PharmaSUG2016 - Paper TT-12

Let’s Make Music: Using SAS® Functions for Music Composition

Kim Truett and Zak Truett, KCT Data, Inc., Alpharetta, GA

ABSTRACT

SAS® has been experimented with for entertaining diversions before, such as the Pegboard Game, Tetris and Solitaire, so we wondered if we could get SAS to write music using improvisational rules. Improvisation in music is not just playing random notes – it follows a defined set of rules - sometimes just one or two, sometimes many.

Following a basic set of these rules for note progression and duration, we use SAS random number functions and weighted probabilities to determine what the next note will be (the pitch), and how long the note is played (the duration). The goal of this light-hearted talk is to illustrate the basic use of SAS functions to improvise a few bars of music that can be played.

Introduction

Improvisation is the art of making up music as you are playing it. But simply playing random notes would not sound like ‘music’, so improvisation follows a basic set of music rules. We set out to teach SAS to improvise using the following set of rules, and using the 12 bar blues, a commonly known structure for improvisation used by musicians for blues music, as the basis:

Key to converting the familiar letter notation to numbers (this is a C scale, so C is #1)

1 = C	5 = E	9 = G#
2 = C#	6 = F	10 = A
3 = D	7 = F#	11 = B flat
4 = E Flat	8 = G	12 = B

We started with 3 basic chords, or groups of notes:

C7 chord, consisting of notes 1, 5, 8, 11 (C-E-G-B flat)
F7 chord, consisting of notes 6, 10, 1, 4 (F-A-C-E flat)
G7 chord, consisting of notes 8, 12, 3, 6 (G-B-D-F)

And then established the number of measures that each chord would be used for selecting notes, giving us our 12 bar blues structure:

C7 x 4
F7 x 2
C7 x 2
G7 x 2
C7 x 2.

So the first chord was to be used 4 times, and the remaining chords 2 times each. Note that the initial chord actually repeats within the sequence.

Within each chord, we used different subgroups of notes. The first subgroup was made up of the notes in the chord, for example, C, E, G and B flat. The second subgroup was made up of the notes in the pentatonic scale associated with the chord. For the C7 chord, these notes are C, D, E, G and A. Lastly, to add extra variety, we defined a subgroup of extra notes, which, for the C7 chord, were C, D, E, F, G, and A. The overlap in the notes within the subgroups reflects the fact that an improvising musician has multiple reasons for choosing what note to play.

All chords are built on a root note, which is the note that gives the chord its name. One of our rules for improvisation, for our purpose, was that the first note of every measure would be the root note. For the C7 measures, the first note would be C; for the F7 measures, the first note would be F; and for the G7 measures, the first note would be G. Starting the measure with the root note helps the listener to recognize the chord being used, since the subsequent notes will add on to this root note.

After the root note, there was a random choice of notes. Notes were mostly likely to be chosen from the pentatonic scale subgroup, less likely to be chosen from the chord subgroup, and least likely to be chosen from the extra subgroup.

To further emphasize the sound of the chord, the last note of a measure before a chord change would also the root note. For example, the last note of the fourth measure would be C, since the chord for the fourth measure is C7, and the chord for the fifth measure is F7.

To establish the duration of the notes, we used eighth notes, so that there were 8 notes per measure, and, when a note was ‘repeated’ it was connected to the other notes.

We also wanted to put this into a macro so that the “SAS musician” could dictate the scales to improvise around, as well as the number of notes generated.

Body

The macro we created is called makenotes.

The macro call consists of sending the macro seven parameters:

```
       %makenotes(likely,unlikely,extra,weight1,weight2,weight3,numberofcycles);
```

The subgroup of notes that are likely are listed in a string (&likely.), the subgroup of notes that are unlikely are listed in a second string (&unlikely), and the extra notes (&extra.) are in a third string. Next the probability, as a percentage, of each of the subgroups being used is listed (&weight1., &weight2., and &weight3.), and last is the number of cycles (or measures of music) we want to generate. For our purposes, we assumed that there would be 8 eighth notes in each cycle, so each cycle produced one measure of music in 4/4 time, but this could be adjusted as well, if desired.

