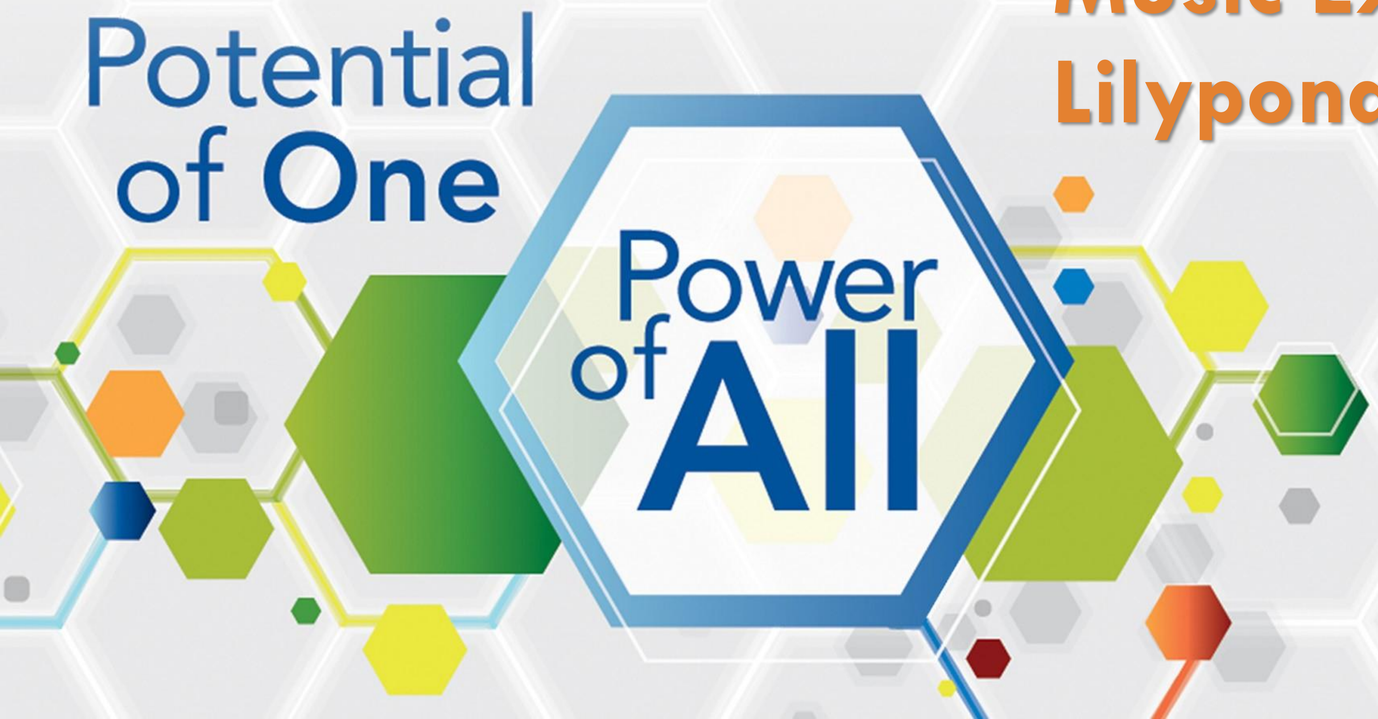
SAS® GLOBAL FORUM

SAS® as a Code Manipulation Language

An Example of Writing a Music Exercise Book with Lilypond and SAS



SAS® as a Code Manipulation Language: An Example of Writing a Music Exercise Book with Lilypond and SAS.

Peter Timusk
Statistics Canada

Abstract

Using Lilypond typesetting software, you can write publication-grade music scores. The input for Lilypond is a text file that can be written once and then transferred to SAS® for patterned repetition, so that you can cycle through patterns that occur in music. The author plays a sequence of notes and then writes this into Lilypond code. The sequence starts in the key of C with only a two-note sequence. Then the sequence is extended to three-, four-, then five-note sequences, always contained in one octave. SAS is then used to write the same code for all other eleven keys and in seven scale modes. The method is very simple and not advanced programming. Lookup files are used in the programming, demonstrating efficient lookup techniques. The result is a lengthy book or exercise for practicing music in a PDF file, and a sound source file in midi format is created that you can hear. This method shows how various programming languages can be used to write other programming languages.

