

Learning to love the SAS LAG function

Phuse 9-12 October 2011

Herman Ament, MSD, Oss NL

Phuse 9-12 October 2011

Contents

- Introduction
- Definition of LAG and DIF function
- LAG explained in detail
- Examples

Introduction

- In order to retrieve the value of a previous observation the function LAG or LAG1 is often used. The previous value is often compared to the most recent value.
- In code:

```
DATA newset;  
  SET oldest;  
  IF VarValue = LAG(VarValue) THEN DO;  
    * value of VarValue equals value of previous  
      observation;  
  END;  
RUN;
```

Examples on retrieving the previous value

- Below you will find other ways to retrieve the previous value of a variable.
 1. By storing the value - at the end of the data step - in a variable that is retained
 2. By storing the value in a **new variable** that is created before the first SET statement in the data step
 3. By using the LAG function.

On the next slide code is shown for these 3 examples

Code examples retrieving previous value

```
DATA d0; INPUT X @@; CARDS;  
1 2 3 4 5  
;
```

```
DATA d1;
```

1

```
A = X;
```

Note:

X is reset just before the SET statement
A is reset at the end of the DATASTEP

2

```
SET d0;
```

```
B = LAG(X) ;
```

Use the **LAG** function

```
OUTPUT;
```

3

```
RETAIN C;
```

```
C = X;
```

Use of the **RETAIN** function

```
RUN;
```

```
PROC PRINT DATA = d1;
```

```
VAR X A B C;
```

```
RUN;
```

Results code examples retrieving previous value

Obs	X	A	B	C
1	1			
2	2	1	1	1
3	3	2	2	2
4	4	3	3	3
5	5	4	4	4

No differences between A, B and C.

They contain all the value of the previous observation

‘Unexpected’ results of LAG

Here is an example of a program giving ‘unexpected’ results, for example increase a counting number for each new subject for a specific assessment.

```
DATA newset;  
  SET oldset;  
  BY assessment;  
  IF NOT first.assessment THEN DO;  
    IF subjid = LAG(subjid) THEN count+1;  
    ELSE count = 1;  
  END;  
END;  
RUN;
```

Because LAG(SUBJID) is executed conditionally, LAG(subjid) does not always contain the value of SUBJID of the previous observation.

In the example above variable COUNT will not always be set to 1 if SUBJID differs from the previous observation.

Definition of LAG

- The LAG functions, LAG1, LAG2, . . . , LAG100 return values from a **queue**. LAG1 can also be written as LAG.
- A LAG_n function stores a value in a queue and **returns a value stored previously in that queue. Each occurrence** of a LAG_n function in a program **generates its own queue** of values.
- It is important to understand that for each LAG function a separate queue with a specific length is created.
- The argument of the LAG function is entered into the queue.
- All values in the queue are moved one position forward
- The oldest value entered will be returned into the expression.
- Hence, for the first n executions of LAG_n, missing values are returned, thereafter the lagged values of the argument begin to appear. For example, a LAG2 queue is initialized with two missing values.
- If the argument of LAG_n is an array name, a separate queue is maintained for each variable in the array