We then created an array of variables to hold the 8 notes;

```
    array notes (8) note1 - note8;
```

and looped those these 8 notes once for each cycle.

```
    do i = 1 to &numberofcycles.;
       do j = 1 to 8;
```

According to the rules that we set up, the first note in a measure should be the first note in the chord, and the very last note of the last cycle should be the same note, so we set those notes specifically, instead of by random choice:

```
    if j = 1 then notes (j)= substr(&likely.,1,2);
    else if j = 8 and i = &numberofcycles. then notes (j)= substr(&likely.,1,2);
```

For the other notes we generated a random number using the RAND function with the “uniform” distribution ensuring that each run would give us different music.

```
    x = int(RAND('UNIFORM') * 100);
```

The likelihood of the next note being chosen from within the pentatonic scale was set at 50%, and within the chord was set at 32%, meaning the odds of an extra note was 18%. There are 4 notes within the chord, 5 notes within the pentatonic scale, and 6 notes in the extra set, so LENGTH/2 was used to calculate the intervals within each probability (so, if there are 4 notes, the length of the string is 8, and the probability of 32% can be divided by (8 / 2 = 4) to uniformly select one of the 4 possible notes). Once the starting position was set (for example 1,3,5,7 for the within-chord notes), the position was used within SUBSTR to select the appropriate note. The same process was used for the pentatonic scale and extra notes, where there were 5 and 6 choices, respectively.

```
    if x < &weight1. then do;
      position = int(x / (&weight1. / (length(&likely.)/2))) * 2 + 1;
      notes (j) = substr(&likely.,position,2);
    end;
    else if x < (&weight1. + &weight2.) then do;
      position = int((x - &weight1.) / (&weight2. / (length(&unlikely.)/2)))*2+1;
      notes (j) = substr(&unlikely.,position,2);
    end;
    else do;
      position = int((x - (&weight1. + &weight2.)) /
                 (&weight3. / (length(&extra.)/2))) * 2 + 1;
      notes (j) = substr(&extra.,position,2);
    end;
```

To create our twelve bar blues, we used five calls to the macro, sending different chords and the number of times to use that chord. Each time, the resulting DATA A was set to a cumulative data set ALL.

```
  %makenotes("01050811","0103050810","010305060810",32,50,18,4);
  data all; set all a; run;

  %makenotes("06100104","0608100103","060810120103",32,50,18,2);
  data all; set all a; run;

  %makenotes("01050811","0103050810","010305060810",32,50,18,2);
  data all; set all a; run;

  %makenotes("08120306","0810120305","081012010305",32,50,18,2);
  data all; set all a; run;

  %makenotes("01050811","0103050810","010305060810",32,50,18,2);
  data all; set all a;run;
```

Lastly, we created a format, so that the music would print out in the more familiar letter notation and dropped the I and j index variables.

We decided to use a program called Lilypond (www.lilypond.org) to generate sheet music from this data, so we used Lilypond notation in the format, but this could easily be changed to use the more familiar letter notation as well. Both are shown here:

(Lilypond notation)
```
    proc format;
      value letters
      1 = "c''8"
      2 = "cis''8"
      3 = "d''8"
      4 = "ees''8"
    Etc.
```

(standard notation)
```
    proc format;
      value letters
      1 = 'C'
      2 = 'C#'
      3 = 'D'
      4 = 'E Flat'
    Etc.

    data all; set all; drop i j;
    format note1-note8 letters.;
    run;
```

Lastly, to be able to input into Lilypond, we added the Pre and Post notations needed.

```
    data pre;
    format fullstring $200.;
```

```
fullstring = "\header{"; output;
fullstring = 'title = "SAS music"'; output;
fullstring = "}"; output;
fullstring = "fixed c' {"; output;
fullstring = "\time 4/4"; output;
fullstring = "\clef treble"; output;
fullstring = "\absolute "; output;
run;

data post;
format fullstring $200.;
fullstring = "c''2"; output;
fullstring = "}"; output;
run;
```

Because Lilypond will also bridge notes if we add a tilde, we ran the notes through one final array to identify where two notes were repeated, and inserted tildes and concatenated the final results.

```
array notes(8) note1-note8;
array tilde(8) $ tilde1-tilde8;
do i = 1 to 7;
  if notes(i) = notes(i+1) then tilde(i) = '~';
end;
fullstring =
catx('',put(note1,letters.),tilde1,put(note2,letters.),tilde2,put(note3,letter
s.),tilde3,put(note4,letters.),tilde4,put(note5,letters.),tilde5,put(note6,let
ters.),tilde6,put(note7,letters.),tilde7,put(note8,letters.),tilde8);
```

