

How can I use the LAG function in SAS with examples?

Authored by
stats writer

July 1, 2024

RECOMMENDED CITATION

stats writer (2024). *How can I use the LAG function in SAS with examples?*.

PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=164467>

The LAG function in SAS is a useful tool for accessing data from a previous observation in a dataset. It allows for the comparison and analysis of data across multiple periods or observations. To use the LAG function, one must specify the number of previous observations to be referenced. For example, using LAG(1) will access data from the immediate previous observation. This function can be applied to various scenarios, such as calculating changes in data over time, identifying trends, and detecting outliers. Overall, the LAG function in SAS provides a powerful means of analyzing and understanding data by referencing past observations.

Use the LAG Function in SAS (With Examples)

You can use the LAG function in SAS to retrieve lagged values of some variable.

This function uses the following basic syntax:

```
lag1_value = lag(value);
```

By default, lag finds the previous value of some variable.

However, you can use lag2, lag3, lagn, etc. to calculate the 2-lagged, 3-lagged, n-lagged, etc. values of some variable.

The following examples show how to use the lag function in practice.

Example 1: Calculated Lagged Values for Some Variable

Suppose we have the following dataset in SAS that shows the total sales made by some store on consecutive days:

```
/*create dataset*/  
data original_data;  
input day $ sales;  
datalines;  
1 14  
2 19  
3 22  
4 20  
5 16  
6 26  
7 40  
8 43  
9 29  
10 30  
11 35  
12 33  
;  
run; /*view dataset*/  
proc printdata=my_data;
```

Obs	day	sales
1	1	14
2	2	19
3	3	22
4	4	20
5	5	16
6	6	26
7	7	40
8	8	43
9	9	29
10	10	30
11	11	35
12	12	33

The following code shows how to calculate the value for sales lagged by 1, 2, and 3 days:

```
/*create new dataset that shows lagged values of sales*/  
data new_data;  
set original_data;  
lag1_sales = lag(sales);  
lag2_sales = lag2(sales);  
lag3_sales = lag3(sales);  
run;  
  
/*view new dataset*/  
proc printdata=new_data;
```

Obs	day	sales	lag1_sales	lag2_sales	lag3_sales
1	1	14	.	.	.
2	2	19	14	.	.
3	3	22	19	14	.
4	4	20	22	19	14
5	5	16	20	22	19
6	6	26	16	20	22
7	7	40	26	16	20
8	8	43	40	26	16
9	9	29	43	40	26
10	10	30	29	43	40
11	11	35	30	29	43
12	12	33	35	30	29

The three new columns (lag1_sales, lag2_sales, lag3_sales) show the sales lagged by one, two, and three days, respectively.

Example 2: Calculated Lagged Values by Group

Suppose we have the following dataset in SAS that shows the total sales made by two stores during consecutive days:

```
/*create dataset*/  
data original_data;  
input store $ sales;  
datalines;
```

A 14

A 19

A 22

A 20

A 16

A 26

B 40

B 43

B 29

B 30

B 35

B 33

;

run;

/*view dataset*/

proc printdata=original_data;

Obs	store	sales
1	A	14
2	A	19
3	A	22
4	A	20
5	A	16
6	A	26
7	B	40
8	B	43
9	B	29
10	B	30
11	B	35
12	B	33

We can use the following code to calculate the 1-day lagged sales values by store:

```
/*create new dataset that shows lagged values of sales  
by store*/
```

```
data new_data;
```

```
set original_data;
```

```
by store;
```

```
lag1_sales = lag(sales);
```

```
if first.store then lag1_sales = .;
```

```
run;
```

```
/*view new dataset*/
```

```
proc printdata=new_data;
```

Obs	store	sales	lag1_sales
1	A	14	.
2	A	19	14
3	A	22	19
4	A	20	22
5	A	16	20
6	A	26	16
7	B	40	.
8	B	43	40
9	B	29	43
10	B	30	29
11	B	35	30
12	B	33	35

The values in the lag1_sales column show the 1-day lagged sales values for each store.

Notice that the value for lag1_sales in row 7 is empty because the 1-day lagged value for that row represents a sales value for a different store.

Additional Resources

The following tutorials explain how to perform other common tasks in SAS: