



Using the descriptor portion of a SAS data file

Dragos Daniel Capan



Outline

- The descriptor portion of a dataset
- Prerequisite- processing a SAS data step
- Examples
 - 2 examples on how making use of the descriptor portion makes your SAS code more efficient
 - 2 examples when using the information in the descriptor makes your life a lot easier

Data set it's made of ...

**Descriptor portion: variable names,
attributes etc.**

		DATA		

Descriptor portion

■ Information on the physical dataset

- Number of observations
- The date that the data set was created and last modified
- Number of indexes
- Whether dataset is sorted or not

■ Information on individual variables

- variable name, type, length, format, informat, label, and whether the variable is indexed.

Descriptor portion

```
proc contents data=datalib.abstract;  
run;
```

OR

```
proc datasets library=datalib nolist;  
  contents data=abstract;  
quit;
```

Processing a SAS data step

A SAS DATA step is processed in two phases

New SAS Data Set

Compilation
Phase



Descriptor Portion

Execution
Phase



Data Portion

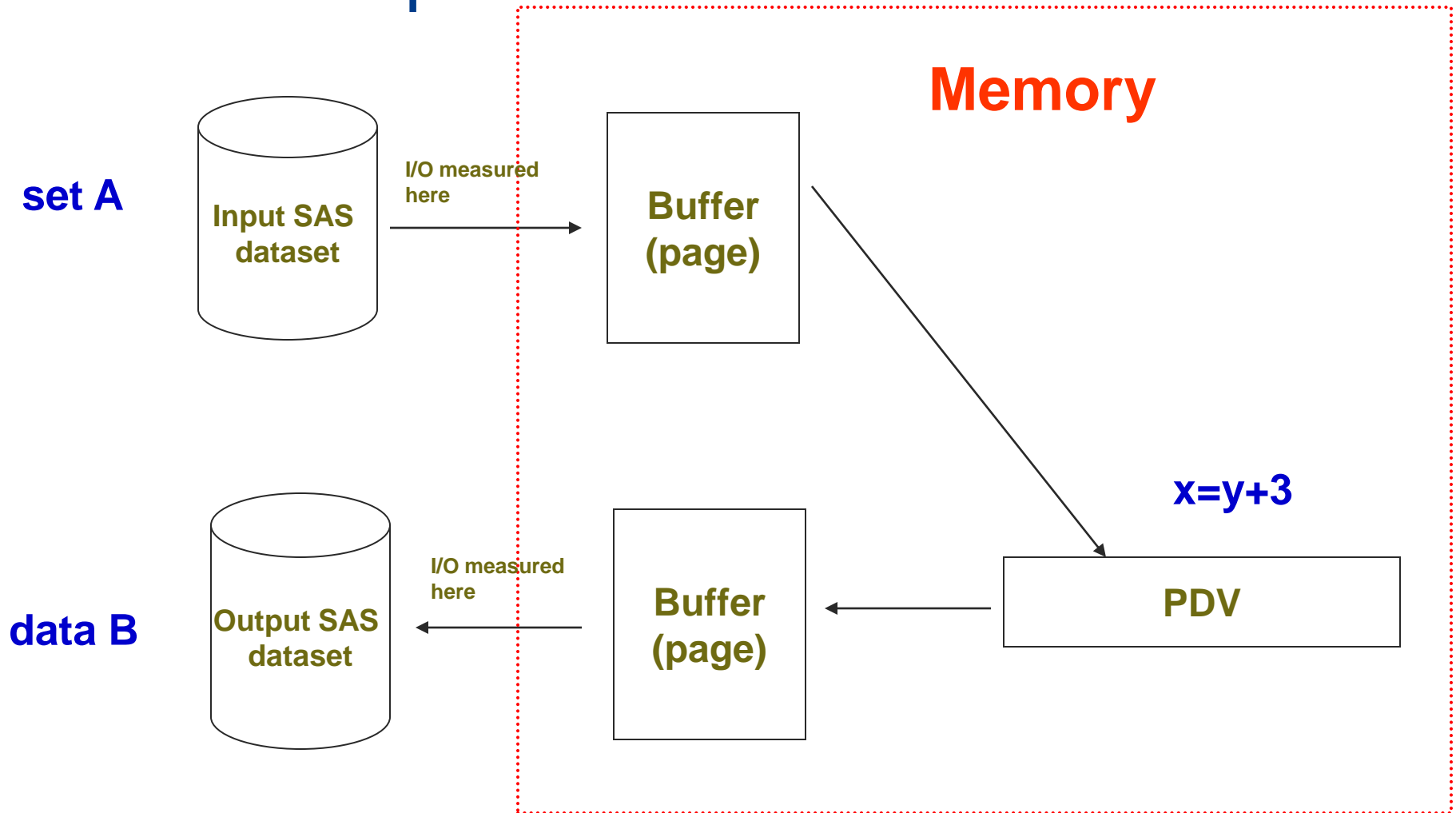
Processing a SAS data step

- During the **compilation phase**, each statement is scanned for syntax errors. When the compilation phase is complete, the descriptor portion of the new data set is created.
- If the DATA step compiles successfully, then the **execution phase** begins. During the execution phase, the DATA step reads and processes the input data.

Execution phase – SAS code example

```
data B;  
  
    set A;  
  
    x=y+3;  
  
run;
```

Execution phase



Example 1

Upon running a SAS code you find out that your new dataset name is misspelled and that some of the variable names are not the same as the ones a macro uses with this dataset. So, you need to **modify the name of the dataset and the variable names**. For consistency you also want to **format** one of them.

The less desirable way...

```
data mylib.DAD;  
  
  set mylib.DSD;  
  
    rename dsd_trans=dad_trans  
           admission_date = admdate;  
  
    format admdate date9. ;  
  
run;
```

Why is this less desirable ?

