David Huron
School of Music
1866 College Road
Ohio State University
Columbus, OH 43210 USA
huron.1@osu.edu

Music Information Processing Using the Humdrum Toolkit: Concepts, Examples, and Lessons

Humdrum is a general-purpose software system intended to assist users in a variety of music-related applications. This article provides an introductory tour of Humdrum emphasizing three goals: to identify the basic elements, operations, and organization of Humdrum; to illustrate the range of applications through a series of tutorial examples; and to identify a number of lessons from Humdrum that might benefit future music software designers.

Humdrum's capabilities are quite broad, so it is difficult to describe concisely what it can do. Since its inception, the principal users of Humdrum have been systematic musicologists, theorists, ethnomusicologists, music librarians, historians, music cognition researchers, and composers. Five features have accounted for this broad interest: the ability of users to concoct or tailor unique representations that pertain to the user's specific interests; a flexible set of analytic and processing tools that can be applied to both established and user-defined representations; a coherent and extensible system for representing reference-related metadata; ease of connectivity to other existing software; and availability of a large volume of high-quality encoded materials.

In Humdrum, new musical representations are easily created. The opportunity to craft representations for specific tasks has led to the development of innumerable representations. For example, Humdrum users have concocted representations for Bugandan xylophone music, Cajun button accordion tablatures, square notation, Persian Ney music, Benesh dance notation, running acoustic spectra, and track index markers for compact discs. For many projects, it is common to generate intermediate or "throw-away" representations that are used only for a single task. For example, in perceptual research, collected data (such as listener responses) are commonly encoded in the same document that contains the stimulus materials.

In addition to specific representation schemes for user "data," Humdrum allows users to define their own types of reference-related information. Conventional reference information includes an artist's name, title, date of performance, copyright status, etc. However, Humdrum allows users both to extend and omit such metadata in ways that maintain compatibility without compelling users to encode task-irrelevant materials. For example, a ballet scholar might define reference information identifying the transcriber of a Laban dance notation while remaining compatible with conventional cataloguing systems. At the same time, users can still process materials with incomplete or absent reference information.

Although Humdrum does not provide its own sound generation or graphic notation capabilities, a number of translators exist that connect Humdrum to existing applications. These include translations to MIDI, Csound (Vercoe 1986), the Finale Enigma format, the Score notation system (Smith 1972; Kornstädt 1996), and the Mup music text formatter (available from www.arkkra.com). In addition, materials can be translated to the Humdrum format from MIDI, the Finale Enigma format, MuseData (Hewlett 1997), DARMS (Erickson 1976; Hall 1997), and the RISM Plaine and Easie representation (Howard 1997).

All Humdrum file formats are ASCII text. This facilitates viewing and editing data, increases cross-platform portability, and eases the writing of task-specific software. The plain text format also facilitates reformatting data for special-purpose uses such as graphics programs or statistical packages.

Musical materials encoded in the Humdrum format span five centuries and six continents. Encoded materials include complete musical works as

Computer Music Journal, 26:2, pp. 11–26, Summer 2002
© 2002 Massachusetts Institute of Technology.

well as collections of themes and incipits. As of 2001, over 40,000 musical works had been encoded in various levels of detail. Some encodings are simple monophonic melodies; others include full orchestral scores including such details as stem-direction and beaming. In addition to the MIDI format, the Humdrum ''kern'' representation is one of the principal distribution formats for the high-quality electronic editions produced by the Center for Computer Assisted Research in the Humanities (www.ccarh.org). An extensive collection of non-Western musics has been encoded; an incomplete alphabetic sample includes Aleutian, Brazilian, Chinese, Dutch, Ethiopian, Fijian, German, Haitian, Indonesian, Japanese, Khmer, Lakota, Mandan, Namibian, Ojibway, Peruvian, Quebecois, Russian, Scottish, Thai, Usarafus, Venda, Xhosa, Yoruba, and Zulu musics. (Huron 1992). Regarding Humdrum's analytic capability, a selection of tools will be discussed next.

Humdrum Syntax

Humdrum data are organized in two-dimensional tables like a spreadsheet. Columns of data can be defined to represent different types of information. Successive lines (records) represent successive moments, so time passes as one moves down the page. The example here encodes an ascending major scale. The leftmost column represents pitch using the International Standards Organization (ISO) pitch designations. The rightmost column represents piano fingerings. The columns are separated by a single tab character. Each column of data begins with an interpretation that identifies the type of data being represented, and ends with a path terminator.

```
**pitch    **fingering
C4         R1
D4         R2
E4         R3
F4         R1
G4         R2
A4         R3
B4         R4
C5         R5
*-         *-
```

The ISO pitch representation is predefined in Humdrum, whereas the fingering representation has been concocted for this example. Humdrum encodings can consist of any number and length of columns.

Unlike the columns in a spreadsheet, Humdrum columns can exhibit complicated paths through the document. Columns can join together, split apart, exchange positions, stop in mid-table, or be introduced in mid-table. Because columns of Humdrum data can roam about the table in a flexible way, they are referred to as *spines*.

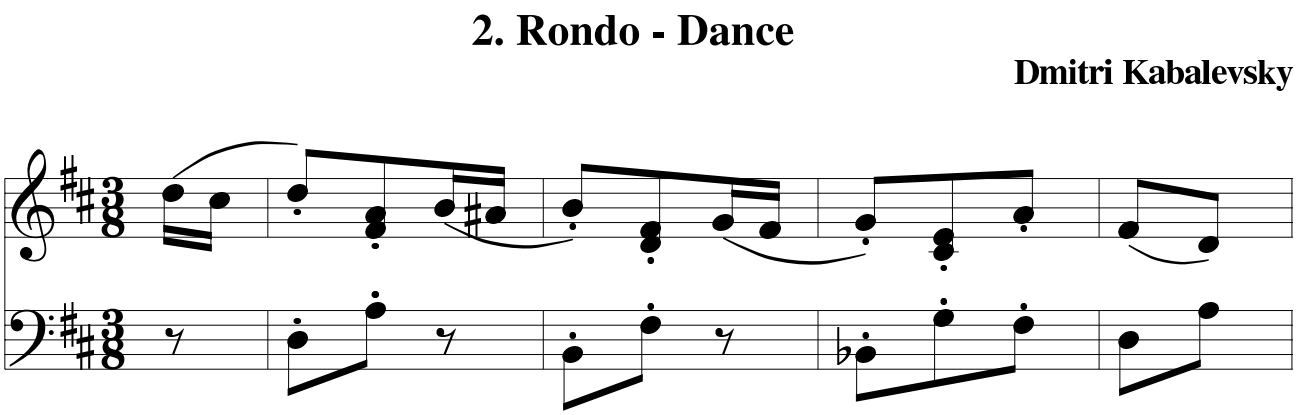
Humdrum spines are formally labeled using interpretations (tokens beginning with an asterisk). Exclusive interpretations (double asterisks) identify the basic type of data being represented. Tandem interpretations (single asterisks) provide supplementary labels or tags that can clarify the state of the data. Any number of tandem interpretations can be added anywhere in a spine. In addition, Humdrum provides ways of adding running commentaries. Comments might pertain to the whole encoding, to a given row or spine, to a given data cell, or to a particular item of information within a cell (such as a single letter or digit). Specially formatted comments are used to encode reference- or cataloguing-related information. All types of comments are designated by a leading exclamation mark.

