Paper 042-2022

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

Richann Watson, DataRich Consulting; Louise Hadden, Independent Consultant

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 Output Delivery System (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 unique outputs such as a bingo card.

The program was successfully tested on SAS version 9.4, maintenance releases 5, 6 and 7, at the date of publication. Note that if you are running SAS version 9.4 m7 in SAS Enterprise Guide, then you will need to make sure you turn off the HTML destination. To do this within SAS Enterprise Guide, navigate to Tools→Options→Results→General and uncheck HTML in the Result Formats section.

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 remarkably 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 has been reduced to the desired 25 records. This process 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 that take 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 uncomplicated design, SRS (Simple Random

Sampling) design was used, and 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; 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 be used when defining the template.
- GRP: (*Optional*) Used to indicate color for that particular value. GRP is only used if you request multiple colors.

Data Display 2: Sample of Texts 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, Getting Started with the Graph Template Language in SAS®: Examples, Tips, and Techniques for Creating Custom Graphs, 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, TEMPLATE Procedure 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 = (family = "&font"
                                     weight = bold size = 6pt)    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 your 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, SAS® Graphics for Clinical Trials by Example, 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, SAS® Graphics for Clinical Trials by Example, 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 the font family and 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 that when selecting the records for your bingo card, pre-processing was done on the text to add 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 of whether or not 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 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

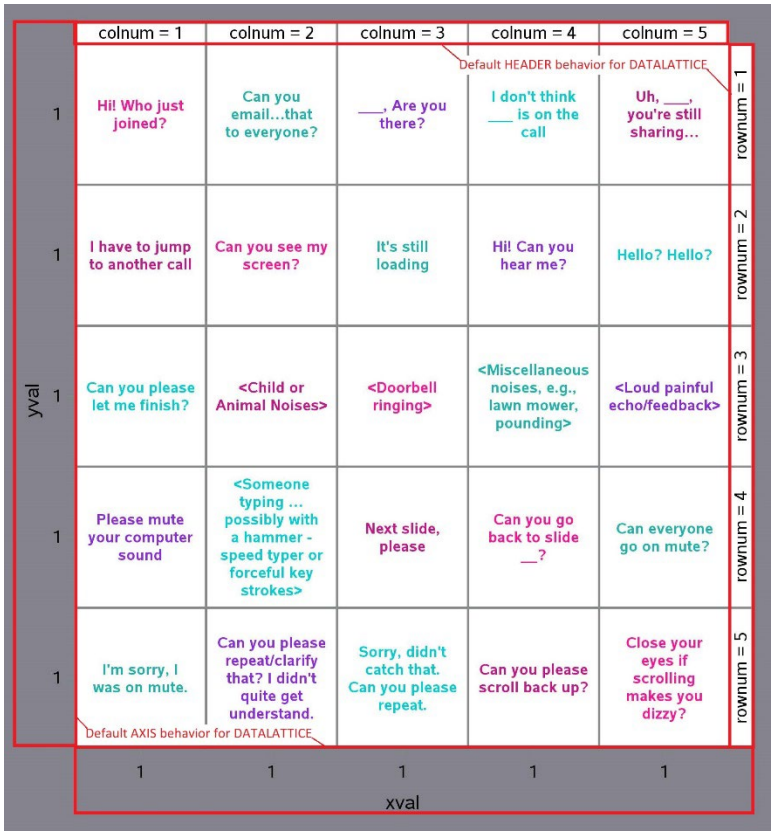
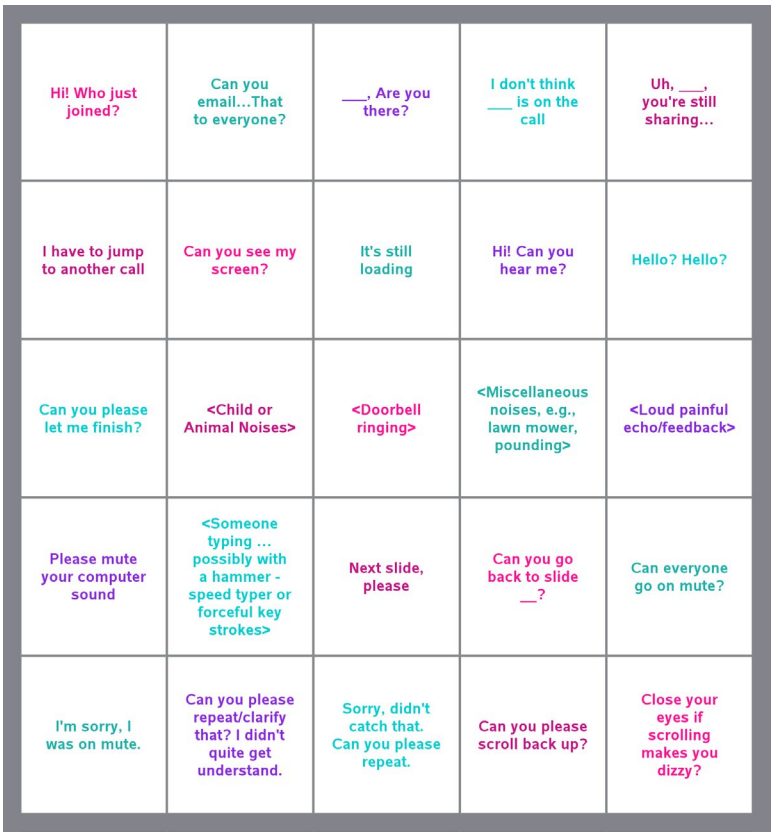


Figure 2: DATALATTICE with HEADERLABELDISPLAY = NONE and HEADERBORDER = FALSE



DEFINING YOUR ATTRIBUTE MAP

There are several methods with which 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 BINGO.BINGOATTR (drop = i);
   ID = 'TXTCLRB';
   array bclrs(5) $20 _TEMPORARY_ ('deeppink' 'lightseagreen'
                  'blueviolet' 'darkturquoise' 'mediumvioletred');
   do i = 1 to 5;
      VALUE = cats(i);
      TEXTCOLOR = bclrs(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 = BINGO.BINGOATTR;  ❸
   dattrvar grp = 'TXTCLRB';  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 PRODUCTIONS OF BINGO CARDS

Now that we have demonstrated the basic concept of creating a bingo card, we can expand on this to create a different variety of bingo cards. In this section, we will illustrate how to create a checkboard bingo card, how to create bingo cards with images and how to use the REPORT procedure to generate a basic bingo card. The method for selecting the items for these alternatives would follow the same approach as the basic one.

CREATING BINGO CARDS WITH CHECKERBOARD

Processing of the items for selection for the checkerboard bingo card is the same as the basic bingo card. There are a few differences with the checkerboard that need to be incorporated. We need to modify the template slightly as well as create an attribute map data set.

Defining Your Template

The template created in SAS Program 3 can be utilized with a few modifications. SAS Program 6 shows the additions that are needed to create the checkerboard effect. Because the code is almost identical to SAS Program 3, we only focus on explaining the new components.

