Paper 094-2019

Fun With The Fun King: The SAS® Solution to the Magic Square

Frank Ferriola, Charles Schwab and Co.

ABSTRACT

My father introduced me to the "Magic Square" as a 3x3 or other odd numbered square in which you could place the integers from 1 to (n*n) and have all rows, across, down and diagonally equal

the same number.

In 2001, I included the concept of the Magic Square in a presentation at WUSS, and how my father taught me a simple pattern that solved it in seconds. In 2005, I presented the Manual Solution to the Magic Square.

In this paper I present the SAS® solution to the problem, which will take you through all the rules that get applied to solve for any square that has an odd number of rows and columns.

INTRODUCTION

When I was a wee lad of around 8 or 9, my father introduced me to a children's encyclopedia called "The Book of Knowledge." In it, there was a section on math problems, from word problems, to matchstick problems to problems that required you to think "outside the box."

I found a lot of the problems fun, but the one that stuck with me was the Magic Square which required you to arrange the numbers 1 to 9 in a 3x3 square in order to have all of the columns, rows and diagonals to add up to 15. I think the reason, it stuck with me is that my father introduced me to a "cheat code" to fill in the boxes in seconds, which came in handy when one of my teachers had us do the problem in class a year or two later, and he was amazed when I had the solution quickly.

THE MAGIC SQUARE CONCEPT

If you search for "Magic Square" on the internet, you will find many different things, so to avoid confusion, I will explain what my definition is. The Magic Square is an odd-numbered square of cells that requires you to place consecutive numbers in the square starting with 1 and ending with n^2 (Where n=the number of rows or columns) in such a way as to have every row, column and diagonal add up to the same number. That number is determined by the formula (! + n^2) * n

A simple magic square is 3x3 like this:

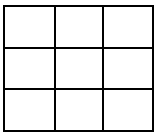

Figure 1. An empty Magic Square 3 x 3

A 5x5 square is thus:

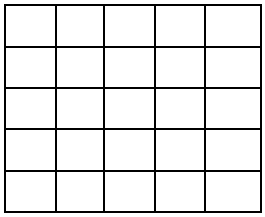

Figure 2. A Blank 5 x 5 Magic Square

Any other odd-numbered square can be made, the total can be calculated and all numbers can be placed easily—but how?

THE RULES

Amazingly, there are only 6 rules to follow and the only math required is the ability to add 1 to a number.

1. The number 1 goes in the middle square of the top row.
2. From that point move to the right one square and up one square (or diagonally) and if it's still inside the square place the next number.
3. If you go outside the square to the top, move to the bottom of that column (last row) and place the next number.
4. If you go outside the square to the right, go to the 1st Column in that row and place the next number.
5. If the cell is full move directly below the last number played and place the next number.

If you are outside the square to the top AND right, then consider it full and place the next number directly below the previous number (as in Rule 5).

So the representation of the rules looks like this.

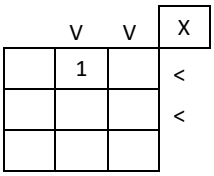

Figure 3. The Rules Illustrated

So following the rules, here is the solution:

8	1	6
3	5	7
4	9	2

Figure 4. The 3 x 3 Solution

Note that all rows, columns and diagonals add to (1 + 9) / 2 * 3 = 15.

Try it yourself following the rules—Here's the form—I'll wait:

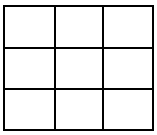

Figure 5. The Reader's Worksheet to Solve 3 x 3 Magic Square

Now let's go on. Apply the rules to the 5x5 Square.

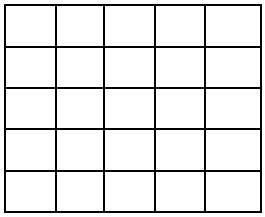

Figure 6. The Reader's Worksheet to Solve 5 x 5 Magic Square

The solution appears in Appendix B to this paper along with the number it adds up to. If you are so inclined, feel free to look at larger squares.

But back to the purpose of this paper.

THE SAS SOLUTION

In preparation for this paper, I started thinking about how I could apply the rules within SAS Code.

I decided to create it in Macro form which allowed more flexibility in building the arrays needed to complete it.