The most common Humdrum files encode the ''score'' of a complete work or movement. Typically, musical notes are encoded in the various cells of the table; the most common use of a spine is to represent a single musical part or instrument.

Humdrum includes some thirty predefined representational schemes. Predefined representations include many common forms of music-related information such as frequency, pitch, cents, scale degree, melodic and harmonic intervals, duration, dynamics, decibels, harmonic function, pitch-class sets, lyrics, etc. However, spines can be used to represent anything, and users can concoct their own representations simply by writing an exclusive interpretation with a unique identifier.

Of the predefined Humdrum representations, the most popular is *kern*, a scheme intended to represent the basic or core musical information of notes,

Figure 1. Musical excerpt from Kabalevsky.



durations, rests, barlines, etc. (Huron 1997). (The kern representation gets its name from the German word for "core.") By way of illustration, the following example encodes the accompanying musical excerpt of Figure 1 using the `**kern` representation.

```
!!!COM: Kabalevsky, Dmitri
!!!OTL: 2. Rondo-Dance
**kern      **kern
*staff2     *staff1
*clefF4     *clefG2
*k[f#c#]    *k[f#c#]
*M3/8       *M3/8
8r          (16dd
.           16cc#
=           =
8D'         8dd')
8A'         8f#' 8a'
8r          (16b
.           16a#
=           =
8BB'        8b')
8F#'        8d' 8f#'
8r          (16g
.           16f#
=           =
8BB-'       8g')
8G'         8c#' 8e'
8F#'        8a'
=           =
8D          (8f#
8A          8d)
*-          *-
```

The representation mirrors the two-dimension structure of the score, rotated clockwise by 90 degrees. The tandem interpretations in this example can be readily deciphered as representing staves, clefs, key, and meter signatures. Reciprocal numbers are used to indicate durations. Pitches are represented using lower-case letters (middle C and above) and upper-case letters (below middle C). Octaves are represented using a system of letter repetition. (See Huron 1995 for a complete description.) In each complete measure, note the appearance of a "multiple stop." Using the space character as a delimiter, more than one data token can appear in a given spine. A single staff can be represented by more than one spine, so complex polyphonic and other textures can be represented.

It is important to understand that there is nothing special about the `**kern` representation. For example, in using the pitch letter names A–G, this particular representation shows a bias towards English-speaking users, while the use of numbers for durations shows an American bias. However, other predefined Humdrum representations allow users to represent music using French "fixed-Do" solfege or the German system of pitch naming (e.g., H = B-natural, Es = S = E-flat, etc.) Encodings are easily translated from one representation to another. Users of Humdrum have tailored other representations that better reflect special representation requirements, from Australian aboriginal music to Zydeco music.

Humdrum Tools

The Humdrum Toolkit contains roughly 70 interrelated yet stand-alone software tools. Each tool can be invoked individually or in connection with other tools. Typically, the tools are invoked in a command-line shell or as statements in a program or script. Humdrum commands conform to the POSIX 2 portability standard (IEEE standard 1003.1, available online at standards.ieee.org/catalog/olis/posix.html) and thus can be used on all popular operating systems. Any data that conforms to the Humdrum syntax can be manipulated using the Humdrum tools. Because the tools can be connected with each other (and can also be connected with non-Humdrum tools), there are many ways to manipulate Humdrum data.

Some Sample Commands

One group of tools is used to extract or select sections of data. Vertical spines of data can be extracted from a Humdrum file using the `extract` command. For example, if a file encodes four musical parts (encoded in four spines), then the `extract` command might be used to isolate one or more given parts. The command

```
extract -f 1,3 filename
```

will extract the first (leftmost) and third column or spine of data. Often it is useful to extract material according to the encoded content without regard to the position of the spine. For example, the following command will extract all spines containing a label indicating the tenor part(s):

```
extract -i '*Itenor' filename
```

Instruments can be labeled by ''instrument class'' and can thus be extracted accordingly. The following command extracts all of the woodwind parts:

```
extract -i '*ICww' filename
```

Any vocal text can be similarly extracted:

```
extract -i '**text' filename
```

Or, if the text is available in more than one language, a specific language may be isolated:

```
extract -i '*LDeutsch' filename
```

As noted above, users are free to define their own interpretations. For example, if a user has coded electro-encephalographic data in a spine (denoted by, say, `**EEG`), the pertinent data can be extracted using the same syntax:

```
extract -i '**EEG' filename
```

Segments or passages of music can be extracted using the yank command. Segments can be defined by sections, phrases, measures, or other any user-specified marker. For example, the following command extracts the section labeled ''Trio'' from a minuet and trio:

```
yank -s Trio -r 1 filename
```

Suppose a user wanted to select the material in measures 114 to 183. In the following command, the-n option specifies a regular expression (=) which might be used in this case as a barline marker. The-r option identifies the range of, in this case, the numbered labels on the barlines:

```
yank -n =-r 114-183 filename
```

In a representation (like `**kern`) that uses curly braces to represent phrases, selecting the penultimate phrase in the work might be achieved as follows:

```
yank -o {-e }-r '$-1' filename
```

Whenever appropriate, users can define general-purpose patterns using the well-known regular expression syntax. This means that the tools can operate in ways that are sensitive to the encoded material, even without any ''knowledge'' of the representational convention.

Some tools translate one representation into another. For example, the mint command generates melodic interval information. The `mint` command only operates on pitch-related data; any non–pitch-related data are ignored.

For example, if we apply the `mint` command to the previous ascending scale, the resulting output would transform the `**pitch` spine while leaving the `**fingering` spine unaffected:

```
**mint     **fingering
[C4]       R1
+M2        R2
+M2        R3
+m2        R1
+M2        R2
+M2        R3
+M2        R4
+m2        R5
*-         *-
```

Two or more commands can be connected into a pipeline. The following command locates all melodic tritones—including compound (octave) equivalents:

```
mint -c filename | egrep -n '((d5)|(A4))'
```

Because the Humdrum tools are stand-alone commands, non-Humdrum tools can be interposed at any point in the processing. For example, the `egrep` command in the above pipeline is a common utility associated with the UNIX operating system but available on many other systems as well.

