

What's in it?

-

SAS arrays to check efficiently for emptiness of variables and datasets

PhUSE 2013

Paper CC05

Guillaume Godin
Statistical Programmer
Business & Decision Life Sciences

Tel +32 2 774 11 00
Fax +32 2 774 11 99
guillaume.godin@businessdecision.com

Rue Saint-Lambert 141
1200 Brussels

www.businessdecision-lifesciences.com



DRIVE YOUR PERFORMANCE

The Fear of the Void



The Fear of the Void

	Patient	Sex	Survive	Shock	Arterial	Heart	Cardiac	Urinary
1	203		1 SURV	NONSHOC	.	95	66	110
2	54		1 DIED	HYPOVOL	.	183	95	0
3	664		2 SURV	CARDIO	.	111	332	12
4	210		2 DIED	BACTER	.	97	369	0
5	101		2 DIED	NEURO	.	130	291	0
6	102		2 SURV	OTHER	.	107	471	65
7	529		1 DIED	CARDIO	.	106	217	15
8	524		2 DIED	CARDIO	.	99	156	10
9	426		1 SURV	OTHER	.	77	410	75
10	509		2 SURV	OTHER	?	84	256	90
11	742		1 DIED	HYPOVOL	.	54	135	0
12	609		2 DIED	NONSHOC	.	101	260	90
13	318		2 DIED	OTHER	.	81	410	405
14	412		1 SURV	BACTER	.	87	296	44
15	601		1 DIED	BACTER	.	101	260	377
16	402		1 SURV	CARDIO	.	137	312	75
17	98		2 SURV	CARDIO	.	87	260	377
18	4		1 SURV	HYPOVOL	.	149	406	200
19	50		2 SURV	HYPOVOL	.	111	332	12
20	2		2 DIED	OTHER	.	114	424	97

The Fear of the Void

Empty data can occur for many reasons:

- Data not collected yet
- Erroneous extraction from a bigger source
- Wrong transformation of the data
- ...

... and all this can change every time an updated version of the data is transferred!

Topics

- 
- 1 The Fear of the Void
 - 2 What do we need?
 - 3 Why SAS arrays?
 - 4 How our macro works
 - 5 So, what's in it?

Topics

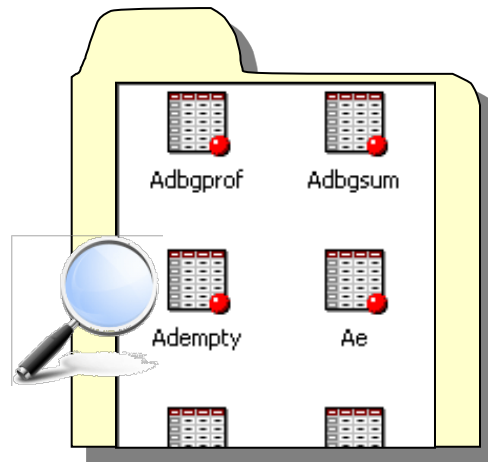
- 
- 1 The Fear of the Void
 - 2 **What do we need?**
 - 3 Why SAS arrays?
 - 4 How our macro works
 - 5 So, what's in it?

What do we need?

To address this issue, we need a macro that:

What do we need?

To address this issue, we need a macro that:
goes through **every dataset** of a target library



What do we need?

To address this issue, we need a macro that:

goes through **every dataset** of a target library

for **every variable**: *"At least one non-null value?"*

What do we need?

To address this issue, we need a macro that:

