“Bored”-Room Buster Bingo - Create Bingo Cards Using SAS® ODS Graphics

Richann Watson, DataRich Consulting; Louise Hadden, Abt Associates Inc.

ABSTRACT

Let’s admit it! We have all been on a conference call that just … well to be honest, it was just bad. Your misery could be caused by any number of reasons – or multiple reasons! The audio quality was bad, the conversation got sidetracked and focus of the meeting was no longer what it was intended, there could have been too much background noise, someone hasn’t muted their laptop and is breathing heavily – the list goes on ad nauseum. Regardless of why the conference call is less than satisfactory, you want it to end, but professional etiquette demands that you remain on the call. We have the answer – SAS®-generated Conference Call Bingo! Not only is Conference Call Bingo entertaining, but it also keeps you focused on the conversation and enables you to obtain the pertinent information the conference call may offer.

This paper and presentation introduce a method of using SAS to create custom Conference Call Bingo cards, moving through brainstorming and collecting entries for Bingo cards, random selection of items, and the production of bingo cards using SAS reporting techniques and the Graphic Template Language (GTL). (You are on your own for the chips and additional entries based on your own painful experiences)! The information presented is appropriate for all levels of SAS programming and all industries.

INTRODUCTION

Most people assume that you can only produce graphs such as series plots, bar charts or scatter plots with SAS graphing tools. However, you can do more with these valuable assets: you can create images and even bingo cards. In this paper we demonstrate how to create your own customized bingo cards using SAS ODS Graphics.

ODS Graphics encompasses more than just the Statistical Graphics (SG) procedures. It also includes Graph Template Language (GTL). ODS Graphics allows for the customization of graphs by allowing you to add or modify various features within a graph. With GTL you can easily incorporate various features or you to create truly unique outputs such as a bingo card.

The program was successfully tested on SAS version 9.4, maintenance releases 5 and 6, at the date of publication.

COMPILING A LIST OF THINGS FOR YOUR BINGO CARD

First, you need to have a list of items that will be used to populate your bingo cards. The items can be stored in an Excel spreadsheet or your favorite data collection software, as long as you can convert the data to a SAS data set.

Your list only needs to have one column which contains the items you would like to be selected for the bingo cards. Refer to Data Display 1 for a sample.

Data Display 1: Sample of Items for Bingo Cards

Things you hear or see on a call
Hi! Who just joined?
Can you email…that to everyone?
___, Are you there?
I don't think ___ is on the call
Uh, ___, you're still sharing…
I have to jump to another call
Can you see my screen?
It's still loading

RANDOMLY SELECTING ITEMS TO FILL YOUR BINGO CARD

SAS has myriad methods for random selection, as do other software packages. Two different methods of random selection are demonstrated: the data step and RAND function, and SAS's PROC SURVEYSELECT. Both are extremely powerful but can also be very simple and easy to use. Either technique works for the purpose of generating the bingo cards.

Random Selection using the Data Step and the RAND Function

Those of us with birthdays of negative SAS date vintage will recall RANUNI, UNIFORM, and other random selection functions. Currently, those older functions have been deprecated, and replace with an umbrella function called the RAND function. The RAND function can be used with arguments such as UNIFORM to select specific sampling algorithms. It is frequently used with CALL STREAMINIT to specify a random number generation method. In SAS Program 1 shown below, the seed is auto-generated with the PCG method (64-bit permuted congruential generator). The RAND function takes an argument describing the desired distribution, from distributions ranging from BERNOULLI to WEIBULL. We use a more typical distribution, UNIFORM. Once random numbers have been generated, the file is sorted by the random number and the data set is subsetted to the desired 25 records. This is repeated until there are cards for all players.

SAS Program 1: Random Selection using the Data Step and the RAND function

```
data bingo&i._a;
   set bingo;
   call streaminit('PCG', 0); /* auto-generate seed */
   __run = rand('uniform');
run;

proc sort data = bingo&i._a;
   by __run;
run;

data bingo&i;
   set bingo&i._a;
   by __run;
   if _n_ le 25;
run;
```

Random Selection using the PROC SURVEYSELECT

PROC SURVEYSELECT is one of a family of SAS procedures which analyze complex survey data taking into account the survey sample design. PROC SURVEYSELECT allows for the selection of samples for surveys, and can range from the extraordinarily simple, shown below in SAS Program 2 to the incredibly complex through the use of options and parameters. For this very simple design, SRS (Simple Random Sampling) design was used, a sample size of 25 specified which is the number of records required for the bingo cards. PROC SURVEYSELECT is truly "one stop shopping" and outputs a data set with the correct parameters in a single step.

