PharmaSUG 2017 - Paper PO17

Power of Interleaving and RETAIN Combination in Data Manipulation

Zongming Pan, ConnectiveRX

ABSTRACT

DATA step with MERGE statement and BY statement is a popular tool in data manipulation. Two or more sorted datasets can also be combined into a new dataset by a SET statement (instead of MERGE statement) and an accompanying BY statement. This process is called interleaving. It has been suggested that interleaving a dataset with itself is a useful technique to finish some kinds of tasks (Schreier, H., 2003). The most popular one is the processing based on group summarization via a sum statement. This 'looking forward' or 'forward looking' character via sum statement can be applied in data manipulation.

RETAIN statement which causes the values of new variables to be retained from one iteration of the DATA step to the next can amplify the 'looking forward' or 'forward looking' character. Incorporating a RETAIN statement into the interleaving process can reduce the use of PROC SORT to a single instance. PROC SORT uses memory and resources extensively and is required multiple times when match merging with the BY statement is performed.

Therefore, it is expected that the method of interleaving & RETAIN is faster than conventional 'Sort and Merge' method in data manipulation.

A series of experiments have been done in this paper to demonstrate that interleaving and RETAIN statement combination is much more efficient than the conventional Sort and Merge method in data manipulation, and the efficiency difference becomes much more significant when the data size and number of variables manipulated increase.

INTRODUCTION

We are familiar with DATA step which contains MERGE statement and BY statement. It joins observations from the input datasets into a single observation in the output dataset and is very frequently used in data manipulation.

```
DATA newdata;
   MERGE inputdata1 inputdata2;
   BY var(s);
RUN;
```

We are also familiar with DATA step which contains SET statement, but no By statement. This DATA step concatenates multiple input datasets in the SET statement into output dataset in DATA statement.

```
DATA newdata;
   SET inputdata1 inputdata2;
RUN;
```

In contrast, DATA step containing a SET statement and an accompanying By statement is not so familiar to many SAS users

```
DATA newdata;
   SET inputdata1 inputdata2;
   BY var(s);
RUN;
```

This also concatenates the datasets in the SET statement but by the variable(s) in the BY statement. This process is called interleaving. Its function or application has not been fully explored.

MERGE STATEMENT and INTERLEAVING

There are two merging procedures which are performed with MERGE statement: (1) One-to-one merging is performed without BY statement, (2) Match merging with BY statement joins the observations only when the values of the BY variables match. The second one is a popular method in data manipulation. Each time MERGE statement with BY statement is performed, the input datasets in MERGE statement must be sorted by BY groups. Unfortunately, the PROC SORT procedure is very

expensive in CPU usage and time. When multiple variables are manipulated, many PROC SORT procedures need to be performed, then the efficiency is reduced.

Interleaving process has a similar syntax as MERGE statement with BY statement, but what they do are different, so are how they do. The following codes are used as a simple example to demonstrate the process of interleaving. During the process, the input datasets (inputd1, inputd2), which have been in the order of BY group, are interleaved by the variable (*id*) in the BY statement. Within each BY group, observations will be read in the order in which input datasets are listed in the SET statement. For demonstration purpose, in this example the two input datasets are in sorted order of *id* when their original data are read in.

```
DATA inputd1;
   INFORMAT id $5. order_date mmddyy10. qty;
   INPUT    id $5. order_date mmddyy10. qty;
   FORMAT   order_date mmddyy10.;
CARDS;
C1111 5/10/2015 30
C1111 10/1/2015 30
C4444 1/1/2016  30
C6666 2/1/2016  30
;
RUN;
DATA inputd2;
   INFORMAT id $5. order_date mmddyy10. qty;
   INPUT    id $5. order_date mmddyy10. qty;
   FORMAT   order_date mmddyy10.;
CARDS;
C1111 6/10/2015 60
C1111 7/1/2015  60
C3333 3/1/2016  60
C3333 4/1/2016  60
C7777 1/15/2016 60
;
RUN;
DATA newdata;
   SET inputd1(IN=ina) inputd2(IN=inb);
   BY id;
   FORMAT flag $27.;
   IF ina AND inb THEN flag="From Both Input Datasets";
   ELSE IF ina THEN flag="From 1st Input Dataset Only";
   ELSE IF inb THEN flag="From 2nd Input Dataset Only";
RUN;

PROC PRINT DATA=newdata;
RUN;
```

