Colagem, recorte e erros em um processo composicional utilizando o Music21.

Guilherme Lunhani[1]

[1]Instituto de Artes e Design – Universidade Federal de Juiz de Fora
Juiz de Fora, MG

gcravista@gmail.com

Abstract. *This article describes a case study in Computer Generated Assistance, for music analysis and didactic composition. A Python command was programmed to automate routines based on [Music21 2015] library, such: selection, cluster and fragmentation from a document in J.S.Bach's corpus. Some compositional exercises were used to test a operation, called* glitch*, in this corpus. In addition, [MuseScore 2015] and [Lilypond 2015] were used to edit and diagram scores. At the end, comment on bugs, compositional problems and future plans of composition.*

Resumo. *Este artigo descreve um estudo de caso em Assitência Gerada por Computador para análise musical e composição didática. Um comando Python foi programado para automatizar rotinas da biblioteca [Music21 2015] como: seleção, agrupamento e fragmentação de um documento no corpus bachiano. Alguns exercícios composicionais foram usados para testar uma operação* glitch*, neste corpus. Adicionalmente, [MuseScore 2015] e [Lilypond 2015] foram utilizados para edição. Ao final comentarei* bugs*, problemas composicionais e planos futuros de composição.*

1. Introdução

Este artigo trata de um protótipo, *m21.py*, desenvolvido a partir da biblioteca [Music21 2015], para composição e análise musical. Na seção 2 contextualizo o que me levou a elaborar o programa. Na seção 3 descrevo as tarefas realizadas para desenvolvimento do *m21*. Na seção 4 a utilização do *m21*.

Este código capacitou a produção de um número considerável de exercícios composicionais para piano. A intenção é oferecer uma ferramenta quase imediatista de material pré-composicional para exercícios criativos. Pode ser útil em cursos de Composição Assistida por Computador, em universidades ou em oficinas de arte. Por economia de espaço,apresentarei na seção 5 um exemplo. Na seção 6 discuto problemas técnicos do *software*, problemas composicionais. Na seção 7 planos futuros.

2. Trabalhos relacionados

Durante uma pesquisa de mestrado, sobre *live coding*, tive contato com o Music21 que, segundo [Soares 2015]:

> É uma biblioteca projetada para trabalhar com manipulação e análise de corpus de arquivos partituráveis. Prepara a conversão entre diversos arquivos de dados musicais. (...) Music21 tem uma abordagem voltada para uma "musicologia assistida por computador"e já tem incorporada em

> suas classes algumas ferramentas comuns a esta prática como: numeração de grau funcional de acorde, numeração de classes de altura usando a classificação de Allen Forte : a implementação dos algoritmos de detecção de tonalidade elaborado por Krumhansl (1990) e aperfeiçoada por Temperley (2001), busca de padrões como transposições e inversões e outros.[Soares 2015, p. 71-72]

Ao invés de compor adicionando informações aos dados musicais, busquei na *Estética do Erro* [Cascone 2000] os procedimentos básicos para composição, explicados com mais detalhes na seção 3. Resultados sonoros procedentes da colagem e do erro dependem muito do *input*. Aplicar o mesmo algoritmo de erro para diferentes materiais, ou, aplicar diferentes erros para um único material, não resulta em um produto homogêneo.

Busquei então utilizar diferentes documentos do *corpus*, e um mesmo erro para realizar um tipo de música que, segundo [Soares 2015, p. 18], existia intensamente "antes da preocupação imediata com os timbres ou da era das manipulações de amostras sonoras - e de certa maneira ainda proto-serialista. Uma música por vezes chamada politonal, polimodal ou usando o termo de Straus (2004): pós-tonal.". Sendo um tipo de composição que não é nova, mas relevante do ponto de vista histórico e didático-composicional, busquei elaborar uma ferramenta para ser usada em processos criativos musicais.

Questões pessoais

Não deixo de mencionar uma antiga conversa com o compositor Franscisco Zmekhol Nascimento de Oliveira, que levou-me a compor segundo regras arbitrárias, na contingência do momento. Isto é, uma música para cada dia da existência.

Ao realizar o mesmo procedimento de composição de um mesmo documento do *corpus* bachiano, o material pré-composicional resultante deve ser diferente de qualquer outro.

Para realizar tecnicamente, valores não-determinados em operações determinísticas são utilizados.

Para compor-improvisar com os materiais resultantes, apliquei princípios de articulação do som pelo silêncio, elaborados por rascunhos de partituras-planimétricas, utilizadas por [Koellreutter 1987].