Objective

- Demonstrate the use of programming to write repetitive structured text such as a program or instruction set for another software.
- Produce a music exercise book
- Exhaustively cover all keys and 7 scale modes in the music

Method

- Method Step 1 **Lilypond sequence input file:** Set the musical idea, a sequence into Lilypond code in a DATASET. We used Microsoft Excel® for entering the code.
- Method Step 2 **Lilypond chromatic scale input file:** Set up a chromatic scale DATASET in Lilypond code that we can use for a LOOKUP file. We used SAS® Enterprise Guide® to write the lookup file.
- Method Step 3 **SAS Program:** Read in the files into a SAS session and program the repetitive patterns with loops and specific modal patterns and output all results to one Excel file.
- Method Step 4 **Compile a Text file in Lilypond:** Copy the Excel file contents to a text file and process the text file in a Lilypond compiler. This outputs the one PDF sheet music book file with added guitar tablature and multiple MIDI files. One MIDI file is created for each sequence, each key and each mode.

Results

- A PDF sheet music document containing all the sequences is produced and separate MIDI files for each key, each sequence and each of the 7 modes are also produced.
- **Examples:**
- View and hear : C Major 4 note sequence
- View and hear : B Locrian 5 note sequence

Conclusions

- Programming allows for patterned and structured repetition.
- One can use programming to make software code files for other software systems.
- Programming is a good approach for large volumes for work.

References

- LilyPond Notation Reference, http://www.lilypond.org/doc/v2.18/Documentation/notation/index.html (Accessed February 2nd, 2014).

Washington, D.C.