Obs	id	order_date	qty	flag
1	C1111	05/10/2015	30	From 1st Input Dataset Only
2	C1111	10/01/2015	30	From 1st Input Dataset Only
3	C1111	06/10/2015	60	From 2nd Input Dataset Only
4	C1111	07/01/2015	60	From 2nd Input Dataset Only
5	C3333	03/01/2016	60	From 2nd Input Dataset Only
6	C3333	04/01/2016	60	From 2nd Input Dataset Only
7	C4444	01/01/2016	30	From 1st Input Dataset Only
8	C6666	02/01/2016	30	From 1st Input Dataset Only
9	C7777	01/15/2016	60	From 2nd Input Dataset Only

The observation number (9 here) of new dataset generated is equal to sum of observation numbers of input datasets. The observations highlighted in green are from the second input dataset—inputd2. No

observation is from the both input datasets simultaneously, as shown in the values of variable *flag*.

The input datasets in the SET statement do not have to be different. When the two input datasets are identical, this kind of interleaving is called self-interleaving which interleaves two identical datasets.

```
DATA newdata;
  SET inputd1 inputd1;
  BY var(s);
RUN;
```

At the first glance, it seems that self-interleaving does not have much practical value in data manipulation. However, it has been suggested that it is a useful technique (Schreier, 2003.). The most popular one is the processing based on group summarization. Each observation of group is populated with group summarization via sum statement. Sum statement has implicit retain function and is equivalent to using SUM function and RETAIN statement. This enables the self-interleaving to possess 'looking forward' character. But sum statement has a limitation—it can only be used for numeric variables.

RETAIN statement causes the values of new variables that are created with assignment statements to be retained from one iteration of the DATA step to the next. Unlike sum statement, RETAIN statement can work on either numeric or character variable. If RETAIN statement is incorporated into the interleaving process, it can amplify the 'looking forward' character of self-interleaving and perform broad functions in data manipulation. Unlike Sort and Merge, self-Interleaving with RETAIN statement needs only one PROC SORT even when multiple variables are manipulated. So it is expected that it will be more efficient than conventional Sort and Merge method, especially when multiple variables in large dataset are manipulated. In this paper, a series of experiments using customer data have been performed to test the hypothesis.

CUSTOMER DATA

Each observation represents an order of a customer. There are 31 variables in the dataset, including 8 key variables:

1. *Id*: de-identified customer ID.
2. *Product*: name of a product.
3. *Order_Date*: date when a product is ordered.
4. *Qty*: quantity of product.
5. *Age*: age of a customer when the customer orders the product.
6. *Gender*: gender of a customer. Valid values: 'F' and 'M'. 'F' stands for Female and 'M' for Male.
7. *State*: a USA state where a customer resides. Valid values: two characters abbreviation, such as 'PA' and 'MD'.
8. *Group*: identifies what kind of group a customer is grouped into. Valid values: 'T' and 'C'. 'T' stands for Test, 'C' for Control. Test customers receive intervention, and control customers do not.

DATA MANIPULATION

MANIPULATION RULE

There are basically three classes of manipulation at customer level:

(1), delete all observations of a customer based on the value of variable, e.g. *qty*.

(2), generate a new variable, e.g. the date when a customer first time ordered a specific product, *first_prod.*

(3), clean variables, such as *gender* and *state*.

Different manipulation rules are applied to variables according to their characters and the requirement by the project.

1. *Qty*: If quantity is missing or zero in any order of a customer, then all the orders of the customer are deleted.

