SAS Dow Loop Approach

Section 1: Introduction

The New York Stock Exchange (NYSE) provides a Trades and Quotes (TAQ) database that contains, in separate files, every trade and quote captured by the consolidated tape. The TAQ data begins in 1993 and is hosted on WRDS in daily files. Trades are stored in files whose name begins with the letters ct followed by the date and the extension used by SAS for its data files. For example, the daily data for 22nd October 2008 is stored in ct_20081022.sas7bdat. The corresponding quotes are stored in cq_20081022.sas7bdat. It is customary for researchers to screen trades and quotes, then combine them using some criteria, then calculate and output statistics from the combined data. The most common procedure followed by researchers is a step-by-step linear approach: screen the trades and save them, screen the quotes and save them in another file, combine the screened trades and quotes either in a DATA step or by using PROC SQL, then calculate and output statistics in a final DATA step. While this procedure may be acceptable when the trades and quotes files are small, it is extremely inefficient and time-consuming when input file sizes are large. For instance, the January 2, 1998 quotes file has a size of 84.4 MB, but the January 2. 2008 quotes file has a size of 17.5 GB.

This note introduces an efficient SAS procedure that consolidates into a single DATA step the screening, merging, and calculation of statistics from the trades and quotes data. In the SAS community, such a DATA step has come to be known as the “DOW Loop”. The SAS code for this procedure is displayed in Table 1. The purpose of this note is to walk the reader through the DOW Loop implemented in Table 1 so that the reader can understand how exactly the loop works and can make appropriate modifications to the program to suit their own preferences. The loop is illustrated by using the trades data that appears in Panel A of Table 2 and by merging it with the quotes data in Panel B of the table.

The procedure outlined here takes advantage of the fact that the monthly trades and quotes files are sorted on symbol, date, and time. The logic of the procedure is very simple: it reads in a trade only if it passes a screen, then looks for a screened matching quote from the quotes file. By default, the quote that matches the trade is the first screened quote that is 5 seconds or more prior to the trade (Lee and Ready, 1991). If the trade does not pass the screen, it is not read in (in SAS it is possible to avoid reading in an observation if that observation does not pass certain criteria – more on this below) and the procedure goes to the next trade. Once the a matching quote is found, the trade is output together with the matching bid and offer. The matching quote is retained in memory just in case it also matches the next trade. At any given time, there are two quotes in memory – they are referred to as the *current* quote and the *retained* quote. If some trade does not match either of the two quotes, a new quote is read in. The current quote is moved to the retained quote, and the new quote is stored as the current quote. This comparison and reading procedure continues until a matching quote is found. The matched trade and quote are written out and the procedure starts all over again with the reading of the next trade. If, for some trade, no quote from the same day matches, the last valid quote in the memory is matched provided the symbol is the same. For example, at 9:30 am every trading day, there is usually an absence of quotes that pass the typical screens applied in the microstructure literature. Such trades will be matched to the quote prevailing at the close of trading the previous trading day.

