

PP07 - Stuck in a Rut?

Prakash Mistry, Phastar, London, UK
prakash.mistry@phastar.com
Jordan Bristow, Phastar, London, UK
jordan.bristow@phastar.com



Count for Non-Existent Values

Occasionally a row is needed for a certain TFL that includes a count (N) value, but the value searched for does not exist. One novel method takes advantage of the way SAS processes data, and side-steps around the issue of non-existence by pre-processing based on current data and allowing for correct processing if present later, thus ensuring a dynamic approach. The essential idea is to work based on what you do have and to use this to your advantage to create a logic process within the SAS framework to process what you don't.

```
if &vars. = "&value." then count = 1;
else count = 0;

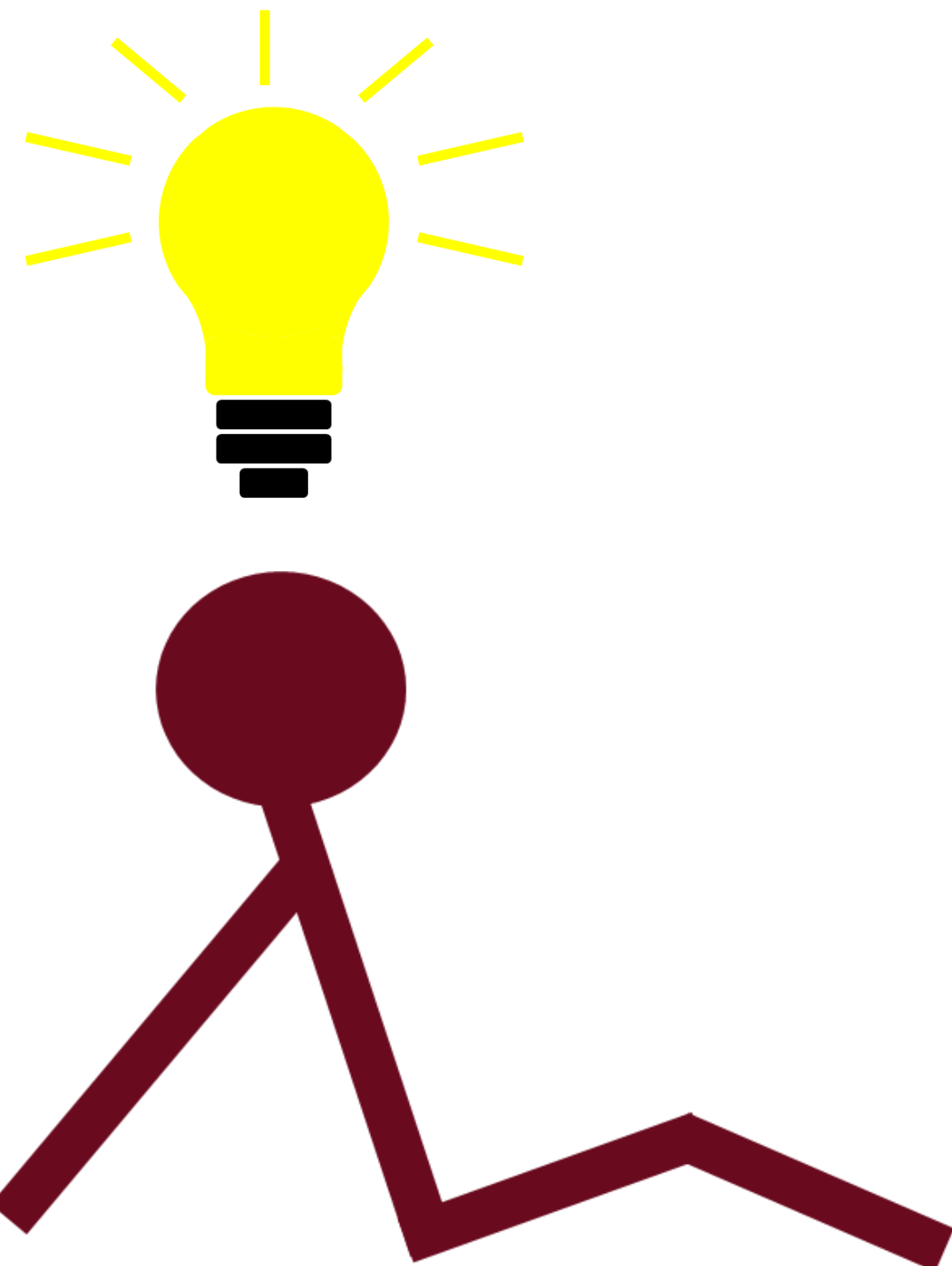
%count_miss(dsin=test, %count_miss(dsin=test,
dsout=count_male, dsout=count_female,
var=sex, var=sex,
value=M); value=F);
```