The input macrovariable which represents the number of rows and columns is called squ:

```
%macro ms_paper(squ=);
```

We can build other macrovariables needed from the &squ. Macrovariable:

```
data _null_;
  /*** Total Observations is the total number of squares in grid ***/
        tot_obs=%eval(&Squ.)**2;
        call symput('tot_obs',tot_obs);
  /*** Mid_pt sets middle column for #1 ****/
       mid_pt=int(%eval(&squ.)/2)+1;
       call symput('mid_pt',mid_pt);
run;
```

Next we will start the data step

This will require us to keep the row and column position

```
data ms_&squ.;
```

First we need to keep the row and column position as well as the last number used from each record as move through the process:

```
retain row_pos col_pos last_num;
```

Next we need to set up a two-dimensional array for the grid to keep track of which cells in the grid have been filled We. will call this variable ‘cell’:

```
/**** Set up Array for Cells ***/
array cell{%eval(&squ.),%eval(&squ.)} $1.;
```

We will populate each cell using nested %do loops and initialize the value of each cell to “E” (For Empty) We will use r for row and c for column:

```
%do r=1 %to %eval(&squ.);
   %do c=1 %to %eval(&squ.);
       format cell&r.&c. $1.;
              cell&r.&c.="E";
   %end;
%end;
```

Now we need to run through another %do loop to populate all of the empty cells and apply the rules. Note we have to apply the rules in a different order than I learned them.

```
%do o=1 %to %eval(&tot_obs.);

/**** Determine Positions     *****/
/**** Rule 1: Put first number in middle of row 1 ****/
         if %eval(&o)= 1 then do;
             last_num=0;
             row_pos=0;
             col_pos=0;


/**** Rule 1: Put first number in middle of row 1 ****/
             r=1;
             c=%eval(&mid_pt.);
         end;

/**** Rule 6: If you are outside the square to the top AND right,
         then consider it full and place the next number directly below
         the previous number (as in Rule 5). ****/

         else if col_pos=%eval(&Squ.) then do;
```

```
            if row_pos=1 then r=2;

/***** Rule 4. If you go outside the square to the right, go to the 1st Column in that
row and place the next number.****/
            else do;
            c=1;
                           r=row_pos-1;
                     end;
                end;

      /**** Rule 3. If you go outside the square to the top, move to the bottom of
that column (last row) and place the next number. ***/
                else if row_pos=1 then do;
                     r=%eval(&squ.);
                       c=col_pos+1;
                end;

      /**** Rule 2. From that point move to the right one square and up one square
(or diagonally) and if it's still inside the square place the next number. ****/
               else do;
                  r=row_pos-1;
                    c=col_pos+1;
               end;

      /**** Rule 5. If the cell is full move directly below the last number played and
place the next number. *****/

                    if cell{r,c}="F" then do;
                       r=row_pos+1;
                       c=col_pos;
                    end;


At this point all of the possible outcomes have been processed.  Now we fill the
information for output.

Cell(r,c) is changed to "F" for Full:
              cell{r,c}="F";

We now set row_pos and col_pos as starting point for next observation.
                row_pos=r;
                col_pos=c;

Set the number for the cell by adding 1 to last_num.
                last_num=last_num+1;

Output the record
                output;
    %end;
run;

Proc sort data=ms_&squ.;
    by row_pos col_pos;
run;

data _null_;
    ln_fmt=length(%eval(&tot_obs.));
       cell_fmt=((%eval(&tot_obs.)/%eval(&tot_cols.))*(ln_fmt+2));
       call symput('ln_fmt',ln_fmt);
       call symput('cell_fmt',cell_fmt);
run;

data ms_order_&squ.;
```

```
   retain cell;
   format cell $%eval(&cell_fmt.).;
   format last_num_c $%eval(&ln_fmt.).;
   set ms;
      by row_pos col_pos;
         last_num_c=last_num;
         If first.row_pos then cell="| ";
         cell=trim(cell)||last_num_C||" | ";
         if last.row_pos then do;
           final_cell=trim(cell)||" |";
              output;
         end;
run;

Title "Magic Square &tot_rows. x &tot_cols.";

       proc print data=ms_order_&squ.;
          var final_cell;
       run;


%mend ms_paper;


Run multiple squares by calling the macro;

%ms_paper(squ=3);
%ms_paper(squ=5);
%ms_paper(squ=7);
%ms_paper(squ=9);

%ms_paper(squ=15);
%ms_paper(squ=25);
%ms_paper(squ=35);

See Appendix B for Results
```