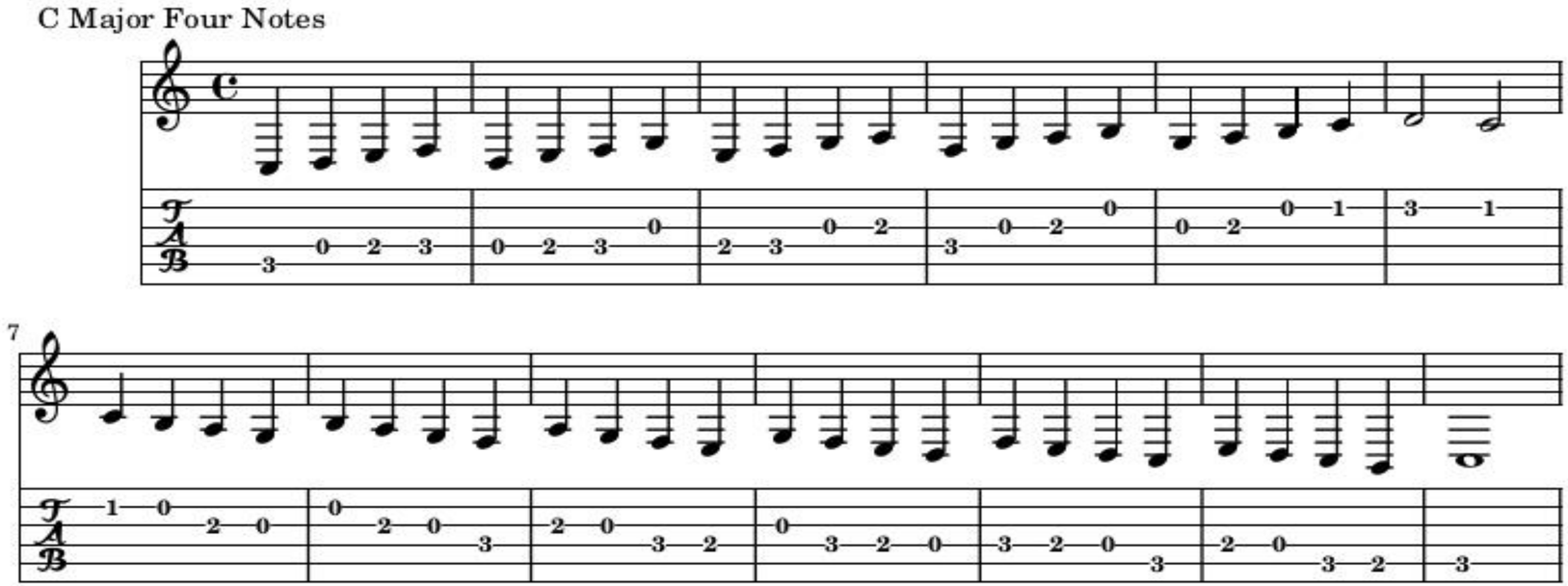
March 23–26, 2014

SAS® as a Code Manipulation Language:
An Example of Writing a Music Exercise Book with Lilypond and SAS.

Peter Timusk
Statistics Canada

Results Slide 1

A MIDI file and a PDF document are produced for each key, each sequence and 7 modes



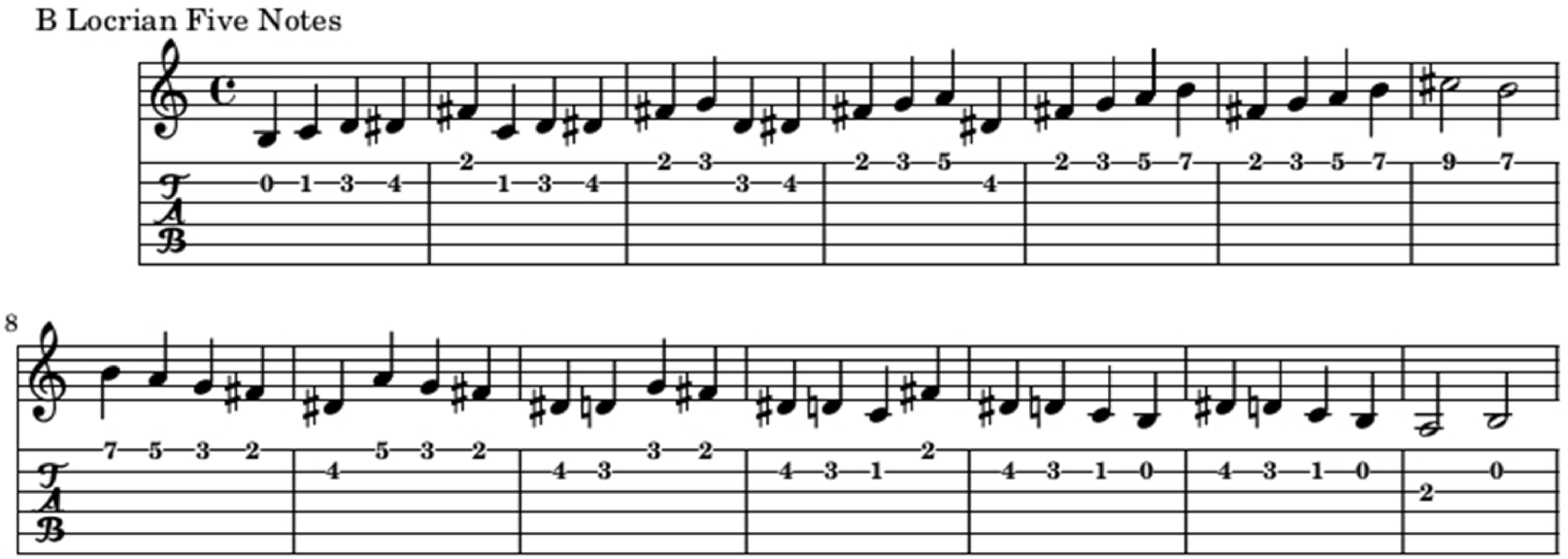
Beginning Main Page End

SAS® as a Code Manipulation Language:
An Example of Writing a Music Exercise Book with Lilypond and SAS.