SAS Program 6: Defining Your Checkerboard Bingo Card Template

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

               layout prototype;
                  blockplot x = xval block = grp / display = (fill); ❷
                  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;
```

❶ Recall in the basic bingo card that some of the text was too long and would get truncated. With the SHRINKFONTS option set to TRUE, this will scale down all the fonts in the current layout as well as any 'children' layouts. Therefore, it will scale down the fonts in the LAYOUT PROTOTYPE as well. By default, SHRINKFONTS is set to FALSE. Notice in Figure 3 that the text in last block on the first row is cut off at the top and the bottom. However, in Figure 4 the same text is found on the fourth block on the fourth row, and it has been shrunk to actual fit in the block.

❷ With the BLOCKPLOT statement blocks are created. The blocks typically contain text, however, on the DISPLAY option, we indicted we only wanted to display the FILL (i.e., the background color). To display the text, we would need to include VALUE in the DISPLAY option, and this would display the value of GRP (i.e., 1 or 2), as demonstrated in Figure 5.

Figure 3: Bingo Card with SHRINKFONTS = FALSE

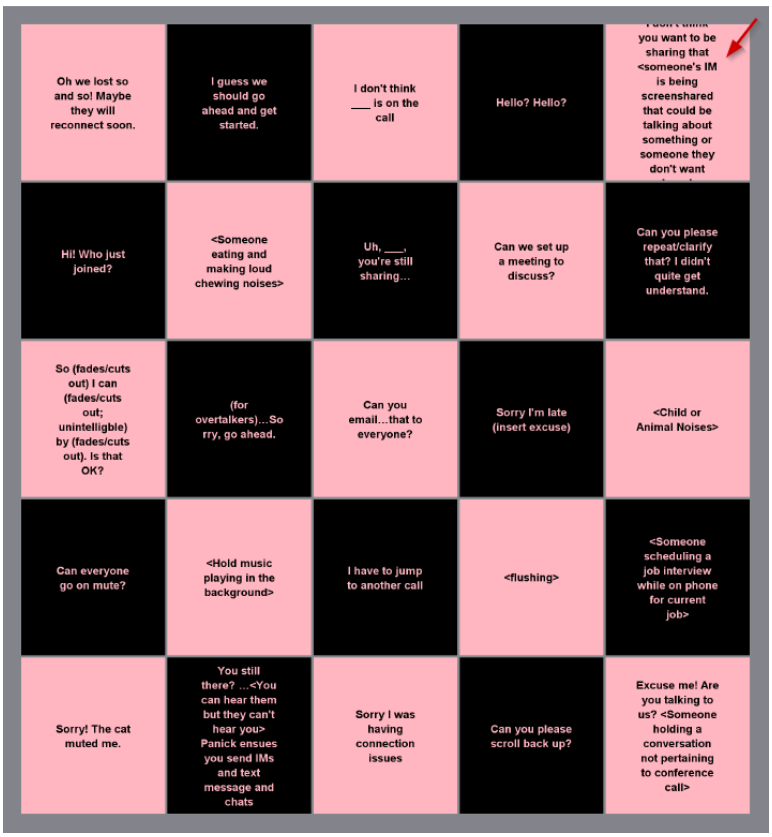


Figure 4: Bingo Card with SHRINKFONTS = TRUE

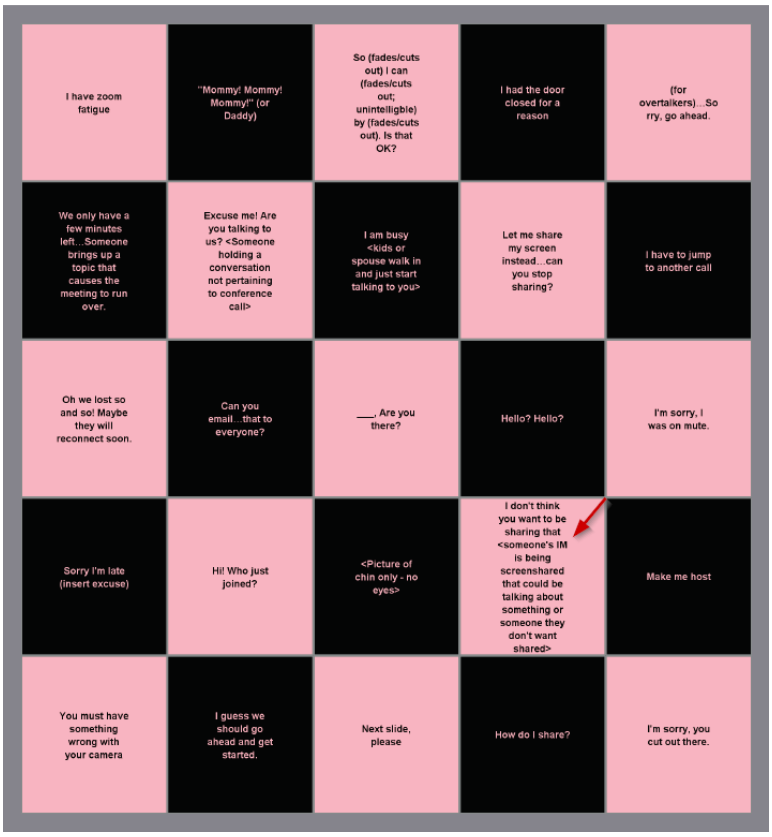
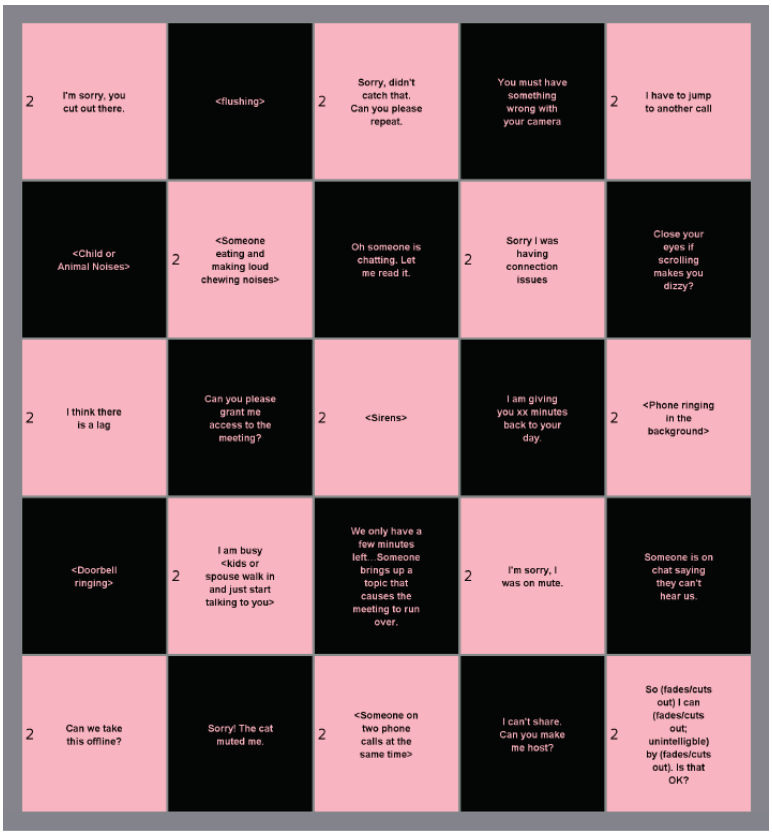


Figure 5: Bingo Card with VALUES Displayed



Defining Your Attribute Map

In the basic bingo, we allowed for all black text or multiple colors and with the multiple colors we used an attribute map data set to manage the multiple colors for the text. For the checkerboard, we will also use the attribute map data set to control the colors for the text as well as the background. In SAS Program 4 we specified the five colors on the ARRAY BCLRS statement, however in SAS Program 6 we specify the two colors that will be used for the checkerboard. In this case we are using lightpink and black. In addition, we utilize the FILLCOLOR attribute which is used to create the checkerboard. Notice that in this attribute data set, we are reversing the TEXTCOLOR and FILLCOLOR based on the two colors. This allows for lightpink text with a black background and black text with a lightpink background (refer to Data Display 4).

SAS Program 7: Attribute Data Set for Checkerboard Bingo