3. Metodologia

Organização dos códigos

O programa foi separado em três arquivos: *i)* um binário em *Python* que realiza tarefas gerais da linha de comando (*m21*); *ii)* rotinas do Music21 (*m21utils.py*); e *iii)* um para rotinas externas (*tools.py*) [1].

Categorização do software

Nas palavras de [Cope 2008, p. x-xiii], o *m21* pode ser classificado como uma ferramenta para uma Assistência Gerada por Computador (*Computer Generated Assistance* ou CGA). Dentro das sub-categorias de CGA propostas por Cope, o *m21* pode ser incuído nos três modos abaixo: *1)* uso de uma Linguagem de Programação em texto (PLs)(*Programming Languages*) ao invés de uma linguagem de programação visual (VPL);

[1] Todos códigos, exemplos e documentação estão disponíveis `https://www.github.com/jahpd/m21`.

2) o material partitural é gerado para performance humana ao invés de uma performance eletroacústica; *3)* abordagens baseadas em regras (*Rules Based*) e Dirigido a dado (*Data-Driven*) são usados para modificar um material existente.

Método de composição

Em geral, o procedimento de composição se deu a partir da execução de um comando *m21* com argumentos explicados na seção 5: *i)* gerar um material musical com base em uma colagem de uma obra no corpus do Music21; *ii)* o material colado será submetido a subtração de compassos, aleatoriamente; *iii)* dos compassos restantes, quaisquer notas serão agrupadas como um evento; *iv)* destes agrupamentos, as oitavas serão embaralhadas; e *v)* do bloco harmônico resultante, pode ser que alguma nota fique deslocada, gerando figuras.

Busquei notar densidades, fraseado e pontos de finalização "naturais" do material harmônico resultante Tais parâmetros eram editados no MuseScore, sendo que algumas interferências não previstas foram incluídas. Após edição, ocorreu o processo de editoração no Lilypound para melhor visualização dos resultados.

Por último, o material pré-composicional foi editado no [MuseScore 2015] e posteriorente diagramado no [Lilypond 2015].

4. M21

4.1. Instalação e uso

Pelo terminal

Considerando que o sistema operacional já possui o *git*[2], podem ser dados os seguintes comando em um terminal.

```
guilherme@R410-L-BP12P1 > git clone https://www.github.com/jahpd/m21.git
guilherme@R410-L-BP12P1 > cd m21
guilherme@R410-L-BP12P1 > chmod u+x ./m21
```

Opções

Ao executar o comando *help* é obtida uma página de ajuda.

```
guilherme@R410-L-BP12P1 > ./m21 --help
Usage: m21 [OPTIONS, [ARGS]]

For computer assisted musicology and composition

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -s, --search-only     Search in corpus for words in --composer and or
                        --index arguments. Ex.: --composer bach --index bwv1x
  -c COMPOSER, --composer=COMPOSER
                        write report with specific corpora. CAUTION: you must
                        use this according http://web.mit.edu/music21/doc/syst
                        emReference/referenceCorpus.html#referencecorpus
  -i INDEX, --index=INDEX
                        Search in corpus specific index of corpora; you must
                        use this with -c option, according available corpora
                        in  http://web.mit.edu/music21/doc/systemReference/ref
                        erenceCorpus.html#referencecorpus
  -C, --CAC             Or simple "Computer Assisted Composition". With this
                        option you will "glitch" a specific piece, as
```

