Paper BB-03-2013

The Secret Life of DATA STEP

Swati Agarwal, OPTUM, Eden Prairie, MN

ABSTRACT

Each SAS® DATA step functions as a self-contained mini program that is compiled and run within your overall SAS® program. Much of DATA step processing is implicit. For example, DATA step statements are executed within an implied loop even though you do not use loop control statements.
This paper discusses several of the implied DATA step statements that control how your DATA step really works. Concepts you will be introduced to:

- the Logical Program Data Vector
- automatic SAS® variables and how are they used
- understanding the internals of DATA step processing
- what happens at program compile time
- what's actually happening at execution time
- how variable attributes are captured and stored.

You have probably written a number of DATA Steps in your codes; some works , some don't; sometimes you know why; and sometimes you are struggling and scratching your head what went wrong. If this describes you, then this talk is for you.

The DATA step has been very well designed, but it is different, and it's worth learning how it works, if you want elegant code. Even people who have been SAS® programmers who have been working with SAS® for years should find, an "Ah ha, so that's why", in this talk.

INTRODUCTION

The SAS® DATA Step is one of the primary methods for creating SAS® data sets. To be a good SAS® programmer it is essential that you understand the intricacies of the DATA Step because some tasks related to data manipulation and data set creation may only be done via the DATA Step (they cannot be done or are too complex to be done via SAS® procedures.)
It is important to learn the life cycle of DATA Step which consists of two phases - Compile phase and Execution phase; it is equally important to understand the Program Data Vector (PDV), which spans across both phases and which determines what information is held in storage and how it changes within the DATA Step.

The Compile phase includes:

- compiling the SAS® statements, including syntax checking
- creating an input buffer (if reading a raw input file), a Program Data Vector (PDV), and descriptor information

The Execution phase includes:

- counting the iterations of the DATA Step (beginning at the DATA statement)
- setting all variables to missing in the Program Data Vector and initializing automatic variables
- reading an input record (from raw file or SAS® data set), if relevant
- executing additional manipulation or computation statements

- writing an output record into an output data set and looping back to the DATA statement

The Program Data Vector spans across the compile and execution phases:

- it is a logical area in memory where SAS® builds a data set, one observation at a time
- contains current values for all variables
- maintains two automatic variables, _N_ and _ERROR_

If you know the rules that govern these important aspects of the DATA Step process, then you will be able to take control, correctly instruct SAS® what to do, and utilize the automatic variables for your benefit.

DATA STEP

A DATA step consists of a group of statements in the SAS® language that can

- read data from external files
- write data to external files
- read SAS® data sets and SAS® views
- create SAS® data sets and SAS® views

Once your data is accessible as a SAS® data set, you can analyze the data and write reports by using a set of tools known as SAS® procedures.

You can use the DATA step for

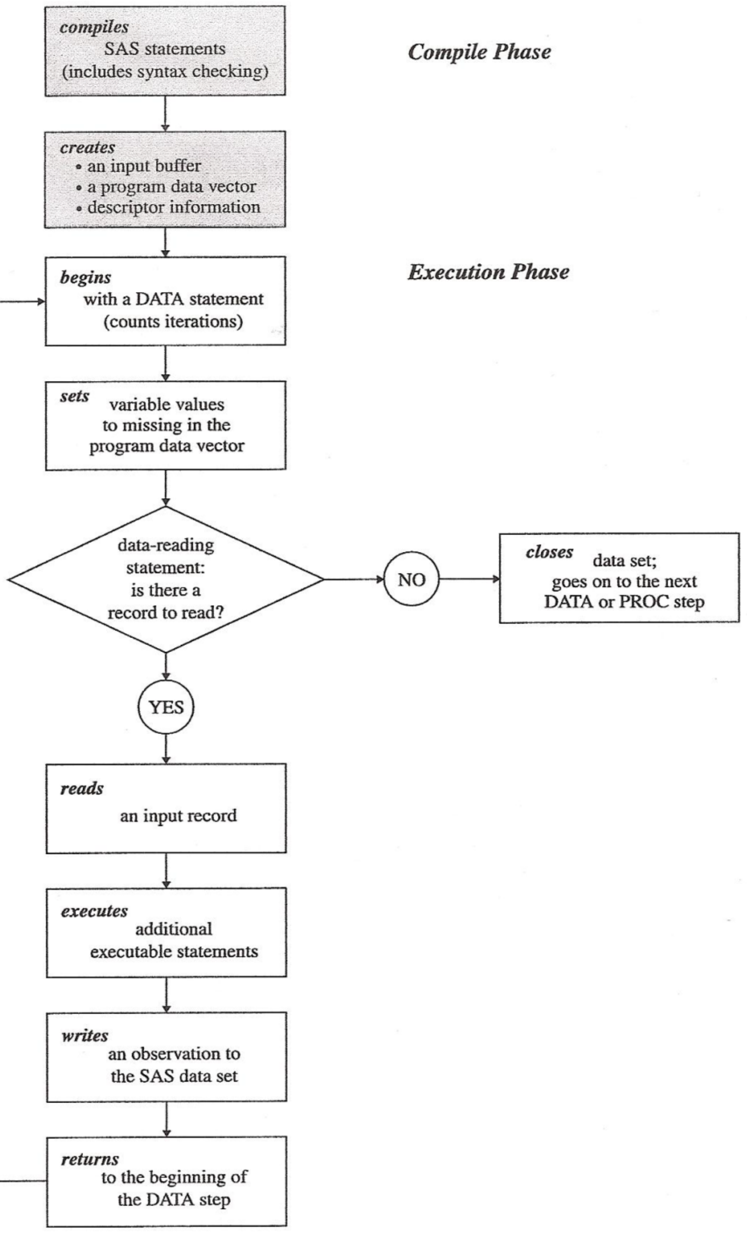
- creating SAS® data sets (SAS® data files or SAS® views)
- creating SAS® data sets from input files that contain raw data (external files)
- creating new SAS® data sets from existing ones by sub setting, merging, modifying, and updating existing SAS® data sets
- analyzing, manipulating, or presenting your data
- computing the values for new variables
- report writing, or writing files to disk or tape
- retrieving information
- file management