Section 2: Program Details: One Symbol, One Date

As it is showed in the below program listing code, the SAS dataset that will contain the output data is declared. The declaration has three options: it *drops* temporary variables that are created during the merging and screening of the data, it *renames* the matching bid and ofr (which are temporarily stored in the variables mbid and mofr), and it turns on the *sortedby* option – because the input data is already sorted, turning on this option will let SAS know that the output dataset is also already sorted and there is no further need for it to be sorted. In lines 2 and 3, temporary variables to store the values of symbols coming from the quotes file are set up to allow the program to distinguish between values of symbol coming from the trades file and from the quotes file. Further, the retain statement in line 4 ensures that two quotes are retained from one iteration of the DATA step to the next.

```
/*****************************************************************************/
/*************** W R D S   R E S E A R C H   A P P L I C A T I O N S ***************/
/*****************************************************************************/
/* Summary    : Table I: Program Listing for the DOW Loop                     */
/* Date       : November 2, 2011                                              */
/* Use the DOW loop to read data from different datasets, clean,             */
/* match and output it all at the same time, saving the I/O resources on the server. */
/*****************************************************************************/

data matched /*Line 1*/
(drop = cqsymbol cqdate cqtime bid bidsiz ofr ofrsiz lagged_match timediff
   exact_match rbid rofr rcqdate cqtime rcqtime rcqsymbol
rename =  (mbid=BID mofr=OFR)
sortedby = symbol date time);
attrib CQSYMBOL length=$10. ; /*Line 2*/
attrib RCQSYMBOL length=$10.; /*Line 3*/
retain CQSYMBOL RCQSYMBOL CQDATE RCQDATE CQTIME RCQTIME BID RBID OFR ROFR
end_of_quotes_file;  /*Line 4*/

set taq.ct_19980102 (where = (time >= '9:30:00'T  AND  time <= '16:00:00'T)) ;  /*Line
5*/
do until (exact_match = 1 OR lagged_match = 1 OR end_of_quotes_file = 1 ); /*Line 6*/
if symbol>cqsymbol OR (symbol = cqsymbol AND date > cqdate) then
  go to READQUOTE; /*Line 7*/
else /*Line  9-12*/
if cqsymbol > symbol OR cqdate>date then do; lagged_match=1;go to END_TIMEOKLOOP;End;
else do; TIMEDIFF = time - cqtime; TIMEDIFF = time - cqtime; /*Line 13-15*/
if timediff = 5 then exact_match = 1; /*Line 16-25*/
else if timediff < 5 then lagged_match = 1;
else do;
  READQUOTE: ;
  RCQSYMBOL = CQSYMBOL;
  RCQDATE = CQDATE;
  RCQTIME = CQTIME;
  RBID = BID;
  ROFR = OFR;

set taq.cq_19980102 (rename = (symbol=CQSYMBOL date=CQDATE time=CQTIME) /*Line 26-30*/
where=(cqtime>='9:30:00'T AND cqtime<='16:00:00'T AND ofr>0 AND (ofr-bid)/bid<= 0.1))
end = end_of_quotes_file ;end;end; END_TIMEOKLOOP: ; end;
/*Line 31-37: Depending on the break point encountered above, select the match array*/
if exact_match then do;
   MBID = bid;  label mbid = 'Matching Bid Price';
   MOFR = ofr;  label mofr = 'Matching Offer Price' ;
   MTIME = cqtime; format mtime time. ;  label mtime = 'Time of the Matching Quote' ;
   Output;end;
else if lagged_match then do; /*Line 38-46*/
  if symbol = rcqsymbol then do;MBID = rbid;MOFR = rofr;MTIME = rcqtime;output;End;
else do; MBID = .; MOFR = .; MTIME = .; output;End;end;              /*Line 47-54*/
else if end_of_quotes_file then do;MBID=.;MOFR=.;MTIME=.;Output;end;/*Line 55-62*/
run;

/* ********************************************************************** */
/* *************  Material Copyright Wharton Research Data Services  *************** */
/* ****************************** All Rights Reserved ****************************** */
/* ********************************************************************** */
```

In line 5, SAS pre-screens for whether the first observation in the trades file should be read into the DATA step (technically into the *program data vector* or *pdv*). The (where = (time >= '9:30:00'T AND time <= '16:00:00'T)) option that appears in this set statement decides to read the next physical observation in the trades file into the *pdv* only if that observation falls between 9:30 am and 4:00 pm. The first observation in the trades file (CTID = 1; CTID and CQID have been added to the illustrative data in Table 2 to ease the discussion in this document – they do not appear in the TAQ data) is recorded at 9:25 am. Because it occurred prior to 9:30 am, SAS will not read it into the *pdv*. This particular where= option is an example of how trades data could be screened before further processing. Researchers can modify this where= option to suit their own criteria for screening trades.

