Paper 3807-2019

Puzzle Me, Puzzle You: How a Thought Experiment Became a Rubik's Cube Among a Set of Fun Puzzles

Amit Patel, Barclays plc; Lewis Mitchell, Barclays plc;

ABSTRACT

How do you help a community of SAS® users become more involved in the software, and learn new ideas without providing direct training? We decided to answer this question by designing a series of puzzles, which require the solutions to be run in SAS. This allowed us to showcase SAS features that were unused by many of our colleagues and at the same time promote a critical thinking mind set when solving the puzzles. Whether its X marks the spot using office locations and geolocational data, solving a Rubik's cube that utilises the SGPLOT procedure, or a dataset based crossword using SAS Procedure names, each puzzle requires a different technique to solve. Our aim is to help users expand their SAS knowledge and develop a different way of thinking that they can apply to their day-to-day work.

INTRODUCTION

The idea of proactively engaging in interactive learning is not a new one, but the methods and tools which are available depend on the environment in question. In this case it's not a classroom, but a SAS® Grid environment with over 1000 users. We want to encourage these users to use SAS in ways they are not typically exposed to in their day-to-day work. By showing people different concepts within SAS or challenging their SAS skills in a fun way, we hope to expand their horizons of what is possible. It is often said that your only limit is your imagination; we hope these puzzles will help our users imagine something new.

THE PUZZLES

There are 10 unique puzzles that have been created to highlight a variety of concepts.

1) ***The Crossword Puzzle*** – Everyone knows the PRINT procedure, FREQ procedure and SORT procedure - but there are some procedures with far more interesting names. This crossword puzzle forms the intersection between SAS procedures and Hollywood movie quotes.
2) ***The SAS Quiz*** – A quick bit of knowledge searching for SAS functions, SAS Dictionary tables and maximum numbers. We encourage internet searching and even provide a *lmgtfy* link for those who may struggle with what to search for.
3) ***The Safe Cracking Puzzle*** – There's a safe that needs cracking! What more needs to be said?
4) ***The Maze Puzzle*** – Time to escape a digital maze! Just as ominous as it may sound; the rules are based on mathematical sequences that tell you where you can and can't move to.
5) ***The Joining Puzzle*** – Can you join the data to get the hidden code? The twist is only a single value in a single column in each dataset allows them to be merged.
6) ***The Encrypted Music Puzzle*** – Can you map and decrypt two datasets in order to get the information needed to play a tune on a digital keyboard?

7) ***The Graph Puzzle*** – You've got the equations but can you identify the image they reveal?

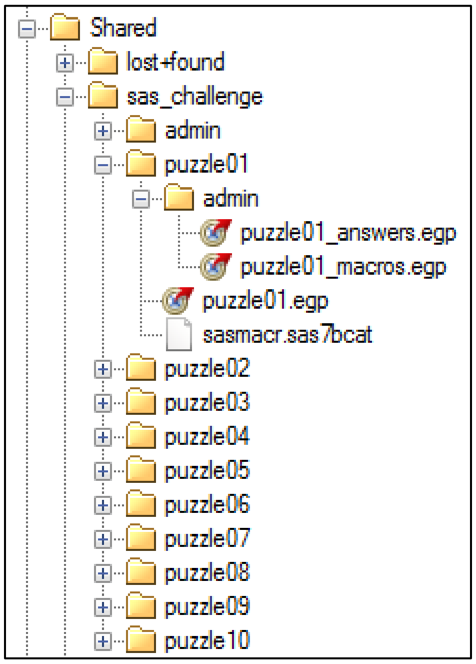
8) ***The Circuit Puzzle*** – With only 60 seconds and a complex set of instructions can you defuse the circuit in time?

9) ***The X Marks the Spot Puzzle*** – It's often said that X marks the spot; this time it's four Barclays offices that give you your X. The answer you seek is the postcode nearest to the intersection of the X

10) ***The Rubik's Puzzle*** – Can you solve a SAS generated Rubik's cube?

SETTING UP THE ENVIRONMENT

In order to make things as accessible as possible for all users, a separate directory structure was set up to store the puzzles together and restrict permissions across the files. Each individual puzzle directory contains the following:



- A copy of the egp which all users have read access to. Each egp contains all the relevant notes and macros the user will need in order to solve it.
- A SAS macro catalogue which holds all the code to run and solve the puzzles. These were compiled using the /secure option so users won't be able to open up the code directly; it's still available for them to call and use within their egp using the SASMSTORE option, but they are prevented from reading the source code.
- An 'admin' directory with restricted permissions holding egp's for the solutions and the raw macro code to create the SAS macro catalogue. These egp's are password protected in addition to the restricted permissions.
- A hidden '.log' directory that is used to record when a user successfully completes a puzzle and enables us to generate usage statistics.

Within each of the egp's there is a standard initialisation piece of code which allows the puzzle macro catalogue to be loaded and then be available for the individual macro calls to solve that puzzle:

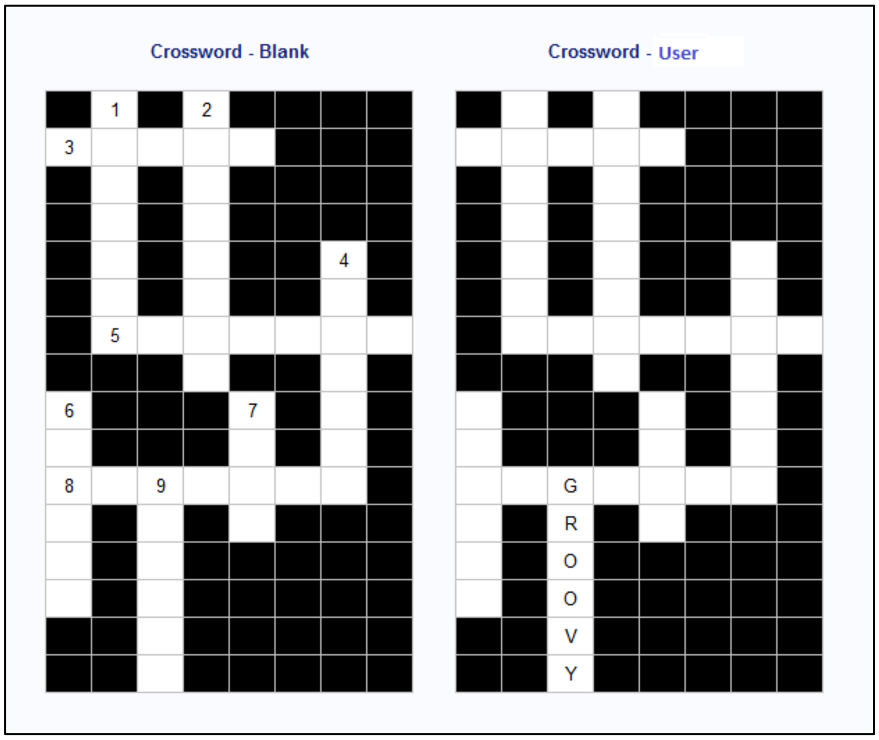
```
%sysmstoreclear;

%let puzzle_root = /sas/Prod/Data/GrpPCB/Shared/sas_challenge;
%let puzzle = puzzle01;

libname &puzzle. "&puzzle_root./&puzzle.";
options mstored sasmstore=&puzzle. pagesize=1024;
```

PUZZLE 1 – THE CROSSWORD PUZZLE

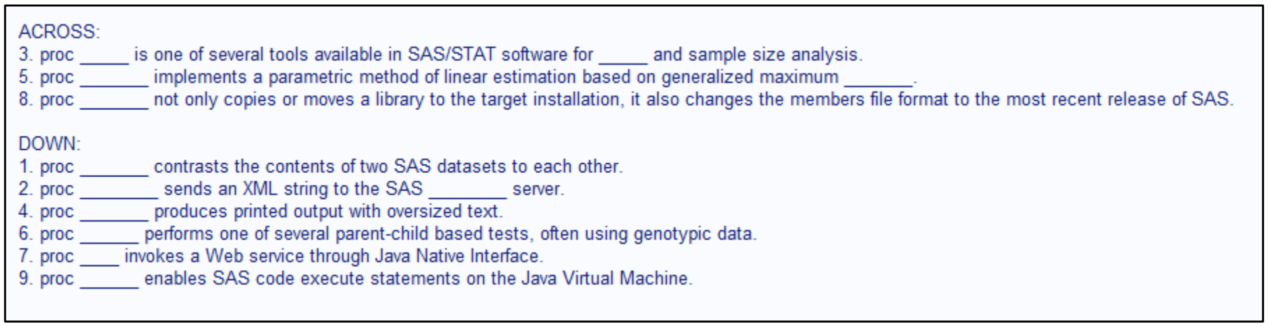
The crossword is built up by using a dataset as a template with a format applied to colour certain cells black. This dataset is displayed using proc report. One version of the dataset, which is blank except for the clue numbers, is displayed on the left-hand side. On the right-hand side, the version of the dataset with the answers the user has entered is shown. Any answer the user enters which is the correct length is displayed, but if the user enters two answers which overlap and have different characters in the overlapping cell, then a '*' is displayed instead of either character to highlight the conflict.



```
%puzzle_crossword(
clue=SAS,
down1=,
down2=,
across3=,
down4=,
across5=,
down6=,
down7=,
across8=,
down9=
);
```

The users enter the answers using the %puzzle_crossword macro.

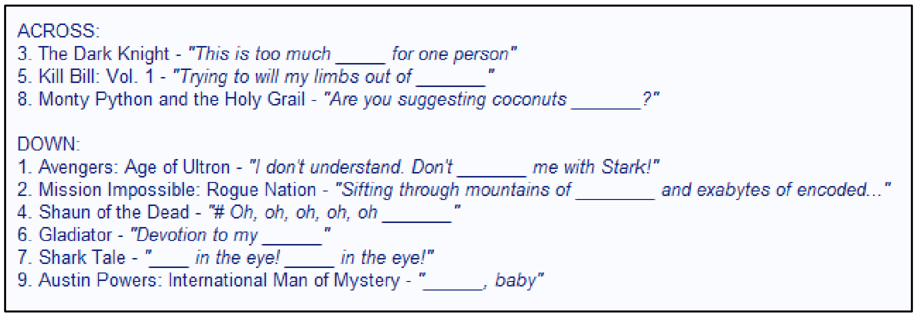
By default, the parameter CLUE is populated with the value SAS, which means that the clues which are generated all relate to SAS procedures.



ACROSS:
3. proc _____ is one of several tools available in SAS/STAT software for _____ and sample size analysis.
5. proc _______ implements a parametric method of linear estimation based on generalized maximum _______.
8. proc _______ not only copies or moves a library to the target installation, it also changes the members file format to the most recent release of SAS.

DOWN:
1. proc _______ contrasts the contents of two SAS datasets to each other.
2. proc ________ sends an XML string to the SAS ________ server.
4. proc _______ produces printed output with oversized text.
6. proc ______ performs one of several parent-child based tests, often using genotypic data.
7. proc ____ invokes a Web service through Java Native Interface.
9. proc ______ enables SAS code execute statements on the Java Virtual Machine.

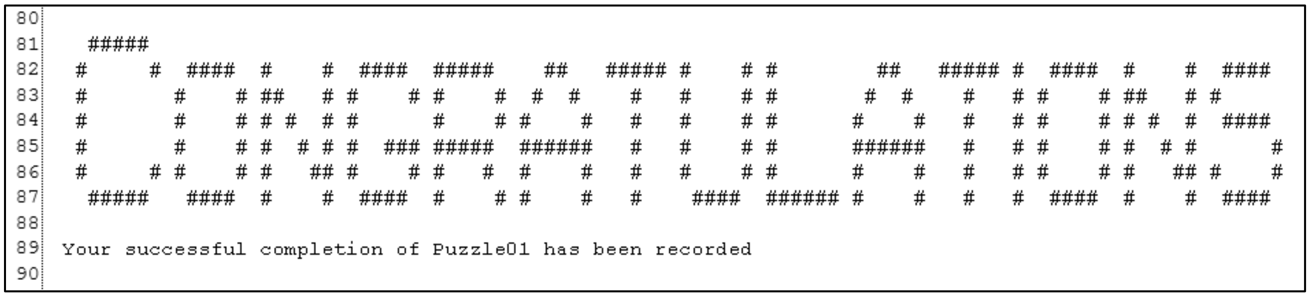
However, there are hints in the background information and log that let users know they can change the CLUE parameter to have a value of MOVIE. If the user feels their pop culture knowledge is better than their SAS procedure knowledge they can instead generate clues that will lead to the same answers based on movie quotes.



ACROSS:
3. The Dark Knight - *"This is too much _____ for one person"*
5. Kill Bill: Vol. 1 - *"Trying to will my limbs out of _______"*
8. Monty Python and the Holy Grail - *"Are you suggesting coconuts _______?"*

DOWN:
1. Avengers: Age of Ultron - *"I don't understand. Don't _______ me with Stark!"*
2. Mission Impossible: Rogue Nation - *"Sifting through mountains of ________ and exabytes of encoded..."*
4. Shaun of the Dead - *"# Oh, oh, oh, oh, oh _______"*
6. Gladiator - *"Devotion to my ______"*
7. Shark Tale - *"____ in the eye! _____ in the eye!"*
9. Austin Powers: International Man of Mystery - *"______, baby"*

Once the macro has been executed with the parameters containing a full set of correct answers, the user will be congratulated in the log and informed that their success has been recorded. This is the same for all 10 puzzles.



```

  #####
 #     #  ####  #    #  ####  #####    ##   ##### #    # #        ##   ##### #  ####  #    #  ####
 #       #    # ##   # #    # #    #  #  #    #   #    # #       #  #    #   # #    # ##   # #
 #       #    # # #  # #      #    # #    #   #   #    # #      #    #   #   # #    # # #  #  ####
 #       #    # #  # # #  ### #####  ######   #   #    # #      ######   #   # #    # #  # #      #
 #     # #    # #   ## #    # #   #  #    #   #   #    # #      #    #   #   # #    # #   ## #    #
  #####   ####  #    #  ####  #    # #    #   #    ####  ###### #    #   #   #  ####  #    #  ####

Your successful completion of Puzzle01 has been recorded

```

PUZZLE 2 – THE SAS QUIZ

The SAS quiz is based around providing questions on key techniques or areas of SAS we have found useful to know ourselves. On running the script, the first question is available:

```
Question 01: Which function compresses multiple blanks to a single blank?
```

The user enters their answer using the macros provided below and a response is given each time a macro is called.

```
%puzzle_quiz_answer_01(answer_01=UNKNOWN);
%puzzle_quiz_answer_02(answer_02=);
%puzzle_quiz_answer_03(answer_03=);
%puzzle_quiz_answer_04(answer_04=);
%puzzle_quiz_answer_05(answer_05=);
%puzzle_quiz_answer_06(answer_06=);
%puzzle_quiz_answer_07(answer_07=);
%puzzle_quiz_answer_08(answer_08=);
%puzzle_quiz_answer_09(answer_09=);
%puzzle_quiz_answer_10(answer_10=);
```

In the event of three incorrect answers for any of the questions, a hint will be provided. For the first few easier questions the hint is a link to "let me google that for you" to promote looking online for SAS queries which may arise.[1] For the trickier questions, more detailed hints are provided.

```
Question 01: Which function compresses multiple blanks to a single blank?

ERROR: UNKNOWN is incorrect!

HINT: http://lmgtfy.com/?q=Which+function+compresses+multiple+blanks+to+a+single+blank%3F
```

Once a question is correctly answered (in numerical order) then the next question is provided in the log.

```
Question 01: Which function compresses multiple blanks to a single blank?
Answer 01: compbl is correct!

Question 02: Which single function produces the same output as combining TRIM and LEFT?
```

PUZZLE 3 – THE SAFE CRACKING PUZZLE

The safe cracking macro is designed to motivate people into using macros and do loops, and will test their ability to implement them correctly. The code to generate the query is relatively simple and only requires three nested loops from 1 to 50 to solve. The three loops will need to cycle through until the values for parameters turn1, turn2 and turn3 are correct; at that point a global macro variable will be populated indicating the correct combination has been found. In the code below the put statements that print the text art for 'Congratulations' to the log are shown using a reduced font size for readability.

```
%macro puzzle_safe_crack(turn1,turn2,turn3) / store secure;

  options nonotes nomlogic nomprint nosymbolgen nosource nosource2;

  %local solved sysmacrolog;

  %let solved = 0;
  %let sysmacrolog = &sysmacroname.;

  %global unlocked;

  %if &turn1. = 37 and &turn2. = 19 and &turn3. = 11 %then %do;

    %let solved = 1;
    %let unlocked = 1;

    %put ;

%put %str()  #####                                                                                             ;
%put %str() #     #  ####  #    #  ####  #####    ##   ##### #    # #        ##   ##### #  ####  #    #  ####  ;
%put %str() #       #    # ##   # #    # #    #  #  #    #   #    # #       #  #    #   # #    # ##   # #      ;
%put %str() #       #    # # #  # #      #    # #    #   #   #    # #      #    #   #   # #    # # #  #  ####  ;
%put %str() #       #    # #  # # #  ### #####  ######   #   #    # #      ######   #   # #    # #  # #      # ;
%put %str() #     # #    # #   ## #    # #   #  #    #   #   #    # #      #    #   #   # #    # #   ## #    # ;
%put %str()  #####   ####  #    #  ####  #    # #    #   #    ####  ###### #    #   #   #  ####  #    #  ####  ;

    %put ;
    %put Your successful completion of Puzzle03 has been recorded ;
    %put ;

    %puzzle_log(puzzle=puzzle03, log=log03, sysmacro=&sysmacrolog., solved=&solved.);

  %end;
  %else %do;

    %let unlocked = 0;
  %end;

%mend puzzle_safe_crack;
```

The macro call the users are given to solve this puzzle only relies on three parameters being passed to it.