[2]https://en.wikipedia.com/git

```
                        instance, with --composer/--index options, --xml
                        option or --tiny-notation option. Adding --fragmentize
                        option with a value (ex: --fragmentize 4, max 6), you
                        will apply a "fragmentation" on input. You can add
                        --no-scramble-notes, --no-scramble-octaves and
                        arguments.
  -n, --no-scramble-notes
                        with this option, the program will not apply a
                        scramble on list of extracted notes, before create
                        chords
  -N, --no-scramble-octaves
                        With this option, the program will not apply a
                        scramble on list of extracted octaves, before create
                        chords
  -g GLITCH, --glitch=GLITCH
                        Apply a'glitch' on extracted notes and octaves; the
                        given argument is the maximum of a random operation.
                        This operation can be a choice of a 6 values: (0) The
                        generated chord will be arranged in closed position;
                        (1) the generated chord will be arranged in semi-
                        closed position; (2) the generated chord will be
                        arranged in super open position; (3) one note will be
                        separated from chord, like an one grace note; (4) two
                        notes will be separated from chord (if this chord have
                        at least, two notes); (5) apply bordadure [? translate
                        ?], and then a chord
  -m MEASURES, --measures=MEASURES
                        Select measures from a stream (corpus, xml or
                        tinynotation)
  -R, --reducte         Reducte some stream to piano staff
  -A TONAL_HARMONIC_ANALYSIS, --tonal-harmonic-analysis=TONAL_HARMONIC_ANALYSIS
                        Analyse some stream in tonal way, given some key
  -S, --show            show stream in some editor (musescore default)
  -x XML, --xml=XML     parse a musicXml file; can be a local one or http url
  -t TINYNOTATION, --tinynotation=TINYNOTATION
                        parse a tiny-notation (ex: --tiny-notation "2/16 E4 r
                        f# g=lastG trip{b-8 a g} c".
  -T TITLE, --title=TITLE
                        used to give a title. (ex: --title "My improvised
                        music"
  -a AUTHOR, --author=AUTHOR
                        used to give a author. (ex: --author "J.S. Bach"
  -r TRANSPOSE, --transpose=TRANSPOSE
                        transpose stream notes by semitones (ex.: --transpose
                        4 will transpose to a ascendent major third and
                        --transpose -4 will transpose to a descendent major
                        third
  -I, --invert          invert stream intervals by semitones. No need
                        arguments
  -L LYTEXIFY, --lytexify=LYTEXIFY
                        Create a .lytex file from a .ly file, and compiles it
                        to a .tex file. Used only in the context of
                        documentation of this software
  --plot-stream         analysis graph with PlotStream class
  --plot-histogram-pitch-space
                        analysis graph with PlotHistogramPitchSpace class
  --plot-histogram-pitch-class
                        analysis graph with PlotHistogramPitchClass class
  --plot-histogram-quarter-length
                        analysis graph with PlotHistogramQuarterLength class
  --plot-scatter-pitch-space-quarter-length
                        analysis graph with PlotScatterPitchSpaceQuarterLength
                        class
  --plot-scatter-pitch-class-quarter-length
                        analysis graph with PlotScatterPitchClassQuarterLength
                        class
  --plot-scatter-pitch-class-offset
                        analysis graph with PlotScatterPitchClassOffset class
  --plot-scatter-pitch-space-dynamic-symbol
                        analysis graph with PlotScatterPitchSpaceDynamicSymbol
                        class
  --plot-horizontal-bar-pitch-space-offset
                        analysis graph with PlotHorizontalBarPitchSpaceOffset
                        class
  --plot-horizontal-bar-pitch-class-offset
```

```
                        analysis graph with PlotHorizontalBarPitchClassOffset
                        class
  --plot-scatter-weighted-pitch-space-quarter-length
                        analysis graph with
                        PlotScatterWeightedPitchSpaceQuarterLength class
  --plot-scatter-weighted-pitch-class-quarter-length
                        analysis graph with
                        PlotScatterWeightedPitchClassQuarterLength class
  --plot-scatter-weighted-pitch-space-dynamic-symbol
                        analysis graph with
                        PlotScatterWeightedPitchSpaceDynamicSymbol class
  --plot3-d-bars-pitch-space-quarter-length
                        analysis graph with Plot3DBarsPitchSpaceQuarterLength
                        class
  --plot-windowed-krumhansl-schmuckler
                        analysis graph with PlotWindowedKrumhanslSchmuckler
                        class
  --plot-windowed-krumhansl-kesslerm
                        analysis graph with PlotWindowedKrumhanslKesslerm
                        class
  --plot-windowed-aarden-essen
                        analysis graph with PlotWindowedAardenEssen class
  --plot-windowed-simple-weights
                        analysis graph with PlotWindowedSimpleWeights class
  --plot-windowed-bellman-budge
                        analysis graph with PlotWindowedBellmanBudge class
  --plot-windowed-temperley-kostka-payne
                        analysis graph with PlotWindowedTemperleyKostkaPayne
                        class
  --plot-windowed-ambitus
                        analysis graph with PlotWindowedAmbitus class
  --plot-dolan          analysis graph with PlotDolan class
```

Alguns usos básicos

Os comandos podem ter sua forma longa e curta; por exemplo `--composer` pode ser abreviado para `-c`. Abaixo listamos alguns dos mais usados durante o trabalho

Procurar por compositores ou obras no *corpus*

:

```
./m21 --search-only --composer bach
./m21 -s -c bach -i bwv1
./m21 --search-only --index bwv1
./m21 -s -i bwv1
```

Mostrar em um *software* de editoração musical uma obra do *corpus*.