Depending on the type of translation, the resulting data can be searched for different things. The following command identifies French sixth chords by first translating the input into a predefined scale degree representation. When provided with scale-degree data, the regular expression `6-.*4+` means "find any lowered sixth scale degree that is concurrent with a raised fourth scale degree. The regular expression `2` simply ensures that the sonority also contains the second scale degree:

```
sdeg file | extract -i '**deg' | ditto | grep
 '6-.*4+' | grep 2
```

The following command locates all sonorities in the music of Machaut where the seventh scale degree has been doubled:

```
deg -t machaut* | grep -n '7[^-+].*7'
```

This command counts the number of phrases that end on the subdominant pitch:

```
deg filename | egrep -c '(}.*4)|(4.*{)'
```

The next command identifies all scores in the current directory whose instrumentation includes a tuba but not a trumpet:

```
grep -sl '!!!AIN.*tuba' * | grep -v 'tromp'
```

Predefined pitch and duration representations can be translated to MIDI using the Humdrum `midi` or `smf` commands; the `perform` command provides a simple command-line MIDI player. For example, the following pipeline generates a MIDI performance of the second phrase in the oboe part:

```
extract -i '*Ioboe' filename | yank -o {-e }
 -r 2 | midi | perform
```

Similarly, the following command will play the first and last measures from a section marked "Coda" at half the notated tempo from a file named `Cui`:

```
yank -s 'Coda' Cui | yank -o ^=-r 1,$ | midi
 | perform -t .5
```

The `ms` command generates input for the Mup notation program written by John Krallmann and William Krauss. The following pipeline generates graphical PostScript output of a notational rendering of the first 20 measures of just the string parts:

```
yank -o =-r 1-20 filename | extract -i
 '*ICstr' | ms
```

Robert Gjerdingen and Po-Yan Tsang have written tools that translate between Humdrum and the Finale Engima format. Other output tools can be used to generate Csound score files.

The context command provides a simple way of searching for patterns in particular contexts. The command

```
context -n 2
```

changes a sequence of tokens, such as the following Roman numeral harmonies:

```
**harm
I
V
iii
VI7
ii;
*-
```

into a sequence of paired tokens, or "digrams," as follows:

```
**harm
I V
V iii
iii VI7
VI7 ii;
.
*-
```

In effect, the resulting representation indicates that a I chord is followed by a V chord; a V chord is followed by a iii chord, and so on. An inventory of the most common two-chord progressions in Bach's chorale harmonizations can be created by simply sorting and counting the number of unique data records.

```
extract -i '**harm' chorales* | context
 -n 2-o = | sort | uniq -c | sort -n
```

An inventory of three-chord progressions can be generated by changing the -n 2 option to -n 3.

Earlier we saw that the `extract` command can be used to isolate one or more spines from a file. The reverse process is achieved using the Humdrum `assemble` command, which amalgamates individual spines into a single file. An obvious use of this would be to assemble a full score from individual parts. However, `assemble` is more commonly used to amalgamate different kinds of information in a single document. Multiple files can be aligned simply by specifying the inputs and output:

```
assemble degree.file interval.file
 > filename
```

Suppose we would like to determine whether descending minor seconds are more likely to occur as Fa–Mi or Do–Ti. We can use the `mint` command to characterize melodic intervals and the `solfa` command to characterize scale degrees:

```
mint melody > file1
solfa melody > file2
```

We can then use `assemble` to amalgamate the two kinds of representations and use `grep` to search for the appropriate combination of interval and scale degree:

```
assemble file1 file2 | grep -c '-m2.*mi'
assemble file1 file2 | grep -c '-m2.*ti'
```

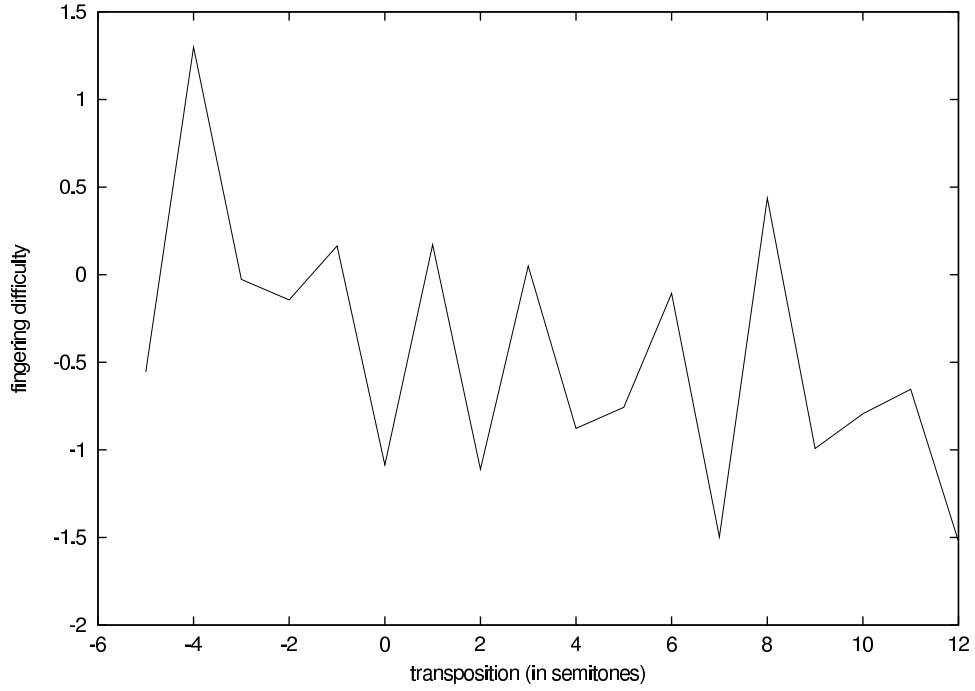
Innumerable variants of this technique are possible. Several different representations of the same music can be coordinated in a single file. For example, a user might search for all instances of a descending minor third in the soprano voice, in preparation for a suspension (occurring in either the alto or tenor), leading to a half cadence, with the soprano terminating on a tonic pitch approached from below. (Such complicated examples are discussed in detail in the *Humdrum User Guide* (Huron, 1999).)

A more general tool, `humsed`, implements a stream editor conforming to the syntax of the popular `sed` stream editor used on UNIX systems. Both `sed` and `humsed` are Turing-complete, which means that any manipulation of strings of characters that can be achieved using a computer can be done using `humsed`. In practice, stream editors such as `humsed` are used for more modest functions like simple substitutions. For example, a user might create a simple script to rewrite the pitches in a trumpet piece as valve combinations.

At the University of Waterloo, Jonathan Berec has collected data from trumpet performers estimating the degree of difficulty for various finger/valve combinations. On a scale from 1 (least difficult) to 10 (most difficult), the valve/finger sequence 1–3 followed by 2 was rated by performers as 9.7 in difficulty, whereas the sequence 1–2 followed by 2 was rated 5.8. Having used `humsed` to translate pitches to valve combinations, the `context` command can be used to assemble pairs of successive valve-combination changes. A subsequent `humsed` script can then be used to replace valve-combination changes by their difficulty ratings. In other words, any monophonic musical score in the range of the trumpet can be processed to generate a new spine indicating the degree of moment-to-moment fingering difficulty:

```
**kern    **valves  **difficulty
B4        2         0
D#5       2         0
A4        1-2       3.0
G#4       2-3       6.0
A4        1-2       5.0
F4        1         1.5
C4        0         1.0
*-        *-        *-
```

Figure 2. Graphical representation of the fingering difficulty for Herbert Clarke's Stars in a Velvety Sky.