Table: Trade-Quote Matching Data

Panel A: Trades

CTID	SYMBOL	DATE	TIME	PRICE
1	A	1/2/2001	9:25:00	43.45
2	A	1/2/2001	9:35:00	44.10
3	A	1/2/2001	15:55:00	44.25
4	A	1/2/2001	15:57:38	44.15
5	A	1/3/2001	9:32:00	45.12
6	A	1/3/2001	10:05:00	45.10
7	AA	1/2/2001	9:45:01	175.83
8	AAA	1/2/2001	14:45:00	12.30

Panel B: Quotes

CQID	SYMBOL	DATE	TIME	BID	OFR
1	A	1/2/2001	9:22:00	0.00	45.00
2	A	1/2/2001	9:32:00	10.00	45.20
3	A	1/2/2001	9:34:45	44.00	44.20
4	A	1/2/2001	9:36:02	44.05	44.25
5	A	1/2/2001	15:54:55	43.95	44.30
6	A	1/2/2001	16:00:01	44.00	44.50
7	A	1/3/2001	8:04:56	44.50	45.50
8	A	1/3/2001	9:35:00	44.45	45.50
9	A	1/3/2001	14:22:51	44.52	44.87
11	AAA	1/2/2001	8:56:23	10.00	2.00
12	AAA	1/2/2001	9:44:56	12.25	12.45
13	AAA	1/2/2001	15:58:38	12.81	13.38
14	AAA	1/3/2001	8:33:15	0.00	0.00
15	AAA	1/3/2001	9:51:42	12.45	13.12

The program then examines whether the next physical observation in the trades file should be read in. Because the observation with CTID = 2 did occur between 9:30 am and 4:00 pm, it is read into the *pdv*. The do loop starting in line 6 and ending at line 30 looks for a suitable quote to match this trade. This do loop will be exited only if an exact match (a quote that was posted 5 seconds before the trade) is found or while reading the quotes file the program reads a quote that is less than 5 seconds before the trade or if the quotes file is completely exhausted. When the program first encounters this do statement, the variables exact_match, lagged_match, and end_of_quotes_file all have missing values. Therefore, the program enters the do loop.

The first two lines (7, 8) of the do loop checks for whether the symbol and date for the current quote in the memory matches the symbol and date for the current trade. Because no quote has yet been read into the *pdv*, the program drops to the do loop beginning in line 14 and ending in line 29. In line 15, the program calculates the difference in time, in seconds, between the trade and quote (TIMEDIFF). Again, because no quote has been read, TIMEDIFF will be missing. In lines 16 and 17 the program checks whether an exact match is found or if the program has read a quote that is less than 5 seconds before the trade. Again, because no quote has been read, none of these conditions are met. Therefore, the program enters the do loop beginning in line 19 and ending in line 28. This is where the program reads and screens quotes. This block of codes has a name, READQUOTE, that appears in line 20. As will be seen later, under some conditions the program will enter this block bypassing the checks above. The naming of the block of code enables the program to enter that block directly.

Before a new quote is read, the program saves the existing quote to memory so that it can be used if necessary. This is accomplished in lines 21-25 by creating temporary variables RCQSYMBOL, RCQDATE, etc. Because no quote has been read yet, these variables will contain missing values.

In line 26, the program decides whether the next quote in the quotes file should be read in. That quote is read into the *pdv* only if it satisfies all the criteria listed in the where= option of the set statement appearing in line 26. Currently, a quote is read into the *pdv* only if it was posted between 9:30 am and 4:00 pm, if both the bid and offer prices are greater than 0.0, if the offer is greater than or equal to the bid and if the bid-ask spread is 10% or less. These conditions can be modified to suit the preferences of the researcher by making suitable changes to the where= option. Note, also, the end = end_of_quotes_file option that appears in this statement. It creates the variable end_of_quotes_file that will indicate if the end of the quotes file is reached.

The program will conclude that the first two observations in the quote file (CQID = 1 and 2) do not meet the criteria in the where= option (the first because it was posted before 9:30 am and the second because the spread > 10%). Therefore, the program reads the third observation (CQID = 3) into the *pdv*.