```
data BINGO.BINGOATTR (drop = i);

   << CODE FROM SAS PROGRAM 4 >>

   ID = 'TXTCLRC';
   array cclrs(2) $20 _TEMPORARY_ ('lightpink' 'black');
   do i = 1 to 2;
      VALUE = cats(i);
      TEXTCOLOR = cclrs(i);
      if i = 1 then FILLCOLOR = cclrs(i+1);
      else if i = 2 then FILLCOLOR = cclrs(1);
      output;
   end;
run;
```

Data Display 4: Color Attribute Map for Checkerboard

ID	VALUE	TEXTCOLOR	FILLCOLOR
TXTCLR	1	lightpink	black
TXTCLR	2	black	lightpink

Associating the Data with the Template

The last step is to generate the data for all the bingo cards and associate the date with the template. This is similar to what was done for the basic. There is one small piece of code that needs to be updated. In the basic bingo code that allowed for multiple colors, we allowed for five colors (i.e., 5 groups). For the checkerboard, we need only two colors (i.e., 2 groups). In addition, on PROC SGRENDER you need to make sure you are referencing the new template (SAS Program 8).

SAS Program 8: Associating the Data with the Checkerboard Template

```
proc sgrender data = bingo&i. template = bingo_ck
              dattrmap = BINGO.BINGOATTR;
   dattrvar grp = 'TXTCLRC';
run;
```

CREATING BINGO CARDS WITH IMAGES

Up to this point we showed how to generate a basic bingo card with black text, a basic bingo card with multiple colored text and a checkerboard bingo card. All of these use text in the bingo block. But what if we want to use images?

Preparing Images and Data

In order to use images, we need to prepare all our images in advance. When preparing the images, you need to make sure they have a high enough resolution. For this illustration all images are 900x900px with a resolution of 900. This has a print size of 1 inch by 1 inch. Instead of using BINGO_TEXT as we did with the other bingo cards, we are going to have a library of images with the names of the files stored in a file that can be converted to a SAS data set (refer to Data Display 5).

Data Display 5: Sample of Images for Bingo Card

English	Spanish
baby	bebé
beach	playa
bicycle	bicicleta
books	libros
boy	el niño
bread	pan
church	iglesia
dog	perro
eggs	huevos
expensive	caro

Selection of the images for the bingo card is similar to the other types of bingo cards. However, you need to specify the full path of the image location so that the program knows where to pull the file from (refer to SAS Program 9 and Data Display 6).

SAS Program 9: Reading in the Data and Specifying Full File Name with Location

```
libname bingo xlsx "&path.\&bingo_file";
data bingo;
   set bingo."&bingo_sheet"n;
   where status = 'done';

   /* create the full file name including path and extension */
   bingo_item = cats("&path.\Images\Final\", english, ".png");
run;
libname bingo clear;
```

Data Display 6: Sample of SAS Data Set with Full File Name.

English	Spanish	bingo_item
baby	bebé	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\baby.png
beach	playa	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\beach.png
bicycle	bicicleta	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\bicycle.png
books	libros	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\books.png
boy	el niño	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\boy.png
bread	pan	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\bread.png
church	iglesia	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\church.png
dog	perro	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\dog.png
eggs	huevos	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\eggs.png
expensive	caro	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\expensive.png

Creating an Annotate Data Set

Once we have all the images in our catalog, we can now create an annotate data set that will utilize the images that are randomly selected for the bingo card. Note that the random selection of the images is the same as if you are selecting text.

SAS Program 10: Creating an Annotate Data Set

```
data anno (keep = id drawspace function image layer height: image:);
    set bingo&i.;
    id = catx('_', 'IMG', rownum, colnum); ❶
    drawspace = 'datavalue';   ❷
    function = 'image';  ❸
    image = bingo_item;
    layer = 'front';
    height = 96;            ❹
    heightunit = "PIXEL";
    imagescale = 'FIT';
run;
```

❶ ID allows you to have multiple annotates within a data set. The ID allows you to reference specific annotations (Heath, 2016).

❷ Need to specify the drawing space and drawing units. There are different drawing spaces: DATA, WALL, LAYOUT and GRAPH. The units can be PIXEL or PERCENT for all four. However, for DATA space an additional unit of VALUE can be used. The drawing space and drawing units are specified together with the DRAWSPACE statement. Refer to SAS documentation for further details on DRAWSPACE.

❸ When creating an annotate, you need to specify a FUNCTION. There are a number of functions that can be used, such as TEXT, LINE, IMAGE. For our purposes we want to use IMAGE, since we want to draw an image on the graph.

❹ Depending on which FUNCTION is used, will determine which additional statements are needed. For FUNCTION = 'IMAGE', some of the additional statements used are (SAS Institute Inc., 2022):

- IMAGE: specifies the location and name of the image (i.e., the full path)
- LAYER: specifies whether it is in front or back of what is on the graph in that space
- HEIGHT: specifies the height of the annotation
- HEIGHTUNIT: specifies the unit associated with the height
- IMAGESCALE: indicates how the image should be scaled within the height and width

As seen in Data Display 7 that a separate ID is created for each IMAGE. These individual IDs will be tied to a specific row and column within the bingo card.

Data Display 7: Sample Annotate Data Set

ID	IMAGE
IMG_1_1	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\books.png
IMG_1_2	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\kitchen.png
IMG_1_3	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\house.png
IMG_1_4	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\church.png
IMG_1_5	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\strawberries.png
IMG_2_1	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\beach.png
IMG_2_2	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\France.png
IMG_2_3	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\flower.png
IMG_2_4	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\ice.png
IMG_2_5	C:\Users\gonza\Desktop\GitHub\SASsy_Bingo\Images\Final\giraffe.png

Not shown, the data set would have the following variables with the same value for all records:

- DRAWSPACE = 'DATAVALUE'
- FUNCTION = 'IMAGE'
- LAYER = 'FRONT'
- HEIGHT = 96
- HEIGHTUNIT = 'PIXEL'
- IMAGESCALE = 'FIT'

Defining Your Template

For bingo cards with images, this is handled a bit differently. Although one template is defined, it is done within a macro to allow for the dynamic creation of all the plot statements. Unlike the bingo cards with text that used only one TEXTPLOT (and BLOCKPLOT for checkerboard), we need to have a TEXTPLOT statement for all 25 cell blocks. This is done so that a unique annotate ID can be associated with each individual block. Recall that for each image that was selected it was assigned a unique annotate ID.

The template is similar to the original template used to generate the basic bingo card. However, there are some key differences. Instead of using DATALATTICE layout, we are now using a LATTICE layout. With DATALATTICE we used a PROTOTYPE layout which repeated the same plot statements in each cell. Essentially, DATALATTICE with PROTOTYPE created the 25 TEXTPLOT statements.

With the images, we need to manually create these 25 TEXTPLOT statements so that we could associate each individual image to a cell. The images are associated with a cell based on the unique ID assigned in the annotate data set.

In SAS Program 11, we utilize the macro facility to loop through each row and column and generate a TEXTPLOT statement with the associated ANNOTATE.

SAS Program 11: Defining Template for Bingo Cards with Images

