFILE= when you've got a PROC EXPORT in your macro).

MULTIPLE PATHS

What if your macro requires that you specify multiple input paths, and all of the paths are equally important? This is similar to the case of multiple input datasets. As with the datasets, there's not necessarily a right answer. But having a holistic approach to naming the parameters would be helpful.

Returning to the same 2017 macro as before, the macro needed 3 input paths: a subject-level dataset path, an adverse events dataset path, and a randomization dataset path. The author's choice was to use the suffix FLDR to name the paths, resulting in SUBJFLDR=, ADAEFLDR=, RANDFLDR=. These are reasonable parameter names. Some other options to consider might be:

- Prefix with the CDISC dataset names: ADSLFLDR=, ADAEFLDR=, RDFLDR=
- Use FLDR as a prefix: FLDRADSL=, FLDRADAE=, FLDRRD=
- Use PATH (in place of FLDR) as a prefix: PATHADSL=, PATHADAE=, PATHRD=
- Use PATH as a suffix instead: ADSLPATH=, ADAEPATH=, RDPATH=

Again, the author's way is not wrong. But since they are passing in the dataset names and folders as separate parameters, they probably should have named the variable pairs consistently. E.g., SUBJLVLDS vs. SUBJFLDR – why does the folder parameter name not include LVL so that the root parameter name for the dataset name and folder match (i.e., SUBJLVLDS and SUBJLVLFLDR)?

SETS OF PARAMETERS

Another consideration is that parameters more naturally group together when a common prefix is used. In the above example:

- Is it more preferable to group by parameter type?
  - DATAADSL, DATAADAE, DATARD
  - PATHADSL, PATHADAE, PATHRD
- Is it more preferable to group by data type?
  - ADSLDATA, ADSLPATH
  - ADAEDATA, ADAEPATH
  - RDDATA, RDPATH

Consider that this macro also has a third set of variables for each data type: the subject ID variable. The author used subject ID parameter names SUBJID= (goes with SUBJLVLDS), AESUBJID= (goes with ADAEDS), RANDSUBJID= (goes with RANDDS). It might have been better to use prefixes that are consistent with the dataset names. For instance, instead of ADAEDS and AESUBJID, why not use a consistent prefix to yield ADAEDS and ADAESUBJID?

Thinking again of grouping, now that there are triples of parameters for each dataset, how would you group them?

- Is it more preferable to group by parameter type?
  - DATAADSL, DATAADAE, DATARD
  - PATHADSL, PATHADAE, PATHRD
  - SUBJIDADSL, SUBJIDADAE, SUBJIDRD
- Is it more preferable to group by data type?
  - ADSLDATA, ADSLPATH, ADSLSUBJID
  - ADAEDATA, ADAEPATH, ADAESUBJID
  - RDDATA, RDPATH, RDSUBJID

EXCEPTIONS

Rules are meant to be broken. I recently saw an in-house macro with parameter names that didn't follow the above suggestions. When I asked the author why, she explained that the macro was written for use by non-programmers, hence the more informal and less syntax-based parameter names. She knows her end users fairly well, so if in her judgment they needed this crutch, who am I to argue?

CONCLUSION

Don't make your macro's users read the documentation every time they want to use your macros. If SAS conventions exist, use them to name your macro parameters. If no convention exists, think holistically about how to name your parameters (within the current macro, across your other macros, across your organization's macros). Exceptions do exist, but they should be given careful consideration.

CONTACT INFORMATION

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

Shane Rosanbalm
Rho, Inc.
srosanba@gmail.com