goes through **every dataset** of a target library

for **every variable**: *"At least one non-null value?"*

produces a **report** with the resulting metadata

	A	B	C	D	E	F	G
1	STUDY	VIEWNAME	VARNAME	LABEL	VARTYPE	FORMAT	EMPTY
2	Pooled	AE	ANTIMIC	How many antimicrobials needed to treat	num	8.	X
3	Pooled	AE	ASSIST	Assistance required	num	YN1F	
4	Pooled	AE	BOULPOS	Blood culture positive	num	NEGPOC1F.	X
5	Pooled	AE	BGLU	Blood glucose value	num	BGLU1F	
6	Pooled	AE	BONEF	Bone fracture	num	BONEF1F.	X
7	Pooled	AE	BONEFRSI	AES1 flag-Bone fracture	num	YN1F	
8	Pooled	AE	BPAIN	Back pain	num	YN1F	X
9	Pooled	AE	CANCERS2	AES1 flag-Cancer 2	num	YN1F	
10	Pooled	AE	CANCERS1	AES1 flag-Cancer	num	YN1F	
11	Pooled	AE	CECADJSI	Flag for CEC adjudicated AEs	num	YN1F	
12	Pooled	AE	CENTRE	Centre	num	10.	
13	Pooled	AE	COSTO	Costo-vertebral angle pain/tenderness	num	YN1F	X
14	Pooled	AE	COUNTRY	Investigator country	char	\$7.	
15	Pooled	AE	CPEVENT	CPE name	char	\$20.	
16	Pooled	AE	DAYNO	Number of days	num	8.	X
17	Pooled	AE	DCMSUBNM	DCM subset name	char	\$8.	
18	Pooled	AE	DECRR1SI	AES1 flag-decr. renal funct. PT search	num	YN1F	
19	Pooled	AE	DECRR2SI	AES1 flag-decr. renal funct. SMQ search	num	YN1F	
20	Pooled	AE	DICTVER	Original MedDRA version	char		
21	Pooled	AE	DOCNUM	Document number	char	\$20.	
22	Pooled	AE	FEVCHI	Fever chills	num	YN1F	X
23	Pooled	AE	FLANKP	Flank pain	num	YN1F	X
24	Pooled	AE	GISI	AES1 flag-GI	num	YN1F	
25	Pooled	AE	GTITYP	Type of genital infection	num	GTITYP1F.	X

What do we need?

To address this issue, we need a macro that:

goes through **every dataset** of a target library

for **every variable**: *"At least one non-null value?"*

produces a **report** with the resulting metadata

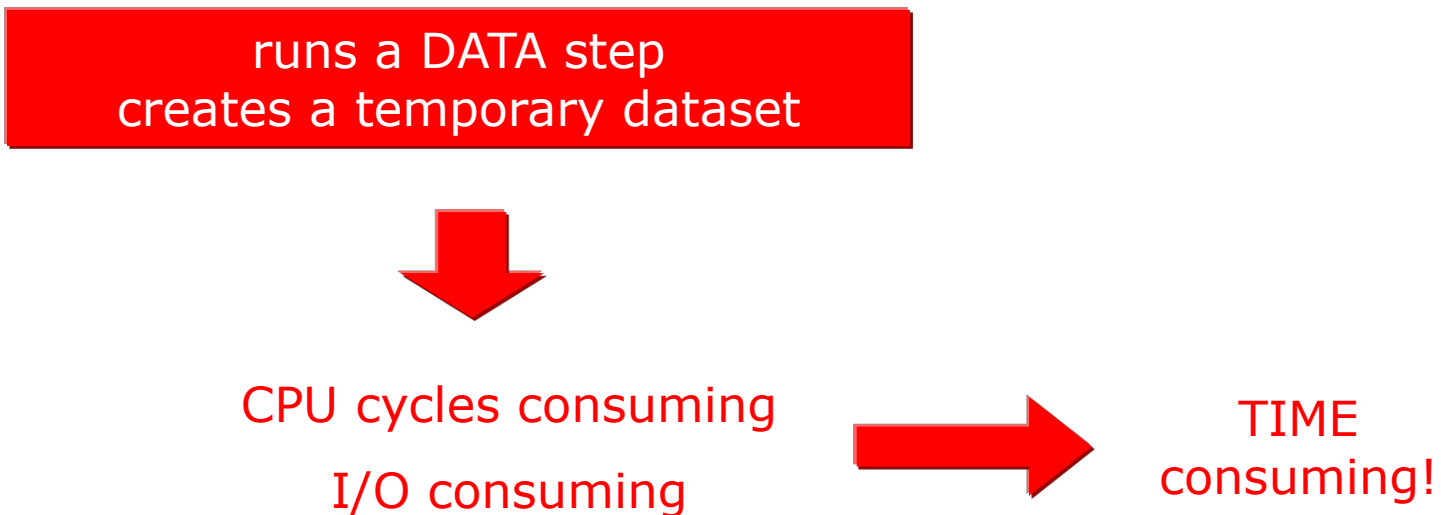
... and does all that **quickly!**

Topics

- 
- 1 The Fear of the Void
 - 2 What do we need?
 - 3 **Why SAS arrays?**
 - 4 How our macro works
 - 5 So, what's in it?

