Paper 7820-2016

Using a SAS® Hash Object to Speed and Simplify the Survey Cell Collapsing Process

Ahmed Al-Attar, AnA Data Warehousing Consulting LLC, McLean, VA

ABSTRACT

This paper introduces an extremely fast and simple implementation of the survey cell collapsing process. Prior implementations had used either several SQL queries or numerous DATA step arrays, with multiple data reads. This new approach uses a single hash object with a maximum of two data reads. The hash object provides an efficient and convenient mechanism for quick data storage and retrieval (sub-second total run time).

INTRODUCTION

While working at the US Census Bureau, a friend and colleague from a different division at the Census Bureau sent me an email, to my personal email address, seeking help with implementing Survey Cell Collapsing solution.

All I got from him are the following collapsing requirements with a spreadsheet of artificial survey results data.

- Calculate the Adjustment factor = popct/wegt.
- Collapse any cell that has < 35 persons or adjustment factor < .67 or > 4 with the cell that is closest scale value.
- Collapse only within sex and race. Do not collapse <15 with 15+
- Collapse means adding the two cells unwegt and weighted cells and calculate the new scale value.

  New scale value= ((cell1 scale_value*cell1 weght) + (cell2 scale_value*cell2 weght))/ (weght1+weght2)

Cell Number	Sex	Race	Age	Scale Value	unwegt	popct	wegt
1	Male	White Alone	<1	601	716	2503603	1552194
2			1	603	12	41959.82682	26014.42458
…			…	…	…	…	…
16			18-19	18	741	2698875	861641
17			20-21	27	683	2737338	1915384
18			22-24	29	1065	4427703	4179955
…			…	…	…	…	…
32			85+	106	260	1446173	358819
33	Female	White Alone	<1	701	358	1122112	504566
34			1	703	321	1201937	1779974
…			…	…	…	…	…
48			18-19	118	694	2725271	2224527
49			20-21	127	625	2646054	1996393
50			22-24	129	964	3953523	1148473
…			…	…	…	…	…
64			85+	206	475	2613759	2111431
65	Male	Black Alone	0-4	821	35	192592.7684	155579.1263
…			…	…	…	…	…
71			30-34	345	34	188427.0097	175348.7282
72			35-39	347	489	2023880	585685
73			40-44	349	302	1011554	106363
74			45-49	355	22	73689.36424	7748.298013
…			…	…	…	…	…
77			65+	365	378	1288817	542540

Figure 1: Artificial survey results table

SOLUTION

Considering the requirements I was given, I implemented a SAS® macro “collapse_data” that treats some the listed requirements as input parameters, in order to generalize it and allow it to be used it with other requirements/conditions.