```
%macro crtcard;
   proc template;
      define statgraph bingo_img;
         begingraph / border = false backgroundcolor = CXC00000;
            layout lattice / rows = 5 rowweights = uniform
                             rowgutter = 0 rowdatarange = union
                             columns = 5 columnweights = uniform
                             columngutter = 0 columndatarange = union;

               %do row = 1 %to 5;
                  %do col = 1 %to 5;
                     layout overlay / xaxisopts = (display = NONE griddisplay = OFF)
                                      yaxisopts = (display = NONE griddisplay = OFF);
                        textplot x = xval y = yval text = english;

                        annotate / id = "IMG_&row._&col";  ❶
                     endlayout;
                  %end;
               %end;
            endlayout;
         endgraph;
      end;
   run;
%mend crtcard;

%crtcard
```

❶ Based on the information stored in the annotation data set, it draws the annotation associated with the ID. Note that ID is an optional argument. However, if you have more than one set of instructions in the annotation data set, then in order to specify which set of instructions to use the ID option needs to be specified.

Although the text from the TEXTPLOT statement is not shown on the bingo card, it is still being generated. It is just not seen because the image drawn by the annotation is placing the image on top of the existing graph; therefore, it is not necessary to specify any text attributes.

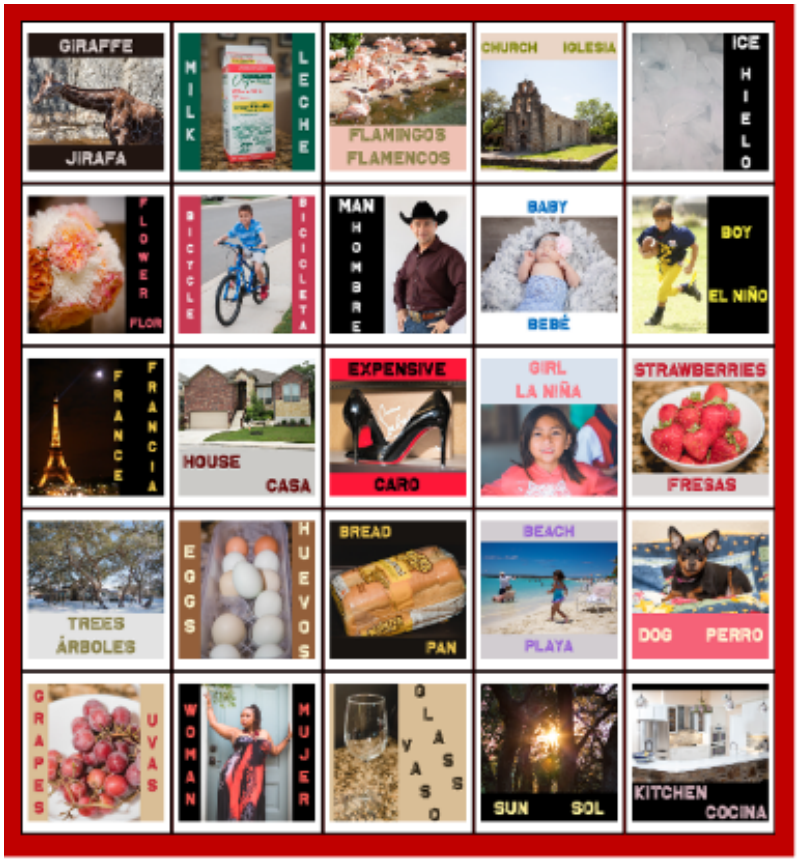
Associating the Data with the Template

As with the other two types of bingo cards, we need to associate the data with the revised template. In addition, we need to associate the appropriate annotate data set that contains all the necessary instructions to draw the images. This is done using SGANNO as shown in SAS Program 12. Figure 6 shows a sample bingo card generated.

SAS Program 12: Associating the Data with the Template and Annotate Data Set

```
proc sgrender data = bingo&i. template = bingo_img
              sganno = anno;
run;
```

Figure 6: Bingo Card with Images



USING PROC REPORT TO CREATE BINGO CARDS

SAS has myriad methods for producing output with style, including the GTL with templates and attribute maps, and the ODS. Using the same bingo card generation system with random selection, we can replicate the bingo cards created with GTL replacing GTL with ODS and PROC REPORT. Bingo card generation with PROC REPORT involves postprocessing the “raw” bingo card data for each card, the use of the ODS PDF destination, ODS text statements to annotate the bingo card logo, ODS style statements in PROC REPORT CALL DEFINE statements, and the use of COMPUTE blocks to create groups with different background and foreground colors.

The first step is to transpose bingo card data into a five column and five row structure. Because we have multiple variables to transpose, a DATA step and merge is used instead of PROC TRANSPOSE (refer to SAS Program 13). The transposed data can be seen in Data Display 8.

SAS Program 13: Transposing Data Using a DATA step and MERGE