Why SAS arrays?

We want to avoid a code that loops through every dataset/variable couple and **at each iteration**:



Why SAS arrays?

SAS arrays:

- are user-defined **groups of variables** that can be handled like matrices
- allow for **compact** coding

```
data mystat;  
    set mysource;  
    array lab_array(30) lab1-lab30;  
    array labsqrt_array(30) lab_sqrt1-lab_sqrt30;  
    do i=1 to 30;  
        labsqrt_array(i)=sqrt(lab_array(i));  
    end;  
    drop i;  
run;
```

- **use the system RAM** for all operations performed on them inside a data step

Topics

- 
- 1 The Fear of the Void
 - 2 What do we need?
 - 3 Why SAS arrays?
 - 4 **How our macro works**
 - 5 So, what's in it?

How our macro works

This macro is organized in 4 main steps:

1. Retrieve information on the current dataset
2. 'Core' submacro: use arrays to check for non-emptiness of all variables of the current dataset
3. Call the steps 1 and 2 for each dataset in the library
4. Produce the metadata report

How our macro works

This macro is organized in 4 main steps:

1. Retrieve information on the current dataset
2. 'Core' submacro: use arrays to check for non-emptiness of all variables of the current dataset
3. Call the steps 1 and 2 for each dataset in the library
4. Produce the metadata report

How our macro works: the 'core' step

In this step, repeated for each dataset in the library, we create 2 arrays for the character variables and 2 arrays for the numeric variables:

```
%if &NOBS > 0 %then %do;

data CHECK_TO_APPEND (keep=VIEWNAME VARNAME EMPTY);
  length VIEWNAME $ 32 VARNAME $ 32 EMPTY $ 1;
  VIEWNAME="&VIEWNAME.";
  EMPTY='X';
  %if &CHARCT. > 0 %then %do;
    array CHARVAR(*) $ &CLIST;
    array C ALLMISS (&CHARCT) $ (&CHARCT*'true');
  %end;
  %if &NUMCT. > 0 %then %do;
    array NUMVAR(*) &NLIST;
    array N ALLMISS (&NUMCT) $ (&NUMCT*'true');
  %end;

(...)
```

How our macro works: the 'core' step

... then the values in each of the 2 'true' arrays are updated and retained to 'false' as soon as a non-null value is encountered for a variable:

```
set SRC.&VIEWNAME. end=DONE;
  %if &CHARCT. > 0 %then %do;
    do i=1 to dim(C ALLMISS);
      if CHARVAR(i) ne ' ' then C ALLMISS(i)='false';
    end;
  %end;
  %if &NUMCT. > 0 %then %do;
    do i=1 to dim(N ALLMISS);
      if NUMVAR(i) ne . then N ALLMISS(i)='false';
    end;
  %end;

(...)
```

How our macro works: the 'core' step

... next, the macro loops through the flag arrays to store the name of every empty variable into a macrovariable:

```

if DONE then do;
    CNT=0;
    %if &CHARCT. > 0 %then %do;
        do i= 1 to dim(C_ALLMISS);
            if C_ALLMISS(i) ='true' then do;
                CNT+1;
                call symput('VAR' || put(cnt,3.-1),vname(CHARVAR(i)));
            end;
        end;
    %end;
    %if &NUMCT. > 0 %then %do;
        do i=1 to dim(N_ALLMISS);
            if N_ALLMISS(i)='true' then do;
                CNT+1;
                call symput('VAR' || put(CNT,3.-1),vname(NUMVAR(i)));
            end;
        end;
    %end;
end;

```

(...)

How our macro works: the 'core' step

... and finally only the names of the empty variables are output in the variable VARNAME, with the variable EMPTY being 'X':

```
do i=1 to CNT;  
    VARNAME=symget('VAR' || strip(put(i,BEST.)));  
    output;  
end;  
run;  
%end;  
  
(...)
```

How our macro works

This macro is organized in 4 main steps:

1. Retrieve information on the current dataset
2. 'Core' submacro: use arrays to check for non-emptiness of all variables of the current dataset
3. Call the steps 1 and 2 for each dataset in the library
4. Produce the metadata report

Topics

- 
- 1 The Fear of the Void
 - 2 What do we need?
 - 3 Why SAS arrays?
 - 4 How our macro works
 - 5 **So, what's in it?**

So, what's in it?