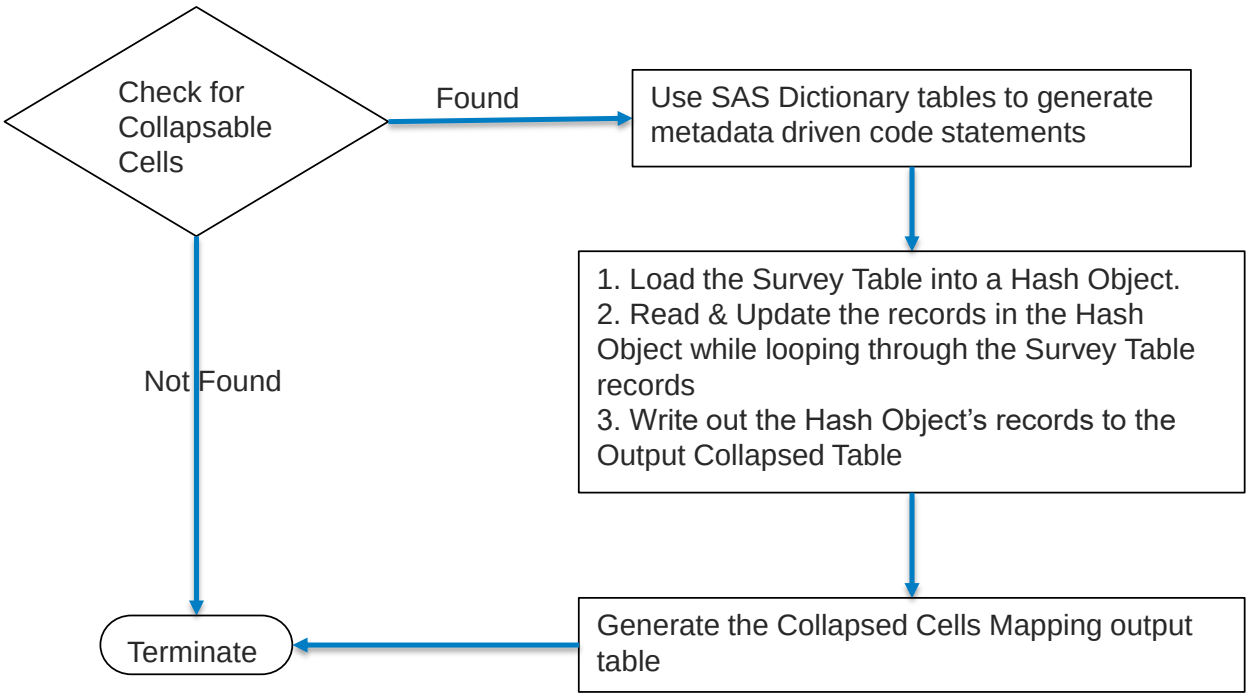

Figure 2: High-level logic flow of the collapse_data macro

MACRO COLLAPSE_DATA DETAILS

```
%macro collapse_data (
      p_inDsName=       /* Input Survey Table */
    , p_outDsName=      /* Output Collapsed Table */
    , p_classVars=      /* By Group Variables */
    , p_prntVar=        /* By Group Boundary variable */
    , p_minLimitCondition= /* Collapsing Condition */
    , p_mapDsName=    /* Collapsed Cells Mapping Output Table */
    );

    %LOCAL
        l_eTime
        l_msg
        l_rc
        l_rTime
        l_sTime
        l_needCollapse
        l_needCollapseCnt
```

```
        l_hashKeyVars
        l_currVarsLenghtStmt
        l_prevVarsLenghtStmt
        l_currVarsAssignStmt
        l_prevVarsAssignStmt
        l_currVarsAssignStmt2
        l_prevVarsAssignStmt2
        ;

    /********** BEGIN -- Main Macro Processing **********/

    /** Record Starting Time **/
    %let l_sTime = %sysfunc(time());

    /* First - Examine we need to Perform the Collapsing Process */
    %let l_needCollapseCnt = 0;
    PROC SQL NOPRINT;
        SELECT STRIP(PUT(COUNT(*),best.))
                ,STRIP(PUT(cell_number,BEST.))
        INTO    :l_needCollapseCnt
                ,:l_needCollapse separated by  ' '
        FROM   &p_inDsName
        WHERE &p_minLimitCondition
        ;
    QUIT;
    %put ;
    %put >>> ---------------------------------------------------------- ;
    %put >>> Cells must be Collapsed Count = &l_needCollapseCnt;
    %PUT >>> Cells must be Collapsed = &l_needCollapse;
    %put >>> ---------------------------------------------------------- ;
    %put ;

    %if(&l_needCollapseCnt EQ 0) %then
    %do;
        %let l_rc  = 1;
        %let l_msg = ERROR>>> collapse_data : Input table did not contain collapsable records! Process
terminated.;
        %goto exit;
    %end;

    /* Initialize Dynamic Variables and Statements to be used later */
    %let l_libName = WORK;
    %let l_dsName  = %UPCASE(%superq(p_inDsName));
    %if (%INDEX(%superq(l_dsName),%str(.)) GT 0) %then
    %do;
        %let l_libName = %SCAN(%superq(l_dsName),1,%str(.));
        %let l_dsName  = %SCAN(%superq(l_dsName),2,%str(.));
    %end;
```

```
PROC SQL NOPRINT;
    SELECT  CATS('curr_',name)||' '|| CASE WHEN type = 'char' then '$' ELSE '' END
            || STRIP(PUT(length,best.))
            ,CATS('prev_',name)||' '|| CASE WHEN type = 'char' then '$' ELSE '' END
            || STRIP(PUT(length,best.))
            ,CATS('curr_',name,' = ',name,';')
            ,CATS('prev_',name,' = ',name,';')
            ,CATS(name,' = curr_',name,';')
            ,CATS(name,' = prev_',name,';')
    INTO    :l_currVarsLenghtStmt separated by ' '
            ,:l_prevVarsLenghtStmt separated by ' '
            ,:l_currVarsAssignStmt separated by ' '
            ,:l_prevVarsAssignStmt separated by ' '
            ,:l_currVarsAssignStmt2 separated by ' '
            ,:l_prevVarsAssignStmt2 separated by ' '
    FROM    DICTIONARY.COLUMNS
    WHERE LIBNAME = "&l_libName"
    AND     MEMNAME = "&l_dsName"
    ;
QUIT;

/* scale_Value unwegt popct wegt origCellNumber */

%let l_hashKeyVars= %str(%')%sysfunc(tranwrd(%superq(p_classVars), %str( ),%str(%',%')))%str(%');

DATA _NULL_;

    /* Define all the variables in the PDV */
    if 0 then SET &p_inDsName;

    LENGTH      &l_currVarsLenghtStmt