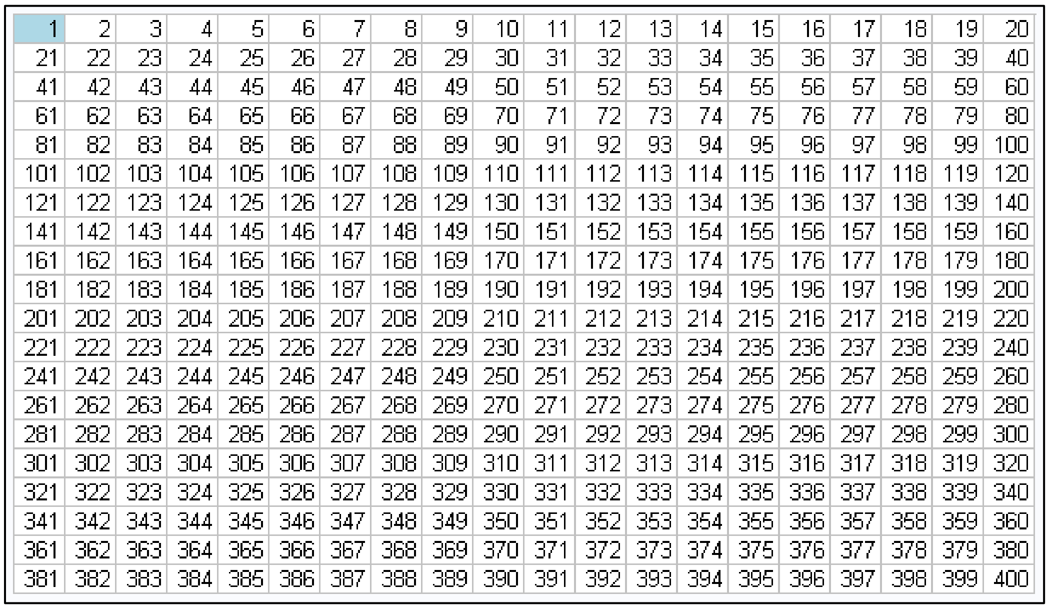
```
%puzzle_safe_crack(turn1=,turn2=,turn3=);
```

PUZZLE 4 – THE MAZE PUZZLE

The maze puzzle represents a step up in complexity and requires the users to use a little more out of the box thinking. The rules for crossing the maze are explained in a rhyme, which details how to move around the maze and which numbers not to move to. Most of the restrictions are based on mathematical sequences.

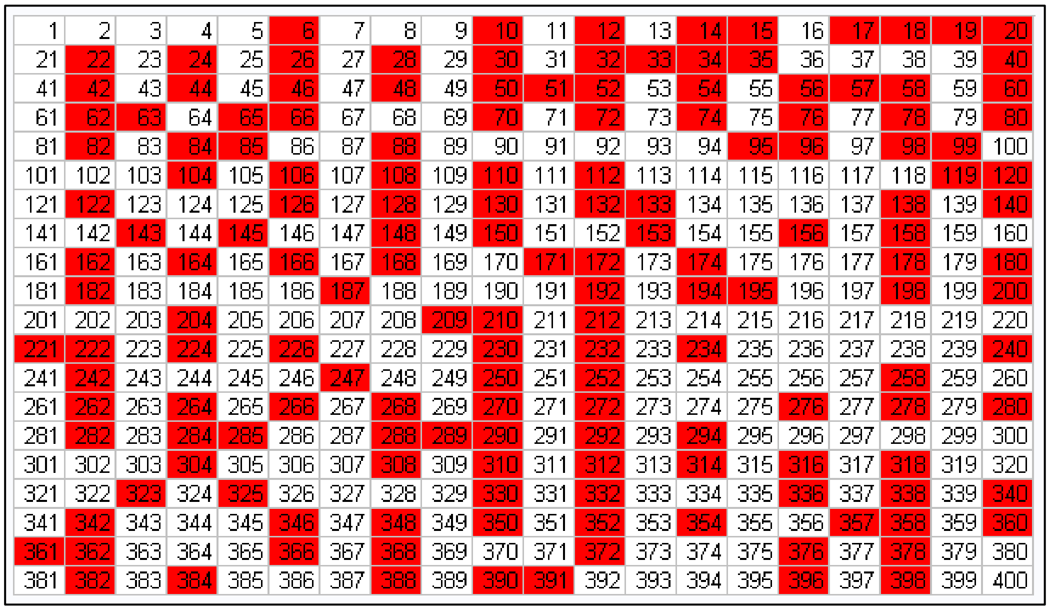
```
/**************************************************************************************/
/*** The next puzzle you face is a maze of sorts,                                   ***/
/*** where the walls are the numbers and will mess with your thoughts.              ***/
/***                                                                                ***/
/*** From one to four-hundred you must go,                                          ***/
/*** up, down, left and right are the moves that you know.                          ***/
/***                                                                                ***/
/*** By applying the rules in the order they come,                                  ***/
/*** you will escape from this maze and avoid feeling glum.                         ***/
/***                                                                                ***/
/*** Rule 1 says you're safe on prime, fibonnaci or square,                         ***/
/*** unless one of those, for one more or one less don't go there.                  ***/
/***                                                                                ***/
/*** Rule 2 says the rest are safe to begin,                                        ***/
/*** although 272 has now changed, to make it harder to win.                        ***/
/***                                                                                ***/
/*** Rule 3 says perfect numbers will give you a chance,                            ***/
/*** on three lots of the second plus the first you can dance.                      ***/
/***                                                                                ***/
/*** Rule 4 says first 17 then 19 make the complexity worse,                        ***/
/*** for all multiples, their safety will be the reverse.                           ***/
/**************************************************************************************/
```

Once the user runs the %puzzle_maze_initialise macro they will be presented with the digital maze and a starting position of location 1. As described in the rhyme they need to move to location 400.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380
381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400

The rhyme should help the user identify that they need to specify up, down, left or right in order to move in those directions. Any illegal moves will be prevented by the macro when it is executed.