The boundaries of a DATA Step are between the DATA statement at the “start” of the step and the implied or explicit RUN statement at the “end” of the step. The “end” of a simple DATA Step serves as an implied RETURN until the last observation of this DATA Step is processed.

LIFE CYCLE OF DATA STEP

SAS® provides the following flowchart to describe the processing of the DATA Step

Figure 20.1 Flow of Action in the DATA Step



compiles SAS statements (includes syntax checking)
Compile Phase
creates • an input buffer • a program data vector • descriptor information
Execution Phase
begins with a DATA statement (counts iterations)
sets variable values to missing in the program data vector
data-reading statement: is there a record to read?
NO
closes data set; goes on to the next DATA or PROC step
YES
reads an input record
executes additional executable statements
writes an observation to the SAS data set
returns to the beginning of the DATA step

COMPILE PHASE

As we learned above, the very first phase in DATA Step processing is the Compile phase. SAS® performs the following tasks during the compile phase:

- Automatically translates the SAS® statements into machine code to be executed later.
- identifies the type and length of each variable
- determines whether a type conversion is necessary for each subsequent reference to a variable
- creates input buffer if there is an INPUT statement for an external file
- creates the Program Data Vector
- creates descriptor information for data sets and variable attributes
- processes statements, which are limited to compile-time; these provide information to the compiler on how to set things up; in fact, they drive how things will be set up within the PDV; they include:
  DROP; KEEP; RENAME; RETAIN; WHERE; LABEL; LENGTH; FORMAT; ARRAY; BY; ATTRIB
- sets up some automatic variables, including:_N_, _ERROR_, END=, IN=, FIRST, LAST, POINT=

Example 1: Syntax Error Found

```
1     data example_1;
2     x = ‖Hi=;
          -
          386
          200
3     y = ‖How=;
          -
          386
          200
4     y = ‖Your=;
          -
          386
          200
5     y = ‖Day=;
          -
          386
          200
ERROR 386-185: Expecting an arithmetic expression.

ERROR 200-322: The symbol is not recognized and will be ignored.

6     run;
```

EXECUTION PHASE

After converting the code into machine code, create the input buffer, PDV and descriptor information. Execution Phase comes into picture and SAS® performs the following functions, in this default order:

- begins with the DATA Statement and sets the _N_ variable to 1 (with each new iteration of the DATA Statement, the _N_ variable gets incremented by 1)
- sets variable values to missing in the Program Data Vector
- reads a data record into the input buffer with the INPUT statement (if reading a raw file)
- reads a SAS® observation from a SAS® data set into the PDV with one of the following statements: SET, MERGE, MODIFY or UPDATE
- executes programming statements found within the DATA Step
- writes an observation into the output data set(s), except in the case of “data _NULL_”
- returns to the DATA Statement
- all actions are repeated per iteration until end of input file (or input data set) is reached
- if the DATA Step includes no reading of input records, the Execution (by default) performs only one iteration
- This default order of execution may be changed through the use of conditional IF-THEN-ELSE statements, DO

- Loops, LINK, RETURN, and GO TO statements, etc.