                &l_prevVarsLenghtStmt
                usedCellsToCollapseWith $3000 recN 8;

    RETAIN usedCellsToCollapseWith;

    FORMAT      scale_Value unwegt popct wegt 16.;

    if (_n_ = 1) then
    do;
        /* declare hash, and load the data set into it */
        dcl hash recs     (dataset:"&p_inDsName",ordered:'a',hashexp: 16) ;
        recs.definekey  (%unquote(&l_hashKeyVars)) ; /* key table with KEY variable */
        recs.definedata (all:'yes') ;  /* store record data */
        recs.definedone () ; /* check validity and instantiate object */

        declare hiter hiRecs("recs");
```

```
    end;

    call missing(of _all_);

    DO recN=1 by 1 UNTIL(LAST.&p_prntVar);

        SET &p_inDsName END=eof;
        BY  &p_classVars;

        /* Load an updated copy of the record */
        rc = recs.find();

        &l_currVarsAssignStmt

        if (MISSING(origCellNumber)) then
            origCellNumber = STRIP(PUT(cell_number,BEST.));

        if (MISSING(curr_origCellNumber)) then
            curr_origCellNumber = STRIP(PUT(curr_cell_number,BEST.));

        if (_N_ GT 1) then
        do;
            /* Reset the Hash Iterator position to current record */
            if (hiRecs.setcur() = 0) then
            do;
                rc = hiRecs.PREV(); /* Load previous record into the PDV */

                &l_prevVarsAssignStmt

                if (MISSING(prev_origCellNumber)) then
                    prev_origCellNumber = STRIP(PUT(prev_cell_number,BEST.));

                rc = hiRecs.NEXT(); /* Load current record into the PDV */
            end;
        end;

        /* Check the Collapsing Condition */
        if (&p_minLimitCondition) then
        do;
            /* Reset the Hash Iterator position to current record */
            if (hiRecs.setcur() = 0) then
            do;
                rc = hiRecs.next(); /* Load next record into the PDV */

                if (MISSING(origCellNumber)) then
                    origCellNumber = STRIP(PUT(cell_number,BEST.));
            end;
```

```
                /* Find the suitable collapsing cell based on closest scale_value.*/
                if (rc = 0) then
                    next_diff = abs(scale_Value - curr_scale_Value);

                prev_diff = abs(curr_scale_Value - prev_scale_Value);

                if ((prev_&p_prntVar = curr_&p_prntVar) AND (prev_diff LT next_diff) OR
                    ((prev_diff EQ next_diff) AND (prev_unwegt LT unwegt))
                    ) then
                do;
                    prev_origCellNumber = CATX(',',prev_origCellNumber,curr_origCellNumber);
                    prev_scale_value =
((curr_scale_value*curr_wegt)+(prev_scale_value*prev_wegt))/(curr_wegt+prev_wegt);
                    prev_unwegt     = SUM(curr_unwegt,prev_unwegt);
                    prev_wegt       = SUM(curr_wegt,prev_wegt);
                    prev_popct      = SUM(curr_popct,prev_popct);

                    &l_prevVarsAssignStmt2

                end;
                else if ((curr_&p_prntVar = &p_prntVar) AND (prev_diff GT next_diff) OR
                    ((prev_diff EQ next_diff) AND (prev_unwegt GT unwegt))
                    ) then
                do;
                    origCellNumber = CATX(',',origCellNumber,curr_origCellNumber);
                    scale_value    =
((curr_scale_value*curr_wegt)+(scale_value*wegt))/(curr_wegt+wegt);