```
data col1 (keep=rownum grp1 col1)
     col2 (keep=rownum grp2 col2)
     col3 (keep=rownum grp3 col3)
     col4 (keep=rownum grp4 col4)
     col5 (keep=rownum grp5 col5);
    set bingo&i.;
    if colnum=1 then do;
        grp1=grp;
        col1=bingo_text;
        output col1;
    end;
    if colnum=2 then do;
        grp2=grp;
        col2=bingo_text;
        output col2;
    end;
    if colnum=3 then do;
        grp3=grp;
        col3=bingo_text;
        output col3;
    end;
    if colnum=4 then do;
        grp4=grp;
        col4=bingo_text;
        output col4;
    end;
    if colnum=5 then do;
        grp5=grp;
        col5=bingo_text;
        output col5;
    end;
run;

data bingo2_&i.;
    merge col1-col5;
    by rownum;
run;
```

Data Display 8: Transpose Bingo Data

ROWNUM	GRP1	COL1	GRP2	COL2	GRP3	COL3
1	2	I'll have to get back to you	1	I have a hard stop at X	2	You must have something wrong with your camera
2	1	Hi! Who just joined?	2	Can you see my screen?	1	<Doorbell ringing>
3	2	I had the door closed for a reason	1	I love you! I love you! I love you! <or some other form of affection> <child or pet comes in begging for attention so you give it to them>	2	I'm sorry, I was on mute.
4	1	<Phone ringing in the background>	2	<Picture of chin only - no eyes>	1	Sorry I'm late (insert excuse)
5	2	<Child or Animal Noises>	1	Can you please grant me access to the meeting?	2	Can everyone go on mute?

ROWNUM	GRP4	COL4	GRP5	COL5
1	1	I am pouring myself a glass of water. <want people to know you are not going to the bathroom>	2	I think there is a lag
2	2	Make me host	1	I think we lost them
3	1	Can you please repeat/clarify that? I didn't quite get understand.	2	I am in the lobby?
4	2	Can we set up a meeting to discuss?	1	Can you email that to everyone?
5	1	<Person sharing screen keeps flickering>	2	Someone is on chat saying they can't hear us.

Next, as shown in SAS Program 14 we set up the output destination and add the bingo card graphic at the top using an ODS TEXT statement and PREIMAGE. Note that we add some spacing between the bingo card graphic and grid using a NEWLINE style statement.

SAS Program 14: Setting Up the Output Destination

```
options nonumber nodate;

title1 ;
run;
ods escapechar='^';

ods listing close;

options topmargin=.5in leftmargin=.8in rightmargin=.8in papersize=letter ;

ods pdf file="&path.\cards\PROCREPORTBingoMulticolor_&i..pdf" style=styles.pearl
    dpi=600 notoc author='Louise Hadden' bookmarklist=hide compress=9;

title1 '^{newline 2}';

ods text='^S={just=c preimage="&path.\images\bbb_resized.png"}';

ods text='^{newline 5}';
```

Thirdly, we set up our PROC REPORT statements (refer to SAS Program 15). Note the use of non-printing GRP (group) variables. These variables were created in the bingo card generation process and are listed in the COLUMN statement but are defined as non-printing in the DEFINE statement. This allows us to pass style information into the COL variables via color definitions set up in non-printing GRP variables by using COMPUTE and CALL DEFINE statements, allowing one variable to control the style of another variable. The ODS PDF destination is closed, and the bingo card is generated. Figure 7 shows a sample of the multi-color bingo card created using PROC REPORT.

SAS Program 15: PROC REPORT to Create Multi-Color Bingo Card

```
proc report nowd data=bingo2_&i noheader
    style(report)=[cellpadding=5pt vjust=b]
    style(header)=[just=center font_face="Helvetica" font_weight=bold font_size=8pt]
    style(lines)=[just=left font_face="Helvetica"] split='|';

    columns rownum grp1 grp2 grp3 grp4 grp5 col1 col2 col3 col4 col5 ;
    define rownum / display ' ' noprint;
    define grp1 / display ' ' noprint;
    define grp2 / display ' ' noprint;
    define grp3 / display ' ' noprint;
    define grp4 / display ' ' noprint;
    define grp5 / display ' ' noprint;
    define col1 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt};
    define col2 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col3 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col4 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col5 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };

    compute col1;
          if grp1=1 then color='purple';
          else if grp1=2 then color='green';
          else if grp1=3 then color='blue';
          else if grp1=4 then color='red';
          else if grp1=5 then color='vipk';
          else color='black';
          call define ('col1','style','style=[foreground='||color||']');
    endcomp;

    <<< REPEAT CODE FOR COL2 - COL5 >>>

run;
```

Figure 7: Multi-Color Bingo Card Generated with PROC REPORT

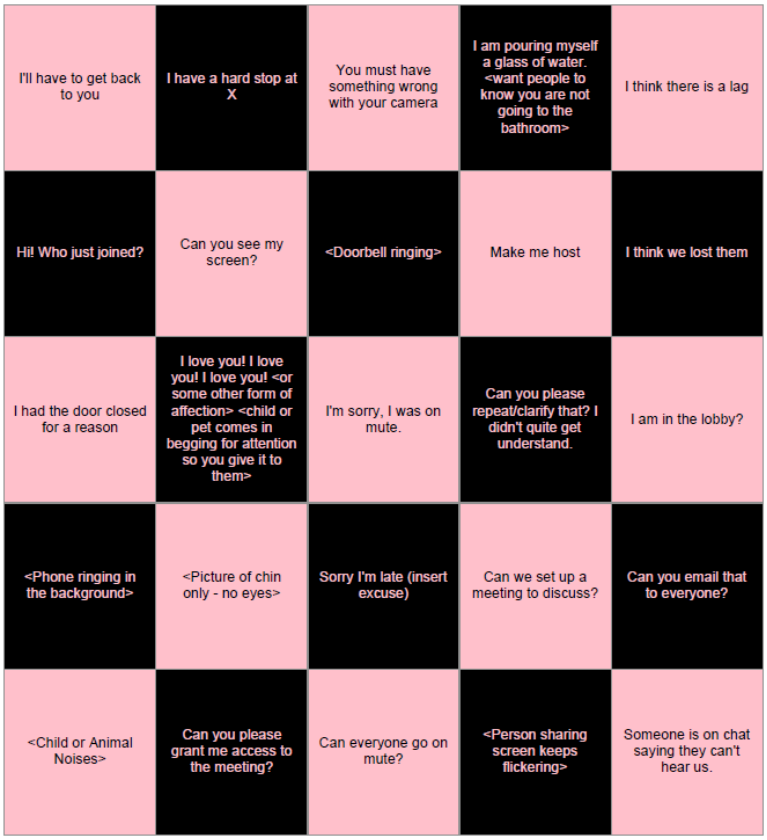
<Doorbell ringing>	Can you please announce your name when speaking? We don't recognize everyone's voices.	GET OFF THE KEYBOARD! <Cat accidentally unmutes and is playing on the keyboard>	You still there? <You can hear them but they can't hear you> Panick ensues you send IMs and text message and chats	<Phone ringing in the background>
Please mute your computer sound	So (fades/cuts out) I can (fades/cuts out; unintelligble) by (fades/cuts out). Is that OK?	I'll have to get back to you	Hi! Who just joined?	I love you! I love you! I love you! <or some other form of affection> <child or pet comes in begging for attention so you give it to them>
I had the door closed for a reason	Think of the door closed as me not being home	I am busy <kids or spouse walk in and just start talking to you>	I think there is a lag	Can you see my screen?
Can you please scroll back up?	<Loud painful echo/feedback>	I'm sorry, I was on mute.	Can you please repeat/clarify that? I didn't quite get understand.	uh huh (feigning interest)
<Hold music playing in the background>	<Heavy breather>	I can't share. Can you make me host?	Make me host	I have a hard stop at X

Just as a checkerboard format bingo card can be generated in GTL, a similar bingo card version can be generated with PROC REPORT, with small variations in the PROC REPORT code (refer to SAS Program 16). Figure 8 shows a sample of the checkerboard bingo card created using PROC REPORT.

SAS Program 16: PROC REPORT to Create Checkerboard Bingo Card