2. *Age*: Because time frame possibly has multiple years, a customer could have different ages in the dataset. For simplicity in this paper, the oldest one is kept as the customer age.
3. *Gender*: If both 'F' and 'M' appear in the orders of a customer, then gender U' (Unknown) is assigned to the customer. If only one of 'F' and 'M' appears in the orders of a customer, then that value is assigned to the customer. Otherwise, 'U' is assigned.
4. *State*: If at least two valid US states appear in the orders of a customer, then state 'ZZ' is assigned to the customer. If only one valid state appears in the orders of a customer, then the state value is assigned to the customer. If no valid US state appear in the orders of a customer, then 'ZZ' is assigned.
5. First order date of a specific product: The date when a customer first time ordered a specific product is identified and assigned to a new variable called *first_prod*.
6. *Group*: Once 'T' appears in the orders of a customer, then assign 'T' to the customer. If 'C' but no 'T', then assign C' to the customer. Otherwise, assign 'U' to the customer. The priority is 'T' > 'C'.
7. Product volume: The total volume of a specific product that a customer ordered is calculated and assigned to a new variable *product_vol*.

MANIPULATION APPROACH

During the interleaving process, according to their specific rules and characters, a variety of approaches were adopted to manipulate variables by means of RETAIN statement and a series of flag variables which are temporary.

Please check the Interleaving & Retain codes in the next section for details.

SAS CODES

The input dataset All has been sorted by *id* and *order_date* before interleaving or merging. All variables which need to be cleaned have been renamed: *age* to *orig_age*, *gender* to *orig_gender*, *state* to *orig_state*, and *group* to *orig_group*. To optimize the Sort & Merge codes, *qty* cleaning which could delete observations was performed first, and new variables *first_prod* and *product_vol* were generated at the end. Besides, only necessary variables were kept and WHERE statement instead of IF statement was used if applicable.

INTERLEAVING and RETAIN CODES

```
DATA thelib.all_int(DROP=flag: del_: all_:);
  SET all(IN=ina KEEP=id qty orig_age orig_gender orig_state
          orig_group product order_date) all(IN=inb);
  BY id;
  RETAIN del_qty age flag_m flag_f all_state group flag1 flag2 first_prod
         flag_prod;
  FORMAT all_state $24. state $2. first_prod mmddyy10.;
  IF orig_state NOT IN (&us_states.) THEN orig_state='ZZ';
  IF ina THEN DO;
    IF FIRST.id THEN DO;
      ***(1) Delete the Customer with Missing or Zero Quantity Order***;
      del_qty='N';

      *****(2) Clean Age: Keep the Oldest One ***;
      age=orig_age;

      ***(3) Clean Gender***;
      flag_m=' ';
      flag_f=' ';

      ***(4) Clean State***;
      all_state=STRIP(orig_state);

      ***(5) Clean Group***;
```

```
      group=' ';
      flag1='0';
      flag2='0';

      ***(6) The First Order Date of a Specific Product of a Customer***;
      first_prod=.;
      flag_prod='0';

      ***(7) All Volume of a Specific Product that a Customer Ordered***;
      product_vol=0;
    END;
    ELSE DO;
      ***(4) Clean State***;
      IF INDEX(all_state, STRIP(orig_state))=0 THEN
                  all_state=STRIP(all_state)||'+'||STRIP(orig_state);
    END;

    ***(1) Delete a Customer with any Missing or Zero Quantity Order***;
    IF qty<=0 THEN del_qty='Y';

    ***(2) Clean Age: Keep the Oldest One***;
    If orig_age>age THEN age=orig_age;

    ***(3) Clean Gender***;
    If orig_gender='M' THEN flag_m='Y';
    ELSE IF orig_gender='F' THEN flag_f='Y';

    ***(5) Clean Group***;
    IF orig_group='T' THEN DO;
      flag1='1';
      group='T';
    END;
    ELSE IF orig_group='C' AND flag1='0' THEN DO;
      flag2='1';
      group='C';
    END;
    ELSE group='U';

    ***(6)The First Order Date of a Specific Product of a Customer***;
    IF product="&product." AND flag_prod='0' THEN DO;
      flag_prod='1';
      first_prod=order_date;
    END;

    ***(7)All Volume of a Specific Product that a Customer Ordered***;
    IF product="&product." THEN product_vol+qty;
  END;
  ELSE IF inb THEN DO;
    ***(1) Delete a Customer with any Missing or Zero Quantity Order***;
    IF del_qty='N';

    ***(3) Clean Gender***;
    IF flag_m='Y' AND flag_f='Y' THEN gender='U';
    ELSE IF flag_m='Y' THEN gender='M';
    ELSE IF flag_f='Y' THEN gender='F';
    ELSE gender='U';
```

```
      ***(4) Clean State***;
      IF COUNTW(STRIP(all_state), '+')=1 THEN state=STRIP(all_state);
      ELSE IF COUNTW(STRIP(all_state), '+')>2 THEN state='ZZ';
      ELSE IF COUNTW(STRIP(all_state), '+')=2 AND INDEX(STRIP(all_state),
                                              'ZZ')=0 THEN state='ZZ';
      ELSE IF COUNTW(STRIP(all_state), '+')=2 AND
                          INDEX(STRIP(all_state),'ZZ')=1 THEN
                             state=SUBSTR(STRIP(all_state), 3, 2);
      ELSE IF COUNTW(STRIP(all_state), '+')=2 AND
                          INDEX(STRIP(all_state),'ZZ')=3 THEN
                            state=SUBSTR(STRIP(all_state), 1, 2);
      OUTPUT;
    END;
  RUN;
```