If instead, the data needs to be output to a midi player, here are the Pre and Post notations needed for that:

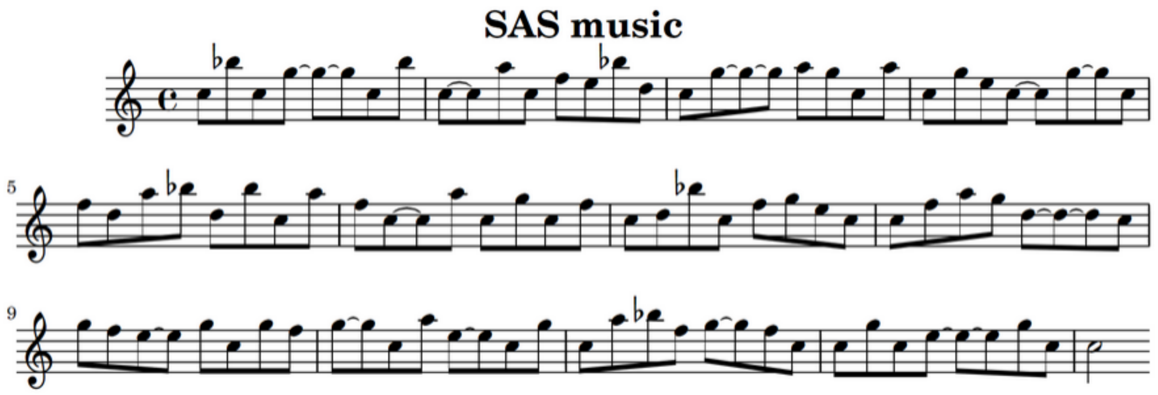
```
data midipre;
format fullstring $200.;
fullstring = "\header{"; output;
fullstring = 'title = "SAS music - midi"'; output;
fullstring = "}"; output;
fullstring = "\score {"; output;
fullstring = "  \new Score {"; output;
fullstring = "{"; output;
fullstring = "\time 4/4"; output;
fullstring = "\clef treble"; output;
fullstring = "\absolute"; output;
run;

data midipost;
format fullstring $200.;
fullstring = "c''2"; output;
fullstring = "}"; output;
fullstring = "}"; output;
fullstring = "\midi { }"; output;
fullstring = " \layout { }"; output;
fullstring = "}"; output;
run;
```

The full text of the program is at the end of this paper.

Here is a sample of the output!

```
\header{
title = "SAS music"
}
fixed c' {
\time 4/4
\clef treble
\absolute
c''8 bes''8 c''8 g''8 ~ g''8 ~ g''8 c''8 bes''8
c''8 ~ c''8 a''8 c''8 f''8 e''8 bes''8 d''8
c''8 g''8 ~ g''8 ~ g''8 a''8 g''8 c''8 a''8
c''8 g''8 e''8 c''8 ~ c''8 g''8 ~ g''8 c''8
f''8 d''8 a''8 bes''8 d''8 bes''8 c''8 a''8
f''8 c''8 ~ c''8 a''8 c''8 g''8 c''8 f''8
c''8 d''8 bes''8 c''8 f''8 g''8 e''8 c''8
c''8 f''8 a''8 g''8 d''8 ~ d''8 ~ d''8 c''8
g''8 f''8 e''8 ~ e''8 g''8 c''8 g''8 f''8
g''8 ~ g''8 c''8 a''8 e''8 ~ e''8 c''8 g''8
c''8 a''8 bes''8 f''8 g''8 ~ g''8 f''8 c''8
c''8 g''8 c''8 e''8 ~ e''8 ~ e''8 g''8 c''8
c''2
}
```

Here is the resulting music, written by SAS, with sheet music generated by Lilypond!



Conclusion

It works! SAS makes music following these rules, and it even sounds a bit like blues. While this simple macro does not compare to true improvisation, it still created something interesting.