```
data col1 (keep=rownum grp1 col1)
     col2 (keep=rownum grp2 col2)
     col3 (keep=rownum grp3 col3)
     col4 (keep=rownum grp4 col4)
     col5 (keep=rownum grp5 col5);
    set bingo1;
    if colnum=1 then do;
        grp1=checker;
        col1=bingo_text;
        output col1;
    end;
    <<< REPEAT CODE FOR COLNUM = 2 - 5 >>>
run;

data bingo2;
    merge col1-col5;
    by rownum;
run;

options nonumber nodate;
title1 ;
ods escapechar='^';
ods listing close;

options topmargin=.5in leftmargin=.8in rightmargin=.8in papersize=letter ;

ods pdf file='PROCREPORTbingoCheckerboard.pdf' style=styles.pearl dpi=600 notoc
    author='Louise Hadden' bookmarklist=hide compress=9;

title1 '^{newline 2}';

ods text='^S={just=c preimage=".\bbb_resized.png"}';
ods text='^{newline 5}';

proc report nowd data=bingo2 noheader
    style(report)=[cellpadding=5pt vjust=b]
    style(header)=[just=center font_face="Helvetica" font_weight=bold font_size=8pt]
    style(lines)=[just=left font_face="Helvetica"] split='|';

    columns rownum grp1 grp2 grp3 grp4 grp5 col1 col2 col3 col4 col5 ;
    define rownum / display ' ' noprint;
    define grp1 / display ' ' noprint;
    define grp2 / display ' ' noprint;
    define grp3 / display ' ' noprint;
    define grp4 / display ' ' noprint;
    define grp5 / display ' ' noprint;
    define col1 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt};
    define col2 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col3 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col4 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col5 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };

    compute col1;
      if grp1=1 then do; color='pink'; rcolor='black'; end;
      else do;  color='black'; rcolor='pink'; end;
      call define ('col1','style','style=[foreground='||color||' background='||rcolor||']');
    endcomp;

    <<< REPEAT CODE FOR COL2 - COL5 >>>
run;

ods pdf close;
```

Figure 8: Checkerboard Bingo Card Generated with PROC REPORT



Similarly, using similar PROC REPORT code in conjunction with PROC FORMAT with image names as value labels, bingo cards with images can be created.

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 as shown in Creating Bingo Cards with Images

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

Future explorations include expanding the techniques demonstrated for business purposes, and employing some additional techniques such as SAS/GRAPH to create bingo cards.

The SAS programs along with the data files, can be found at https://github.com/rwatson724/SASsy_Bingo.

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 AND RECOMMENDED READINGS

Eslinger, J. (2016). *The SAS® Programmer's PROC REPORT Handbook Basic to Advanced Reporting Techniques.* Cary: SAS Institute Inc.

Eslinger, J. (2018). *The SAS® Programmer's PROC REPORT Handbook ODS Companion.* Cary: SAS Institute Inc.

Harris, K., & Watson, R. (2018). "Great Time to Learn GTL". *PharmaSUG*. Seattle: PharmaSUG. Retrieved from https://www.pharmasug.org/proceedings/2018/EP/PharmaSUG-2018-EP18.pdf

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

Heath, D. (2016). "Annotating the ODS Graphics Way!". *Proceedings of the SAS Global Forum 2016 Conference.* Las Vegas: SAS Institute Inc. Retrieved from https://support.sas.com/resources/papers/proceedings16/SAS5300-2016.pdf

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.

Matange, S. (2016). *Clinical Graphs Using SAS®.* Cary: SAS Institute Inc.

SAS Institute Inc. (2016). *SAS® 9.4 Graph Template Language: User's Guide.* Cary, NC: SAS Institute, Inc. Retrieved from https://documentation.sas.com/api/docsets/grstatug/9.4/content/grstatug.pdf?locale=en

SAS Institute Inc. (2020, Jul 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

SAS Institute Inc. (2022, Jun 08). *ANNOTATE Statement*. Retrieved 2022, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatgraph/p0fnr3r0fc0jfjn1rtp6di3ygpom.htm

SAS Institute Inc. (2022, May 26). *Creating an SG Annotation Data Set*. Retrieved 2022, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatug/p1qlqpavhh0e1pn10oon6blo7dt6.htm

SAS Institute Inc. (2022, Jun 8). *IMAGE Function*. Retrieved 2022, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatproc/p1cdsjhbc8dk20n1jp3xsozin5pe.htm

SAS Institute Inc. (2022, Jun 08). *SAS® 9.4 Graph Template Language: Reference, Fifth Edition*. Retrieved 2022, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatgraph/titlepage.htm

SAS Institute Inc. (2022, May 26). *Selecting the Drawing Space and Units*. Retrieved 2022, from SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatug/n16ot4xwouv7hon1em203otkde10.htm

Watson, R. (2019). "Great Time to Learn GTL: A Step-by-Step Approach to Creating the Impossible". *SAS Global Forum.* Dallas: SAS. Retrieved from https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3170-2019.pdf

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. Retrieved from https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4660-2020.pdf

ACKNOWLEDGMENTS

Thank you to Lisa Mendez for helping us collect the variety of photos needed to create the images for Mexican Bingo.

All photos for Mexican Bingo were taken by Richann Watson, Louise Hadden and Lisa Mendez.

CONTACT INFORMATION

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

Richann Watson
DataRich Consulting
richann.watson@datarichconsulting.com

Louise Hadden
Independent Consultant
saslouisehadden@gmail.com

APPENDIX A: BINGO CARD

The code to generate the bingo card is broken into three parts. The three parts run in sequence will produce bingo cards.

APPENDIX A.1: MACRO VARIABLES INITIALIZATION AND INPUT OF DATA

The template and the macro rely on global macro variables that are initialized at the beginning of the program. In addition, the file where the texts (or image names) are stored, need to be converted to a SAS data set, so that it can be processed in the macro that will randomly select the items for the bingo card.

```
/*************************************************************************/
/*** BEGIN SECTION TO INITIALIZE ALL MACRO VARIABLES AND DEFINE FORMATS ***/
/*************************************************************************/
%let path = C:\Users\gonza\Desktop\GitHub\SASsy_Bingo;
%let hdrimg = bbb_resized_v2.png;
%let bingo_file = Conference Call Bingo.xlsx;
%let bingo_sheet = Bingo;
%let font = 'Helvetica';

%let bingo_card = MyBingoCard;  /*** prefix used to name the bingo card ***/
%let textlen = 15;
%let splitchr = ^;

options validvarname = v7;
ods path(prepend) work.TEMPLAT (update);
/*************************************************************************/
/*** END SECTION TO INITIALIZE ALL MACRO VARIABLES AND DEFINE FORMATS ***/
/*************************************************************************/

/*************************************************************************/
/*** BEGIN SECTION TO READ IN DATA USED FOR RANDOM SELECTION FOR CARDS  ***/
/*************************************************************************/
/* read in the list of bingo text options */
libname bingo xlsx "&path.\Programs\&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 DATA USED FOR RANDOM SELECTION FOR CARDS  ***/
/*************************************************************************/
```

APPENDIX A.2: TEMPLATE AND ATTRIBUTE MAP FOR BINGO CARD USING TEXT

The template to produce the bingo card is generated using PROC TEMPLATE. In addition, the attribute map data set is created. The attribute map is only needed if you want to create bingo cards with multiple colors. The attribute map is where the multiple colors used for the text are defined.