Count_sexM	Count_sexF
10	0

INTRODUCTION

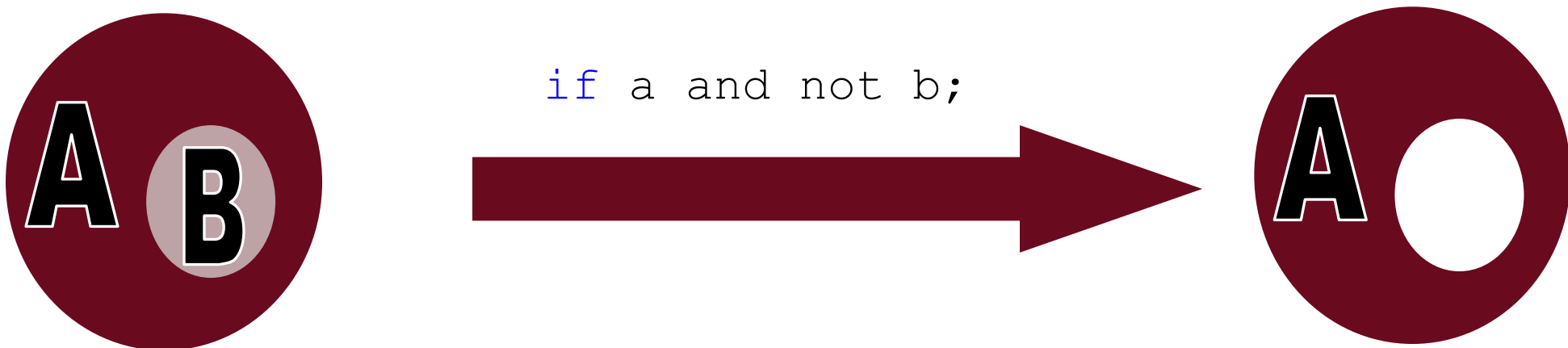
When programming code, occasionally we can encounter challenges that we will have not faced before. The aim would be to try and solve these issues as quickly and efficiently as possible.

However, sometimes these challenges will cause us to become 'Stuck in a rut', either by favouring an approach we have been using previously or by causing us to become stuck altogether. It is important to remember to keep an open mind; there may be many ways in overcoming the issues.



Removing Unwanted Observations

Usually when removing data, we would use dataset filtering. This approach can be made considerably more difficult depending on the complexity of the filters we need to implement and the state of the data that we have. Sometimes the simpler way of filtering would be to create a dataset containing the data we want to remove from the parent dataset, then merging this dataset with the parent dataset and using the condition "If a and not b". This can be substantially easier and quicker to work with than processing using multiple dataset steps and further filtering applied with sorting.



Preserving Order

Having constructed a dataset exactly the way we desire, we can come across issues with duplicate results which we need to remove. A method of removing these issues can be found by applying a new variable and setting it to be equal to _n_.

Value	Value	Ord	Value	Ord
D	D	1	D	1
N	N	2	N	2
N	N	3	A	4
A	A	4		

Late-In-The-Day Additions

Situations can occur when programming SDTMs and ADaMs, of a variable that will need to be present but the predecessor data relating to this variable does not exist within the RAW nor SDTM datasets. We could set the variable to missing at the start to get around this issue and just update the code when the data relating to these variables becomes present. However, there is a more dynamic way of programming this that uses the dictionary datasets or SAS help views. If the variable that needs to be programmed stems from a predecessor dataset then using the tables dictionary/SAS help dataset is the best option for use. If it is a predecessor variable, then the columns dataset should be used. Filtering when calling these two datasets should be based on the libname and memname variables.

```
proc sql;
create table exist as
select *
from dictionary.tables (or sashelp.vtable)
where libname = "SDTM" and memname = "SUPPAE";
quit;
```

Call Execute

Generally, macros are executed in open code as simple calls. But, by using the CALL EXECUTE function we can execute macros within a data step. The benefits of using CALL EXECUTE are that we can execute macros conditionally as well as use existing values within the dataset as values in the macro call.

```
data _null_;
set flags;
CALL EXECUTE ("&pop(pop="||strip(name)||");");
run;
```

Name	
FASFL	%pop(pop=FASFL);
RANDFL	%pop(pop=RANDFL);
SAFFL	%pop(pop=SAFFL);
ITTFL	%pop(pop=ITTFL);
PPROTFL	%pop(pop=PPROTFL);

```
data _null_;
set medhist1 (where=(occur="Y"));
CALL EXECUTE ('%nrstr(%demographics(
flag='||strip(name)||
ord='||strip(put(ord, 2, 0))||'))');
run;
```

Name	Occur	Ord	
MH01FL	Y	1	%demographics(flag=MH01FL, ord=1);
MH02FL		2	✗
MH03FL	Y	3	%demographics(flag=MH03FL, ord=3);
MH04FL	Y	4	%demographics(flag=MH04FL, ord=4);