Example 2: Default and forced sequence

The first example shows a simple DATA Step, which follows the standard execution of statements and output of records. Here, one iteration produces one observation in the output data set:

```
data example_2;
x = 'Minnesota';
y = 'Columbus';
run;
```

```
The log presents the following message:
NOTE: The data set WORK.EXAMPLE_2 has 1 observation and 2 variables.
```

The second example shows how a single iteration of the data step produces 5 observations in the output data set. The default order of execution was changed via the DO Loop:

```
data example_3;
do i = 1 to 5;
x = 'Minnesota';
y = 'Columbus';
output;
end;
run;
```

```
The log presents the following message:
NOTE: The data set WORK.EXAMPLE_2 has 5 observations and 3 variables.
```

THE PROGRAM DATA VECTOR

The SAS® Language Reference defines the program data vector as:

"A logical area in memory where SAS® builds a data set, one observation at a time. When a program executes, SAS® reads values from the input buffer or creates them by executing SAS® language statements. The data values are assigned to the appropriate variables in the program data vector. From here, SAS® writes the values to a SAS® data set as a single observation."

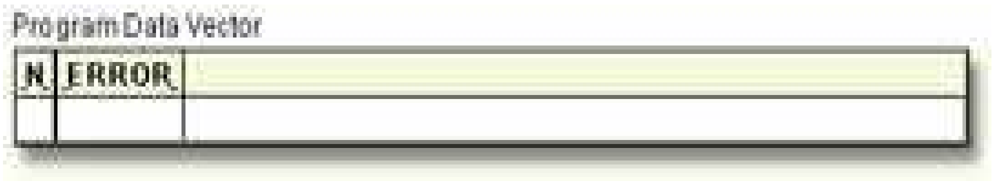
Stated another way: The program data vector is a storage place in memory that contains all of the variables encountered by the data step. The order of the variables in the program data vector is determined by when they are encountered. These variables may have indicators flagging them to be kept or dropped or renamed. When the program runs, the program data vector contains the observation currently being processed. At the end of the step, the data are output according to the drop, keep, or rename instructions encountered in the program.

After the input buffer is created, the **program data vector** is created.

The **program data vector** is the area of memory where SAS® builds a data set, one observation at a time.

Like the term **input buffer**, the term **program data vector** refers to a logical concept.

- The program data vector contains two **automatic variables** that can be used for processing but which are not written to the data set as part of an observation.
- **_N_** counts the number of times that the DATA step begins to execute.
- **_ ERROR_** signals the occurrence of an error that is caused by the data during execution. The default value is *0*, which means there is no error. When one or more errors occur, the value is set to *1*
- First.BY-variable and last.*BY-variable*: These temporary variables always appear in pairs, one pair for each BY-variable on a BY statement. They have the value of 1 or 0 if the condition is true or false respectively.



Temporary variables are created through the use of certain SAS® options and statements, and like automatic variables, are not written to the output data set. Some common temporary variables include:

- IN=*variable*: IN= is a data set option that indicates whether a particular data set has contributed to the current observation. The user specifies a variable name with the option that will have the value of 1 if the data set contributed to the observation or 0 if it did not.
- END=*variable*: END= is an option to the SET statement that indicates the end of the input data has been reached. The user defines a variable name that will have the value of 1 if the end of the data has been reached, 0 if it has not. Only one END= can be specified on the SET statement. If multiple data sets are given in the SET statement, the END= variable will be set to 1 only when the last observation of the last data set has been read.

User defined variables can be assigned the value of automatic and temporary variables if it is necessary to have their values in the output data set.
The program data vector holds all variables, user defined, automatic, and temporary, from the data sets, input sources, or created during the execution of the data step. The program data vector is used to populate the output data sets. All variables in the output data sets are in the program data vector, but not all variables on the program

data vector are written to the output data sets.
During execution SAS® will set the non-retained and non-input variables to missing in the program data vector at the beginning of each iteration of the DATA step. SAS® reads data from an input file or from a SAS® data set directly into the program data vector, replacing the previously existing values.