SAS Program 2: Random Selection using PROC SURVEYSELECT

```
proc surveyselect data = bingo method = srs n = 25 out = bingo&i;
run;
```

FORMATTING DATA FOR YOUR BINGO CARD

After items for the bingo cards have been randomly selected, the data will need to be formatted in order to use the template. Because bingo cards are typically 5 x 5, you will need to create ROWNUM and COLNUM variables to indicate which cell a particular value is placed. In addition, because the template uses TEXTPLOT (refer to SAS Program 3), you will need to create variables that will be associated with the X and Y axes within each cell. The last required variable for the graph structure defined in SAS Program 3 is B_TEXT. Below is a list of the variables and a sample of the data set is seen in Data Display 2.

- BINGO_TEXT: Value that was selected from the list of possible items (Data Display 1).
- ROWNUM: Represents which row on the bingo card the value will be displayed.
- COLNUM: Represents which column on the bingo card the value will be displayed.
- XVAL and YVAL: Default to 1 because each cell in the grid contains only one value.
- B_TEXT: Modified text to incorporate split character variable in order to force the text to wrap within the cell. This is the value that will actually be used when defining the template.
- GRP: (*Optional*) Used to indicate color for that particular value. This is only used if you request multiple colors.

Data Display 2: Sample of Used to Produce Bingo Card

BINGO_TEXT	ROWNUM	COLNUM	XVAL	YVAL	B_TEXT	GRP
Hi! Who just joined?	1	1	1	1	Hi! Who just^joined?	1
Can you email…that to everyone?	1	2	1	1	Can you^email…that^to everyone?	2
___, Are you there?	1	3	1	1	___, Are you^there?	3
I don't think ___ is on the call	1	4	1	1	I don't think^___ is on the^call	4
Uh, ___, you're still sharing…	1	5	1	1	Uh, ___,^you're still^sharing…	5

CREATING YOUR BINGO CARD

After you have randomly selected your items to fill in the bingo card, you can begin to actually build the bingo card. We took the approach of using Graph Template Language which is part of ODS Graphics.

DEFINING YOUR TEMPLATE

One of the things to understand about using GTL is that it is a two-step process (Matange, 2013):

1. First, you need to define the structure of the graph using the STATGRAPH template. In the creation of the template, no graph is actually produced.
2. Second, you need to associate the data in order to render the template that will produce the graph.

Based on this, PROC TEMPLATE does not actual produce a graph, or in this case a bingo grid. Rather it allows you to design how you would like your output to look. Once you have designed your output via PROC TEMPLATE, you can then associate any data that has the necessary variables specified in the template with the structure.

SAS Program 3 illustrates the structure defined to produce the bingo card.

SAS Program 3: Defining Your Bingo Card Template

```
proc template;
   define statgraph bingo;
      begingraph / border = false backgroundcolor = bgr;
         layout datalattice columnvar = colnum rowvar = rownum / ❶
                        headerlabeldisplay = NONE    ❷
                        rowaxisopts = (display = NONE)
                        columnaxisopts = (display = NONE);  ❸

               layout prototype;  ❹
                  textplot x = xval y = yval text = b_text / ❺
                        textattrs = (weight = bold size = 8pt)  ❻
                                          splitpolicy = split
                                          splitchar = "^"     ❼
                                          group = grp;  ❽
               endlayout;

         endlayout;
      endgraph;
   end;
run;
```

❶ Within GTL, there are several different layouts to choose from. You would need to decide how you want our graph to look in order to determine the appropriate layout. For the bingo card, the DATALATTICE layout is used because you want to repeat the same graph/output in a grid. Because you are repeating a similar graph you will need to use PROTOTYPE, refer to call-out ❹.
DATALATTICE uses PROTOTYPE to repeat the graph based on the number of crossings of 1 or 2 classification variables specified in the data. (Harris & Watson, 2020)
In this example, the classification variables are specified by COLUNMVAR = COLNUM and ROWVAR = ROWNUM. COLNUM and ROWNUM are the variables within the data and they each have values ranging from one to five. The crossing of COLNUM and ROWNUM make a 5 x 5 grid.

❷ Each layout has a variety options to help you further control the structure and appearance of the output. By default, in a DATALATTICE the values of the classification variables will be displayed in the header (Figure 1). With the bingo card you want to suppress the printing of those classification values, by using HEADERLABELDISPLAY = NONE (Figure 2).

❸ Axis options allow you to control what is displayed along each axis. Refer to Figure 1 to see the default behavior. Notice that both the X and Y axes have '1' for each cell. This represents the axes within each cell. Because you want to create a bingo grid, you would need to use the (DISPLAY = NONE) option on both ROWAXISOPTS and COLUMNAXISOPTS statements in order to suppress the axis values and axis label.

❹ In order to repeat a plot based on the data, you need to use PROTOTYPE layout. Note that PROTOTYPE is dependent on the data so it can only be used with DATAPANEL or DATALATTICE layouts (Harris & Watson, 2020).

❺ Within the TEXTPLOT statement, you specify the point on the graph where the text should be displayed using X = and Y = options. You need to also specify the text by using TEXT = option.

❻ You can control the appearance of the text using TEXTATTRS options. You are able to indicate if the font should be bold with WEIGHT = BOLD. You can also indicate the size of the font. Color of the font can be controlled as well. However, in this particular template, we will use GROUP to control the color. Refer to call-out explanation ❽.

❼ Recall when selecting the records for your bingo card, that some pre-processing was done on the text to add in a split character. The split character can be any value that will not occur naturally in the data. Regardless of what you use as your split character, it needs to be specified using SPLITCHAR = option. In addition, you will need to use SPLITPOLICY = option to indicate what the policy is for avoiding data collisions. The default is NONE. However, when using SPLITCHAR you need to set the policy to either SPLIT or SPLITALWAYS. The difference is that SPLIT will only split if the data does not fit within the allotted space. SPLITALWAYS will always split at the SPLITCHAR regardless if the data could fit.

❽ The GROUP option can use a variable in the data or can be used to reference an attribute map variable. In this instance, it is used to reference an attribute variable named GRP. The attribute map contains the color that is used to for each specific group. If GRP is not specified in the data set, then it will default to black.

Figure 1: DATALATTICE with Default Header Behavior

Figure 2: DATALATTICE with HEADERLABELDISPLAY = NONE and HEADERBORDER = FALSE

DEFINING YOUR ATTRIBUTE MAP

There are several methods to control the color for each group. One such method is using an attribute map data set. With an attribute map data set you can control a variety of features outside of the plot statements. (Watson, 2020)

An attribute map data set contains an ID that will be referenced within the plot statements as well as a VALUE which contains the unique values found within your data. Those are the two required variables. The remaining variables specified are dependent on what type of attributes you are defining. You can specify things such as line patterns, markers, size, and color. Text color is the only attribute needed for the bingo cards, therefore, only TEXTCOLOR is specified. SAS Program 4 illustrates how an attribute map data set can be created. Data Display 3 shows the values contained within the attribute map data set.

SAS Program 4: Defining Color Attribute Map

```
data textclr (drop = i);
   ID = 'TXTCLR';
   array clrs(5) $20 _TEMPORARY_ ('deeppink' 'lightseagreen'
                  'blueviolet' 'darkturquoise' 'mediumvioletred');
   do i = 1 to 5;
      VALUE = cats(i);
      TEXTCOLOR = clrs(i);
      output;
   end;
run;
```

Data Display 3: Color Attribute Map

ID	VALUE	TEXTCOLOR
TXTCLR	1	deeppink
TXTCLR	2	lightseagreen
TXTCLR	3	blueviolet
TXTCLR	4	darkturquoise
TXTCLR	5	mediumvioletred

To learn more on attribute map data set, you can visit Defining a Discrete Attribute Map at SAS® Graph Template Language: Reference, Fifth Edition documentation (SAS Institute Inc., 2020).

ASSOCIATING THE DATA WITH THE TEMPLATE

As mentioned previously, using GTL to produce an output is a two-step process. The structure of the graph is defined using PROC TEMPLATE. In order to associate the data with the template, you need to use the SGRENDER procedure. SAS Program 5 shows the syntax used in PROC SGRENDER.

SAS Program 5: Associating the Data with the Template

```
proc sgrender data = bingo&i ❶
              template = bingo  ❷
              dattrmap = textclr;  ❸
   dattrvar grp = 'TXTCLR';  ❹
run;
```

❶ You need to indicate what the input data set is that is used to produce the output. The input data needs to contain all the necessary variables specified in the template.

❷ TEMPLATE is a required option. This will allow you to associate the desired template with the data.

❸ If you have an attribute map data set that is used to control various features such as color in your graph, then you need to use the DATTRMAP option to specify the attribute map data set.

❹ If you are using DATTRMAP, you will either need to have the DISCRETEATTRVAR statement within the template specified or you will need to use the DATTRVAR statement to indicate what input variable is to be associated with an ID in the attribute map data set.

ALTERNATIVE USES

The beauty of this is that it can be used to create bingo cards for a variety of things. Here are few alternative uses for the program.