                    unwegt        = SUM(curr_unwegt,unwegt);
                    wegt          = SUM(curr_wegt,wegt);
                    popct         = SUM(curr_popct,popct);
                end;

                /* Add/Replace the record in the Hash Object */
                recs.replace();

                /* Need to Remove Currently collapsed record from the Hash Object */
                if ((prev_&p_prntVar = curr_&p_prntVar) OR (curr_&p_prntVar = &p_prntVar)) then
                do;

                    &l_currVarsAssignStmt2

                    if (recs.find() = 0) then
                    do;
                        rc= recs.remove();
                    end;
                end;

            end;
```

```
            else
            do;
                /* Replace the record in the Hash Object */
                recs.replace();
            end;
        END;

        if (eof) then
            recs.output(dataset:"&p_outDsName");
    RUN;

    /* Produce Cell_number & Collapse Index mapping data set */
    DATA &p_mapDsName(keep=index);
        LENGTH index 3;
        SET &p_outDsName(keep=cell_number origCellNumber);

        index = _n_;
        i=1;
        do until(scan(origCellNumber,i,',') eq '');
            OUTPUT;
            i+1;
        end;
    RUN;

    %goto finished;

    %exit:
        %put *** collapse_data   ***;
        %put *** l_RC must be zero (0). ***;
        %put *** l_RC= &l_RC. ***;
        %put *** &l_MSG ***;
        %put *** collapse_data   ***;

    %finished:
        /** Record Finish Time **/
        %let l_eTime = %sysfunc(time());

        %if (%superq(l_sTime) NE ) %then
        %do;
            /* Calculate Run Time and display it */
            %let l_rTime = %sysfunc(putn(%sysevalf(&l_eTime - &l_sTime),time12.2));
            %put;
            %put >>> ---------------------------------------------- ;
            %put >>> collapse_data :>>> Total RunTime = &l_rTime;
            %put >>> ---------------------------------------------- ;
            %put;
        %end;
```

```
%mend collapse_data;


/* Sample macro execution call */
%collapse_data (p_inDsName=myLib.input, p_outDsName=work.lev1
, p_classVars=%STR(cell_number sex age), p_prntVar=sex
, p_minLimitCondition=%str(unwegt LT 35 OR  (popct/wegt) LT 0.67 OR (popct/wegt) GT 4)
, p_mapDsName=work.map1);

/* Sample Output */
>>> ------------------------------------------------------------
>>> Cells must be Collapsed Count = 25
>>> Cells must be Collapsed = 2 6 10 14 20 32 39 41 42 43 54 59 60 66 71 73 74 75 80 84 87 93 98 102
107
>>> ------------------------------------------------------------
 NOTE: There were 108 observations read from the data set MYLIB.INPUT.
 NOTE: The data set WORK.LEV1 has 83 observations and 9 variables.
 NOTE: There were 108 observations read from the data set MYLIB.INPUT.
 NOTE: DATA statement used (Total process time):
      real time          0.02 seconds
      cpu time           0.03 seconds
 NOTE: There were 83 observations read from the data set WORK.LEV1.
 NOTE: The data set WORK.MAP1 has 108 observations and 1 variables.
 NOTE: DATA statement used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
 >>> ------------------------------------------------------------------
 >>> collapse_data :>>> Total RunTime = 0:00:00.10
 >>> ------------------------------------------------------------------
```