At this stage the variables of interest in the *pdv* have the following values:

SYMBOL	**DATE**	**TIME**	**PRICE**	
A	1/2/2001	9:35:00	44.10	
RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
missing	missing	missing	missing	missing
CQSYMBOL	**CQDATE**	CQTIME	**BID**	**OFR**
A	1/2/2001	9:34:45	44.00	44.20

The program then goes to the top of the do loop beginning in line 6. The if and else if conditions in lines 7 and 8 are not met. Therefore the program enters the else block beginning in line 13. The program drops to line 15 where TIMEDIFF is calculated (TIMEDIFF = TIME – CQTIME = 9:35:00 – 9:34:45 = 15 seconds). While this is a potential match between trade and quote, the program will check whether a better match can be made. Neither the if or the else if conditions in lines 16 and 17 are met. Therefore, the program will go to the READQUOTE block to read another quote. Before another quote is read, however, the current quote is retained in the variables whose names are prefixed with R. In line 26, the next quote is read into the *pdv*. At this stage the *pdv* contains (among other things):

SYMBOL	DATE	TIME	PRICE
A	1/2/2001	9:35:00	44.10

RCQSYMBOL	RCQDATE	RCQTIME	RBID	ROFR
A	1/2/2001	9:34:45	44.00	44.20

CQSYMBOL	CQDATE	CQTIME	BID	OFR
A	1/2/2001	9:36:02	44.05	44.25

The program goes to the top of the do loop beginning in line 6. The if and else if conditions in lines 7 and 8 are not met. Therefore, the program enters the else block beginning in line 13. The program drops to line 15 where it calculates TIMEDIFF = -62. It is now known that the retained quote is the best quote to match this trade. In line 17 the program detects that TIMEDIFF < 5.0 and sets the value of lagged_match = 1. The program goes to the top of the do loop beginning in line 6. Because lagged_match = 1, one of the conditions for exiting the do loop has been triggered, and the program extis the do loop and drops to line 31. The if condition in line 31 is not met but the else if condition in line 38 is met. Therefore the block of code in lines 39 – 54 is executed. This block of code copies the values of the retained quote to the “match” variables. The match variables hold the values of the quote that match the trade in the *pdv*. In line 40, the program ensures that the symbol of the trade and the retained quote are the same. Because they are, the retained quote values are copied to the match variables in lines 42 – 44 and the result is output in line 45. In the output dataset, the names of the matching bid and offer price are changed from MBID and MOFR to BID and OFR. Therefore, the following variables with their given values will be written to the output dataset:

SYMBOL	DATE	TIME	PRICE	BID	OFR
A	1/2/2001	9:35:00	44.10	44.00	44.20

While no statistics based on the matched trade and quote are output in this example, the program could have calculated, e.g., the bid-ask spread in the block of code in lines 42 – 44 where the match variables are given their values. This could be accomplished by inserting one or more program statements in that block. Any program statements inserted here should also be inserted in the block of code in lines 33 – 35 as match variables sometimes receive their values in this block.

The program now goes to the top of the DATA step and reads the next trade (CTID=3) in line 5. The retain statement in line 4 retains the values of the variables that appear in the statement from one DATA step to the next. Therefore, the *pdv* contains:

SYMBOL	DATE	TIME	PRICE
A	1/2/2001	15:55:00	44.25

RCQSYMBOL	RCQDATE	RCQTIME	RBID	ROFR
A	1/2/2001	9:34:45	44.00	44.20

CQSYMBOL	CQDATE	CQTIME	BID	OFR
A	1/2/2001	15:54:55	43.95	44.30

exact_match	lagged_match	end_of_quotes_file
missing	Missing	missing

The program calculates TIMEDIFF (=22,738) then reads the next quote (CQID=5) and goes to the top of the do loop in line 6. The *pdv* contains:

SYMBOL	**DATE**	**TIME**	**PRICE**	
A	1/2/2001	15:55:00	44.25	
RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
A	1/2/2001	9:36:02	44.05	44.25
CQSYMBOL	**CQDATE**	**CQTIME**	**BID**	**OFR**
A	1/2/2001	15:54:55	43.95	44.30
exact_match	**lagged_match**	**end_of_quotes_file**		
missing	Missing	missing		

The program calculates TIMEDIFF (=5). The if condition in line 16 is met because the current quote is exactly 5 seconds prior to the trade in the *pdv*. The variable exact_match takes on a value of 1 and the program transfers control to the top of the do loop in statement 6. Because one of the conditions that triggers the exit of the program from this do loop is met, the program exits the do loop and program control is transferred to line 31. The if condition in line 31 is met. The values of the current quote are copied to the match variables and the following is written to the output dataset:

SYMBOL	DATE	TIME	PRICE	BID	OFR
A	1/2/2001	15:55:00	44.25	43.95	44.30

Section 3. Program Details: Date and Symbol Transitions

In this section three examples demonstrate how the program transitions from one date to the next, how the program transitions from a symbol that appears in the trade data but not in the quotes, and finally how the program transitions from one symbol to the next.

After writing the last matching trade and quote to the output dataset, the program goes to the top of the DATA step and reads in the next screened trade from the trades file (CTID = 6). The *pdv* contains:

SYMBOL	**DATE**	**TIME**	**PRICE**	
A	1/3/2001	10:05:00	45.15	
RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
A	1/2/2001	15:54:55	43.95	44.30
CQSYMBOL	**CQDATE**	**CQTIME**	**BID**	**OFR**
A	1/3/2001	9:35:00	44.45	45.50
exact_match	**lagged_match**	**end_of_quotes_file**		
missing	missing	missing		

The program calculates TIMEDIFF (=1,800), then reads the next quote (CQID=9). The variables of interest in the *pdv* are:

SYMBOL	**DATE**	**TIME**	**PRICE**	
A	1/3/2001	10:05:00	45.15	

RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
A	1/3/2001	9:35:00	44.45	45.50

CQSYMBOL	**CQDATE**	**CQTIME**	BID	OFR
A	1/3/2001	14:22:51	44.52	44.87

exact_match	**lagged_match**	**end_of_quotes_file**
missing	missing	missing

The program calculates TIMEDIFF (= –15,471). The else if condition in line 17 is met, lagged_match=1 and the program outputs:

SYMBOL	**DATE**	**TIME**	**PRICE**	**BID**	**OFR**
A	1/3/2001	10:05:00	45.15	44.45	45.50

Thus, the program transitions smoothly from one date to another. The program goes to the top of the DATA step and reads in the next screened trade (CTID=7). The *pdv* contains:

SYMBOL	**DATE**	**TIME**	**PRICE**	
AA	1/2/2001	9:45:01	175.83	

RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
A	1/3/2001	9:35:00	44.45	45.50

CQSYMBOL	**CQDATE**	**CQTIME**	**BID**	**OFR**
A	1/3/2001	14:22:51	44.52	44.87

exact_match	**lagged_match**	**end_of_quotes_file**
missing	missing	missing

The program enters the do loop in line 6. The if condition in line 7 is met as symbol > cqsymbol and program control is transferred to the READQUOTE block (line 20). The current quote is copied to the retained quote and the set statement in line 26 brings CQID=12 into the *pdv.* The *pdv* contains:

SYMBOL	**DATE**	**TIME**	**PRICE**	
AA	1/2/2001	9:45:01	175.83	

RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
A	1/3/2001	14:22:51	44.52	44.87