	A	B	C	D	E	F	G
1	STUDY	VIEWNAME	VARNAME	LABEL	VARTYPE	FORMAT	EMPTY
2	Pooled	AE	ANTIMIC	How many antimicrobials needed to treat	num	8.	X
3	Pooled	AE	ASSIST	Assistance required	num	YN1F.	
4	Pooled	AE	BCULPOS	Blood culture positive	num	NEGPOC1F.	X
5	Pooled	AE	BGLU	Blood glucose value	num	BGLU1F.	
6	Pooled	AE	BONEF	Bone fracture	num	BONEF1F.	X
7	Pooled	AE	BONEFRSI	AESI flag-Bone fracture	num	YN1F.	
8	Pooled	AE	BPAIN	Back pain	num	YN1F.	X
9	Pooled	AE	CANCERS2	AESI flag-Cancer 2	num	YN1F.	
10	Pooled	AE	CANCERSI	AESI flag-Cancer	num	YN1F.	
11	Pooled	AE	CECADJSI	Flag for CEC adjudicated AEs	num	YN1F.	
12	Pooled	AE	CENTRE	Centre	num	10.	
13	Pooled	AE	COSTO	Costo-vertebral angle pain/tenderness	num	YN1F.	X
14	Pooled	AE	COUNTRY	Investigator country	char	\$7.	
15	Pooled	AE	CPEVENT	CPE name	char	\$20.	
16	Pooled	AE	DAYNO	Number of days	num	8.	X
17	Pooled	AE	DCMSUBNM	DCM subset name	char	\$8.	
18	Pooled	AE	DECRR1SI	AESI flag-decr. renal funct. PT search	num	YN1F.	
19	Pooled	AE	DECRR2SI	AESI flag-decr. renal funct. SMQ search	num	YN1F.	
20	Pooled	AE	DICTVER	Original MedDRA version	char		
21	Pooled	AE	DOCNUM	Document number	char	\$20.	
22	Pooled	AE	FEVCHI	Fever chills	num	YN1F.	X
23	Pooled	AE	FLANKP	Flank pain	num	YN1F.	X
24	Pooled	AE	GISI	AESI flag-GI	num	YN1F.	
25	Pooled	AE	GTITYP	Type of genital infection	num	GTITYP1F.	X

So, what's in it?

	A	B	C	D	E	F	G
1	STUDY	VIEWNAME	VARNAME	LABEL	VARTYPE	FORMAT	EMPTY
2	Pooled	AE	ANTIMIC	How many antimicrobials needed to treat	num	8.	X
3	Pooled	AE	ASSIST	Assistance required	num	YN1F.	
4	Pooled	AE	BCULPOS	Blood culture positive	num	NEGPOC1F.	X
5	Pooled	AE	BGLU	Blood glucose value	num	BGLU1F.	
6	Pooled	AE	BONEF	Bone fracture	num	BONEF1F.	X
7	Pooled	AE	BONEFRSI	AESI flag-Bone fracture	num	YN1F.	
8	Pooled	AE	BPAIN	Back pain	num	YN1F.	X
9	Pooled	AE	CANCERS2	AESI flag-Cancer 2	num	YN1F.	
10	Pooled	AE	CANCERSI	AESI flag-Cancer	num	YN1F.	
11	Pooled	AE	CECADJSI	Flag for CEC adjudicated AEs	num	YN1F.	
12	Pooled	AE	CENTRE	Centre	num	10.	
13	Pooled	AE	COSTO	Costo-vertebral angle pain/tenderness	num	YN1F.	X
14	Pooled	AE	COUNTRY	Investigator country	char	\$7.	
15	Pooled	AE	CPEVENT	CPE name	char	\$20.	
16	Pooled	AE	DAYNO	Number of days	num	8.	X
17	Pooled	AE	DCMSUBNM	DCM subset name	char	\$8.	
18	Pooled	AE	DECRR1SI	AESI flag-decr. renal funct. PT search	num	YN1F.	
19	Pooled	AE	DECRR2SI	AESI flag-decr. renal funct. SMQ search	num	YN1F.	
20	Pooled	AE	DICTVER	Original MedDRA version	char		
21	Pooled	AE	DOCNUM	Document number	char	\$20.	
22	Pooled	AE	FEVCHI	Fever chills	num	YN1F.	X
23	Pooled	AE	FLANKP	Flank pain	num	YN1F.	X
24	Pooled	AE	GISI	AESI flag-GI	num	YN1F.	
25	Pooled	AE	GTITYP	Type of genital infection	num	GTITYP1F.	X