APPENDIX A: THE FULL CODE

Here is the full code from start to finish:

```
       %macro ms_paper(squ=);

       data _null_;
       /*** Total Observations is the total number of squares in grid ***/
              tot_obs=%eval(&Squ.)**2;
              call symput('tot_obs',tot_obs);
          /*** Mid_pt sets middle column for #1 ****/
        mid_pt=int(%eval(&squ.)/2)+1;
        call symput('mid_pt',mid_pt);
       run;

       data ms_&squ.;
           retain row_pos col_pos last_num ;
       /**** Set up Array for Cells ***/
       array cell{%eval(&squ.),%eval(&squ.)} $1.;
       %do r=1 %to %eval(&squ.);
           %do c=1 %to %eval(&squ.);
                  format cell&r.&c. $1.;
```

```
     cell&r.&c.="E";
      %end;
%end;

last_num=last_num+1;

         output;
    %end;
run;

Proc sort data=ms_&squ.;
    by row_pos col_pos;
run;

data _null_;
    ln_fmt=length(%eval(&tot_obs.));
       cell_fmt=((%eval(&tot_obs.)/%eval(&tot_cols.))*(ln_fmt+2));
       call symput('ln_fmt',ln_fmt);
       call symput('cell_fmt',cell_fmt);
run;

data ms_order_&squ.;
   retain cell;
   format cell $%eval(&cell_fmt.).;
   format last_num_c $%eval(&ln_fmt.).;
   set ms;
      by row_pos col_pos;
         last_num_c=last_num;
         If first.row_pos then cell="| ";
         cell=trim(cell)||last_num_C||" | ";
         if last.row_pos then do;
           final_cell=trim(cell)||" |";
              output;
         end;
run;

Title "Magic Square &tot_rows. x &tot_cols.";

proc print data=ms_order_&squ.;
    var final_cell;
run;


%mend ms_paper;

%ms_paper(squ=3);
%ms_paper(squ=5);
%ms_paper(squ=7);
%ms_paper(squ=9);
%ms_paper(squ=15);
```

APPENDIX B

Solutions to the above:

3 x 3 (Sum=15)

row_pos	col1	col2	col3
1	8	1	6
2	3	5	7
3	4	9	2

5 x 5 (Sum=65)

row_pos	col1	col2	col3	col4	col5
1	17	24	1	8	15
2	23	5	7	14	16
3	4	6	13	20	22
4	10	12	19	21	3
5	11	18	25	2	9

7 x 7 (Sum=175)

row_pos	col 1	col 2	col 3	col 4	col 5	col 6	col 7
1	30	39	48	1	10	19	28
2	38	47	7	9	18	27	29
3	46	6	8	17	26	35	37
4	5	14	16	25	34	36	45
5	13	15	24	33	42	44	4
6	21	23	32	41	43	3	12
7	22	31	40	49	2	11	20

9 x 9 (Sum=369)

row_pos	col1	col2	col3	col4	col5	col6	col7	col8	col9
1	47	58	69	80	1	12	23	34	45
2	57	68	79	9	11	22	33	44	46
3	67	78	8	10	21	32	43	54	56
4	77	7	18	20	31	42	53	55	66
5	6	17	19	30	41	52	63	65	76
6	16	27	29	40	51	62	64	75	5
7	26	28	39	50	61	72	74	4	15
8	36	38	49	60	71	73	3	14	25
9	37	48	59	70	81	2	13	24	35

15 x 15 (Sum=1,695)

row_pos	col 1	col 2	col 3	col 4	col 5	col 6	col 7	col 8	col 9	col10	col11	col12	col13	col14	col15
1	122	139	156	173	190	207	224	1	18	35	52	69	86	103	120
2	138	155	172	189	206	223	15	17	34	51	68	85	102	119	121
3	154	171	188	205	222	14	16	33	50	67	84	101	118	135	137
4	170	187	204	221	13	30	32	49	66	83	100	117	134	136	153
5	186	203	220	12	29	31	48	65	82	99	116	133	150	152	169
6	202	219	11	28	45	47	64	81	98	115	132	149	151	168	185
7	218	10	27	44	46	63	80	97	114	131	148	165	167	184	201
8	9	26	43	60	62	79	96	113	130	147	164	166	183	200	217
9	25	42	59	61	78	95	112	129	146	163	180	182	199	216	8
10	41	58	75	77	94	111	128	145	162	179	181	198	215	7	24
11	57	74	76	93	110	127	144	161	178	195	197	214	6	23	40
12	73	90	92	109	126	143	160	177	194	196	213	5	22	39	56
13	89	91	108	125	142	159	176	193	210	212	4	21	38	55	72
14	105	107	124	141	158	175	192	209	211	3	20	37	54	71	88
15	106	123	140	157	174	191	208	225	2	19	36	53	70	87	104

CONCLUSION

Using SAS to solve puzzles can be challenging and fun. Working through the problems and refining the code teaches you new concepts (or sharpens old ones). Once learned you can use the concepts to impress your friends.

Take other logic and math puzzles and try to come up with your own SAS Solutions to them.

ACKNOWLEDGMENTS

I would like to thank the following

My father, Frank Ferriola, Sr. who encouraged me as a child in my fascination with numbers and logic, which has led to a lifetime of working with numbers and logic in a career I never expected to have.

The members and former members off the WUSS Executive Committee including Rebecca Ottesen, Ethan Miller, and Chelsea Lofland who first gave me the “Fun King” nickname at SAS Global Forum in Las Vegas, 2016 and the many others who have perpetuated it.

The WUSS and SAS communities who encourage others to present, share and excel at SAS Concepts.

CONTACT INFORMATION

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

Frank Ferriola
Charles Schwab & Co.
faferriola@gmail.com