When the output is piped to a sound synthesis program such as Csound, the "difficulty" information might be used to modify performance parameters so that difficult fingerings are associated with hesitation, fluctuations in intonation, or other acoustic cues that add realism to the performance.

A composer might try different transpositions to determine the easiest (or most difficult) key for a work. For example, Figure 2 shows a graph of the average fingering difficulty for Herbert Clarke's *Stars in a Velvety Sky*, a virtuoso trumpet work composed by a skilled trumpet performer. The work was methodically transposed covering all keys in which the work might be theoretically played. In general, the average fingering difficulty tends to decrease as the work is transposed higher in pitch owing to the greater variety of alternative fingerings available for high notes compared with low notes. Apart from this general trend, significant changes in average fingering difficulty occur from key to key. In works composed by trumpet virtuosi, Mr. Berec and I have found a marked tendency for the lowest fingering difficulty to correspond with the key in which the work was actually written. This pattern was not evident in trumpet works written by non-trumpet players.

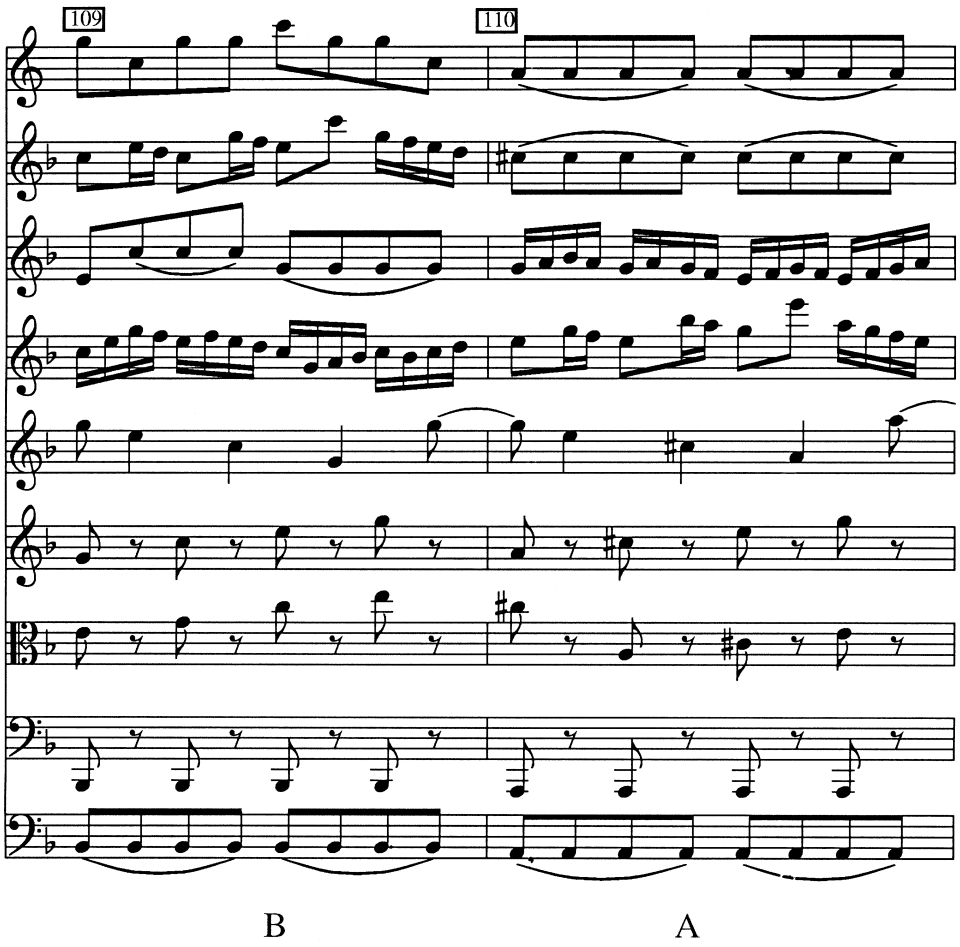
The values for Figure 2 used the Humdrum `trans` tool, which allows passages to be transposed in various ways. The `trans` tool permits independent diatonic and chromatic offsets, so it is possible to transpose works into different modes as well as different keys. For example, the following command causes a simple diatonic shift, and so performs a Joplin rag in D Dorian rather than C major:

```
trans -d 1 Joplin | midi | perform
```

Two-Dimensional Patterns

Tools such as `grep` allow users to carry out string searches where the pattern of interest can be found on a single line. The two-dimensional structure of Humdrum representations means that important musical patterns can stretch over many lines or records. The Humdrum `patt` command provides a two-dimensional equivalent to `grep` that allows users to search for patterns spanning multiple records.

Figure 3. A formerly unknown instance of B–A–C–H in Bach's Concerto No. 2.



Consider, for example, the task of searching for the pitch pattern B–A–C–H. As a sequential pattern, the Humdrum syntax will cause such a pattern to span several lines. The `patt` command allows users to specify a multi-record pattern template. A suitable template might be as follows:

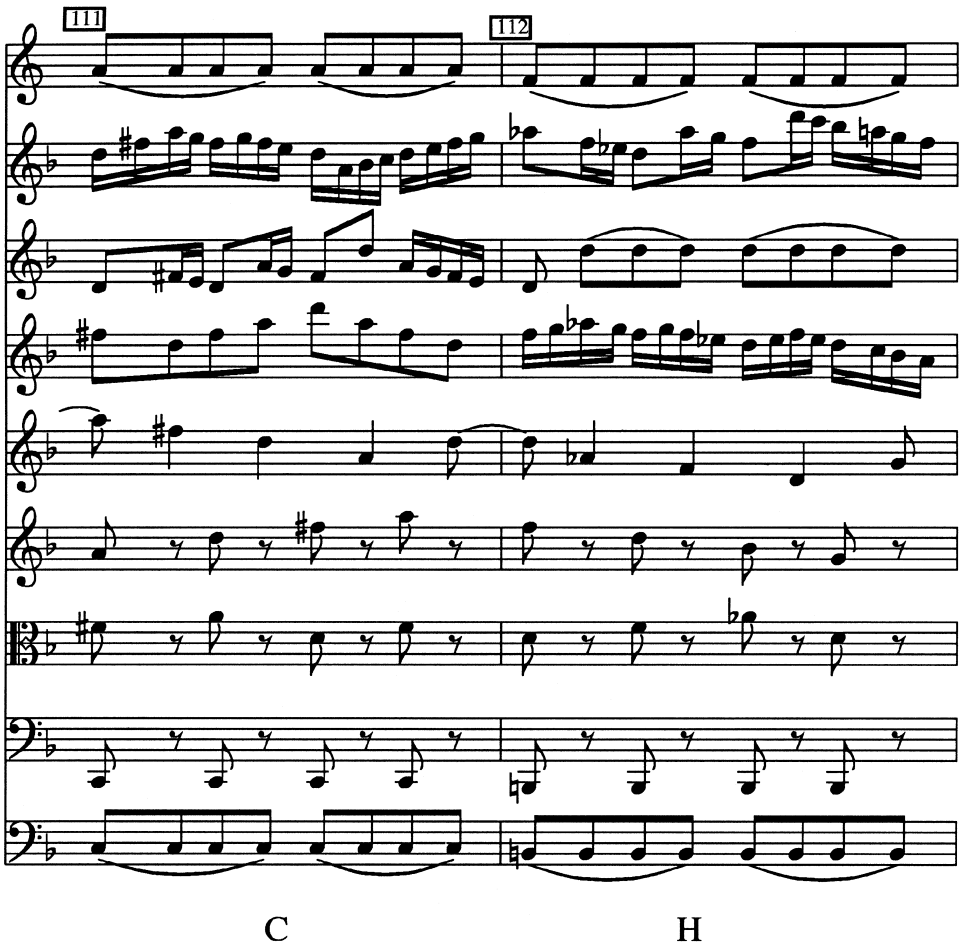
```
[Bb]  +
[Aa]  +
[Cc]  +
[Hh]  +
```

The square brackets are used to indicate character classes (e.g. "B" or "b"). The plus sign following the tab means "one or more matching records." Consequently, the above template refers to "one or more (data) records matching either a lower- or uppercase character 'B' followed by one or more records matching either a lower- or uppercase character 'A' followed by one or more records matching either a lower- or uppercase character 'C' followed by one or more records matching either a lower- or uppercase character 'H'." We can use this template after we have translated our input into the German system for pitch representation. Assuming the file `BACH` contains the above template, the following two commands will search for all instances of B–A–C–H and play each instance using MIDI:

```
tonh -x file | ditto -s = > file.german
patt -e -f BACH file.german | midi | perform
```

Figure 3 shows an example of a formerly unknown instance of B–A–C–H discovered by Walter

Figure 3. Continued.



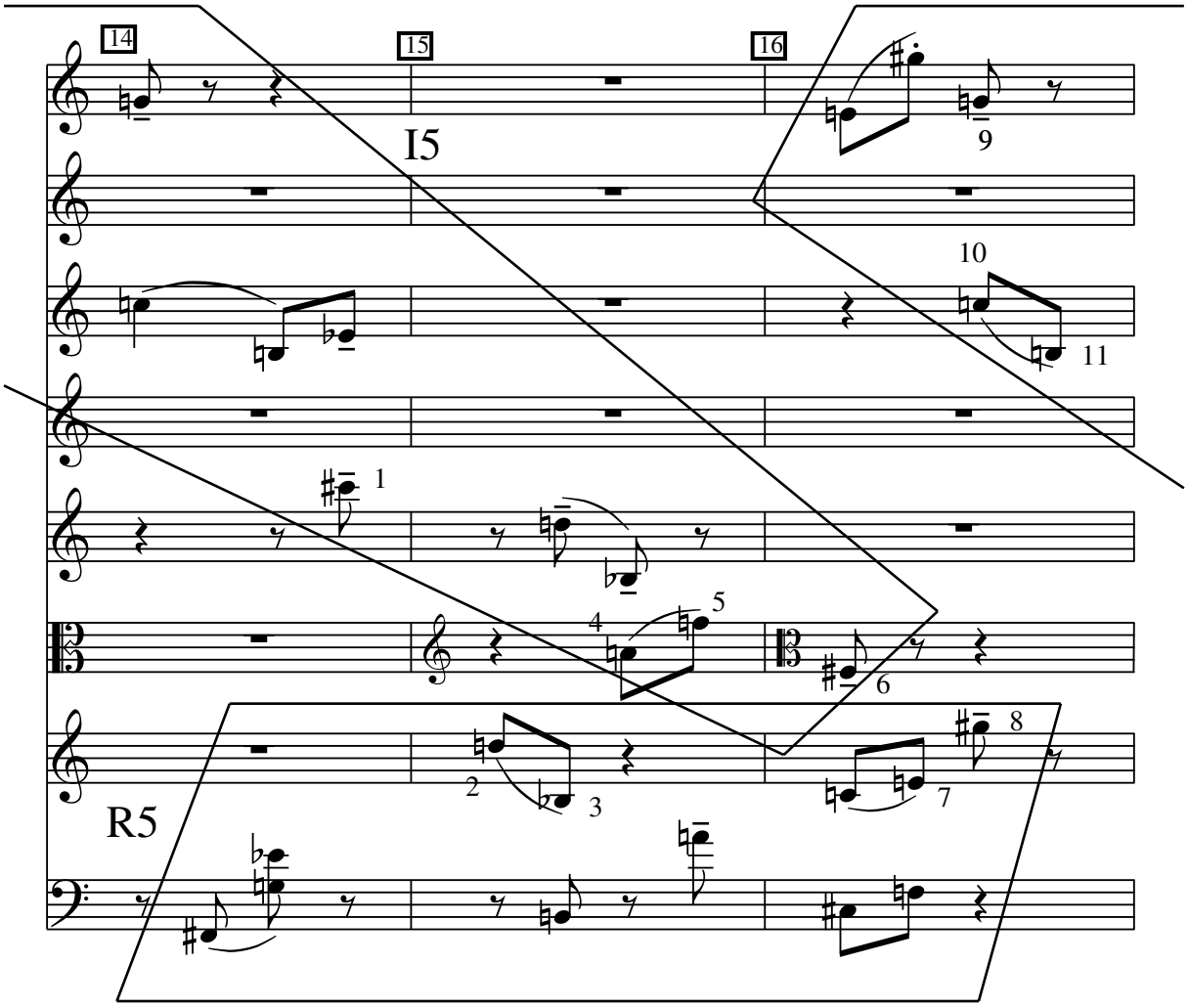
Hewlett in Bach's *Brandenburg Concerto No. 2*—just six measures before the end of the first movement. The B–A–C–H pattern occurs in the cello and contrabass parts.

Rather than searching for pitch patterns, a musically more useful search might focus on melodic intervals. Suppose, for example, we were looking for motivic statements in the opening movement of Beethoven's Fifth Symphony. The following template will locate all passages that begin with any sonority ("`.*`"), followed by two perfect unisons, and terminated by either a major or minor descending third:

```
.*
P1
P1
-[mM]3
```

In some applications, it is often useful to generate pattern templates automatically. For example, in serial and twelve-tone compositions, the Humdrum `reihe` command can be used to generate all set variants from some prime form. Then the `patt` command can be iteratively invoked to search for each set form. Normally, a pitch-class set (`**PC`) representation would be used to conduct the search.

Figure 4. A passage from the first movement of Webern's Opus 24 Concerto, *with the row statements identified using a script that iteratively invokes* `patt` *for each row variant.*



A unique characteristic of set-related practices is the constructing of simultaneities using successive elements from a row. That is, an abstract sequence of tones (1, 2, 3, 4, 5) might appear as tone 1, followed by tones 2, 3, and 4 (sounding concurrently), followed by tone 5. The `patt` command provides an option that instructs `patt` to match each input record with the maximum number of possible contiguous template patterns.