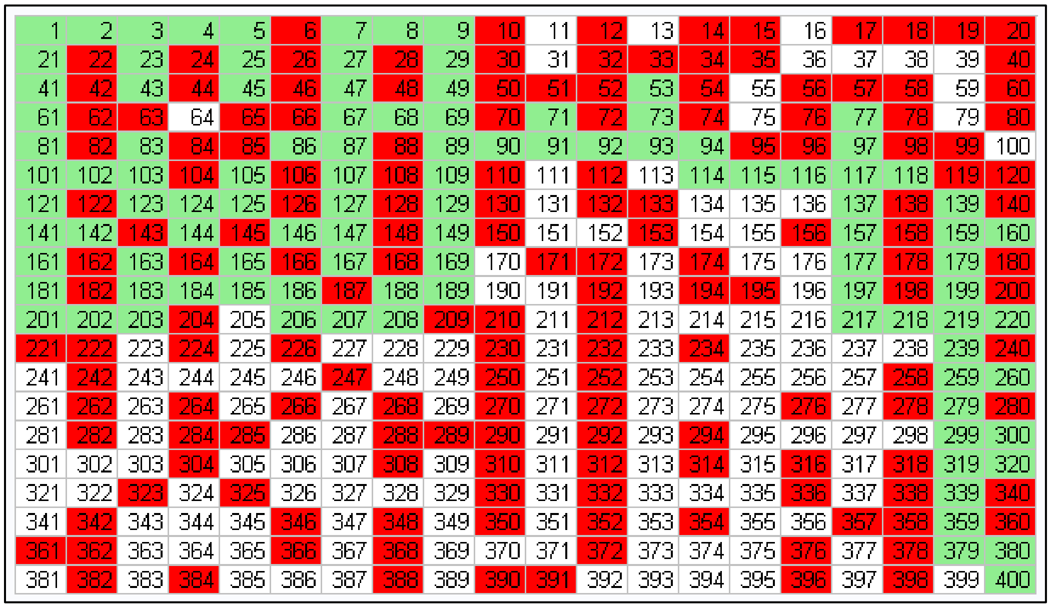


1	2	3	4	5	**6**	7	8	9	**10**	11	**12**	13	**14**	**15**	16	**17**	**18**	**19**	**20**
21	**22**	23	**24**	25	**26**	27	**28**	29	**30**	31	**32**	**33**	**34**	**35**	36	37	38	39	**40**
41	**42**	43	**44**	45	**46**	47	**48**	49	**50**	**51**	**52**	53	**54**	55	**56**	**57**	**58**	59	**60**
61	**62**	**63**	64	**65**	**66**	67	68	69	**70**	71	**72**	73	**74**	75	**76**	77	**78**	79	**80**
81	**82**	83	**84**	**85**	86	87	**88**	89	90	91	92	93	94	**95**	**96**	97	**98**	**99**	100
101	102	103	**104**	105	**106**	107	**108**	109	**110**	111	**112**	113	114	115	116	117	118	**119**	**120**
121	**122**	123	124	125	**126**	127	**128**	129	**130**	131	**132**	**133**	134	135	136	137	**138**	139	**140**
141	142	**143**	144	**145**	146	147	**148**	149	**150**	151	152	**153**	154	155	**156**	157	**158**	159	160
161	**162**	163	**164**	165	**166**	167	**168**	169	170	**171**	**172**	173	**174**	175	176	177	**178**	179	**180**
181	**182**	183	184	185	186	**187**	188	189	190	191	**192**	193	**194**	**195**	196	197	**198**	199	**200**
201	202	203	**204**	205	206	207	208	**209**	**210**	211	**212**	213	214	215	216	217	218	219	220
221	**222**	223	**224**	225	**226**	227	228	229	**230**	231	**232**	233	**234**	235	236	237	238	239	**240**
241	**242**	243	244	245	246	**247**	248	249	**250**	251	**252**	253	254	255	256	257	**258**	259	260
261	**262**	263	**264**	265	**266**	267	**268**	269	**270**	271	**272**	273	274	275	**276**	277	**278**	279	**280**
281	**282**	283	**284**	**285**	286	287	**288**	**289**	**290**	291	**292**	293	**294**	295	296	297	298	299	300
301	302	303	**304**	305	306	307	**308**	309	**310**	311	**312**	313	**314**	315	**316**	317	**318**	319	320
321	322	**323**	324	**325**	326	327	328	329	**330**	331	**332**	333	334	335	**336**	337	**338**	339	**340**
341	**342**	343	344	345	**346**	347	**348**	349	**350**	351	**352**	353	**354**	355	356	**357**	**358**	359	**360**
361	**362**	363	364	365	**366**	367	**368**	369	370	371	**372**	373	374	375	**376**	377	**378**	379	380
381	**382**	383	**384**	385	386	387	**388**	389	**390**	**391**	392	393	394	395	**396**	397	**398**	399	400

In order to solve this puzzle the user first needs to understand which of the numbers represent the walls in the digital maze; they will soon discover if they move onto a space that is defined as a wall they will be reset to the starting location. By following the rules laid out they should be able to generate a formatted dataset that highlights the walls.

```
* Run the puzzle_maze_move to move orthogonally;
%puzzle_maze_move(move=up,    display=YES);
%puzzle_maze_move(move=down,  display=YES);
%puzzle_maze_move(move=left,  display=YES);
%puzzle_maze_move(move=right, display=YES);
```

At this point the user can manually enter the directions to navigate from 1 to 400 based on their observations. An alternate strategy to solve this automatically would be to use the classic maze solving technique of placing one of your hands on the maze wall and not removing it until you reach the exit. This is more of a brute force approach and will mean you explore a lot of dead ends as you traverse the maze, but eventually through an exhaustive approach you will reach the destination. This method is often called 'wall follower' but there are various algorithms for solving mazes.[2] The below diagram is the implementation of a macro to replicate the 'wall follower' method. The first twenty moves of the route it took were 1 -> 2 -> 3 -> 4 -> 5 -> 25 -> 45 -> 25 -> 5 -> 4 -> 3 -> 23 -> 43 -> 23 -> 3 -> 2-> 1-> 21 -> 41 -> 61 -> etc. Once you understand how it is moving it is easy to see how it ended up at this solution.



🟩 1	🟩 2	🟩 3	🟩 4	🟩 5	🟥 6	🟩 7	🟩 8	🟩 9	🟥 10	11	🟥 12	13	🟥 14	🟥 15	16	🟥 17	🟥 18	🟥 19	🟥 20
🟩 21	🟥 22	🟩 23	🟥 24	🟩 25	🟥 26	🟩 27	🟥 28	🟩 29	🟥 30	31	🟥 32	🟥 33	🟥 34	🟥 35	36	37	38	39	🟥 40
🟩 41	🟥 42	🟩 43	🟥 44	🟩 45	🟥 46	🟩 47	🟥 48	🟩 49	🟥 50	🟥 51	🟥 52	🟩 53	🟥 54	55	🟥 56	🟥 57	🟥 58	59	🟥 60
🟩 61	🟥 62	🟥 63	64	🟥 65	🟥 66	🟩 67	🟩 68	🟩 69	🟥 70	🟩 71	🟥 72	🟩 73	🟥 74	75	🟥 76	🟩 77	🟥 78	79	🟥 80
🟩 81	🟥 82	🟩 83	🟥 84	🟥 85	🟩 86	🟩 87	🟥 88	🟩 89	🟩 90	🟩 91	🟩 92	🟩 93	🟩 94	🟥 95	🟥 96	🟩 97	🟥 98	🟥 99	100
🟩 101	🟩 102	🟩 103	🟥 104	🟩 105	🟥 106	🟩 107	🟥 108	🟩 109	🟥 110	111	🟥 112	113	🟩 114	🟩 115	🟩 116	🟩 117	🟩 118	🟥 119	🟥 120
🟩 121	🟥 122	🟩 123	🟩 124	🟩 125	🟥 126	🟩 127	🟥 128	🟩 129	🟥 130	131	🟥 132	🟥 133	134	135	136	🟩 137	🟥 138	🟩 139	🟥 140
🟩 141	🟩 142	🟥 143	🟩 144	🟥 145	🟩 146	🟩 147	🟥 148	🟩 149	🟥 150	151	152	🟥 153	154	155	🟥 156	🟩 157	🟥 158	🟩 159	🟩 160
🟩 161	🟥 162	🟩 163	🟥 164	🟩 165	🟥 166	🟩 167	🟥 168	🟩 169	170	🟥 171	🟥 172	173	🟥 174	175	176	🟩 177	🟥 178	🟩 179	🟥 180
🟩 181	🟥 182	🟩 183	🟩 184	🟩 185	🟩 186	🟥 187	🟩 188	🟩 189	190	191	🟥 192	193	🟥 194	🟥 195	196	🟩 197	🟥 198	🟩 199	🟥 200
🟩 201	🟩 202	🟩 203	🟥 204	205	🟩 206	🟩 207	🟩 208	🟥 209	🟥 210	211	🟥 212	213	214	215	216	🟩 217	🟩 218	🟩 219	🟩 220
🟥 221	🟥 222	223	🟥 224	225	🟥 226	227	228	229	🟥 230	231	🟥 232	233	🟥 234	235	236	237	238	🟩 239	🟥 240
241	🟥 242	243	244	245	246	🟥 247	248	249	🟥 250	251	🟥 252	253	254	255	256	257	🟥 258	🟩 259	🟩 260
261	🟥 262	263	🟥 264	265	🟥 266	267	🟥 268	269	🟥 270	271	🟥 272	273	274	275	🟥 276	277	🟥 278	🟩 279	🟥 280
281	🟥 282	283	🟥 284	🟥 285	286	287	🟥 288	🟥 289	🟥 290	291	🟥 292	293	🟥 294	295	296	297	298	🟩 299	🟩 300
301	302	303	🟥 304	305	306	307	🟥 308	309	🟥 310	311	🟥 312	313	🟥 314	315	🟥 316	317	🟥 318	🟩 319	🟩 320
321	322	🟥 323	324	🟥 325	326	327	328	329	🟥 330	331	🟥 332	333	334	335	🟥 336	337	🟥 338	🟩 339	🟥 340
341	🟥 342	343	344	345	🟥 346	347	🟥 348	349	🟥 350	351	🟥 352	353	🟥 354	355	356	🟥 357	🟥 358	🟩 359	🟥 360
🟥 361	🟥 362	363	364	365	🟥 366	367	🟥 368	369	370	371	🟥 372	373	374	375	🟥 376	377	🟥 378	🟩 379	🟩 380
381	🟥 382	383	🟥 384	385	386	387	🟥 388	389	🟥 390	🟥 391	392	393	394	395	🟥 396	397	🟥 398	399	🟩 400

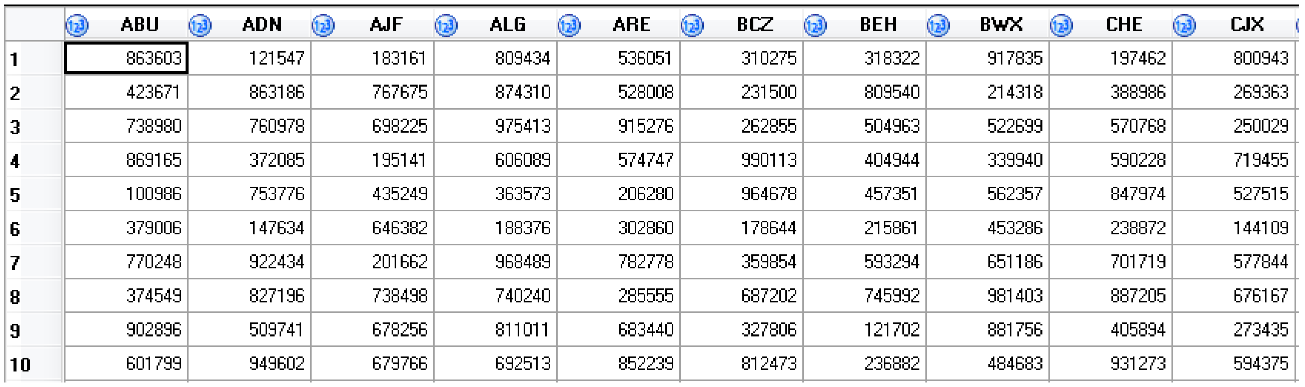
PUZZLE 5 – THE JOINING PUZZLE

The joining puzzle touches some of the same areas as the safe puzzle, requiring an understanding of loops and macros to solve. However, there is an added degree of complexity that will mean users have to push their technical skills a little further this time. The code the user is initially presented with is as follows:

```
/*************************************************************************************/
/*** Run the puzzle_join_initialise to build the join datasets. There are three      ***/
/*** datasets each with 100 columns and 1000 rows.                                   ***/
/*** ~ Only one column in each dataset will allow the datasets to be joined          ***/
/*** ~ Once the data is joined the columns will reveal the positive code             ***/
/*************************************************************************************/
%puzzle_join_initialise();

/*************************************************************************************/
/*** Enter the positive code into puzzle_join_check_code to solve the puzzle         ***/
/*************************************************************************************/
%puzzle_join_check_code(code=???);
```

By running the %puzzle_join_initialise macro the users will randomly generate three datasets, each with 100 columns and 1000 rows. Every cell will contain a 6-digit number but only a single 6-digit number is repeated across all three datasets. This means there is only a single cell that belongs to a single column in each dataset that can actually be joined. The users need to develop a way of finding out what columns can be joined in order to determine the code and solve the puzzle.