Peter Timusk
Statistics Canada

Results Slide 2

A MIDI file and a PDF document are produced for each key, each sequence and 7 modes



Beginning Main Page End

Washington, D.C.

March 23–26, 2014

Beginning Main Page End

Method Step 1 Slide 1

First the 2 to 5 notes sequences are coded manually in text consisting of LilyPond code (shown in the second line of the three fields below.)

xar1…xar170 are variable names we will use to access and manipulate the text in SAS. Each sequence takes one line of text and is xar1 to xar170

We will follow this one sequence (C Major 2 note) of "mynotes" section along on the next slides. We will skip some variables so we can see more .

xar1	xar2	xar3	xar4	xar5	xar6
\header { title = "		Sequences	"	composer = "	Peter Timusk

xar7	xar8	xar9	xar10	xar11	xar12
" }		mynotes = {		c	4

xar13	xar14	xar15	xar16	xar17	xar18	xar19
d	d	e	|	e		4

One bar has passed of 4 quarter notes. The notes are c, d, d and e.

xar11 is really the first note.

Beginning Main Page End

Peter Timusk
Statistics Canada

Method Step 1 Slide 2

Following along, we reach the C note one octave above our starting C and we use a single quote ‘ after the letter to indicate we are one octave higher in xar38

We go one note higher than the octave to D. We start our descent at xar45 and in xar46 we switch from quarter notes (a 4) as in xar37, to one whole note (a 1) xar46 to accent the key of C.

We continue descending from C to B now again using quarter notes by setting xar59 to a 4.

xar35	xar36	xar37	xar38	xar39	xar40
b		4	c'		c'
xar41	xar42	xar43	xar44	xar45	xar46
	d'		|	c'	1
xar57	xar58	xar59	xar60	xar61	xar62
c'		4	b		b

Beginning Main Page End

Peter Timusk
Statistics Canada

Method Step 1 Slide 3

Following along descending we reach the starting C note again at xar87 and we use a single comma on the B note at xar91 as we go one note lower than the C so indicate this is a lower B.

We stop at the C note at xar93 and again we switch from quarter notes (a 4) as in xar86 to one whole note (a 1) at xar95 to accent the key of C.

The remainder of the variables are either for longer 3, 4 and 5 notes sequences or for Lilypond formatting code that we will not work with here.
On the next page, we will look at the end of the 5 note sequence.

xar84	xar85	xar86	xar87	xar88	xar89
d		4	c		c

xar90	xar91	xar92	xar93	xar94	xar95
	b,	|	c		1

Beginning Main Page End

Peter Timusk
Statistics Canada

Method Step 1 Slide 4

We have looked at the first line of Lilypond code which covered the C Major 2 note sequence.
Here we briefly view the end of the C Major 5 note sequence.

Here we pass 5 notes starting at xar117 with F to xar127 and lower B. Here instead of whole notes we have used half notes (a 2) at xar128 and xar132 for accents.

This concludes the first step of the method.

xar115	xar116	xar117	xar118	xar119	xar120
c	|	f		4	
xar121	xar122	xar123	xar124	xar125	xar126
e		d		c	|
xar127	xar128	xar129	xar130	xar131	xar132
b,	2		c		2

Beginning Main Page End

Washington, D.C.

March 23–26, 2014

Beginning Main Page End

Peter Timusk
Statistics Canada

Method Step 2 Slide 1

Here in one slide is the lookup file we use to reference a Lilypond chromatic scale. The file starts with note 1 as a C. cis means a C sharp (C# in music notation).

The highest note we will need is when playing a B key, 11 Keys up from a C, reaching the higher B octave (b' note 24) and going one tone higher to a C# (cis'' note 26) before resolving back to the B.

The notes 27, 28 and 29 are needed for programming convenience.
This chromatic scale uses only sharps not flats. Again this is for programming convenience.