De acordo com a documentação oficial, um *software* externo pode ser usado para abrir aquivos formatados em *musicXml*. Como é um formato padrão, diferentes editores de partitura podem ser usados. Entre eles Finale e Sibelius. Neste artigo utilizamos o [MuseScore 2015]. Para alterar qual deles será usado, é necessário modificar a variável `musicxmlPath` no arquivo `~/.music21.rc` (em sistemas Unix), criado durante a instalação.

```
./m21 --show --composer bach --index bwv1
./m21 -S -c bach -i bwv1
```

Mostrar no Musescore um arquivo do tipo musicxml.

```
./m21 --show --xml bwv1.xml
./m21 -S -x bwv1.xml
```

Mostrar no Musescore uma idéia musical improvisada.

É possível escrever rapidamente uma idéia musical. Um pequeno exemplo de codificação foi realizado abaixo com um motivo da primeira invenção de Bach.

```
./m21 -show --tinynotation "2/4 r16 c d e f d e c g8 c' b c' d'"
./m21 -S -t "2/4 r16 c d e f d e c g8 c' b c' d'"
```

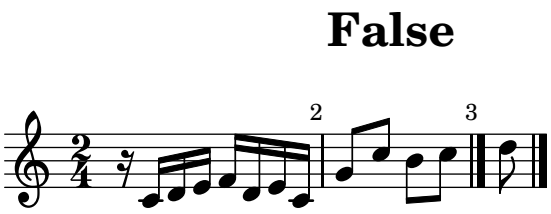

Figura 1: Motivo principal da invenção em dó maior J.S. Bach, BWV X, escrito rapidamente n terminal. Metadados necessitam de correções Fonte: autor

Mostrar no Musescore um recorte de uma obra no *corpus*.

Apresentar um fragmento de uma peça. Abaixo recortei os cinco primeiros compassos do BWV1, como na figura 2:

```
./m21 --show --composer bach --index bwv1 --measures 0 4
./m21 -S -c bach -i bwv1 -m 0 4
```

Analisar uma peça tonal

Por exemplo, os cinco primeiros compassos anteriores (quatro compassos e anacruse) podem ser analisados em qualquer tom. Considerando que o grau I é Fá, indicamos na linha de comando a representação alfabética, ou a letra **F** será apresentado o resultado como na figura 3. Qualquer outra nota utilizada para análise irá gerar um "processo proporcional", mas incorreto do ponto de vista da tradição didática musical. Certamente informações podem aparecer incorretamente e outras são redundantes. Tais critérios, ainda a serem discutidos e programados, serão importantes para desenvolvimentos futuros.

```
./m21 --show --composer bach --index bwv1 --tonal-harmonic-analysis F
./m21 -S -c bach -i bwv1 -m 0 9 -A F
```

Plotar histogramas.

Comandos de plotagem não possuem versões minificadas. Muitas delas possuem problemas de compatibilidade com o fragmento a ser analisado e a opção fornecida. Um exemplo de análise da classe de altura mais usada em um fragmento, ou peça inteira, é demonstrado na figura 4.

```
./m21 --show --composer bach --index bwv1 --plot-histogram-pitch-space
./m21 -S -c bach -i bwv1 --plot-histogram-pitch-space
```