```
/*************************************************************************/
/*** BEGIN SECTION TO DEFINE GRAPH TEMPLATE FOR BINGO CARDS WITH TEXT ***/
/*************************************************************************/
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 FOR BINGO CARDS WITH TEXT ***/
/*************************************************************************/

/*************************************************************************/
/*** BEGIN SECTION TO CREATE ATTRIBUTE MAP TO MAKE TEXT DIFFERENT COLORS ***/
/*************************************************************************/
data BINOG.BINGOATTR (drop = i);
   ID = 'TXTCLRB';
   array bclrs(5) $20 _TEMPORARY_ ('deeppink' 'lightseagreen' 'blueviolet'
                                   'darkturquoise' 'mediumvioletred');
   do i = 1 to 5;
      VALUE = cats(i);
      TEXTCOLOR = bclrs(i);
      output;
   end;

   /** code from Appendix B-2 is included **/

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

APPENDIX A.3: MACRO TO RANDOMLY SELECT ITEMS AND PRODUCE BINGO CARD

The macro has three macro parameters with default values as indicated in Appendix Table 1.

Appendix Table 1: Macro Parameters Descriptions

Macro Parameter	Description	Default
MAXCARDS	Number of bingo cards to generate	5
MULTICLR	Indication if you want to use multiple colors or if you want to use all black text. If parameter is set to Y then multiple colors will be used. Otherwise, all black text is used.	Y
EXT	Extension for the file. Possible values are PDF or RTF.	PDF

```
/*************************************************************************/
/*** 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.\cards\&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 = BINGO.BINGOATTR;
         dattrvar grp = 'TXTCLRB';
      run;
      ods &ext close;
   %end;

%mend bingo;

%bingo(maxcards = 4)
```

APPENDIX B: CHECKERBOARD BINGO CARD

Similar to the basic bingo card, the program can be broken into three parts.

APPENDIX B.1: MACRO VARIABLES INITIALIZATION AND INPUT OF DATA

The code found in Appendix A.1 can be used for the checkerboard.

APPENDIX B.2: TEMPLATE AND ATTRIBUTE MAP FOR CHECKERBOARD BINGO CARD

Since the main difference is only the template and attribute map, only that is provided here.

```
/*************************************************************************/
/*** BEGIN SECTION TO DEFINE GRAPH TEMPLATE FOR CHECKERBOARD BINGO CARDS ***/
/*************************************************************************/
proc template;
   define statgraph bingo_ck;
      begingraph / border = false backgroundcolor = bgr;

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

              layout prototype;
                 blockplot x = xval block = grp / display = (fill);
                 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 FOR CHECKERBOARD BINGO CARDS ***/
/*************************************************************************/

/*************************************************************************/
/*** BEGIN SECTION TO CREATE AN ATTRIBUTE MAP TO MAKE TEXT DIFFERENT COLORS ***/
/*************************************************************************/
data BINGO.BINGOATTR (drop = i);

   /** code from Appendix A-2 is included **/

   ID = 'TXTCLRC';
   array cclrs(2) $20 _TEMPORARY_ ('lightpink' 'black');
   do i = 1 to 2;
      VALUE = cats(i);
      TEXTCOLOR = cclrs(i);
      if i = 1 then FILLCOLOR = cclrs(i+1);
      else if i = 2 then FILLCOLOR = cclrs(1);
      output;
   end;
run;
/*************************************************************************/
/*** END SECTION TO CREATE AN ATTRIBUTE MAP TO MAKE TEXT DIFFERENT COLORS ***/
/*************************************************************************/
```

APPENDIX B.3: MACRO TO RANDOMLY SELECT ITEMS AND PRODUCE BINGO CARD

The code found in Appendix A.3 can be used for the checkerboard with the exception of the portion highlighted in green in Appendix A.3. This portion would need to be modified as indicated in Appendix SAS Program 1.

Appendix SAS Program 1: Multicolor Code for Checkerboard

```
/* have text be different colors - 2 different colors */
%if &multiclr = Y %then %do;
   data bingo&i.;
      set bingo&i.;
      grp = mod(_n_, 2) + 1;
   run;
%end;
```

In addition, for the portion highlighted in turquoise, the ID value for the attribute map would need to be updated to use the correct one, 'TXTCLRC.

APPENDIX C: BINGO CARD USING IMAGES

Similar to the basic bingo card, the program can be broken into three parts.

APPENDIX C.1: MACRO VARIABLES INITIALIZATION AND INPUT OF DATA

The macro variable initialization code found in Appendix A.1 can be used for the bingo cards with images. The DATA step used to read in the catalog of items is slightly different as shown in SAS Program 9 found in Preparing Images and Data.

APPENDIX C.2: TEMPLATE FOR BINGO CARD WITH IMAGES

Notice that for a bingo card with images there is no need for an attribute map, since there are no texts or fill colors to control.

```
/*****************************************************************************/
/*** BEGIN SECTION TO DEFINE GRAPH TEMPLATE FOR BINGO CARDS WITH IMAGES ***/
/*****************************************************************************/
%macro crtcard;
   proc template;
      define statgraph bingo_img;
         begingraph / border = false backgroundcolor = CXC00000;
            layout lattice / rows = 5 rowweights = uniform
                             rowgutter = 0 rowdatarange = union
                             columns = 5 columnweights = uniform
                             columngutter = 0 columndatarange = union;

               %do row = 1 %to 5;
                  %do col = 1 %to 5;
                     layout overlay / xaxisopts = (display = NONE griddisplay = OFF)
                                      yaxisopts = (display = NONE griddisplay = OFF);
                        textplot x = xval y = yval text = english;
                        annotate / id = "IMG_&row._&col";
                     endlayout;
                  %end;
               %end;
            endlayout;
         endgraph;
      end;
   run;
%mend crtcard;

%crtcard
/*****************************************************************************/
/*** END SECTION TO DEFINE GRAPH TEMPLATE FOR BINGO CARDS WITH IMAGES ***/
/*****************************************************************************/
```

APPENDIX C.3: MACRO TO RANDOMLY SELECT ITEMS AND PRODUCE BINGO CARD

The code found in Appendix A.3 can be used for the bingo card with images with the exception of the portions highlighted in Appendix A.3. The portion highlighted in grey can be removed since there is no text that will need to go through a 'splitting' process to make the text fit. The portion highlighted in grey would need to be replaced with the code used to create the annotate data set as shown in SAS Program 10. In addition, it may be best to set the DPI to 150 instead of 300 (yellow highlighted portion).