The best way

```
proc datasets lib=mylib;  
  change DSD=DAD;  
  modify DAD;  
    rename dsd_trans = dad_trans  
          admission_date=admdate;  
  format admdate date9.;
```

quit;

run;

Some stats ...

The data step approach:

NOTE: DATA statement used (Total process time):

real time	30.64 seconds
cpu time	12.07 seconds

The proc datasets approach:

NOTE: PROCEDURE DATASETS used (Total process time):

real time	0.01 seconds
cpu time	0.01 seconds

Example 2

You need to **create a macro variable that holds the number of observations in a certain dataset.**

You can later use this macro variable to check whether the dataset is empty or for some DO loops.

The less efficient way

```
data mylib.dad;  
  set mylib.dad end=last;  
  if last then call symput('nr_obs', _N_);  
run;
```

```
real time          28.26 seconds  
cpu time           13.19 seconds
```

```
data _null_;  
  set mylib.dad end=last;  
  if last then call symput('nr_obs', _N_);  
run;
```

```
real time          4.18 seconds  
cpu time           3.44 seconds
```

The most efficient way

```
data _null_;  
    if 0 then set mylib.dad nobs=nr;  
    call symput('nr_obs',nr);  
    stop;  
  
run;
```

real time	0.00 seconds
cpu time	0.00 seconds

Example 3

Create a macro variable that holds all the variable names in a dataset (or only the numeric/character ones).

Useful for PROC SCORE, for KEEP/DROP options etc.

PROC CONTENTS & SCORE

```
proc score data=hsmr_data  
  score=parameters out=scored_data  
  TYPE=PARMS;  
  
var &covariates;  
  
run;
```

Parameters is a dataset that was created after running proc logistic. It consists of 1 row of coefficients for every covariate in the model.

Macro variable *covariates* resolves to:

```

LOS_group1 LOS_group2 LOS_group4 LOS_group5
LOS_group6 admcatU age charlson_group_new1
charlson_group_new2 diag_flagA04 diag_flagA41
diag_flagC16 diag_flagC18 diag_flagC22 diag_flagC25
diag_flagC34 diag_flagC50 diag_flagC61 diag_flagC67
diag_flagC71 diag_flagC78 diag_flagC79 diag_flagC80
diag_flagC83 diag_flagC85 diag_flagC90 diag_flagC91
diag_flagC92 diag_flagE11 diag_flagE86 diag_flagE87
diag_flagF03 diag_flagG30 diag_flagG93 diag_flagI21
diag_flagI25 diag_flagI26 diag_flagI46 diag_flagI48
diag_flagI50 diag_flagI60 diag_flagI61 diag_flagI62
diag_flagI63 diag_flagI64 diag_flagI71 diag_flagJ18
diag_flagJ44 diag_flagJ69 diag_flagJ80 diag_flagJ84
diag_flagJ90 diag_flagJ95 diag_flagJ96 diag_flagK55
diag_flagK56 diag_flagK57 diag_flagK63 diag_flagK65
diag_flagK70 diag_flagK72 diag_flagK74 diag_flagK85
diag_flagK92 diag_flagN17 diag_flagN18 diag_flagN19
diag_flagN39 diag_flagR57 diag_flagS06 diag_flagS72
diag_flagT81 diag_flagZ54

sexM transfer1

```

Creating the *covariates* macro variable

```
proc contents data=parameters
    out=var_names (keep = name where=(name not
    in ('Intercept', '_LINK_', '_LNLIKE_',
    '_NAME_', '_STATUS_', '_TYPE_'))) noprint;
run;
```

```
proc sql noprint;
select name into: covariates separated by" "
    from var_names;
quit;
```

Example 4 ... and the last one

We have a dataset that has 120 numeric variables and a few character ones. This dataset is going to be used in the e-publication so what we need to do is **replace the values of -2009 to †**

To solve this we need to transform all the numeric variables to character, replace the value above, while keeping format, the name and the order of the variables the same.

Transforming numeric to character

```

proc contents data=epub noprint
    out=descriptor (keep = name type varnum formatl
formatd);

run;

data variables_num;
set descriptor (where=(type=1));
length varformat $ 7;
if formatl=0 then varformat='best12.'; else
varformat=put(formatl,1.) || "." || put(formatd,1.);

run;

```

Transforming numeric to character

■ Create the macro variables:

- for1, for2, for3 ... that will hold the FORMAT of each variable that needs to be transformed
- mac1, mac2, mac3 ... that will hold the NAME of each variable
- ch1, ch2, ch3 ... that will be the temporary names for the variables

Transforming numeric to character

```

data _null_;
  set variables_num end=end;
  call symput('for' || left(put(_N_,4.)), varformat);
  call symput('mac' || left(put(_N_,4.)), name);
  call symput('ch' || left(put(_N_,4.)), 'ch' || left(put(_N_,4.)));
  if end then call symput('end', put(_N_,8.));
run;

```

```

data epub_ch (drop= %do p=1 %to &end; &&mac&p %end;
  rename=(%do p=1 %to &end; &&ch&p=&&mac&p %end;));
set epub ;
  %do p=1 %to &end;
    &&ch&p=strip(put(&&mac&p, &&for&p));
    if &&ch&p in: ("-2009") then &&ch&p ="+";
  %end;
run;

```

Keeping variables in the same order

```
proc sort data=descriptor; by varnum; run;
```

```
proc sql noprint;
```

```
  select name into :inorder separated by ', '  
  from descriptor;
```

```
quit;
```

```
proc sql noprint;
```

```
  create table epub_table as  
  select &inorder from epub_ch;
```

```
quit;
```

.....

QUESTIONS ?