Note1	Note2	Note3	Note4	Note5	Note6	Note7	Note8	Note9	Note10
c	cis	d	dis	e	f	fis	g	gis	a

Note11	Note12	Note13	Note14	Note15	Note16	Note17	Note18	Note19	Note20
ais	b	c'	cis'	d'	dis'	e'	f'	fis'	g'

Note21	Note22	Note23	Note24	Note25	Note26	Note27	Note28	Note29
gis'	a'	ais'	b'	c''	cis''	b'	b,	ais,

Beginning Main Page End

Washington, D.C.

March 23–26, 2014

Beginning Main Page End

Method Step 3 Slide 1

Here in the SAS program we import the Excel music sequence file from Step 1 and use a PROC SQL join to join it to the chromatic scale from Step 2.

```
PROC IMPORT OUT= IN2.IMPORTEDLILYPONDCMAJORSEQ
            DATAFILE=
"C:\Users\ptimusk\Dropbox\Lilypondcode\Sequences_Lists of
scales2.txt"
            DBMS=TAB REPLACE;
     GETNAMES=NO;
     DATAROW=1;
RUN;
PROC SQL;
create table in2.LilypondwithChrom
as select *
from
 in2.IMPORTEDLILYPONDCMAJORSEQ as a left join
in2.lilypond_lookup_chromatic as b
on a.Var11=b.Note1 where substr(a.Var1,1,5) ne 'Dummy' ;
QUIT;
```

Beginning Main Page End

Method Step 3 Slide 2

Here in the SAS program we start a MACRO program and a %DO loop j to cover the other 11 keys in Western music. Then a DATASTEP reads in our input file and creates Major scale DATASETS called majtranpose0 to majtranspose11. We create some arrays for old and new notes and start a DO loop inside the DATASTEP to loop through our variables. We do some changes to the Lilypond title and end code. The DATASTEP continues on the next slide.

```
%MACRO loopmajors();
%DO j=0 %TO 11;

DATA majtranspose&j.;
SET in2.LilypondwithChrom;
FORMAT xar1-xar170 $147.;
   ARRAY old (170) var1 - var170;
ARRAY Notes (29) Note1 - Note29;
    ARRAY new (170) xar1 - xar170;

DO i= 1 TO 170;
new(i)=old(i);
IF  old(i)="C Major Sequences" THEN new(i)="Sequences";
IF  old(i)='"}}' THEN new(i)='"} \midi{} \layout {}}';
```

Beginning Main Page End

Method Step 3 Slide 3

Here in the SAS program we change the title of the sheet music Key from capital C to the appropriate capital letter for the present Key in the loop. This will not effect the Lilypond "mynotes" code as these are small letters for note names. We are only calling Keys by sharps which is perhaps not technically or musically correct.

```
IF &j.=1 and old(i)="C" THEN new(i)="C#";
IF &j.=2 and old(i)="C" THEN new(i)="D";
IF &j.=3 and old(i)="C" THEN new(i)="D#";
IF &j.=4 and old(i)="C" THEN new(i)="E";
IF &j.=5 and old(i)="C" THEN new(i)="F";
IF &j.=6 and old(i)="C" THEN new(i)="F#";
IF &j.=7 and old(i)="C" THEN new(i)="G";
IF &j.=8 and old(i)="C" THEN new(i)="G#";
IF &j.=9 and old(i)="C" THEN new(i)="A";
IF &j.=10 and old(i)="C" THEN new(i)="A#";
IF &j.=11 and old(i)="C" THEN new(i)="B";
```