SORT and MERGE CODES

```
  ***(1) Delete a Customer with any Missing or Zero Quantity Order***;
  PROC SORT DATA=all(KEEP=id qty) OUT=no_qty(KEEP=id) NODUPKEY;
    WHERE qty<=0;
    BY id;
  RUN;
  DATA all;
    MERGE all(IN=ina) no_qty(IN=inb);
    IF ina AND NOT inb;
    BY id;
  RUN;

  ***(2) Clean Age: Keep the Oldest One ********;
  PROC SORT DATA=all(KEEP=id orig_age) OUT=age NODUPKEY;
    BY id orig_age;
  RUN;
  DATA age;
    SET age;
    BY id orig_age;
    IF LAST.id;
    RENAME orig_age=age;
  RUN;
  DATA all;
    MERGE all(IN=ina) age(IN=inb);
    BY id;
    IF ina AND inb;
  RUN;

  ***(3) Clean Gender**;
  DATA m(KEEP=id) f(KEEP=id) u(KEEP=id);
    SET all(KEEP=id orig_gender);
    IF orig_gender='M' THEN OUTPUT m;
    ELSE IF orig_gender='F' THEN OUTPUT f;
    ELSE OUTPUT u;
  RUN;
  PROC SORT DATA=m NODUPKEY;
    BY id;
  RUN;
  PROC SORT DATA=f NODUPKEY;
    BY id;
  RUN;
```

```
PROC SORT DATA=u NODUPKEY;
  BY id;
RUN;
DATA gender;
  MERGE m(IN=ina) f(IN=inb) u(IN=inc);
  BY id;
  IF ina AND inb THEN gender='U';
  ELSE IF ina THEN gender='M';
  ELSE IF inb THEN gender='F';
  ELSE gender='U';
RUN;
DATA all;
  MERGE all(IN=ina) gender(IN=inb);
  BY id;
  IF ina AND inb;
RUN;

***(4) Clean State***;
PROC SORT DATA=all(KEEP=id orig_state) OUT=state NODUPKEY;
  BY id orig_state;
  WHERE orig_state IN (&us_states.);
RUN;
PROC FREQ DATA=state NOPRINT;
  TABLES id/LIST MISSING OUT=st_num(DROP=percent);
RUN;
DATA all;
  MERGE all(IN=ina) st_num(IN=inb);
  BY id;
  IF ina;
  IF count=1 THEN state=STRIP(orig_state);
  ELSE state='ZZ';
  DROP count;
RUN;

***(5) Clean Group***;
DATA grp_t(KEEP=id) grp_c(KEEP=id) grp_u(KEEP=id);
  SET all(KEEP=id orig_group);
  IF orig_group='T' THEN OUTPUT grp_t;
  ELSE IF orig_group='C' THEN OUTPUT grp_c;
  ELSE OUTPUT grp_u;
RUN;
PROC SORT DATA=grp_t NODUPKEY;
  BY id;
RUN;
PROC SORT DATA=grp_c NODUPKEY;
  BY id;
RUN;
PROC SORT DATA=grp_u NODUPKEY;
  BY id;
RUN;
DATA group;
  MERGE grp_t(IN=ina) grp_c(IN=inb) grp_u(IN=inc);
  BY id;
  IF ina THEN group='T';
  ELSE IF inb THEN group='C';
  ELSE group='U';
RUN;
```

```
DATA all;
   MERGE all(IN=ina) group(IN=inb);
   BY id;
   IF ina AND inb;
RUN;

***(6)The First Order Date of a Specific Product of a Customer***;
DATA first_prod;
   SET all(KEEP=id product order_date);
   BY id order_date;
   WHERE product =”&product.”;
   IF FIRST.id;
   DROP product;
   RENAME order_date=first_prod;
RUN;
DATA all;
   MERGE all(IN=ina) first_prod(IN=inb);
   IF ina;
   BY id;
RUN;

***(7)All Volume of a Specific Product that a Customer Ordered***;
PROC MEANS DATA=all(KEEP=id qty product) SUM NOPRINT;
   WHERE product =”&product.”;
   VAR qty;
   CLASS id;
   OUTPUT OUT=id_qty(DROP=_:) SUM=product_vol;
RUN;

DATA thelib.all_merg;
   MERGE all(IN=ina) id_qty(IN=inb);
   BY id;
   IF ina;
   IF product_vol=. THEN product_vol=0;
RUN;
```