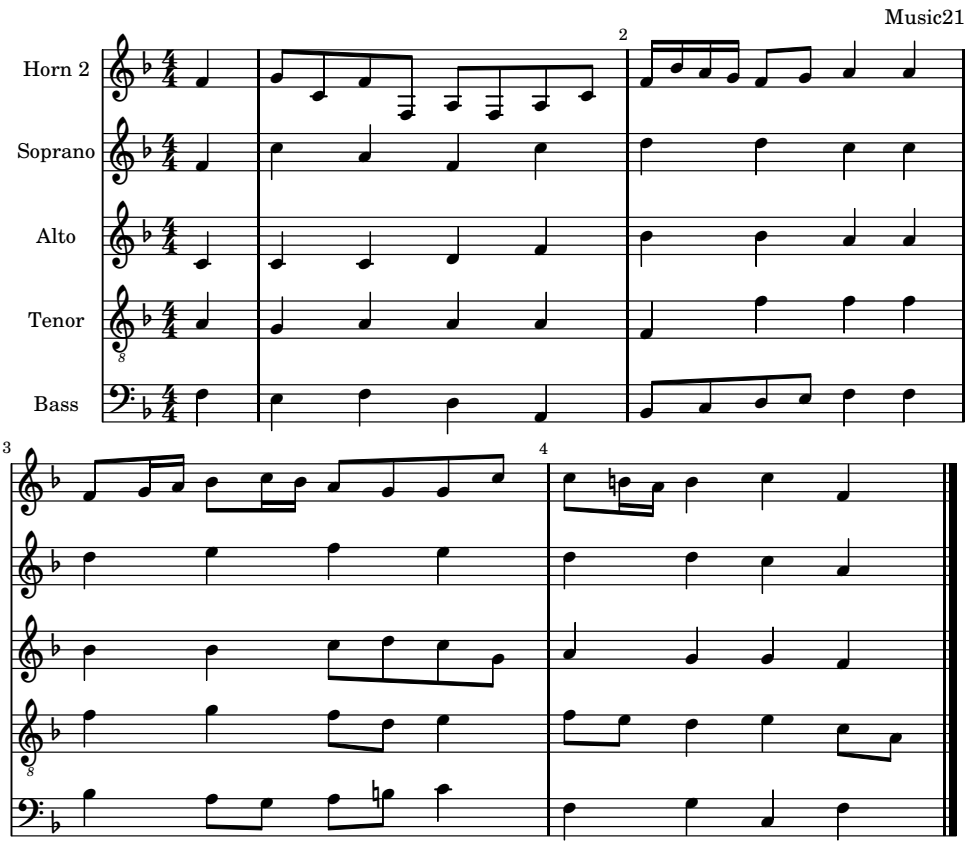

Figura 2: Extração dos cinco primeiros compassos (*measures*) do BWV1.6, utilizando o comando *./m21 -c bach -i bwv1 -m 0 4*

5. Resultados

Comandos foram utilizados para a estruturação de "Corais", pequenas peças didáticas de, por colagem de materiais da bachianos, recorte por técnica *glitch*[3], e readequações de discurso harmônico pós-tonal. Resultados foram possíveis com a implementação da a opção `--CAC` ou `-C`, como no código abaixo. No entanto esta é apenas uma *flag* indicativa que um material será reconfigurado. Inclui, até o momento, a *flag* `--glitch` ou `-g`, que recorta o material fornecido.

```
./main.py --show --CAC --composer bach --index bwv1 --glitch 2
./main.py -S -C -c bach -i bwv1 -g 2
```

5.1. BWV1

O sexto movimento de *Wie schün leucht der Morgenstern* (Cantata para a festa da Anunciação, 1725, ver figura 2)[4] foi utilizado segundo procedimentos explicados na seção 3. Foi gerado um conjunto de simultanóides[5], apresentado na figura 5.

Cada bloco harmônico são notas de um determinado compasso, escolhido ao acaso pelo programa, comprimidas em um único evento. Algumas notas podem ter sido omitidas por erros de codificação no *script* Python. Mas isso é coerente com o princípio estético. É também interessante observar uma direcionalidade da tessitura, que vai da região média aos graves, percebida após repetidos usos do comando descrito.

Realizei algumas intervenções com este material:

[3]Aplicação de recortes indeterminados sobre um material pré existente.

[4]Vídeo disponível em `https://www.youtube.com/watch?v=POe2fBjbswA`.

[5]Utilizo aqui a nomenclatura de [Koellreutter 1987] para identificar blocos harmônicos.