Beginning Main Page End

Method Step 3 Slide 4

Here in the SAS program we change the note variables. In musical terms this is called a transposition. This is an example for the Major scale mode. We have 7 modes, so 7 different transposition DATASTEPS. A Phrygian mode example is on slide 6 in this method step. We look up the old note in the chromatic scale file variables Notes() and change it by the loop increment j, in other words each pass through the j loop, we move all notes up one half step or one semitone. Note(28) deals with the extreme ends of the sequence. We here end the loop through the 170 variables in one sequence. This is not the most efficient lookup technique yet effective because we want to change values the variable takes not just group or format the variables.

```
IF old(i)=Notes(1) THEN new(i)=Notes(&j.+1);
IF old(i)=Notes(3) THEN new(i)=Notes(&j.+3);
IF old(i)=Notes(5) THEN new(i)=Notes(&j.+5);
IF old(i)=Notes(6) THEN new(i)=Notes(&j.+6);
IF old(i)=Notes(8) THEN new(i)=Notes(&j.+8);
IF old(i)=Notes(10) THEN new(i)=Notes(&j.+10);
IF old(i)=Notes(12) THEN new(i)=Notes(&j.+12);
IF old(i)=Notes(13) THEN new(i)=Notes(&j.+13);
IF old(i)=Notes(15) THEN new(i)=Notes(&j.+15);

IF old(i)=Notes(28) and &j. ne 0 THEN new(i)=Notes(&j.);
IF old(i)=Notes(28) and &j. eq 0 THEN new(i)=Notes(28);
END;
```

Beginning Main Page End

Peter Timusk
Statistics Canada

Method Step 3 Slide 5

At the end of the DATASTEP (at the first RUN;) we keep only the needed xar1-xar170 variables and we trim off some extra lines with an inclusive IF, including only lines without `'Dummy'` in variable xar1. Then using a PROC APPEND we join the Key we just completed to a BASE DATASET allscalev1. Then we start the j loop again for the next Key with a %END.

```
KEEP xar1-xar170;
IF substr(xar1,1,5) ne 'Dummy';
RUN;
PROC APPEND BASE=allscalev1 DATA=majtranspose&j.;
RUN;
%END;
```

Beginning Main Page End

Method Step 3 Slide 6

We just saw the DATASTEP for the 12 Major scales. Here below is the code for the Phrygian mode scales. Knowing that the second note in our original scale C Major is `Notes(3)`, 2 semitones above the C or `Notes(1)` and that the second note for the Phrygian is only 1 semitone above the note C, we change `Notes(3)`, to `Notes(&j.+2)` only 1 semitone above the note C. The Phrygian mode has other characteristics reflected in our code.

```
IF old(i)=Notes(1) THEN new(i)=Notes(&j.+1);
IF old(i)=Notes(3) THEN new(i)=Notes(&j.+2);
IF old(i)=Notes(5) THEN new(i)=Notes(&j.+4);
IF old(i)=Notes(6) THEN new(i)=Notes(&j.+6);
IF old(i)=Notes(8) THEN new(i)=Notes(&j.+8);
IF old(i)=Notes(10) THEN new(i)=Notes(&j.+9);
IF old(i)=Notes(12) THEN new(i)=Notes(&j.+11);
IF old(i)=Notes(13) THEN new(i)=Notes(&j.+13);
IF old(i)=Notes(15) THEN new(i)=Notes(&j.+15);

IF old(i)=Notes(28) and &j. ne 0 and &j. ne 1 THEN
new(i)=Notes(&j.-1);
IF old(i)=Notes(28) and &j. eq 0 THEN new(i)=Notes(29);
IF old(i)=Notes(28) and &j. eq 1 THEN new(i)=Notes(28);
```

Beginning Main Page End

Method Step 3 Slide 7

After passing through a DATASTEP for each scale mode and appending each mode and each key into the DATASET allscalesv1, we close our MACRO program with a %MEND and then call the MACRO with the line %loopscales ();. We then export it to an Excel file.

```
%MEND;
%loopscales();
PROC EXPORT DATA= allscalesv1
            OUTFILE=
"C:\Users\ptimusk\Dropbox\Lilypondcode\test7midi
tranposeAllScales.xls"
            DBMS=EXCEL REPLACE;
     SHEET="SASEXPORT";
RUN;
```

Beginning Main Page End

Washington, D.C.

March 23–26, 2014

Beginning Main Page End