Figure 4 shows a sample passage from the first movement of Webern's *Opus 24 Concerto*, with the row statements identified using a script that iteratively invokes `patt` for each row variant. Notice that `patt` has identified patterns both within and across instruments. When appropriate, alternative names for tone-row statements will be identified. In the second system, `patt` also identified a statement of I1 (identified only by numbered pitches). Although this statement seems to be questionable, the pitches of I1 are contiguous across the participating instruments.

The `patt` command provides an option that causes a new spine to be output containing user-defined tags (such as P5, RI6, Theme A, etc.). These tagged outputs can be used as input for another pattern search, hence users can use `patt` to search for

patterns of patterns, etc. For example, the following Humdrum input might match a template whose purpose is to identify sonata allegro forms:

```
**sections
Introduction
Exposition
Development
Recapitulation
Coda
*-
```

Similarity

A characteristic of pattern searches is that only two states are possible: either a passage matches the defined template or it does not. Because music relies extensively on the art of variation, however, it is more important for many musical applications to be able to characterize degrees of resemblance than identify absolute matches.

While the concept of ''similarity'' seems intuitively obvious, as an operational concept it proves remarkably complex. A basic distinction can be made between similarity for numerical (or parametric) data and non-numeric (non-parametric) data. Measurements of similarity for parametric data can be devised using variants of mathematical correlation, such as Pearson's *r*. Measurements of similarity for non-parametric data have been explored using various algorithms in the field of approximate string matching (see Hall and Dowling 1980).

Humdrum provides a parametric similarity tool (`correl`) for characterizing numerical similarity, such as determining similar pitch contours. A non-parametric Humdrum tool (`simil`) allows users to define similarity according to an extended class of Damereau–Levenshtein edit-distances (see Orpen and Huron 1992 for details).

The `simil` tool allows users to define penalties for various edit actions such as deletions, insertions, and substitutions; users are also free to define the basic representation used by `simil`. Depending on the application, a user might choose pitch similarity, interval similarity, contour/duration similarity, articulation similarity, harmonic similarity, or other forms or combinations of similarity. Note that `simil` is not restricted to notation-based applications; it is equally adept at all non-parametric similarity tasks, such as characterizing spectral similarity, fingering similarity, similarity of conducting gestures, etc. Any representation that a user can concoct can be used as input for `simil`.

A unique property of the Humdrum `simil` tool is that it allows users to define asymmetrical similarity measures, where A is judged more similar to B than B is to A. Psychologically, the capacity for asymmetrical similarity measures is important. In prototype theory, for example, it is known that people tend to judge the color pink as more similar to the color red (prototype) than vice versa. Similarly, listeners tend to judge a musical variation as more similar to the theme (prototype) than the reverse comparison. Several contrasting illustrations using `simil` are described in Orpen and Huron (1992).

Other Tools

The above examples have introduced only a handful of the Humdrum tools and have only scratched the surface in illustrating the information processing capabilities of Humdrum. As we have seen, the strength of the Humdrum tools lies not in their individual capabilities, but in their endless variety of configurations and interactions. The Humdrum web site (humdrum.net) provides hundreds of additional tutorial examples that illustrate a variety of music-related information-processing tasks.

Scripts

Humdrum's command-line interface is frequently regarded as the principal impediment to more widespread use. However, programmers will readily understand that the command-oriented structure is Humdrum's principal strength, because it allows complex scripts to be written and embedded in one's favorite programming language. Scripts can be created both to carry out routine operations and

to package a complex process as a stand-alone application.

An example of the latter can be found in Themefinder, a ''name-that-tune'' web application located at www.themefinder.com. Themefinder provides a simple interface that allows users to search a database of approximately 25,000 musical themes and incipits by specifying various search keys, such as the up/down melodic contour (Kornstädt 1998). The engine underlying Themefinder is a Humdrum script that was written in a single afternoon. The actual searching is done by the UNIX `grep` command. It is important to understand that Themefinder actually limits users' searching capabilities. Humdrum itself allows far more ways of accessing and searching the data, but a form-based web interface provides greater convenience.

Another example of the value of scripting is evident in the simplicity with which Humdrum can be connected to other software tools. Figure 5 shows a musical map generated by connecting Humdrum to GMT, a free ''generic mapping tool'' that is used by professional geographers to generate high-quality PostScript maps. Like the Humdrum commands, GMT is invoked as a POSIX-format command with a potentially large number of map-drawing options. As part of a project to study culture-related musical features, Bret Aarden wrote a script that formats Humdrum's geographical reference records as input to GMT (Aarden and Huron 2001). Figure 5 shows a simple contour map indicating the density of Germanic folksongs in minor keys. The map was generated using data from the Essen Folksong Collection (Schaffrath 1995). The important point is that any type of music-related search can be ''piped'' to GMT, resulting in a tailor-made musical map.

Technical Specifications

Humdrum is written in a combination of languages, including C, C++, awk, kornshell, yacc, lex, and perl. The toolkit is available free of charge via the web, and source code is included in the distribution. Humdrum can be used with all popular operating systems; however, a POSIX-conformant command shell is required. Complete documentation is available, including information describing all pre-defined representations, software documentation for all Humdrum tools, introductory tutorials, examples, and newsletters. A developers kit is available that includes a syntax checker, input parser, and test suites. Documentation is available in both printed and online versions. The best general introduction to the Humdrum Toolkit is *Music Research Using Humdrum: A User's Guide* (Huron 1999).

Drawing Lessons from the Humdrum Experience

Whether or not one uses Humdrum, there are a number of broad lessons arising from the Humdrum experience that are pertinent to creating general-purpose music-related software.