- Planning a road trip and need to keep kiddos occupied.
- In charge of putting on a bridal shower or a baby shower and want to create a game for people to play
- Just a fun evening with the family and friends
- Learning a foreign language

All you need to do is create a list of items that are specific to your desired bingo card.

Future explorations include annotating images in the center of the bingo card and expanding some of the techniques used for business purposes.

CONCLUSION

ODS Graphics in combination with SAS random generation techniques have the winning touch. Hopefully these techniques can provide entertainment for your zoom calls for weeks to come, and enhance your daily work through the lessons learned in our project. SAS graphics are not just for statistical analysis.

REFERENCES

Harris, K., & Watson, R. (2020). *SAS® Graphics for Clinical Trials by Example.* Cary, NC: SAS Institute Inc.

Matange, S. (2013). *Getting Started with the Graph Template Language in SAS®: Examples, Tips, and Techniques for Creating Custom Graphs.* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2020, July 30). *Defining a Discrete Attribute Map*. Retrieved from SAS® 9.4 Graph Template Language: Reference, Fifth Edition: https://documentation.sas.com/?docsetId=grstatgraph&docsetTarget=n1oq0axfb1tc1yn10kgw6dyuitk2.htm&docsetVersion=9.4&locale=en

Watson, R. (2020). "What's Your Favorite Color? Controlling the Appearance of a Graph". *Proceedings of the SAS Global Forum 2020 Conference.* Washington, D.C.: SAS Institute Inc.