	ABU	ADN	AJF	ALG	ARE	BCZ	BEH	BWX	CHE	CJX
1	863603	121547	183161	809434	536051	310275	318322	917835	197462	800943
2	423671	863186	767675	874310	528008	231500	809540	214318	388986	269363
3	738980	760978	698225	975413	915276	262855	504963	522699	570768	250029
4	869165	372085	195141	606089	574747	990113	404944	339940	590228	719455
5	100986	753776	435249	363573	206280	964678	457351	562357	847974	527515
6	379006	147634	646382	188376	302860	178644	215861	453286	238872	144109
7	770248	922434	201662	968489	782778	359854	593294	651186	701719	577844
8	374549	827196	738498	740240	285555	687202	745992	981403	887205	676167
9	902896	509741	678256	811011	683440	327806	121702	881756	405894	273435
10	601799	949602	679766	692513	852239	812473	236882	484683	931273	594375

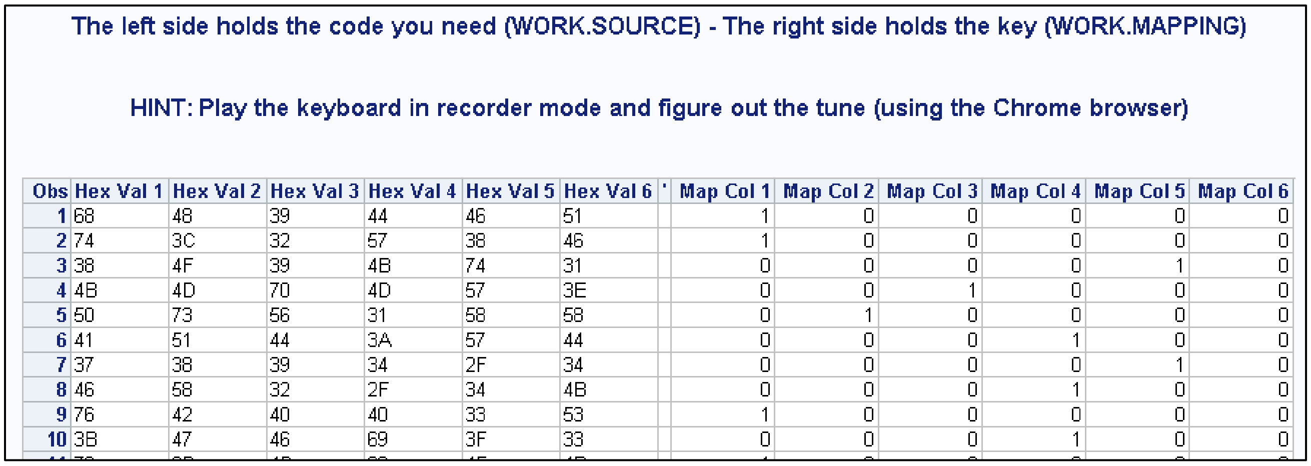
PUZZLE 6 – THE ENCRYPTED MUSIC PUZZLE

There are two key components to this puzzle. Firstly, the puzzle is designed to test a user's knowledge of arrays and ability to find the functions they need to resolve an issue. Secondly, once they've cracked the code, they then have to be able to follow the instructions in order to play a tune on a virtual piano before identifying the tune. If they don't recognise the tune themselves they may have to reach out to someone else to see if they can help them identify it – knowing when to ask for help is also a skill!

```
/*****************************************************************************************/
/*** Run the puzzle_music macro as many times as you like.                             ***/
/***                                                                                   ***/
/*** Your aim is combine the SOURCE and MAPPING datasets in such a way that you:       ***/
/*** ~ a) Find which website you need to access (recommended browser is Chrome)        ***/
/*** ~ b) Enter the code into the website (using the recorder function within the     ***/
/***      website)                                                                     ***/
/*** ~ c) Identify the resultant tune that is played and submit your answer using     ***/
/***      the macro                                                                    ***/
/***                                                                                   ***/
/*****************************************************************************************/

%puzzle_music(tune_name=);
```

The encryption part of this puzzle requires the user to convert hexadecimal codes to ASCII and then use the mapping dataset to only keep the values from the hexadecimal array, which have a 1 in their mapping column counterpart. There is only one mapping column with a value of 1 per row.

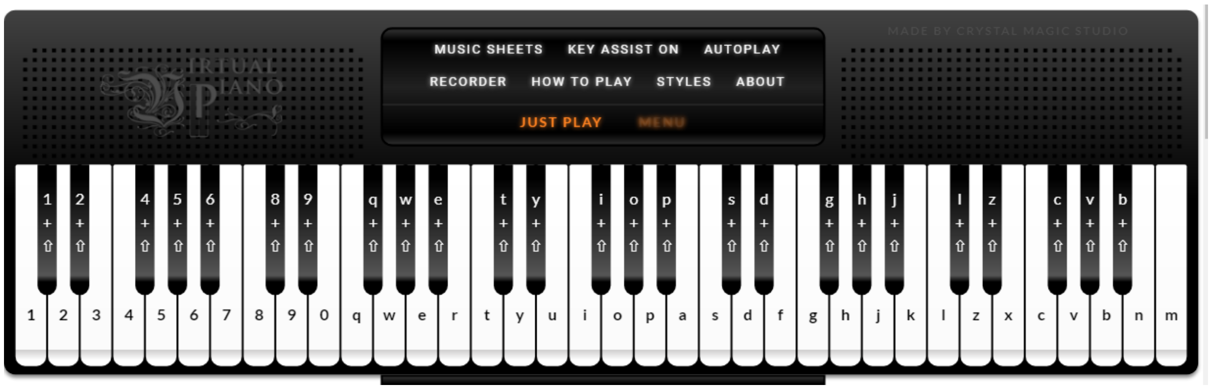


The left side holds the code you need (WORK.SOURCE) - The right side holds the key (WORK.MAPPING)

HINT: Play the keyboard in recorder mode and figure out the tune (using the Chrome browser)

Obs	Hex Val 1	Hex Val 2	Hex Val 3	Hex Val 4	Hex Val 5	Hex Val 6	'	Map Col 1	Map Col 2	Map Col 3	Map Col 4	Map Col 5	Map Col 6
1	68	48	39	44	46	51		1	0	0	0	0	0
2	74	3C	32	57	38	46		1	0	0	0	0	0
3	38	4F	39	4B	74	31		0	0	0	0	1	0
4	4B	4D	70	4D	57	3E		0	0	1	0	0	0
5	50	73	56	31	58	58		0	1	0	0	0	0
6	41	51	44	3A	57	44		0	0	0	1	0	0
7	37	38	39	34	2F	34		0	0	0	0	1	0
8	46	58	32	2F	34	4B		0	0	0	1	0	0
9	76	42	40	40	33	53		1	0	0	0	0	0
10	3B	47	46	69	3F	33		0	0	0	1	0	0

Once the users have resolved the encryption part, they then need to go to the URL they are directed to and enter the code as instructed to be played on the virtual piano.[3] The name of the tune is the puzzles answer.


MADE BY CRYSTAL MAGIC STUDIO
VIRTUAL PIANO
MUSIC SHEETS
KEY ASSIST ON
AUTOPLAY
RECORDER
HOW TO PLAY
STYLES
ABOUT
JUST PLAY
MENU
1 2 4 5 6 8 9 q w e t y i o p s d g h j l z c v b
1 2 3 4 5 6 7 8 9 0 q w e r t y u i o p a s d f g h j k l z x c v b n m

PUZZLE 7 – THE GRAPH PUZZLE

As expected of any SAS education piece we couldn't not include graphs. The GPLOT procedure and SGPLOT procedure are extremely useful when wanting to quickly view trends of data; so we wanted to make use of them within a puzzle. In order to solve this puzzle, users will need to utilise the equations provided and plot a graph that reveals an image. Only the equations for the first image are provided initially.

