MistyR: Solving Music Theory 1

Taqi

MistyR Package

Written for MATH241. Needed to count frequencies of scale degrees to find summary statistics mutated from these frequencies, which we called "tonal features". The summary statistics were computted then faceted by composer/genre, from which we gain inferential insight. Here are the features we came up with:

$$Consonance : \{\hat{3}, \hat{5}, \hat{6}\}$$

$$Dissonance : \{\hat{2}, \hat{7}\}$$

Introduction

Notes: Letters and Incidentals

In Music Theory I, we are taught how to write down chords in any key we are asked to. We are able to do this because behind every chord, there are intervals that build it up. As such, to solve music theory computationally, we need to boil down the task to its most basic components and reduce it to a string manipulation puzzle.

We will dive right into it. First, we need to know what keys have **black key sharps** and **black key flats**. Look at the piano below.

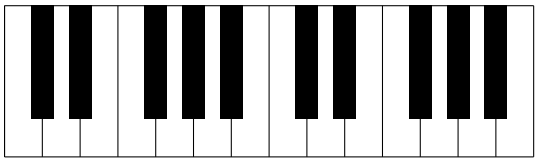

Figure 1: The Chromatic Scale

Now, the piano is not labelled, but assuming that you've memorized the 7 note names of the C Major scale on the piano, you should be able to answer this easily. But a little review cannot hurt. The 7 keys of the piano are each labelled by a letter. We will call these letters **the musical alphabet** which is essentially the first 7 letters of the Latin alphabet wrapping back (hinting at the octave).

```
MUS_ALPH
```

```
## [1] "A" "B" "C" "D" "E" "F" "G"
```

This is peeking way in the future, but any 7-note scale (defined as set of notes that partition the octave with respect to some root/tonic) will probably be easy expressible with *ordered pairs* consisting of **all the letters of the musical alphabet** and **an incidental**. For example, the note $B^\flat$ can be resolved as the ordered pair $(B, \flat)$. If there is no incidental, it is understood as a natural ($\natural$). So, $C = (C, \natural)$

Now, once you know the 7 notes, you look at its neighbouring notes and record which notes have black key flats/sharps. do so, you then make a dataframe for those two sets of notes. Note that any note has a "flat" or a "sharp", by "having a flat/sharp", we really mean "having a black key flat/sharp".[1].

```
HAS_FLATS
```

```
## [1] "A" "B" "D" "E" "G"
```

```
HAS_SHARPS
```

```
## [1] "A" "C" "D" "F" "G"
```

Gathering all that information in a nice "musical alphabet" hash table which we call `ALPHABET`:

```
ALPHABET
```

```
##   LETTER HAS_FL HAS_SH
## 1      A   TRUE   TRUE
## 2      B   TRUE  FALSE
## 3      C  FALSE   TRUE
## 4      D   TRUE   TRUE
## 5      E   TRUE  FALSE
## 6      F  FALSE   TRUE
## 7      G   TRUE   TRUE
```

That being said, let's now consider the study of intervals.

Intervals

That being said, now we can easily construct intervals a given number of semitones above a given note just by using the hash table above! This is easy for *white notes* at least, which is what we will solve first. The answer is also visual, if we choose two white keys, we just count how many musical letters or "steps" we take chromatically; we take the number of white notes (just the number of letters in the alphabet, by construction) and the black keys in between. When we do so, we obtain a function like this:

```
# Find the white-key range number of semitones between a note and a given scale degree
.WKsemitones <- function(note, degree){
  # Get the alphabet range for the chosen note and degree
  note_idx <- note_index(note)
  range <- .ALPH_RANGE(note_idx, degree)
  letter_range <- ALPHABET[range,]
  # Get the number of flats in range
  FLATS <- sum(letter_range[2:degree,]$HAS_FL)
  # Get the number of white keys in the range (exclude tonic)
  WHITES <- degree - 1
  # Return the number of semitones
  semitones <- FLATS + WHITES
  semitones
}
```

Its usage, and some helper functions:

```
# Convert a note into a note index in the musical alphabet
note_index <- function(note){ which(MUS_ALPH == note) }
```

[1]Otherwise, white key flats/sharps are known as **enharmonics** or enharmonic spellings of the note. This is critical to note because remember, we are trying to reduce the scale degree question to a matter of string manipulation; enharmonics cause some trouble so it is important to keep them in mind. That being said, the context in which enharmonics usually arise is when the key is "very far from C". Some spelling enharmonics are thus less common because if available, the simplest spelling will be used and double-sharps/flats are avoided when possible

```
# Generates a wrapped range of indices on the musical alphabet of a given a length
.ALPH_RANGE <- function(i, length){
  range <- i:(i + (length - 1))
  range <- mod7(range - 1)
  range + 1
}
```

We compute the number of semitones between A and F:

```
.WKsemitones("A", 6)
```

```
## [1] 8
```

We compute the number of semitones between C and B:

```
.WKsemitones("C", 7)
```

```
## [1] 11
```

Now here is the clever part, since we obtained the results for the white keys, we can also use them for the black keys if we are clever to reduce the problem to a white key problem. To be concrete, say we want the note which is 5th scale degree of $F^\sharp$, to do so, we obtain the 5th scale degree for F, then sharpen it!

Furthermore, we cleverly exploit the encoded incidental information in the quality of our interval. For instance, if we want the $\flat\hat{3}$, or the minor third, we simply flatten $\hat{3}$. Below is our interval construction function:

```
.intervalNote <- function(tonic, degree){
  # Coerce the degree as a numeric
  degree_n <- parse_number(degree)
  # Get the number of semitones from the scale degree
  semitones <- SCALE_DEGREES[[degree]]
  # Get the target note letter (by stepping up n - 1 times)
  white <- tonic %STEP+% (degree_n - 1)
  # Find the semitones spanned
  semitones_white <- .WKsemitones(tonic, degree_n)
  # Use incidentals to find the difference
  incidental <- .incidentalSemitones(semitones - semitones_white)
  # Paste incidental the correct target note
  note <- paste(white, incidental, sep = "")
  # Return the interval
  note
}
```

With the helper function:

```
# Return either the number of semitones of a incidental or incidental from semitones
.incidentalSemitones <- function(x, inverse = T){
  INCIDENTAL <- c("&", "-", "", "#", "x")
  VALUE <- c(-2,-1,0,1,2)
  if(inverse) { return(INCIDENTAL[which(VALUE == x)]) }
  VALUE[which(INCIDENTAL == x)]
}
```

And the scale degree dataframe:

```
SCALE_DEGREES
```

```
## $`1`
## [1] 0
##
```

```
## $b2
## [1] 1
##
## $`2`
## [1] 2
##
## $b3
## [1] 3
##
## $`3`
## [1] 4
##
## $`4`
## [1] 5
##
## $`#4`
## [1] 6
##
## $b5
## [1] 6
##
## $`5`
## [1] 7
##
## $`#5`
## [1] 8
##
## $`6`
## [1] 9
##
## $b7
## [1] 10
##
## $`7`
## [1] 11
```

Resulting in the functionality:

```
.intervalNote("C", "5")
```

```
## [1] "G"
```