CONTACT INFORMATION

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

Richann Watson
DataRich Consulting
richann.watson@datarichconsulting.com

Louise Hadden
Abt Associates, Inc.
louise_hadden@abtassoc.com

APPENDIX

```
/*************************************************************************/
/*** BEGIN SECTION TO INITIALIZE ALL MACRO VARIABLES AND DEFINE FORMATS ***/
/*************************************************************************/
%let path = C:\Users\gonza\Desktop\Conferences\Draft Papers\Boredom Buster
Bingo - Conference Call Bingo\Development;
%let hdrimg = bbb_resized.png;
%let ctrimg = phone-resize-h235.png;
%let bingo_file = Conference Call Bingo.xlsx;
%let bingo_sheet = Bingo;
%let font = 'AMT Albany';

%let bingo_card = MyBingoCard;
%let textlen = 15;
%let splitchr = ^;
options validvarname = v7;
/*************************************************************************/
/*** END SECTION TO INITIALIZE ALL MACRO VARIABLES AND DEFINE FORMATS ***/
/*************************************************************************/

/*************************************************************************/
/*** BEGIN SECTION TO READ IN AND RANDOMLY SELECT TEXT FOR BINGO CARDS  ***/
/*************************************************************************/
/* read in the list of bingo text options */
libname bingo xlsx "&path.\&bingo_file";
data bingo (drop = things_you_hear_or_see_on_a_call);
   set bingo."&bingo_sheet"n;
```

```
   /* format the text so it will display accurately */
   bingo_text = strip(things_you_hear_or_see_on_a_call);

   bt_len = length(bingo_text);
run;
libname bingo clear;

/* determine the number of possible splits based on the max length of all
values */
proc sql noprint;
   select ceil(max(length(bingo_text)) / &textlen) + 5 into :numsplit
   from bingo;
quit;
/*************************************************************************/
/*** END SECTION TO READ IN AND RANDOMLY SELECT TEXT FOR BINGO CARDS  ***/
/*************************************************************************/

/*************************************************************************/
/*** BEGIN SECTION TO DEFINE GRAPH TEMPLATE AND GENERATE BINGO CARDS    ***/
/*************************************************************************/
proc template;
   define statgraph bingo;
      begingraph / border = false backgroundcolor = bgr;

         layout datalattice columnvar = colnum rowvar = rownum /
                                        headerborder = false
headerlabeldisplay = NONE
                                        rowaxisopts = (display = NONE)
                                        columnaxisopts = (display = NONE);

            layout prototype;
               textplot x = xval y = yval
                        text = b_text / textattrs = (family = "&font"
                                                     weight = bold
                                                     size = 6pt)
                                        splitpolicy = split
                                        splitchar = "^"
                                        group = grp;
            endlayout;

         endlayout;
      endgraph;
   end;
run;
/*************************************************************************/
/*** END SECTION TO DEFINE GRAPH TEMPLATE AND GENERATE BINGO CARDS    ***/
/*************************************************************************/

/*************************************************************************/
/*** BEGIN SECTION TO CREATE ATTRIBUTE MAP TO MAKE TEXT DIFFERENT COLORS ***/
/*************************************************************************/
data textclr (drop = i);
   ID = 'TXTCLR';
   array clrs(5) $20 _TEMPORARY_ ('deeppink' 'lightseagreen' 'blueviolet'
'darkturquoise' 'mediumvioletred');
   do i = 1 to 5;
      VALUE = cats(i);
      TEXTCOLOR = clrs(i);
```

```
      output;
   end;
run;
/*************************************************************************/
/*** END SECTION TO CREATE ATTRIBUTE MAP TO MAKE TEXT DIFFERENT COLORS ***/
/*************************************************************************/

/*************************************************************************/
/*** BEGIN SECTION TO SELECT TEXT FOR BINGO CARDS AND RENDER BINGO CARDS***/
/*************************************************************************/
%macro bingo(maxcards = 5, multiclr = Y, ext = pdf);
   %do i = 1 %to &maxcards;

      data bingo&i._a;
         set bingo;
         call streaminit('PCG', 0); /* auto-generate seed */
         __run = rand('uniform');
      run;

      proc sort data = bingo&i._a;
         by __run;
      run;

      data bingo&i.;
         set bingo&i._a;
         by __run;
         if _n_ le 25;

         retain rownum 0 colnum;
         if mod(_n_, 5) = 1 then do;
           rownum + 1;
           colnum = 0;
         end;
         colnum + 1;

         /* defaulted to 1 so that they can be used on textplot statement */
         xval = 1;
         yval = 1;

         /*************************************************************/
         /*** BEGIN SECTION TO ADD SPLIT CHARACTERS TO BINGO VALUES ***/
         /*************************************************************/
         /* insert split characters */
         array __split(&numsplit) $ &textlen;
         __var = bingo_text;
         __len = length(__var);
         __i = 0;
         /* if text will not fit then split otherwise get out of loop */
         /* check character (_chr) needs to be initialized to some    */
         /* non-delimiter/split value just to get into loop           */
         do while (__len > &textlen);
            __chk = &textlen + 1;
            __chr = '+';
            /* look for split character within the first _chk characters */
            /* if it exists then reset _chk to location + 1              */
            if find(substr(__var, 1, __chk), "&splitchr") then
                    __chk = find(substr(__var, 1, __chk), "&splitchr") + 1;

            /* need to look for a delimiter or natural split */
```

```
      /* if there is a natural split want to keep it   */
      /* get out of the loop once a delimiter or split */
      /* character is found or until there are no more */
      /* characters to check                           */
      do until (__chr in (' ' '/' '-' "&splitchr") or __chk = 0);
         __chk = __chk - 1;
         if __chk ne 0 then __chr = substr(__var, __chk, 1);
      end;

      __i + 1;
      /* if exit because of split or delimiter then create a split */
      if __chk > 0 then do;
         if substr(__var, __chk, 1) = "&splitchr" then
                __split(__i) = substr(__var, 1, __chk - 1);
         else __split(__i) = substr(__var, 1, __chk);
         __var = substr(__var, __chk + 1);
      end;
      /* no logical delimiter or split character found within */
      /* specified length so split at specified length        */
      else do;
         __split(__i) = substr(__var, 1, &textlen);
         __var = substr(__var, &textlen + 1);
      end;

      /* __var has been reset so need to reset the __len as well */
      __len = length(__var);
   end;
   /* once out of the loop need to assign the remainder */
   /* of the text to the next array variable            */
   __split(__i + 1) = __var;

   /* combine all the vars into one separated by the split character */
   length b_text $1000;
   b_text = catx("&splitchr", of __split:);
   /*************************************************************/
   /*** END SECTION TO ADD SPLIT CHARACTERS TO BINGO VALUES ***/
   /*************************************************************/
run;

/* have text be different colors - 5 different colors */
%if &multiclr = Y %then %do;
   data bingo&i.;
      set bingo&i.;
      if _n_ in (1 7 13 19 25) then grp = 1;
      else if _n_ in (2 8 14 20 21) then grp = 2;
      else if _n_ in (3 9 15 16 22) then grp = 3;
      else if _n_ in (4 10 11 17 23) then grp = 4;
      else grp = 5;
   run;
%end;

/* render each bingo card */
options nodate nonumber;
ods graphics on / width = 6.5in height = 7in;
ods &ext file = "&path.\card\&bingo_card._&i._m5.&ext"
        %if &ext = pdf %then dpi ; %else image_dpi; = 300;

ods escapechar = "^";
title "^S = {preimage=""&path.\images\&hdrimg"" }";
```

```
      proc sgrender data = bingo&i. template = bingo
                dattrmap = textclr;
         dattrvar grp = 'TXTCLR';
      run;
      ods &ext close;
   %end;

%mend bingo;

%bingo(maxcards = 4);
```