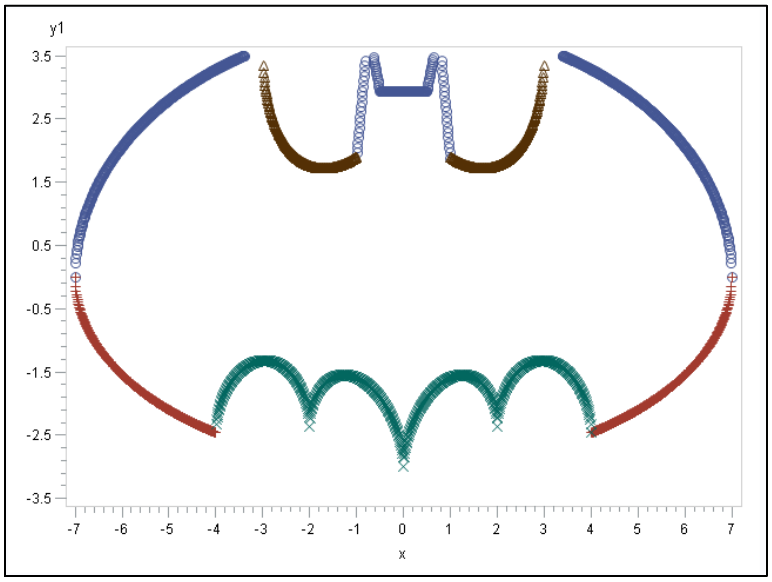
```
/**************************************************************************************/
/*** 1) The answer to image1 is what you see when you plot the below equations:      ***/
/***    x = 16sin(t)^3                                                               ***/
/***    y = 13cos(t)-5cos(2t)-2cos(3t)-cos(4t)                                       ***/
/***    for 0<t<6.3                                                                  ***/
/***                                                                                 ***/
/*** 2) To be able to get the answer to image2 you will need to await further        ***/
/***    instructions!                                                                ***/
/**************************************************************************************/
%puzzle_graphs(image1=???,image2=???);
```

Once the users have identified the first image they will be presented with a series of more complex equations that they will need to graph.[4]

```
INFO: The answer to image2 is what you see when you plot the below equations:

y1 = 2sqrt(-abs(abs(x)-1)*abs(3-abs(x))/((abs(x)-1)*(3-abs(x))))*(1+abs(abs(x)-3)/(abs(x)-
3))*sqrt(1-(x/7)^2)+(5+0.97*(abs(x-.5)+abs(x+.5))-3(abs(x-.75)+abs(x+.75)))*(1+abs(1-
abs(x))/(1-abs(x)))
y2 = -3sqrt(1-(x/7)^2)*sqrt(abs(abs(x)-4)/(abs(x)-4))
y3 = abs(x/2)-0.0913722(x^2)-3+sqrt(1-(abs(abs(x)-2)-1)^2)
y4 = (2.71052+(1.5-.5abs(x))-1.35526sqrt(4-(abs(x)-1)^2))*sqrt(abs(abs(x)-1)/(abs(x)-1))+0.9
for -7<x<7
```

Once the equations have successfully been graphed, it should be clear what the second image is.



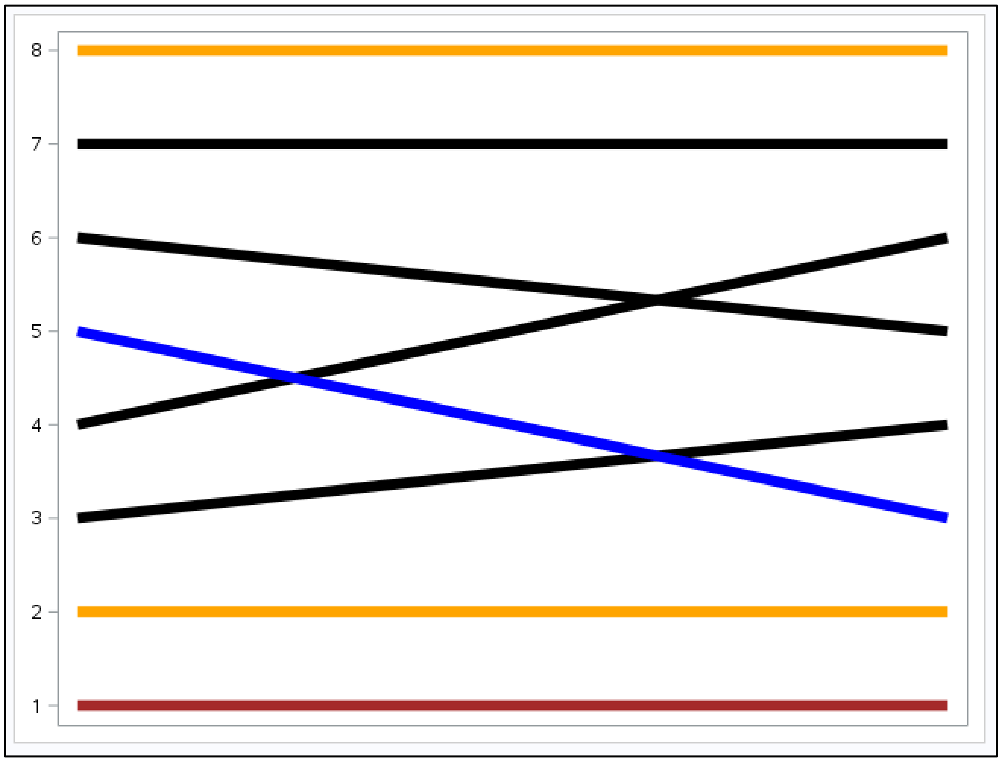
PUZZLE 8 – THE CIRCUIT PUZZLE

The circuit puzzle involves a change of pace with it being the only timed puzzle. Like several of the other puzzles, you begin by initialising a start position and then have a second macro to interact with the output produced by the first.

```
/********************************************************************************/
/*** Run puzzle_circuit_initialise to generate a circuit and start the countdown     ***/
/********************************************************************************/
%puzzle_circuit_initialise();

/********************************************************************************/
/*** Run puzzle_circuit_cut_wires to cut (Y) or not (N) wires for an active circuit  ***/
/********************************************************************************/
%puzzle_circuit_cut_wires(cut_wire1=N,cut_wire2=N,cut_wire3=N,cut_wire4=N,
  cut_wire5=N,cut_wire6=N,cut_wire7=N,cut_wire8=N);
```

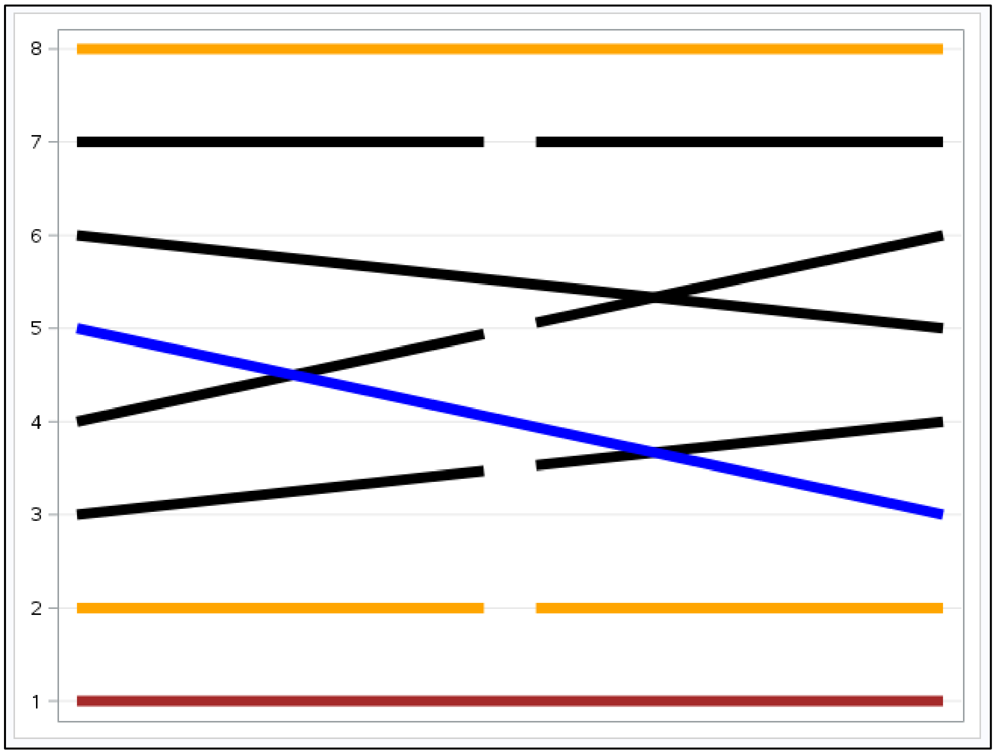
Once you have initialised a circuit you will have 60 seconds to deactivate it before it is destroyed. When the circuit is initialised, an rsubmit is used to spawn an additional session that will output a file after 60 seconds that prevents the user from being able to solve the circuit at that point. Each circuit is randomly generated. There are always 8 wires but those wires can cross one or more of their neighbours. Each wire has a chance of randomly being assigned 1 of 8 colours. After the 60 seconds are up the user can initialise another circuit.