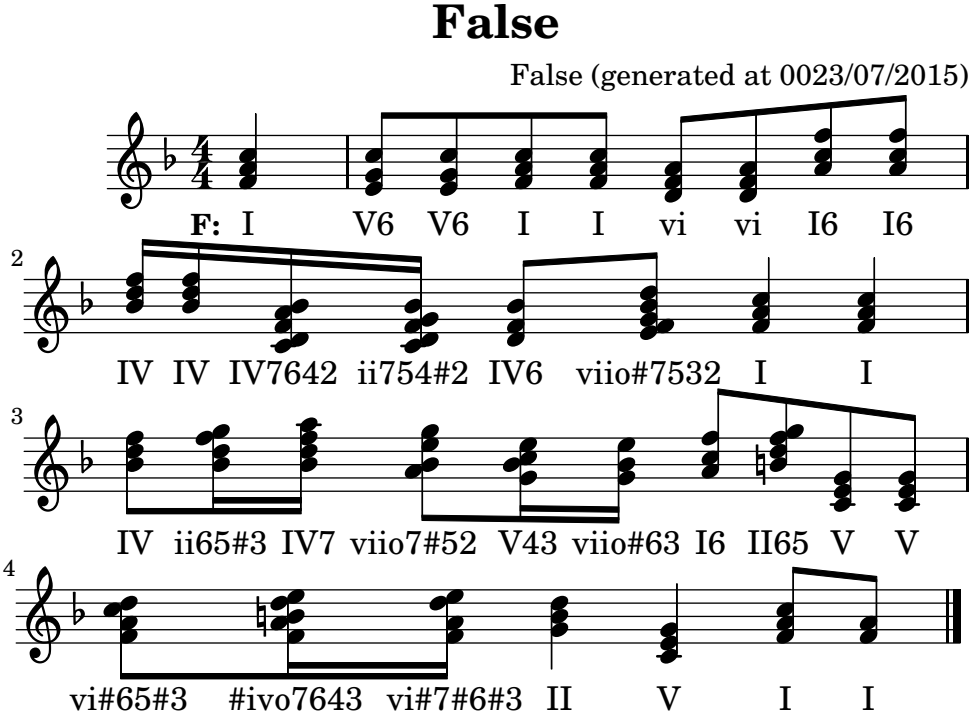

Figura 3: Análise harmônica automática dos cinco primeiros compassos, na tonalidade de Fá Maior

- manutenção da ordem dos blocos;
- escolha de pontos que delimitam fraseados (silêncio como articulado de frases;);
- observação de densidades na quantidade de notas como fator para o fim do fraseado;
- dinâmicas como um dispositivo de ênfase do fraseado harmônico;
- a modificação de uma oitava de dois si bemoís para um intervalo de sétima ou nona de si bequadro e si bemol compasso, inserindo um intervalo de sétima com outro sib;
- deslocamento ou subtração de elementos da figura 5.

A peça finalizada está na figura 6.

As ferramentas analíticas auxiliaram a observação de semelhanças e diferenças entre uma peça original e a peça variada. Por exemplo, com BWV1 de J.S.Bach, o histograma da figura 4 revela fatos analíticos comuns, como uma ênfase do *pitch class* 5, Fá, I grau; e depois *pitch class* 0, Dó, V grau, seguido de outras classes, como Lá (III grau) e Sol (ii grau ou V/V); outros, como Ré (vi grau) e Si bemol, possuem a mesma quantidade. Por último Mi (vii grau) e si natural possuem menor número, talvez como dispositivos cadenciais(como por exemplo na "modulação" Fá⇒Dó do quinto compasso da figura 2).

A partir da artesania do material pré-composicional gerado por fragmentação do BWV1, notei, no histograma da figura 7 que algumas proporções foram mantidas de maneira aproximada. A quantidade de eventos diminui drasticamente. Mesmo com a segmentação da peça, um centro tonal ainda pode ser notado (embora de maneira bastante ambígua).

6. Conclusão

Duas funcionalidades do Music21 foram observadas: Musicologia Assistida por Computador e Composição Assistida por Computador (CAC). Da documentação e do

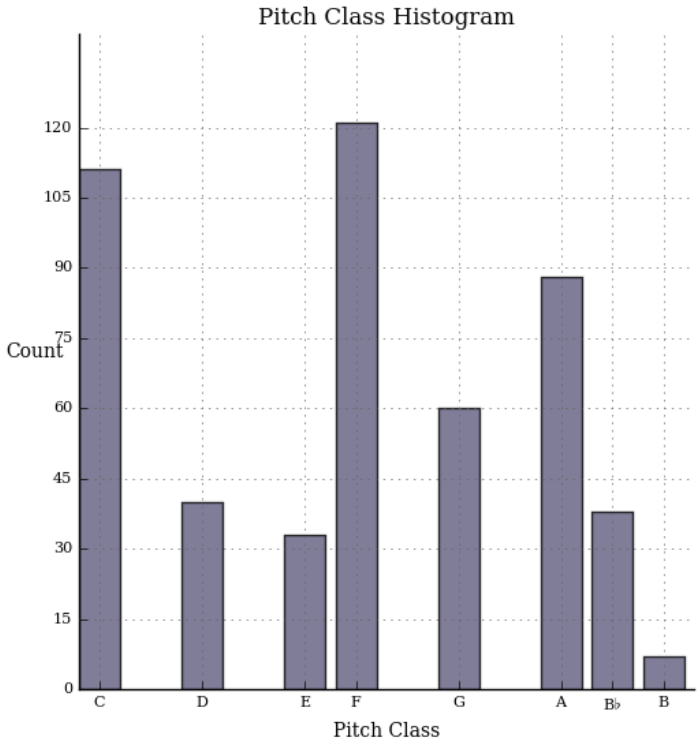

Figura 4: Histograma de *pitch-class* do BWV1.6. Fonte: autor utilizando Music21.