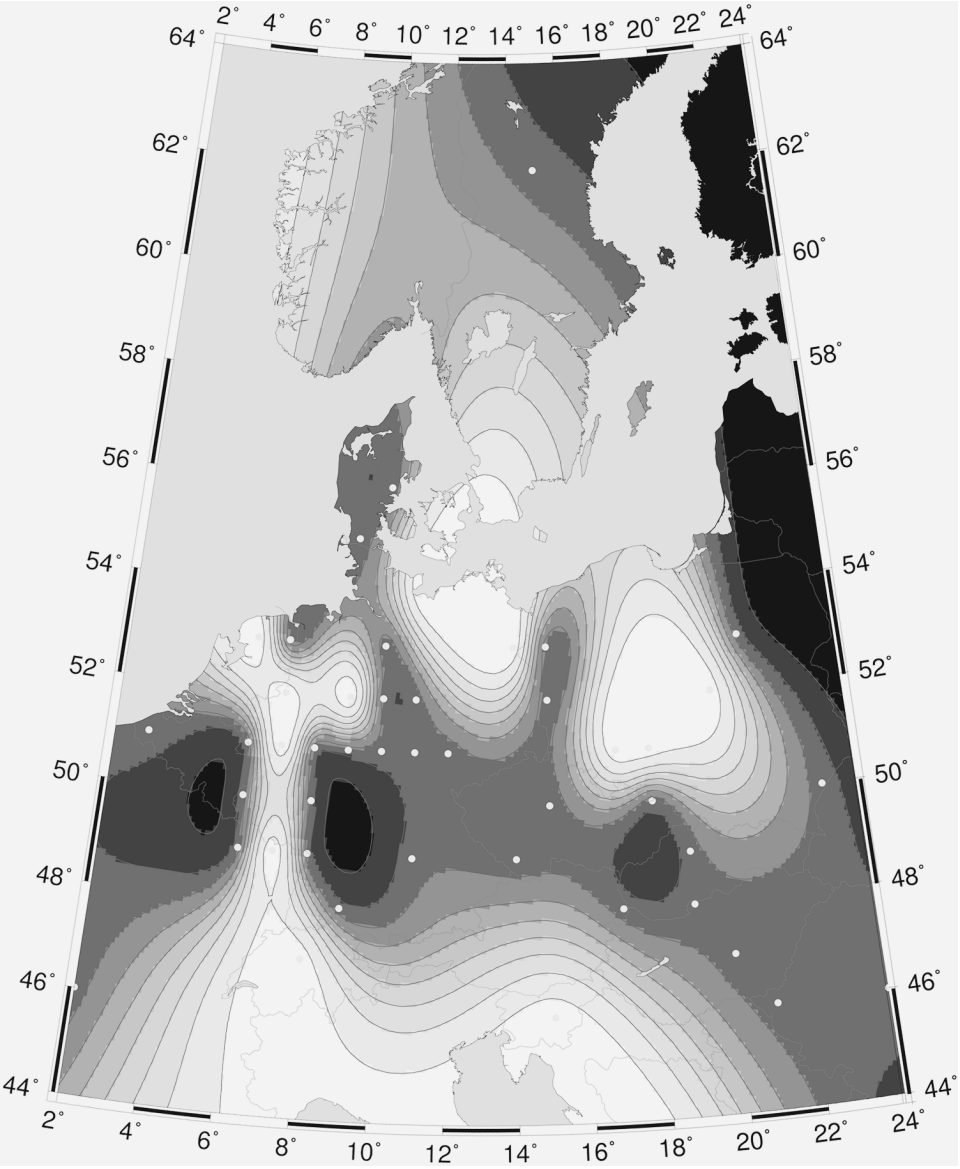
In software design, a useful distinction can be made between *closed* and *open* applications. A closed application can be defined as one serving a well-defined problem space where all tasks can be specified in advance. In closed applications, it is possible to construct efficient stand-alone software that provides an intuitive and effective user interface. However, in research or development-related environments, the types of tasks users pursue are unpredictable and thus open-ended. In such environments, it is impossible to conceive of a single integrated application that will serve all purposes, and so a software tools approach is preferable.

Whenever possible, use flat text (ASCII) representations. This spares programmers the trouble of having to design special-purpose editors and other general tools for each representation. Using text-based representation also avoids erecting barriers for users who wish to define and use their own representations.

Do not attempt to represent everything in one gigantic encoding scheme. As in the case of structured programming, it is better to segment the representational problem so encoding, editing, and processing remain manageable.

Do not force users to encode information (such as pitch, duration, title, etc.) that is not of interest to the user.

Figure 5. A musical map generated by connecting Humdrum to GMT, a free generic mapping tool.



The effectiveness of text-based representations is greatly increased by pushing the data structures into the data representation. In its use of a tabular format of spines and records, Humdrum representations are inherently two-dimensional and thus closely echo the structure of sequential and concurrent events typical of music. In effect, Humdrum representations form a linked-list lattice structure. Instead of creating an internal data structure accessible only within a program, Humdrum places the linked-list structure within the data representation. This approach greatly simplifies access to and processing of the data.

Label or tag the representations so each software tool can identify which data the tool can legitimately manipulate, which data it should leave untouched, and which data constitute errors.

In research and development situations, create

software tools that can be executed within any programming language or script.

When processing information, at each step ensure that users can intercept the data and invoke either commercial tools or user-crafted scripts to massage the information.

Do not assume that users will perform a given task using the tool you have provided. Users may disagree with the approach or be more familiar with the operation of another tool.

Avoid writing software that already exists, such as general ''sort'' routines or music notation editors. Having to write such software usually means that insufficient attention has been paid to allowing users to connect easily with other software.

Provide comprehensive and complete documentation for all tools so that conscientious users can comprehend and anticipate the limits of operation, and so programmers can identify suspected bugs with confidence. In many ways, basic Humdrum tools like `extract`, `assemble`, `patt`, and `humsed` are structured equivalents of the popular UNIX utilities `cut`, `paste`, `grep`, and `sed`. Whereas the UNIX tools operate indiscriminately on all data in the standard input, the Humdrum equivalents pass comments and reference information intact, preserve the structure and syntax of the input, and operate only on specified data types within a data stream. In addition, the Humdrum equivalents update the interpretation information where appropriate so that subsequent tools recognize that the data has been modified in a particular manner.

The importance of data typing is evident in the burgeoning conventions for filename extensions such as ''.jgp,'' ''.gif,'' ''.html,'' ''.mid,'' ''.mp3,'' etc. These tags are often used by application software to recognize files that can be processed, and they are appended to output files to specify the data format. The Humdrum ''interpretation'' records provide a more nuanced and powerful way of data tagging. By placing the type tags within the file or data stream itself, Humdrum allows multiple forms of information to co-exist within a single document.

The benefit of this approach is that it preserves structural relationships between various data types, thereby facilitating contextually sensitive processing of multiple forms of data concurrently. For example, with Humdrum it is straightforward to process notation-related and performance-related data concurrently. By way of illustration, in Humdrum it would be relatively easy for a user to search for all notated mordents where the key velocity of a MIDI performance is higher for the second note than for the third note of the mordent. By contrast, if the notation data and MIDI data were stored in separate files, performing such a search would be technically challenging.

Conclusion

This article has provided a cursory introduction to Humdrum. Humdrum provides a syntax that allows users to represent arbitrary forms of time-dependent information. It also provides a set of utilities or tools that manipulate Humdrum representations in various ways. The tools perform operations such as displaying, performing, searching, editing, transforming, extracting, linking, classifying, labeling, and comparing. The tools can be used individually or linked together to carry out a variety of tasks. Users can intercept representations at any point and write their own tools that augment the functioning of Humdrum. In addition, the Humdrum tools can be accessed from within standard programming languages.

After more than a decade, the Humdrum Toolkit remains without peer in music information processing applications. Its principal attraction has been its broad scope for musical problem-solving and its flexibility of operation. Its principal detractions have been its command-line interface, and its presumption that users have facility with UNIX-like shell commands. Especially for users with no prior programming experience, the learning curve for Humdrum can seem intimidating. While not all computer music researchers will benefit from using Humdrum, a number of design features and criteria provide useful lessons for developing future music-related software.