CQSYMBOL	**CQDATE**	**CQTIME**	**BID**	**OFR**
AAA	1/2/2001	9:44:56	12.25	12.45

exact_match	**lagged_match**	**end_of_quotes_file**
missing	missing	missing

Program control transfers to line 6, the if condition in line 8 is met, lagged_match=1, and the do loop in lines 6–30 is exited. Program control is transferred to line 31, the if condition in line 38 is met and the program enters the do loop beginning in line 39. The if condition in line 40 that checks whether the trade and retained quote symbols are the same is unmet, therefore the program enters the do loop in line 49. This situation is triggered when the quote read into the *pdv* is for a symbol that lexically appears after the symbol for the trade in memory. Because, there are no matching quotes for this trade, therefore the following is written to the output dataset:

SYMBOL	**DATE**	**TIME**	**PRICE**	**BID**	**OFR**
AA	1/2/2001	9:45:01	175.83	missing	missing

The program "recognizes" that symbol AA has no quotes and writes out the one trade with the bid and offer prices as missing. If additional trades for this symbol appeared in the data, they would all be written out to the output dataset with missing values for bid and offer. Additionally, if there exist, in the quotes file, quotes for a symbol for which no trades exist in the trades file, they are simply ignored by the program.

The program next goes to the top of the DATA step and reads in the next screened trade (CTID=8). The *pdv* contains:

SYMBOL	**DATE**	**TIME**	**PRICE**	
AAA	1/2/2001	14:45:00	12.30	
RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
A	1/3/2001	14:22:51	44.52	44.87
CQSYMBOL	**CQDATE**	**CQTIME**	**BID**	**OFR**
AAA	1/2/2001	9:44:56	12.25	12.45
exact_match	**lagged_match**	**end_of_quotes_file**		
missing	missing	missing		

The program enters the do loop in line 6, calculates TIMEDIFF (=18,004) and reads the next quote. After reading the next quote the *pdv* contains:

SYMBOL	**DATE**	**TIME**	**PRICE**	
AAA	1/2/2001	14:45:00	12.30	
RCQSYMBOL	**RCQDATE**	**RCQTIME**	**RBID**	**ROFR**
AAA	1/2/2001	9:44:56	12.25	12.45
CQSYMBOL	**CQDATE**	**CQTIME**	**BID**	**OFR**
AAA	1/2/2001	15:58:38	12.81	13.38
exact_match	**lagged_match**	**end_of_quotes_file**		
missing	missing	missing		

Again, the program enters the do loop in line 6 and calculates TIMEDIFF (= –4,418). The if condition in line 17 is met, lagged_match=1, the program goes to the top of the do loop in line 6. Because lagged_match=1, the do loop in lines 6-30 is exited and the program goes to line 31. The if condition in line 38 is met, the if condition in line 40 is also met and program writes the following to the output dataset:

SYMBOL	**DATE**	**TIME**	**PRICE**	**BID**	**OFR**
AAA	1/2/2001	14:45:00	12.30	12.25	12.45

Therefore, the program transitions seamlessly from one symbol to the next. The program now goes to the top of the DATA statement and attempts to read the next trade. Because there are no additional trades in the trades file the DATA step is exited and the program ends. The output dataset will contain the following observations:

SYMBOL	DATE	TIME	PRICE	BID	OFR
A	1/2/2001	9:35:00	44.10	44.00	44.20
A	1/2/2001	15:55:00	44.25	43.95	44.30
A	1/2/2001	15:57:38	44.15	43.95	44.30
A	1/3/2001	10:05:00	45.15	44.45	45.50
AA	1/2/2001	9:45:01	175.83	missing	missing
AAA	1/2/2001	14:45:00	12.30	12.25	12.45

The trade-quote matching algorithm takes advantage of the fact that the input datasets were both sorted on symbol, date, and time. The output dataset is guaranteed to be sorted on symbol, date, and time. Thus, the sortedby=symbol date time option should be turned on for the output dataset (line 1). Any further processing of the output dataset in DATA steps or PROC steps can then take advantage of the fact that this dataset is already sorted and no resorting is required. While resorting of the output dataset that appears in this note would be a trivial exercise for SAS, when an output dataset becomes large the DOW loop matching of trades and quotes results in a very substantial reduction in time taken to process the data.