Interleaving a Dataset with Itself: How and Why

Howard Schreier
U.S. Dept. of Commerce

1

Purpose

When two or more SAS datasets are combined by means of a SET statement and an accompanying BY statement, they are said to be interleaved. It is also possible tto interleave a dataset with itself, and that technique is useful in solving a fairly common type of problem.

2

DEMO

Obs	id	value
1	a	2
2	a	1
3	a	3
4	b	3
5	b	5

3

MORE

Obs	id	value
1	a	0
2	b	0

4

Simplest SET: Code

```
data _null_;
set demo;
put _all_;
run;
```

5

Simplest SET: Log

```
id=a value=2 _N_=1
id=a value=1 _N_=2
id=a value=3 _N_=3
id=b value=3 _N_=4
id=b value=5 _N_=5
NOTE: There were 5 observations
 read from the data set
 WORK.DEMO.
```

6

Multiple Datasets: Code

```
data _null_;
set demo more(in=from_more);
put _all_;
run;
```

7

Multiple Datasets: Log

```
from_more=0 id=a value=2 _N_=1
from_more=0 id=a value=1 _N_=2
from_more=0 id=a value=3 _N_=3
from_more=0 id=b value=3 _N_=4
from_more=0 id=b value=5 _N_=5
from_more=1 id=a value=0 _N_=6
from_more=1 id=b value=0 _N_=7
```

8

Multiple Datasets: Log

```
NOTE: There were 5 observations
 read from the data set
 WORK.DEMO.
NOTE: There were 2 observations
 read from the data set
 WORK.MORE.
```

9

BY statement: Code

```
data _null_;
set demo more(in=from_more);
by id;
put _all_;
run;
```

10

BY statement: Log

```
from_more=0 id=a value=2
 FIRST.id=1 LAST.id=0 _N_=1
from_more=0 id=a value=1
 FIRST.id=0 LAST.id=0 _N_=2
from_more=0 id=a value=3
 FIRST.id=0 LAST.id=0 _N_=3
from_more=1 id=a value=0
 FIRST.id=0 LAST.id=1 _N_=4
from_more=0 id=b value=3
 FIRST.id=1 LAST.id=0 _N_=5
```

11

BY statement: Log

```
from_more=0 id=b value=5
 FIRST.id=0 LAST.id=0 _N_=6
from_more=1 id=b value=0
 FIRST.id=0 LAST.id=1 _N_=7
NOTE: There were 5 observations
 read from the data set
 WORK.DEMO.
NOTE: There were 2 observations
 read from the data set
 WORK.MORE.
```

12

Self-interleave: Code

```
data _null_;
set demo(in=firstpass) demo;
by id;
put _all_;
run;
```

13

Self-interleave: Log

```
firstpass=1 id=a value=2
 FIRST.id=1 LAST.id=0 _N_=1
firstpass=1 id=a value=1
 FIRST.id=0 LAST.id=0 _N_=2
firstpass=1 id=a value=3
 FIRST.id=0 LAST.id=0 _N_=3
firstpass=0 id=a value=2
 FIRST.id=0 LAST.id=0 _N_=4
firstpass=0 id=a value=1
 FIRST.id=0 LAST.id=0 _N_=5
```

14

Self-interleave: Log

```
firstpass=0 id=a value=3
 FIRST.id=0 LAST.id=1 _N_=6
firstpass=1 id=b value=3
 FIRST.id=1 LAST.id=0 _N_=7
firstpass=1 id=b value=5
 FIRST.id=0 LAST.id=0 _N_=8
firstpass=0 id=b value=3
 FIRST.id=0 LAST.id=0 _N_=9
firstpass=0 id=b value=5
 FIRST.id=0 LAST.id=1 _N_=10
```

15

Self-interleave: Log

```
NOTE: There were 5 observations
 read from the data set
 WORK.DEMO.
NOTE: There were 5 observations
 read from the data set
 WORK.DEMO.
```

16

Applications

- Processing which requires both preservation of detail and summarization or evaluation across observations
- Ordering without sorting

17

Scale transformation

- Task: for each ID level in DEMO, create variable SCALE which transforms variable VALUE to a zero-one scale
- Traditional strategy
  - PROC SUMMARY produces MIN and MAX stats
  - MERGE these with original data
  - Compute
- SQL suitable
- Self-interleave is alternative

18

Scale transformation: Code

```
data zero_one_scale;
set demo(in=firstpass)
    demo;
by id;
retain min max;
drop   min max;
if first.id then do;
   min = .;
   max = .;
   end;
```

19

Scale transformation: Code

```
if firstpass then do;
   min = min(min,value);
   max = max(max,value);
   end;
else do;
   scale = (value-min)/
           (  max-min);
   output;
   end;
run;
```

20

Scale transformation: Results

Obs	id	value	scale
1	a	2	0.5
2	a	1	0.0
3	a	3	1.0
4	b	3	0.0
5	b	5	1.0

21

Ordering without sorting

- Task: for each ID level in DEMO, present observations with odd values of VALUE first, followed by observations with even values of VALUE
- Traditional strategy
  - DATA step to create an odd/even indicator variable
  - PROC SORT
- SQL suitable
- Self-interleave is alternative

22

Ordering without sorting: Code

```
data odd_then_even;
set
 demo
  (where=(mod(value,2)=1))
 demo
  (where=(mod(value,2)=0));
by id;
run;
```

23

Ordering without sorting: Results

Obs	id	value
1	a	1
2	a	3
3	a	2
4	b	3
5	b	5

24

Detecting the boundary : Code

```
data _null_;
set demo(in=p1) demo(in=p2);
by id;
boundary = p2 and lag(p1);
put _all_;
run;
```

25

Detecting the boundary : Log

```
p1=1 p2=0 id=a value=2
 FIRST.id=1 LAST.id=0
 boundary=0 _N_=1
p1=1 p2=0 id=a value=1
 FIRST.id=0 LAST.id=0
 boundary=0 _N_=2
p1=1 p2=0 id=a value=3
 FIRST.id=0 LAST.id=0
 boundary=0 _N_=3
```

26

Detecting the boundary : Log

```
p1=0 p2=1 id=a value=2
 FIRST.id=0 LAST.id=0
 boundary=1 _N_=4
p1=0 p2=1 id=a value=1
 FIRST.id=0 LAST.id=0
 boundary=0 _N_=5
p1=0 p2=1 id=a value=3
 FIRST.id=0 LAST.id=1
 boundary=0 _N_=6
```

27

Detecting the boundary : Log

```
p1=1 p2=0 id=b value=3
 FIRST.id=1 LAST.id=0
 boundary=0 _N_=7
p1=1 p2=0 id=b value=5
 FIRST.id=0 LAST.id=0
 boundary=0 _N_=8
p1=0 p2=1 id=b value=3
 FIRST.id=0 LAST.id=0
 boundary=1 _N_=9
```

28

Detecting the boundary : Log

```
p1=0 p2=1 id=b value=5
 FIRST.id=0 LAST.id=1
 boundary=0 _N_=10
```

29

About the speaker

Howard Schreier

U.S. Dept. of Commerce
Washington DC 20230

(202) 482-4180

Howard_Schreier@ita.doc.gov

30