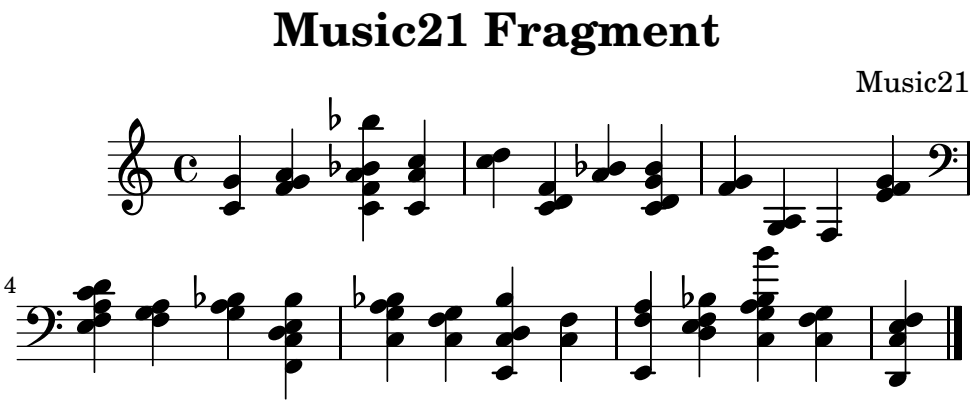

Figura 5: Sequência de simultanóides gerados. Fonte: Autor.

corpus do *software*, programei rotinas que lidassem com aquilas tarefas que considerei demasiadamente laboriosas. Para isso proponho comandos que podem auxiliar em tarefas cotidianas, composicionais ou analíticas.

Da rotina para composição/análise, cito a busca no corpus, de uma obra específica ou obras pelo nome do compositor. Das rotinas analíticas, enumero *a)* identificação dos graus *b)* estruturação intervalar de blocos harmônicos *c)* plotamento de histogramas de classes de altura. Das rotinas composicionais, enumerei procedimentos para uma composição por fragmentação (*glitch*): *d)* colagem de fragmentos de uma partitura *e)* compressão de melodias em blocos harmônicos dos fragmentos resultantes *f)* troca de oitavas com as notas deste bloco *g)* possível fragmentação do bloco resultante em figuras ou arpejos Existe adicionalmente um bug que ocorre por fatores lógicos, próprio da atividade de programação realizada, mas ainda não resolvido. Possivelmente o compositor pode lidar com dados nulos. Caso a peça de entrada seja muito grande, pode acontecer do programa não responder.