Explanation for LAG3

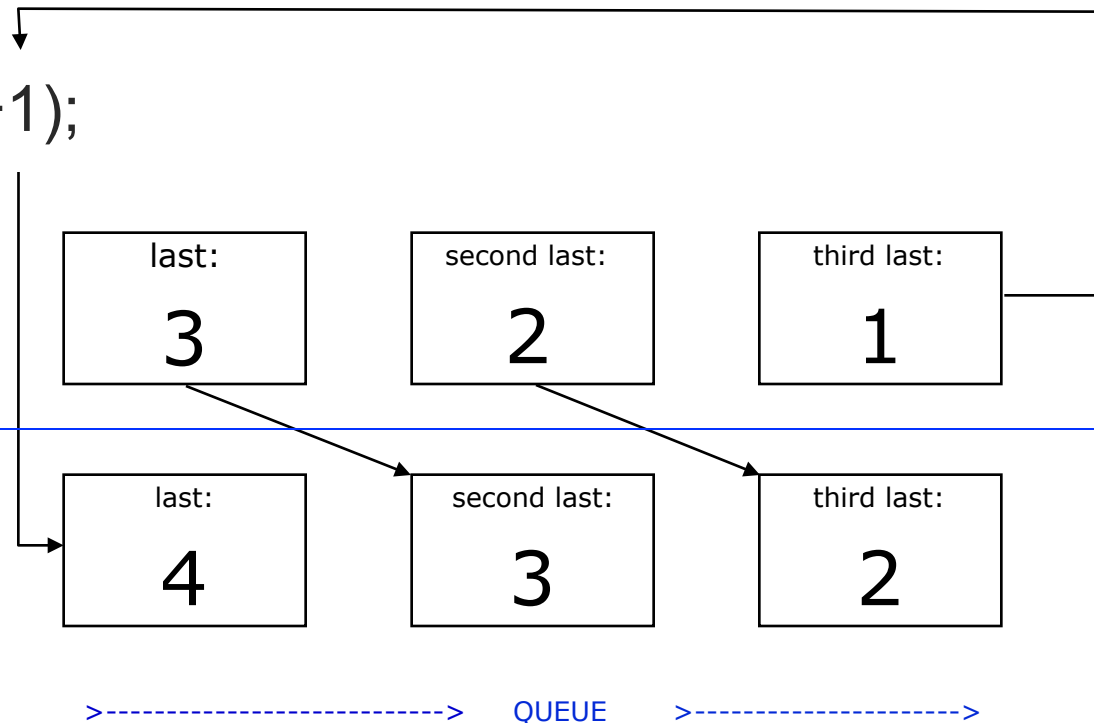
$A = 3;$

$Y = \text{LAG3}(A+1);$

Result

$Y = 1$

1 is returned



Definition of DIF

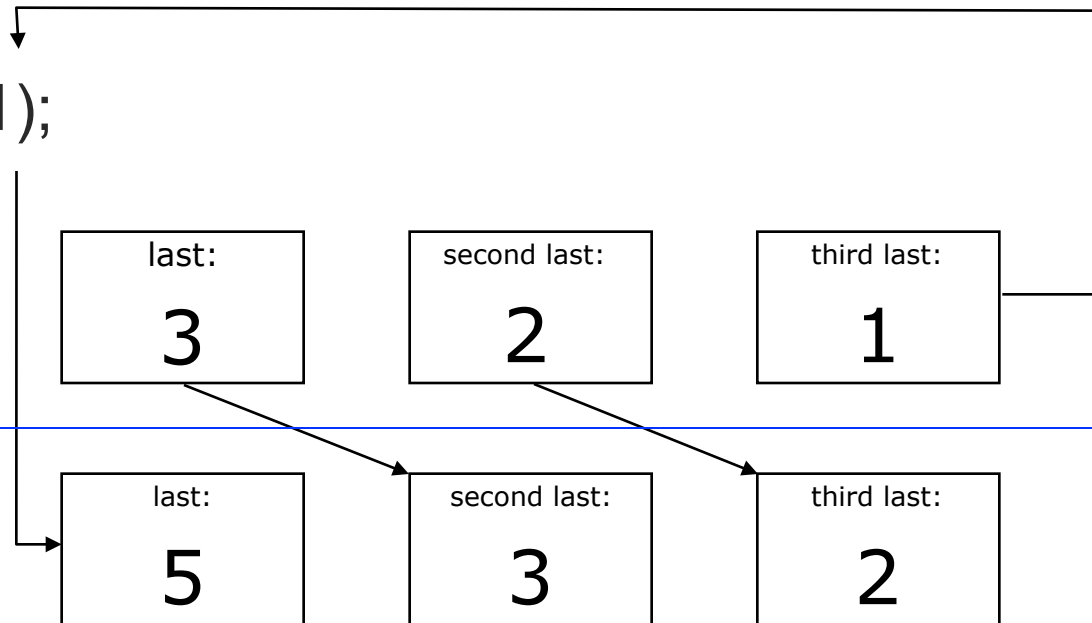
- The DIF functions, DIF1, DIF2, ..., DIF100, return the first differences between the argument and its nth lag. DIF1 can also be written as DIF. DIFn is defined as $DIFn(x) = x - LAGn(x)$.
- The DIF function is almost the same as the LAG function. The difference is that returned value from LAGn is subtracted from the argument of the DIF function.

Explanation for DIF3

$A = 4;$

1 subtracted from 5 is returned

$Y = \text{DIF3}(A+1);$



Result

$Y = 4$

>-----> QUEUE >----->

Example 1, two queues

Here are two separate identical queues. The content of both LAG queues are the same after each iteration of the data step.

```
DATA a;  
INPUT a;  
x = LAG(a) ;  
y = LAG(a) ;  
CARDS;  
1  
2  
;  
  
PROC PRINT;  
VAR a x y;  
RUN;
```

Result

Obs	a	x	y
1	1		
2	2	1	1

Example 2, one queue executed in a DO loop

Here is one queue. The value entered in the queue in the first iteration of J is returned in the second iteration. The result is that X has the same value as A and that the content of the queue is a missing value.

```
DATA a;  
INPUT a;  
DO J=1 to 2;  
x = LAG(a);  
END;
```

```
CARDS;  
1  
2  
;
```

```
PROC PRINT;  
VAR a x ;  
RUN;
```

Result

<u>Obs</u>	<u>a</u>	<u>x</u>
1	1	1
2	2	2

Example 3, one queue used in a LINK statement

Here is one queue, stored in a subroutine and called via LINK statement.

The value entered after the first call is returned in the second call. The result is the same as for example 2: X has the same value as A.