CODES RUNNING

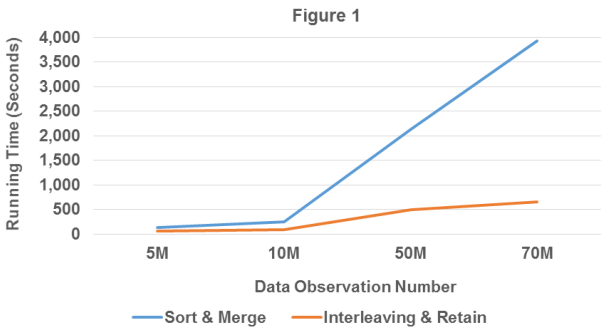
To avoid possible noise and get very similar environments for fair/accurate comparison, all the codes were run in the night. A pair of programs (Interleaving & RETAIN vs Sort & Merge) were run in a row with an interval as short as possible. All the log files were checked to make sure that there were no errors and warnings. The output datasets generated by the pair of programs using the same input data were compared. It turned out that they all were identical, meaning that the pair of programs did identical jobs as intended.

Two sets of codes, which are same in structure or algorithm except the number of variables manipulated, were run. The first set, which is listed in the last section, manipulated 7 variables. The second set manipulated only two variables: *qty* and *gender*.

The datasets with different observation number: 5M, 10M, 50M and 70M (M stands for million) were used. Multiple duplicate runs were performed with same code and same dataset. There were some variations in the time amount for each case probably due to different server usage conditions. However the trends were always the same. The results of one run for different code sets and dataset sizes are shown in the next section.

RESULTS

The first set: manipulated 7 variables.