para Glerm Soares

Coral #1

Após uma extração de alturas do BWV1 em 08/7/2015

Bach / music21

Guilherme Lunhani

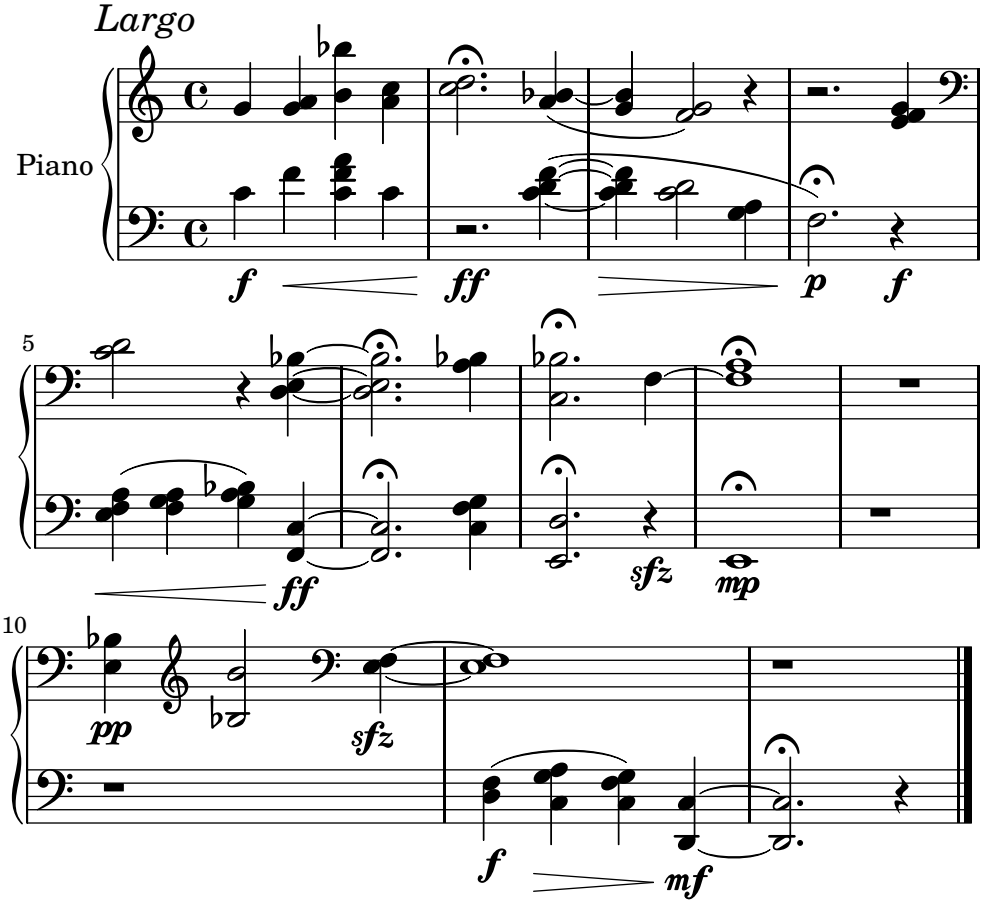

Figura 6: Peça resultante das intervenções. Fonte: autor.

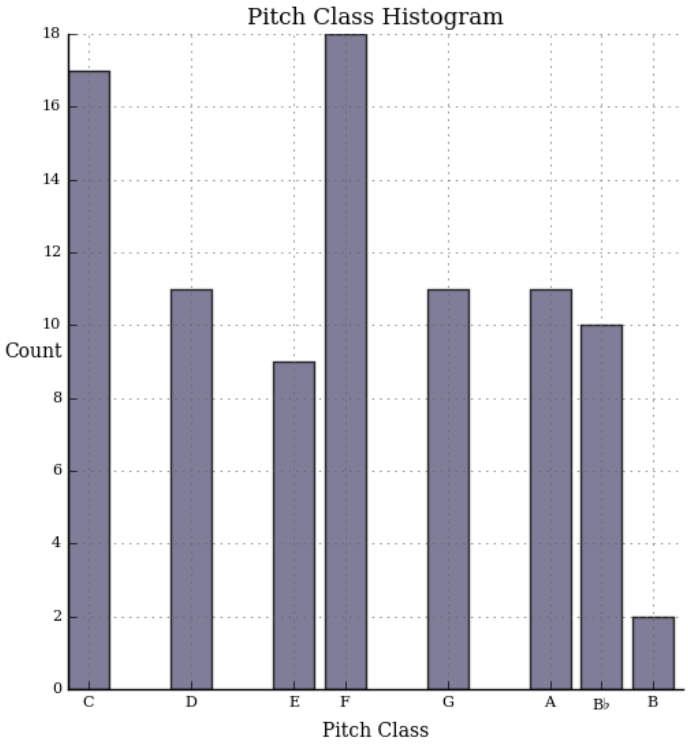

Figura 7: Histograma de *pitch-class* da peça feita, utilizando o comando *./main.py -x examples/bwv1/bwv1.xml –plot-histogram-pitch-class*

Foi observado uma versatilidade de materiais pré-composicionais gerados, bem como a possibilidade de improvisar com estes materiais. O exemplo apresentado na seção 5 foi feito em algumas horas. Com isso espero oferecer aos docentes e discentes em composição musical uma ferramenta alternativa para atividades diáticas em processos criativos.

7. Planos Futuros

Correção do *bug* e implementação de um novo comando que fragmente várias obras do corpus em um único material pré-composicional (comando `--poop` ou `-p`). Continuação de novas composições para o ciclo.

8. Agradecimentos

Ao Guilherme Rafael Soares por apresentar a biblioteca *music 21*. Aos desenvolvedores do *musescore* e *lilypound*. FAPEMIG pelo financiamento da pesquisa.

Referências

Cascone, K. (2000). The aesthetics of failure: 'post-digital' tendencies in contemporary computer music. *Computer Music Journal*.

Cope, E. D. (2008). Prefacio a OM composer's book vol. 2. In *OM Composer's Book*, volume 2, pages ix–xv. Editions Delatour.

Koellreutter, H. J. (1987). *Introdução à estética e à composição musical contemporânea;*. Movimento, 2 edition.

Lilypond (2015). Lilypond.

MuseScore (2015). Musescore.

Music21 (2015). Music21.

Soares, G. R. (2015). Luteria composicional de algoritmos pós-tonais v1.1final.