References

Contact Information

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

Kim Truett (SAS questions) / Zak Truett (music questions)
11585 Jones 11877 Douglas Rd, Ste 102-146
Alpharetta, GA 30022
770.372.0989
Email: KCTData03@kctdm.com

Full Program:

```
%macro makenotes(likely,unlikely,extra,weight1,weight2,weight3,numberofcycles);
data a;
array notes (8)  note1 - note8;
do i = 1 to &numberofcycles.;
   do j = 1 to 8;
     if j = 1 then notes (j)= substr(&likely.,1,2);
     else if j = 8 and i = &numberofcycles. then notes (j)=
substr(&likely.,1,2);
     else do;

     x = int(RAND('UNIFORM') * 100);
     if x < &weight1. then do;
       position = int(x / (&weight1. / (length(&likely.)/2))) * 2 + 1;
       notes (j) = substr(&likely.,position,2);
    end;
    else if x < (&weight1. + &weight2.) then do;
       position = int((x - &weight1.) / (&weight2. / (length(&unlikely.)/2))) *
2 + 1;
       notes (j) = substr(&unlikely.,position,2);
    end;
    else do;
       position = int((x - (&weight1. + &weight2.)) / (&weight3. /
(length(&extra.)/2))) * 2 + 1;
       notes (j) = substr(&extra.,position,2);
    end;
  end;
  end;
  output;
end;
%mend;

data all;
format note1-note8 8.;
run;

%makenotes("01050811","0103050810","010305060810",32,50,18,4);
data all; set all a; run;

%makenotes("06100104","0608100103","060810120103",32,50,18,2);
data all; set all a; run;

%makenotes("01050811","0103050810","010305060810",32,50,18,2);
data all; set all a; run;

%makenotes("08120306","0810120305","081012010305",32,50,18,2);
data all; set all a; run;

%makenotes("01050811","0103050810","010305060810",32,50,18,2);
data all; set all a;run;

proc format;
  value letters
  1 = "c''8"
  2 = "cis''8"
  3 = "d''8"
  4 = "ees''8"
  5 = "e''8"
  6 = "f''8"
  7 = "fis''8"
  8 = "g''8"
  9 = "gis''8"
  10 = "a''8"
  11 = "bes''8"
  12 = "b''8";

data pre;
format fullstring $200.;
fullstring = "\header{"; output;
fullstring = 'title = "SAS music"'; output;
fullstring = "}"; output;
fullstring = "fixed c' {"; output;
fullstring = "\time 4/4"; output;
fullstring = "\clef treble"; output;
fullstring = "\absolute "; output;
run;
data post;
format fullstring $200.;
fullstring = "c''2"; output;
fullstring = "}"; output;
run;

data all; set all; drop i j;
format note1-note8 letters. fullstring $200.;
if note1 = '' then delete;
array notes(8) note1-note8;
array tilde(8) $ tilde1-tilde8;
do i = 1 to 7;
  if notes(i) = notes(i+1) then tilde(i) = '~';
end;
fullstring =
catx('',put(note1,letters.),tilde1,put(note2,letters.),tilde2,put(note3,letters
.),tilde3,put(note4,letters.),tilde4,put(note5,letters.),tilde5,put(note6,lette
rs.),tilde6,put(note7,letters.),tilde7,put(note8,letters.),tilde8);

data final;
set pre all post;
proc print noobs;
var fullstring;
run;

data midipre;
format fullstring $200.;
fullstring = "\header{"; output;
fullstring = 'title = "SAS music - midi"'; output;
fullstring = "}"; output;
fullstring = "\score {"; output;
fullstring = "  \new Score {"; output;
fullstring = "{"; output;
fullstring = "\time 4/4"; output;
fullstring = "\clef treble"; output;
fullstring = "\absolute"; output;
run;

data midipost;
format fullstring $200.;
fullstring = "c''2"; output;
fullstring = "}"; output;
fullstring = "}"; output;
fullstring = "\midi { }"; output;
fullstring = " \layout { }"; output;
fullstring = "}"; output;
run;

data allmidi; set all; drop i j;
format note1-note8 letters. fullstring $200.;
if note1 = '' then delete;
array notes(8) note1-note8;
array tilde(8) $ tilde1-tilde8;
do i = 1 to 7;
  if notes(i) = notes(i+1) then tilde(i) = '~';
end;
fullstring =
catx('',put(note1,letters.),tilde1,put(note2,letters.),tilde2,put(note3,letters
.),tilde3,put(note4,letters.),tilde4,put(note5,letters.),tilde5,put(note6,lette
rs.),tilde6,put(note7,letters.),tilde7,put(note8,letters.),tilde8);

data final;
set midipre allmidi midipost;
proc print noobs;
var fullstring;
```

```
run;
```