So, what's in it?

This macro is an example of how arrays can be combined with macro language and a few SCL statements in order to

- go through a library and analyze every dataset inside it
- check if every variable contains at least one non-null value
- produce a report/dataset containing the relevant metadata

... at a minimum resource cost.

Thanks to the SAS arrays we can do that in a compact, readable and reusable code that will write mainly in the system RAM.

So, what's in it?





**Thank you
for your attention**

Guillaume Godin
Statistical Programmer
Business & Decision Life Sciences

Tel +32 2 774 11 00
Fax +32 2 774 11 99
guillaume.godin@businessdecision.com

Rue Saint-Lambert 141
1200 Brussels

www.businessdecision-lifesciences.com



DRIVE YOUR PERFORMANCE

Additional slides



How our macro works: step 1

In this step, we use SCL statements to retrieve information on the current dataset at a minimum resource cost:

```
%let DSID=%sysfunc(open(SRC.&VIEWNAME,in));  
%let NOBS=%sysfunc(attrn(&DSID,nobs));  
%if &DSID > 0 %then %let RC=%sysfunc(close(&DSID));  
  
(...)
```

How our macro works: step 1

... then we store the names of the character and numeric variables, and their respective number, into macrovariables:

```
proc sql noprint;
  select NAME, put(count(NAME),5.-L) into :CLIST separated by ' ' , :CHARCT
  from DICTIONARY.COLUMNS
  where LIBNAME='SRC' and MEMNAME=upcase("&VIEWNAME.") and TYPE='char';

  select NAME, put(count(NAME),5.-L) into :NLIST separated by ' ' , :NUMCT
  from DICTIONARY.COLUMNS
  where LIBNAME='SRC' and MEMNAME=upcase("&VIEWNAME.") and TYPE='num';
quit;

(...)
```

How our macro works: step 3

We take information from the read-only dictionary tables `DICTIONARY.COLUMNS` and `DICTIONARY.TABLES` to list all the dataset/variable combination in the target library:

```
proc sql noprint;
  create table LISTCOL as
  select distinct MEMNAME as VIEWNAME, NAME as VARNAME, LABEL as LABEL,
    TYPE as VARTYPE, FORMAT as FORMAT
  from DICTIONARY.COLUMNS
  where LIBNAME eq "SRC"
  ;
  create table LISTTABLES as
  select MEMNAME
  from DICTIONARY.TABLES
  where libname eq "SRC"
  ;
  create table EMPTY_VAR (VIEWNAME char (32), VARNAME char (32)
    , EMPTY char (1))
  ;
quit;

(...)
```

How our macro works: step 3

... then, with a %DO loop, we will call the “inner” macro once for each record of the dataset LISTTABLES:

```
data _null_;  
  set LISTTABLES;  
  NUMBER_OF_DS +1;  
  call symput("TABLE" || strip(put(NUMBER_OF_DS,3.)),strip(MEMNAME));  
  call symputx("NUMBER_DS_MVAR",NUMBER_OF_DS);  
run;  
  
%do i=1 %to &NUMBER_DS_MVAR;  
  %CHECK_EMPTY_VAR (VIEWNAME =&&TABLE&i);  
%end;  
  
(...)
```

How our macro works: step 3

... and finally, we join the table containing the list of all domains and variables with the list of the all-missing variables across all domains:

```
proc sql noprint;
  create table contents as
  select distinct "Pooled" as STUDY, A.*, B.EMPTY
  from LISTCOL A left join EMPTY_VAR B
  on A.VIEWNAME = B.VIEWNAME
  and A.VARNAME = B.VARNAME
  ;
quit;

(...)
```