Acknowledgments

Continued software development has benefited from the efforts of many individuals. Two graphic

user interfaces have be written: one by Michael Taylor at the University of Belfast (Taylor 1996), and a second by Andreas Kornstädt at the University of Hamburg (Kornstädt 1996). Robert Gjerdingen at Northwestern University and Po-Yan Tsang at the University of Waterloo wrote Humdrum/Finale translators. Kyle Dawkins at McGill University wrote extensions for coordinating Humdrum with compact discs. Utilities for making Humdrum Lisp-compatible have been written by Jörg Garbers and Thomas Noll at the Technical University of Berlin. Craig Sapp at Stanford University has written a number of utilities, including a harmonic analyzer. Another harmonic analyzer developed by David Temperley (Temperley 2001) at Columbia University and the Eastman School of Music was rendered Humdrum-compatible by Bret Aarden at the Ohio State University. Michael Good of Recordare Inc. has fashioned an XML scheme inspired by Humdrum (Good 2000).

The Center for Computer Assisted Research in the Humanities has been indispensable in providing direct and indirect support for Humdrum. I am grateful to Drs. Walter Hewlett and Eleanor Selfridge-Field for making the MuseData editions available online in Humdrum format. Other scholars have provided databases initially encoded using other representation schemes, including John Miller at North Dakota State University, Helmut Schaffrath at the University of Essen, and Harry Lincoln at the State University of New York at Binghamton.

More than 80 music collections have been encoded directly in the Humdrum format. Among those who have encoded music using Humdrum are Suzi Wint, Sandra Serafini, Lonney Young, Ben Koen, Eric Berg, Norma Welch, Franz Wiering, Joshua Veltman, Igor Karaca, Natasa Kara, Paul von Hippel (von Hippel 1998), Stefan Morent (Morent 2000), Matthew Royal (Huron and Royal 1996), and Denis Collins (Collins and Huron, in press).

Further thanks are due to Keith Orpen, Keith Mashinter, Bill Thompson (Thompson and Stainton 1995–96), Jasba Simpson (Simpson and Huron 1993), Jonathan Wild (Wild 1996), Randall Howard, Simon Clift, Maki Ishizaki, Bo Alphonce, Bruce Pennycook, David Wessel, Chris Chafe, Max Mathews, John Howard, Gregory Sandell, Perry Roland, and Jordi Martin. I am especially indebted to my former research assistants Tim Racinsky and Kyle Dawkins for their professional programming efforts.

References

Aarden, B., and D. Huron. 2001. ''Mapping European Folksong: Geographical Localization of Musical Features.'' *Computing in Musicology* 12:169–183.

Collins, D., and D. Huron. In press. ''Voice-leading in Cantus Firmus-Based Canonic Composition: A Comparison Between Theory and Practice in Renaissance and Baroque Music.'' *Computers in Music Research*.

Erickson, R. 1976. ''DARMS: A Reference Manual.'' Binghamton, NY: typescript.

Good, M. 2000. ''MusicXML for Notation and Analysis.'' *Computing in Musicology* 12:113–124.

Hall, P., and G. Dowling. 1980. ''Approximate String Matching.'' *ACM Computing Surveys* 12:381–402.

Hall, T. 1997. ''DARMS: The A-R Dialect.'' In E. Selfridge-Field, ed. *Beyond MIDI: The Handbook of Musical Codes*. Cambridge, Massachusetts: MIT Press, pp. 573–580.

Hewlett, W. B. 1997. ''MuseData: A Multipurpose Representation.'' In E. Selfridge-Field, ed. *Beyond MIDI: The Handbook of Musical Codes*. Cambridge, Massachusetts: MIT Press, pp. 402–447.

Howard, J. 1997. ''Plaine and Easie Code: A Code for Music Bibliography.'' In E. Selfridge-Field, ed. *Beyond MIDI: The Handbook of Musical Codes*. Cambridge, Massachusetts: MIT Press, pp. 343–361.

Huron, D. 1992. ''Design principles in computer-based music representation.'' In A. Marsden and A. Pople, eds. *Computer Representations and Models in Music*. London: Academic Press, pp. 5–59.

Huron, D. 1995. *The Humdrum Toolkit: Reference Manual*. Stanford, California: Center for Computer Assisted Research in the Humanities.

Huron, D. 1997. ''Humdrum and Kern: Selective Feature Encoding.'' In E. Selfridge-Field, ed. *Beyond MIDI: The Handbook of Musical Codes*. Cambridge, Massachusetts: MIT Press, pp. 375–401.

Huron, D. 1999. *Music Research Using Humdrum: A User's Guide*.

Huron, D., and M. Royal. 1996. ''What is Melodic Accent? Converging Evidence from Musical Practice.'' *Music Perception* 13(4):498–516.
Kornstädt, A. 1996. ''SCORE-to-Humdrum: A Graphical Environment for Musicological Analysis.'' *Computing in Musicology* 10:105–122.
Kornstädt, A. 1998. ''Themefinder: A Web-Based Melodic Search Tool.'' *Computing in Musicology* 11:231–236.
Morent, S. 2000. ''Representing a Medieval Repertory and its Sources: The Music of Hildegard von Bingen.'' *Computing in Musicology* 12:19–33.
Orpen, K., and D. Huron. 1992. ''Measurement of Similarity in Music: A quantitative Approach for Non-Parametric Representations.'' *Computers in Music Research* 4:1–44.
Schaffrath, H. 1995. *The Essen Folksong Collection*. Stanford, California: Center for Computers Assisted Research in the Humanities.
Simpson, J., and D. Huron. 1993. ''The Perception of Rhythmic Similarity: A Test of a Modified Version of Johson-Laird's Theory.'' *Canadian Acoustics* 21(3): 89–90.
Smith, L. 1972. ''SCORE : A Musician's Approach to Computer Music.'' *Journal of the Audio Engineering Society* 20:7–14.
Taylor, W. M. 1996. *Humdrum Graphical User Interface*. MA Thesis, Music Technology, Queen's University of Belfast, Ireland.
Temperley, D. (2001). *The Cognition of Basic Musical Structures*. Cambridge, Massachusetts: MIT Press.
Thompson, W.F., and M. Stainton. 1995–96. ''Using Humdrum to Analyze Melodic Structure: An Assessment of Narmour's Implication-Realization Model.'' *Computing in Musicology* 10:24–33.
Vercoe, B. 1986. *Csound: A Manual for the Audio Processing System and Supporting Programs with Tutorials*. Cambridge, Massachusetts: MIT Media Lab.
von Hippel, P. 1998. *42 Ojibway Folksongs in the Humdrum **kern Representation: Electronic Transcriptions from the Densmore Collections*. Stanford, California: Center for Computer Assisted Research in the Humanities.
Wild, J. 1996. ''A Review of the Humdrum Toolkit: UNIX Tools for Musical Research, Created by David Huron.'' *Music Theory Online* 2(7). Available online at www.societymusictheory.org/mto/.