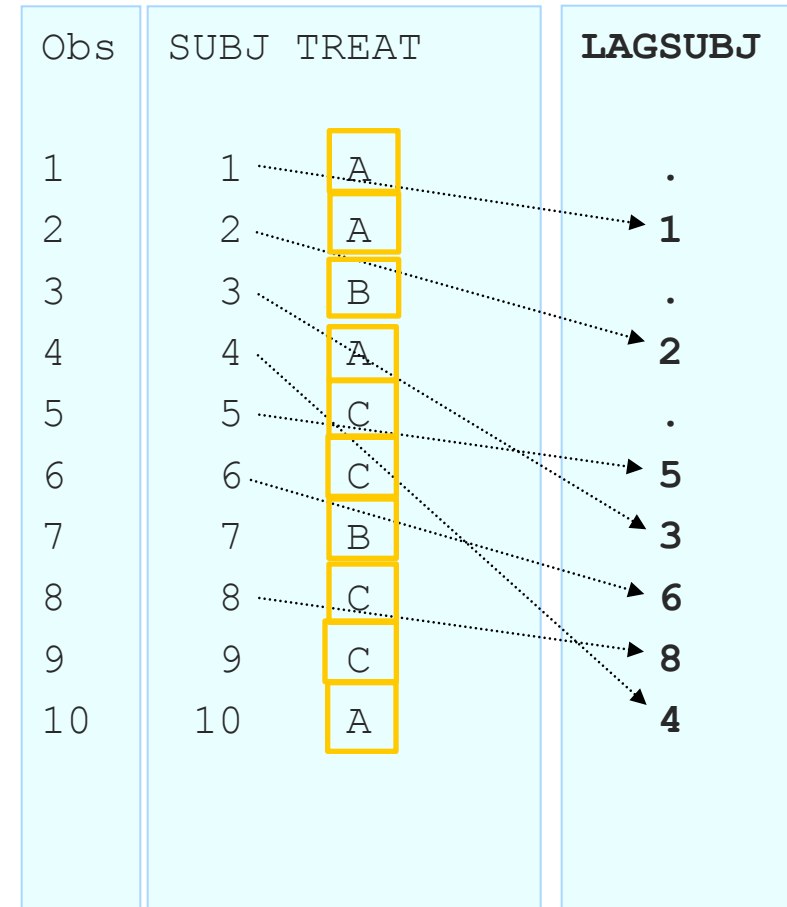
```
DATA a;  
INPUT a;  
  LINK example;  
  LINK example;  
RETURN;  
  Example: x = LAG(a);  
RETURN;  
CARDS;  
1  
2  
;  
PROC PRINT;  
VAR a x ;  
RUN;
```

Result

<u>Obs</u>	<u>a</u>	<u>x</u>
1	1	1
2	2	2

Example 4, each LAG has its own queue

```
DATA b;  
  SET a;  
  SELECT (treat);  
  WHEN ('A') lagsubj = lag(subj);  
  WHEN ('B') lagsubj = lag(subj);  
  WHEN ('C') lagsubj = lag(subj);  
END;  
RUN;  
  
PROC PRINT;  
RUN;
```



Example golden ratio (cont' d)

- The golden ratio is mentioned in the novel “The Da Vinci Code” of Dan Brown.
- By definition, the first two Fibonacci numbers are 0 and 1, and each subsequent number is the sum of the previous two. The quotient of the last 2 numbers converges to the golden ratio.
- The golden ratio is exactly $(1+\sqrt{5})/2$, which is approximately 1.618033988. The Fibonacci row converges rapidly to this number

Example golden ratio (cont'd)

```
DATA fib;  
fib = 1;  
OUTPUT;  
DO UNTIL (fib > 1000);  
    quotient = SUM(fib, LAG(fib)) /fib;  
    fib      = SUM(fib, LAG(fib));  
    OUTPUT;  
END;  
RUN;  
  
PROC PRINT;  
VAR fib quotient;  
RUN;
```

Example golden ratio (cont'd)

Output:

Obs	fib	quotient	
1	1		
2	1	1.0000000000	= 1/1
3	2	2.0000000000	= 2/1
4	3	1.5000000000	= 3/2
5	5	1.6666666667	= 5/3
6	8	1.6000000000	= 8/5
7	13	1.6250000000	= 13/8
8	21	1.6153846154	= 21/13
9	34	1.6190476190	= 34/21
10	55	1.6176470588	= 55/34
11	89	1.6181818182	= 89/55
12	144	1.6179775281	= 144/89
13	233	1.6180555556	= 233/144

Conclusion

CONCLUSION

- The LAG and DIF function are powerful functions. If well understood they can be used in many ways.
- If the previous value in a data step has to be retrieved and the code is simple, the LAG function can be used.
- If the code is more complex, e.g. when previous values are used within a conditional section, the RETAIN statement is recommended.