APPENDIX D: BINGO CARD USING PROC REPORT

The code to generate the bingo card using PROC REPORT is broken into four parts.

APPENDIX D.1: MACRO VARIABLES INITIALIZATION AND INPUT OF DATA

The code found in Appendix A.1 can be used for the both the multi-color and checkerboard bingo cards produced using PROC REPORT.

APPENDIX D.2: MACRO TO RANDOMLY SELECT ITEMS AND PRODUCE BINGO CARD

The code found in Appendix A.3 can be used for the bingo card with images with the exception of the portions highlighted in grey in Appendix A.3. The portion highlighted in grey can be removed for both the multi-color and checkerboard.

For the multi-color bingo card, the portion highlighted in green would need to be kept; however, for the checkerboard it needs to be modified as shown in Appendix SAS Program 2.

Appendix SAS Program 2: Multicolor Code for Checkerboard for PROC REPORT

```
%if &multiclr = Y %then %do;
    data bingo&i.;
       set bingo&i.;
       if mod(_n_,2)=0 then checker=1;
       else checker=2;
    run;
%end;
```

APPENDIX D.3: TRANSPOSING THE DATA FOR PROC REPORT

The data set from generated in Appendix D.2 needs to be transposed as shown in SAS Program 13. For the checkerboard version of the program, there is a slight revision needed as seen by the text in red font in Appendix SAS Program 3.

Appendix SAS Program 3: Transposing Data Using DATA Step and MERGE for Checkerboard

```
data col1 (keep=rownum grp1 col1)
     col2 (keep=rownum grp2 col2)
     col3 (keep=rownum grp3 col3)
     col4 (keep=rownum grp4 col4)
     col5 (keep=rownum grp5 col5);
    set bingo&i;
    if colnum=1 then do;
        grp1=checker;
        col1=bingo_text;
        output col1;
    end;
    if colnum=2 then do;
        grp2=checker;
        col2=bingo_text;
        output col2;
    end;
    if colnum=3 then do;
        grp3=checker;
        col3=bingo_text;
        output col3;
```

```
    end;
    if colnum=4 then do;
        grp4=checker;
        col4=bingo_text;
        output col4;
    end;
    if colnum=5 then do;
        grp5=checker;
        col5=bingo_text;
        output col5;
    end;
run;
```

APPENDIX D.4: PRODUCE THE BINGO CARDS WITH PROC REPORT

Appendix SAS Program 4 is the full PROC REPORT code for generating the bingo cards with multiple color text. Appendix SAS Program 5 is the code for generating the checkerboard bingo cards with PROC REPORT.

Appendix SAS Program 4: PROC REPORT for Multi-Color Bingo Card

```
proc report nowd data=bingo&i._2 noheader
    style(report)=[cellpadding=5pt vjust=b]
    style(header)=[just=center font_face="Helvetica" font_weight=bold font_size=8pt]
    style(lines)=[just=left font_face="Helvetica"] split='|';

    columns rownum grp1 grp2 grp3 grp4 grp5 col1 col2 col3 col4 col5 ;
    define rownum / display ' ' noprint;
    define grp1 / display ' ' noprint;
    define grp2 / display ' ' noprint;
    define grp3 / display ' ' noprint;
    define grp4 / display ' ' noprint;
    define grp5 / display ' ' noprint;
    define col1 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt};
    define col2 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col3 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col4 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col5 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };

    compute col1;
          if grp1=1 then color='purple';
          else if grp1=2 then color='green';
          else if grp1=3 then color='blue';
          else if grp1=4 then color='red';
          else if grp1=5 then color='vipk';
          else color='black';
          call define ('col1','style','style=[foreground='||color||']');
    endcomp;

    compute col2;
          if grp2=1 then color='purple';
          else if grp2=2 then color='green';
          else if grp2=3 then color='blue';
          else if grp2=4 then color='red';
          else if grp2=5 then color='vipk';
          else color='black';
          call define ('col2','style','style=[foreground='||color||']');
    endcomp;

    compute col3;
          if grp3=1 then color='purple';
```

```
          else if grp3=2 then color='green';
          else if grp3=3 then color='blue';
          else if grp3=4 then color='red';
          else if grp3=5 then color='vipk';
          else color='black';
          call define ('col3','style','style=[foreground='||color||']');
    endcomp;

    compute col4;
          if grp4=1 then color='purple';
          else if grp4=2 then color='green';
          else if grp4=3 then color='blue';
          else if grp4=4 then color='red';
          else if grp4=5 then color='vipk';
          else color='black';
          call define ('col4','style','style=[foreground='||color||']');
    endcomp;

    compute col5;
          if grp5=1 then color='purple';
          else if grp5=2 then color='green';
          else if grp5=3 then color='blue';
          else if grp5=4 then color='red';
          else if grp5=5 then color='vipk';
          else color='black';
          call define ('col5','style','style=[foreground='||color||']');
    endcomp;
run;

ods pdf close;
```

Appendix SAS Program 5: PROC REPORT for Checkerboard Bingo Card

```
proc report nowd data=bingo&i._2 noheader
    style(report)=[cellpadding=5pt vjust=b]
    style(header)=[just=center font_face="Helvetica" font_weight=bold font_size=8pt]
    style(lines)=[just=left font_face="Helvetica"] split='|';

    columns rownum grp1 grp2 grp3 grp4 grp5 col1 col2 col3 col4 col5 ;
    define rownum / display ' ' noprint;
    define grp1 / display ' ' noprint;
    define grp2 / display ' ' noprint;
    define grp3 / display ' ' noprint;
    define grp4 / display ' ' noprint;
    define grp5 / display ' ' noprint;
    define col1 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt};
    define col2 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col3 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col4 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };
    define col5 / style(COLUMN)={just=c vjust=c font_face="Helvetica"
           font_size=8pt cellwidth=195 cellheight=195 borderwidth=4pt };

    compute col1;
          if grp1=1 then do; color='pink'; rcolor='black'; end;
          else do;  color='black'; rcolor='pink'; end;
          call define ('col1','style','style=[foreground='||color||'
background='||rcolor||']');
    endcomp;

    compute col2;
      if grp2=1 then do; color='pink'; rcolor='black'; end;
      else do;  color='black'; rcolor='pink'; end;
```

```
      call define ('col2','style','style=[foreground='||color||' background='||rcolor||']');
    endcomp;

    compute col3;
      if grp3=1 then do; color='pink'; rcolor='black'; end;
      else do;  color='black'; rcolor='pink'; end;
      call define ('col3','style','style=[foreground='||color||' background='||rcolor||']');
    endcomp;

    compute col4;
      if grp4=1 then do; color='pink'; rcolor='black'; end;
      else do;  color='black'; rcolor='pink'; end;
      call define ('col4','style','style=[foreground='||color||' background='||rcolor||']');
    endcomp;

    compute col5;
      if grp5=1 then do; color='pink'; rcolor='black'; end;
      else do;  color='black'; rcolor='pink'; end;
      call define ('col5','style','style=[foreground='||color||' background='||rcolor||']');
    endcomp;
run;
```