The puzzle gives the user a comprehensive set of instructions to disarm the circuit. However, the design is such that it should be near impossible to manually apply all 17 instructions in 60 seconds. The developer will have to use the WIRES and COLOURS datasets output on initialisation to develop an automated solution. The instructions are as follows:

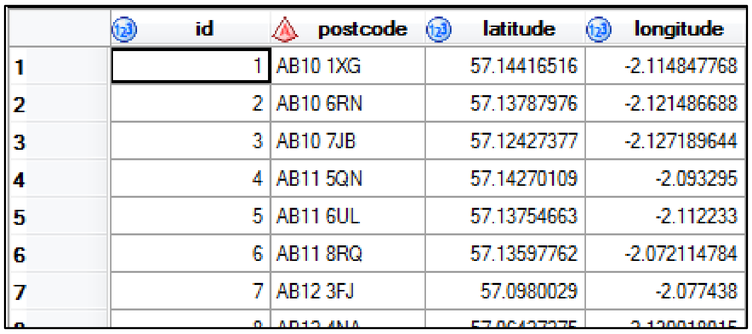
1) Cut wire 1 if at least two of the following colours are not present in the circuit: BLACK, BLUE, BROWN & GREEN
2) Cut wire 2 if at least two of the following colours are not present in the circuit: ORANGE, PURPLE, RED & YELLOW
3) Cut wire 3 if there is exactly one BLACK wire and one BROWN wire in the circuit unless there is also exactly one ORANGE wire and one RED wire in the circuit
4) Cut wire 4 if there is exactly one BLUE wire and one GREEN wire in the circuit unless there is also exactly one PURPLE wire and one YELLOW wire in the circuit
5) Cut wire 5 if at least two of the following colours have exactly two wires present in the circuit: BLACK, GREEN, ORANGE & YELLOW
6) Cut wire 6 if at least two of the following colours have exactly two wires present in the circuit: BLUE, BROWN, PURPLE & RED
7) Cut wire 7 if any of the following colours have three or more wires present in the circuit: BLACK, BLUE, ORANGE & PURPLE
8) Cut wire 8 if any of the following colours have three or more wires present in the circuit: BROWN, GREEN, RED & YELLOW
9) If wire 1 crosses a BLACK or PURPLE wire then do the opposite of what you would have done to wire 1 based on the previous instructions (e.g. cut becomes don't cut)
10) If wire 2 crosses exactly 1 wire then do the opposite of what you would have done to wire 2 based on the previous instructions (e.g. cut becomes don't cut)
11) If wire 3 crosses a BLUE or RED wire then do the opposite of what you would have done to wire 3 based on the previous instructions (e.g. cut becomes don't cut)
12) If wire 4 crosses exactly 2 wires then do the opposite of what you would have done to wire 4 based on the previous instructions (e.g. cut becomes don't cut)
13) If wire 5 crosses a BROWN or YELLOW wire then do the opposite of what you would have done to wire 5 based on the previous instructions (e.g. cut becomes don't cut)
14) If wire 6 crosses exactly 2 wires then do the opposite of what you would have done to wire 6 based on the previous instructions (e.g. cut becomes don't cut)
15) If wire 7 crosses a GREEN or ORANGE wire then do the opposite of what you would have done to wire 7 based on the previous instructions (e.g. cut becomes don't cut)
16) If wire 8 crosses exactly 1 wire then do the opposite of what you would have done to wire 8 based on the previous instructions (e.g. cut becomes don't cut)
17) If after following all other instructions the solution is to cut no wires then instead cut all wires

In this instance, the instructions are pointing us to cut wires 2, 3, 4 and 7 to disarm the circuit. Once the user runs the macro and specifies which wires to cut, they will be presented with a second image of the circuit and their success or failure to disarm the circuit in 60 seconds will be put to the log.



PUZZLE 9 – THE X MARKS THE SPOT PUZZLE

The penultimate puzzle marks a significant increase in difficulty. By this point, the users should be looking out for any clues that will help them. The background information tells them that a formation of four Barclays offices will provide them with an X and that the solution is the postcode nearest to the intersecting lines of that X. It also tells them that the Earth has been assumed to be a sphere and the value used for the radius of the Earth. When the users run %puzzle_x_marks_initialise to begin this puzzle they will be provided with a dataset with 1,762,315 UK postcodes as well as the latitude and longitude points for those postcodes. This dataset was downloaded from an open source site.[5]

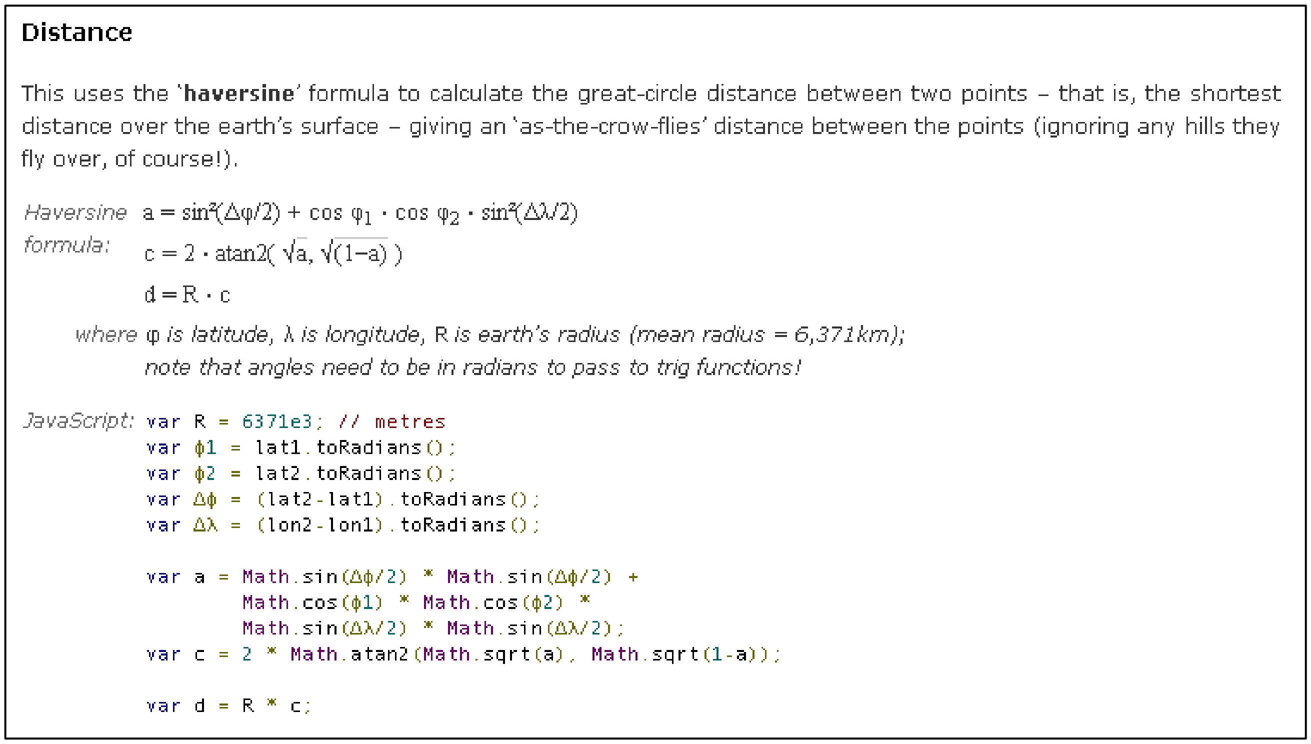


	id	postcode	latitude	longitude
1	1	AB10 1XG	57.14416516	-2.114847768
2	2	AB10 6RN	57.13787976	-2.121486688
3	3	AB10 7JB	57.12427377	-2.127189644
4	4	AB11 5QN	57.14270109	-2.093295
5	5	AB11 6UL	57.13754663	-2.112233
6	6	AB11 8RQ	57.13597762	-2.072114784
7	7	AB12 3FJ	57.0980029	-2.077438

Once the user has seen the dataset they are likely to try and solve this puzzle one of three ways.

1) Try and genuinely solve the puzzle using some reasonably complex equations which will involve
    a. Finding the great-circle distance between the offices which are diagonal to each other
    b. Finding the initial bearing to travel between the pairs of offices
    c. Finding the intersection point based on the start points and bearings
    d. Finding the postcode nearest to the intersection point
2) Try and genuinely solve the puzzle but mistakenly use equations for a plane and not a sphere
3) Try to brute force solve the puzzle

The first approach uses equations like the Haversine formula; for some this might be too much mathematics.



Distance

This uses the '**haversine**' formula to calculate the great-circle distance between two points – that is, the shortest distance over the earth's surface – giving an 'as-the-crow-flies' distance between the points (ignoring any hills they fly over, of course!).

Haversine formula:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

where φ *is latitude,* λ *is longitude,* R *is earth's radius (mean radius = 6,371km);*
note that angles need to be in radians to pass to trig functions!