PROCESSING A DATA STEP: A WALKTHROUGH

Example 4 – simple default – no reading of input data

```
data example_4;
put "after compile before execution: " _all_;
x = "GOOD";
y = "MORNING";
z = "COLUMBUS";
n = 4;
k = n*2;
put "at end of execution: " _all_;
run;
```

The Compile phase sets up the PDV as follows:

x	y	z	n	k	_N_	_ERROR_
			.	.	1	0

At the end of the DATA Step Execution the PDV appears as follows:

x	y	z	n	k	_N_	_ERROR_
GOOD	MORNING	COLUMBUS	4	8	1	0

One single record is output into the example1 data set. Only a single iteration of the Step is done. The output record includes all the variables except for the automatic _N_ and _ERROR_

Example 5 – reading of raw input data

```
data total_points (drop=Class);
  input class $ StudentName $ Subject1 Subject2 Subject3;
  Total = (Subject1 + Subject2 + Subject3);
  datalines;
Knights Sue    6  8  8
Kings Jane     9  7  8
Knights John   7  7  7
Knights Lisa   8  9  9
Knights Fran   7  6  6
Knights Walter 9  8 10
;
```

Since this example is having the input file which is containing raw data; SAS® creates an input buffer to hold the data before moving the data to the program data vector (PDV)

Note - (If the input file is a SAS® data set, however, SAS® does not create an input buffer. SAS® writes the input data directly to the PDV.)

Input Buffer

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0

Program Data Vector

Class	Student Name	Subject 1	Subject 2	Subject 3	Total	_N_	_ERROR_
		.	.	.	.	1	0
Drop						Drop	Drop

Note that in this representation, numeric variables are initialized with a period and character variables are initialized with blanks. The automatic variable _N_ is set to 1; the automatic variable _ERROR_ is set to 0.

After reading the first observation, PDV looks like

Class	Student Name	Subject 1	Subject 2	Subject 3	Total	_N_	_ERROR_
Knights	Sue	6	8	8	22	1	0

When SAS® executes the last statement in the DATA step, all values in the PDV, except those marked to be dropped, are written as a single observation to the data set TOTAL_POINTS.

SAS® then returns to the DATA statement to begin the next iteration. SAS® resets the values in the PDV in the following way:

- The values of variables created by the INPUT statement are set to missing.
- The value created by the Sum statement is automatically retained.
- The value of the automatic variable _N_ is incremented by 1, and the value of _ERROR_ is reset to 0.

Below table shows how each record would look in the PDV before it is written out to the example2 SAS® data set. Remember that the PDV only holds one record at a time and writes it to the output data set before a new record is built in the PDV.

Class	Student Name	Subject 1	Subject 2	Subject 3	Total	_N_	_ERROR_
Knights	Sue	6	8	8	22	1	0
Kings	Jane	9	7	8	24	2	0
Knights	John	7	7	7	21	3	0
Knights	Lisa	8	9	9	26	4	0
Knights	Fran	7	6	6	19	5	0
Knights	Walter	9	8	10	27	6	0

CONCLUSION

Visualizing the PDV is a very helpful way to increase your understanding of what is going on in DATA step processing. It may not seem necessary for very simple DATA steps, but even then, it can help. If you visualize on some level, a record being read in and a record being output for every observation in a data set, you will be less likely to write minuscule DATA steps like:

```
data a;
set perm.a ;

data b;
set a ;
x = y + 2;
```

As long as each observation of PERM.A is to be copied into the PDV, why not create X while it is there?

When you want to do something more complex, you need concrete, not vague, knowledge of the information the PDV is holding, and the rules SAS® observes in manipulating that information. When your program is not doing what you intend, use PUT statements to write parts of the PDV to the log, or use the DATA step.

ACKNOWLEDGMENTS

I would like to Thank Saiful Islam and Steve Roloff who reviewed my paper and provided comments and encouragements.

I would also like to Thank Andrew for his guidance on his valuable feedback for completing the paper.

RECOMMENDED READING

Arthur Li (2013), *“Essentials of the Program Data Vector (PDV): Directing the Aim to Understanding the DATA Step!”*, Proceedings of the 2013 SAS® Global Forum

Whitlock, Ian, *“How to Think Through the SAS® DATA Step”*, Proceedings of the 31st Annual SAS® Users Group International Conference

Kahane, Dalia C (2011), *“SAS®® DATA Step Merge – A Powerful Tool”*, Proceedings of the 2011 North East SAS®
Users Group Conference

SAS®® 9.1.3 Language Reference: Concepts, Third Edition. 2005, SAS® Institute Inc., Cary, NC, USA

CONTACT INFORMATION

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

Name: Swati Agarwal
Enterprise: Optum
Address: 12982 Valley View Road
City, State ZIP: Eden Prairie, MN, 55344
Work Phone: 952-974-1932
E-mail: mailtoswatiagarwal@yahoo.com