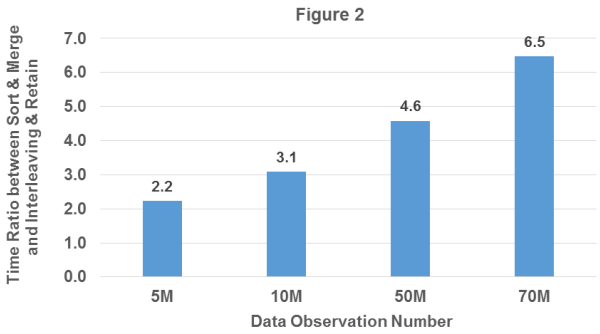
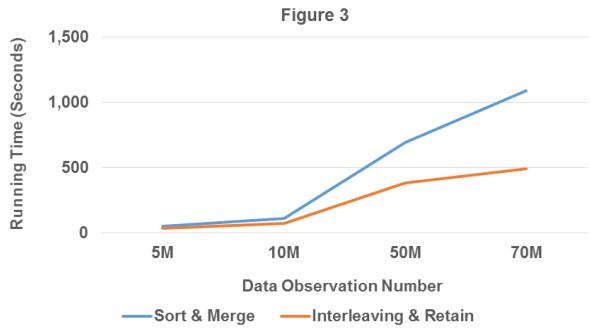
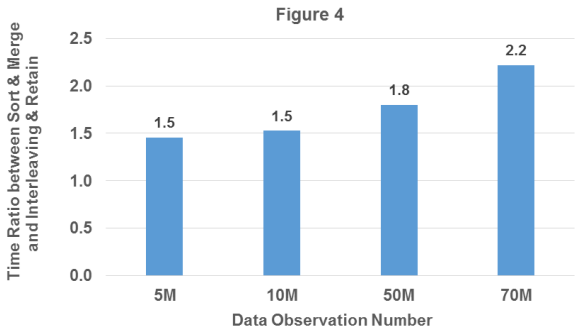


Figure 1 shows the running time of each case. For the same data, conventional Sort & Merge method took more time than Interleaving & Retain method to finish the job. In other words, conventional Sort & Merge method was slower than Interleaving & Retain method when they did the same job. When data size increased from 5M to 70M, the time needed to finish the task using conventional Sort & Merge method increased much more significantly than the one using Interleaving & Retain method. This phenomenon is clearly shown in Figure 2. With data of 5 million observations, the Interleaving & Retain method was 2.2 times faster than conventional Sort & Merge method. With the data of 70M observations, the Interleaving & Retain method was 6.5 times faster than conventional Sort & Merge method. It implies that efficiency difference between Interleaving & Retain method and conventional Sort & Merge method was related to data size and became even more significant with larger data.

The second set: manipulated 2 variables (*qty* and *gender*)





Like the first set, the Interleaving & Retain method in the second set was always faster than the conventional Sort & Merge method with same data (Figure 3), and the time ratio between conventional Sort & Merge method and Interleaving & Retain method increased with data size but not dramatically. As shown in Figure 4, the time ratio increased from 1.5 with 5M data to 2.2 with 7M data, again showing that efficiency difference between Interleaving & Retain and conventional Sort & Merge method was related to data size and became larger with larger data.

Compared to the first set, as expected, the second set was much faster correspondingly due to fewer variables manipulated. Besides, the efficiency difference between Interleaving & Retain method and conventional Sort & Merge method in the second set was smaller than that in the first set, implying that efficiency difference between Interleaving & Retain method and Sort & Merge method was also related to number of variables manipulated, and became more significant with more variables manipulated.

CONCLUSION

In addition to its concise codes (just one DATA step), the Interleaving & Retain method is more efficient than conventional Sort & Merge method in manipulating data. The efficiency saves much more significantly when both the data size and number of variables manipulated increase. The Interleaving & Retain method is a very optimal method in manipulating multiple variables in large dataset.

REFERENCES

Schreier, Howard. 2003. “Interleaving a Dataset with Itself: How and Why.” *Proceedings of the NESUG 16 Conference*, Arlington VA. Available at http://www.howles.com/saspapers/cc002.pdf.

ACKNOWLEDGMENTS

The author would like to thank Sangeeta Tendon for her encouragement, support and help.

CONTACT INFORMATION

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

Zongming Pan
ConnectiveRx,
Address: 480 Norristown Road, Suite 100, Blue Bell, PA 19422
Email: Zongming.pan@connectiveRx.com

TRADEMARKS