JavaScript:

```
var R = 6371e3; // metres
var φ1 = lat1.toRadians();
var φ2 = lat2.toRadians();
var Δφ = (lat2-lat1).toRadians();
var Δλ = (lon2-lon1).toRadians();

var a = Math.sin(Δφ/2) * Math.sin(Δφ/2) +
        Math.cos(φ1) * Math.cos(φ2) *
        Math.sin(Δλ/2) * Math.sin(Δλ/2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

var d = R * c;
```

The second approach gets users pretty close to the right answer when they treat the latitude and longitude as points on a plane. To reward their efforts a special note is put to the log in this scenario, which lets them know they are within 5km of the correct answer.

```
INFO: This is the answer if you treat the coordinates as a plane and not a sphere - but
you're within a 5km radius!
 * Increasing time penalty for incorrect answers by a second!
```

The final approach of brute force is something they will have been able to utilise in other challenges and so this time we've made it a little trickier. As per the above message in the log, every incorrect answer yields an increase in time penalty of a second; as such, it would take around 49,241 years to brute force all of the postcodes. The key to brute force working here is the user needs to realise there is something keeping track of their efforts from one round to the next, it can't all be local to that one macro call. If they look at the current global macro variables they will see one called sleep_secs and if they set this to 0 before every macro call they will sleep for 0 seconds instead of what would have been the penalty amount.

Calling a macro up to 1,762,315 times is still going to take a little while so there are more hints available to help speed up the process. When entering most incorrect postcodes you will get the following message.

```
INFO: You're cold!
```

The message “You’re cold!” is meant to serve two purposes; first to tell them they are incorrect and second to imply if they get in the right ball park they will be told something like “You’re getting warmer!”. In the UK, there are six different formats a postcode can come in but they are all made up of an area, district, sector and unit. There are only 125 unique areas and if a postcode containing the correct area is entered, you get.

```
INFO: That's the correct POSTCODE AREA - You're getting warm!
```

In total there are 2,983 unique districts, but within the correct area there are only 24. If a postcode containing the correct area and district is entered, you get.

```
INFO: That's the correct POSTCODE DISTRICT - You're pretty hot!
```

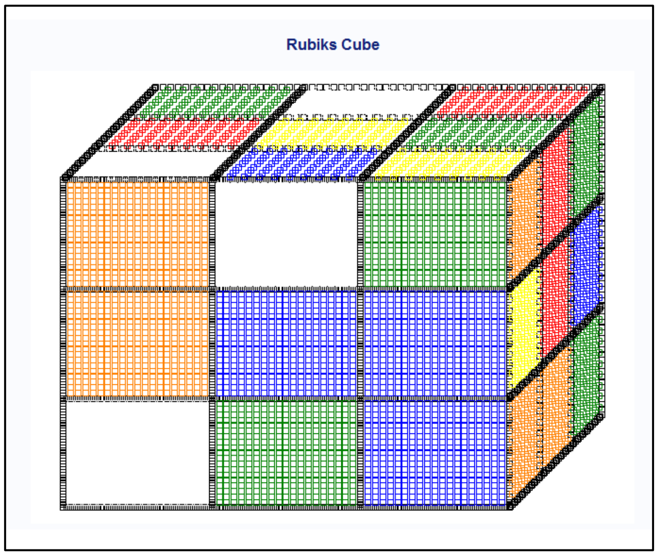
While there might be 11,241 sectors in the UK, within the correct area and district there are only 6. If a postcode containing the correct area, district and sector is entered, you get.

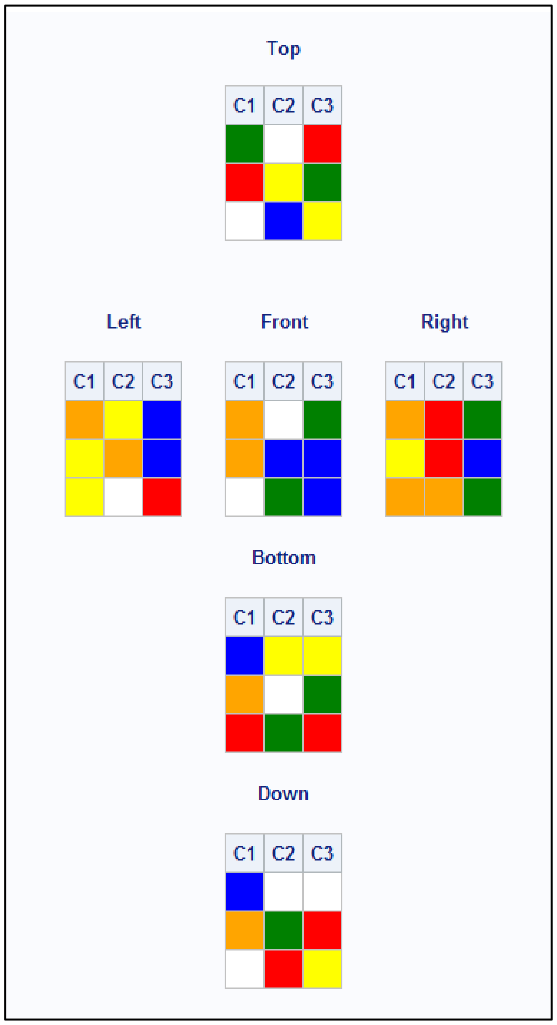
```
INFO: That's the correct POSTCODE SECTOR - You're burning up!
```

Whilst this puzzle is one of the most challenging, the variety of solutions available to people with a keen eye makes it an interesting addition.

PUZZLE 10 – THE RUBIK'S PUZZLE

In time-honoured tradition, we saved the best puzzle until last. For the final puzzle, we have a fully functional digital Rubik's cube. In the background, the cube consists of 54 macro variables that define the current state of the cube. The cube is displayed using a proc sgplot that is applied to a dataset with tens of thousands of data points. The data points give the illusion of a cube in the exact same way pixels work on a monitor. The users start by initialising a randomly generated cube. The default is to display the cube in 3D but it can also be displayed in 2D.





Top

C1	C2	C3
🟩	⬜	🟥
🟥	🟨	🟩
⬜	🟦	🟨

Left

C1	C2	C3
🟧	🟨	🟦
🟨	🟧	🟦
🟨	⬜	🟥

Front

C1	C2	C3
🟧	⬜	🟩
🟧	🟦	🟦
⬜	🟩	🟦

Right

C1	C2	C3
🟧	🟥	🟩
🟨	🟥	🟦
🟧	🟧	🟩

Bottom

C1	C2	C3
🟦	🟨	🟨
🟧	⬜	🟩
🟥	🟩	🟥

Down

C1	C2	C3
🟦	⬜	⬜
🟧	🟩	🟥
⬜	🟥	🟨

The objective for this puzzle is what you would expect, solve the Rubik's cube. In order to do this the users are given a couple of tools. The %puzzle_rubiks_turn macro allows the cube to be turned up, right, left and down in order to allow the cube to be viewed from other angles when displayed in 3D.

The %puzzle_rubiks_rotate macro allows you to rotate each of the faces (front, up, left, right, down, back) in either direction (clock-wise or anti-clockwise). Every option has a definition for how to manipulate the 54 macro variables to represent that rotation.

For those who already know how to solve a Rubik's cube the puzzle will be relatively simple. For those less well versed with the Rubik's cube, if they can muster up some search engine skills they will easily find websites that offer to solve Rubik's cubes in the minimum number of moves necessary.[6] For those looking to solve this purely in SAS but lacking the skills to do, there is the "pull the stickers off method". With the Rubik's cube being tracked using 54 macro variables, if the users can figure out which macro variables to change and update then they too will solve the cube and beat the puzzle.

CONCLUSION

The puzzles in this paper were designed with two purposes in mind. First off, we wanted to find a fun way to help people learn and explore some of the classic programming features within SAS. There is nothing new about getting people to use loops, arrays or macros but that doesn't mean we should stop trying to find new ways to teach old techniques. The second reason for doing this was to try and teach people a new way of thinking. Our hope is that once they've seen a dataset displayed as a crossword, maze or a Rubik's cube, that they will start to visualise and think of data differently.

Like all other software, SAS has limitations; but far too often, what users perceive to be a limitation in the software is actually a limitation in their own imagination. The functions needed to generate a Rubik's cube have existed within SAS for years but we didn't imagine it was possible to construct one until we challenged ourselves to create these puzzles. In that same way I hope the puzzles will help expand other people's perceptions of what is possible so they too can create things they wouldn't have otherwise imagined.

REFERENCES

1. Let me google that for you, https://lmgtfy.com/
2. Wikipedia, https://en.wikipedia.org/wiki/Maze_solving_algorithm
3. Virtual Piano, https://virtualpiano.net/
4. Stack Exchange, https://math.stackexchange.com/questions/54506/is-this-batman-equation-for-real
5. GitHub, https://github.com/dwyl/uk-postcodes-latitude-longitude-complete-csv
6. Rubiks Cube Solver, https://rubiks-cube-solver.com/

CONTACT INFORMATION

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

Amit Patel, Email: amit.xc.patel@barclayscorp.com
Amit Patel has been developing solutions across a range of SAS products since 2011 and has worked as a SAS technical lead within several organisations including two of the big four banks in the UK.

Lewis Mitchell, Email: lewis.mitchell@barclayscorp.com
Lewis Mitchell has been developing solutions across a range of SAS products since 2009 and has worked as a SAS technical lead within several organisations including three of the big four banks in the UK.