Cell_Number	Sex	Race	Age	Scale_Value	unwegt	popct	wegt	origCellNumber
1	Male	White Alone	<1	601.033	728	2545563	1578208	1,2
…			…	…	…	…	…	
5			4	611.0277	625	2635370	1475543	5,6
…			…	…	…	…	…	
9			8	620.1548	323	1360688	470413.7	9,10
..			…	…	…	…	…	
15			16-17	15.8706	995	3951597	1236326	15,14
…			…	…	…	…	…	
19			25-29	47.44487	3316	13798844	3474992	19,20
21			35-39	57	1573	5866860	6241955	
…			…	…	…	…	…	
31			80-84	104.6845	702	3161600	1048419	31,32
…	Female	White Alone	…	…	…	…	…	
38			5	712	303	1211654	954199	
40			7	716.9633	980	3930691	962505.2	40,39,42,41
44			12 to 13	729.0286	1678	6465308	9633235	44,43
45			14	733	335	1248982	615495	
…			…	…	…	…	…	
52			30-34	149	1834	6524873	2809038	
53			35-39	157.1725	3382	13191510	9003230	53,54
55			45-49	163	1887	7439013	2740975	
56			50-54	165	2026	7818318	2342809	

Figure 3: Collapsed artificial survey results table

CONCLUSION

The Hash Object and the Hash Iterator Object are two component objects provided by SAS® for use in a Data Step. They enable us to rapidly store, search, and retrieve data based on lookup keys efficiently there by cutting down the run time.

The predefined attributes and methods of these two component objects, provide great functionality out of the box, simplify and reduce the length of the SAS® programs, thereby enhance their maintainability.

REFERENCES

Hagemeier, Tamara and Chang, Yue-Hwa 1999. "Use PROC SQL to Collapse Cells – It's Easy". Proceedings of SAS Users Group International 1999, Miami Beach, Florida. Available at http://www2.sas.com/proceedings/sugi24/Coders/p074-24.pdf

Yeh, Shi-Tao 2000, "Collapsing Adverse Experiences Records". Proceedings of Pharmaceutical Industry SAS® Users Group 2000, Seattle, Washington. Available at http://www.lexjansen.com/pharmasug/2000/Posters/p08.pdf

Holle, Ann Von 2000. "USING PROC SQL TO SELECTIVELY COLLAPSE CELLS FOR WEIGHTING PROCESS". Proceedings of the NorthEast SAS Users Group 2000, Philadelphia, Pennsylvania. Available at http://www.lexjansen.com/nesug/nesug00/cc/cc4022.pdf

Dorfman, Paul and Vyverman, Koen 2005. "Data Step Hash Objects as Programming Tools". Proceedings of SAS Users Group International 2005, Philadelphia, Pennsylvania. Available at http://www2.sas.com/proceedings/sugi30/236-30.pdf

Rhoades, Stephen and Dodoo, Lee 2008. "Another Helping of HASH". Proceedings of the NorthEast SAS Users Group 2008, Pittsburgh, Pennsylvania. Available at http://www.lexjansen.com/nesug/nesug08/po/po12.pdf

Loren, Judy and DeVenezia, Richard 2011. "Building Provider Panels: An Application for the Hash of Hashes". Proceedings of SAS Global Forum 2011, Las Vegas, Nevada. Available at http://support.sas.com/resources/papers/proceedings11/255-2011.pdf

CONTACT INFORMATION

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

Ahmed Al-Attar
AnA Data Warehousing Consulting, LLC.
McLean, VA 22101
Cell Phone: 703-477-7972
E-mail: ahmed.al-attar@anadwc.com
Web: www.anadwc.com

TRADEMARKS