CHAPTER 6

Computational and Comparative Musicology

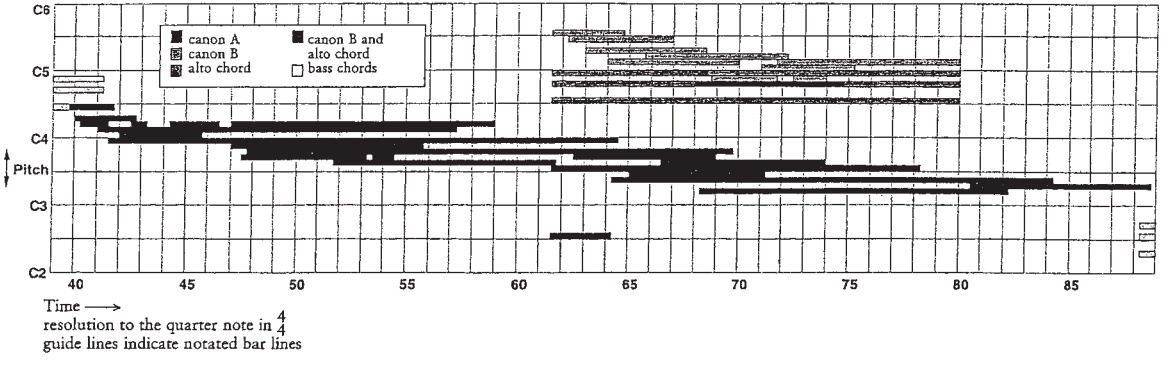
Nicholas Cook

Introduction

The middle of the twentieth century saw a strong reaction against the comparative methods that played so large a part in the disciplines of the humanities and social sciences in the first half of the century, and musicology was no exception. The term "comparative musicology" was supplanted by "ethnomusicology," reflecting a new belief that cultural practices could only be understood in relation to the particular societies that gave rise to them: it was simply misleading to compare practices across different societies, the ethnomusicologists believed, and so the comparative musicologist was replaced by specialists in particular musical cultures. A similar reaction took place in theory and analysis: earlier style-analytical approaches (largely modeled on turn-of-the-century art history) gave way to a new emphasis on the particular structural patterns of individual musical works. Perversely, this meant that the possibility of computational approaches to the study of music arose just as the idea of comparing large bodies of musical data—the kind of work to which computers are ideally suited—became intellectually unfashionable. As a result, computational methods have up to now played a more or less marginal role in the development of the discipline.

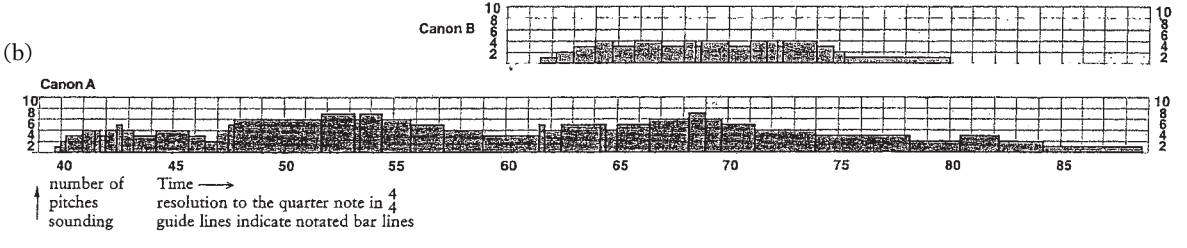
In this chapter I suggest that recent developments in computational musicology present a significant opportunity for disciplinary renewal: in the terms introduced in chapter 1, there is potential for musicology to be pursued as a more data-rich discipline than has generally been the case up to now, and this in turn entails a re-evaluation of the comparative method. Central to any computational approach, however, are the means by which data are represented for analysis, and so I begin with some examples of "objective" data representations before introducing the issue of comparison. (The examples I discuss are graphic, but the same points could have been made in terms of numerical representations.) This is followed by an extended case study of an important current software package for musicological research, the Humdrum Toolkit, and the chapter concludes with a brief consideration of the prospects for computational methods in musicology.

(a)



Range Graph, *Lux aeterna*, Section 2

(b)



Pitch Count Graph, *Lux aeterna*, Section 2

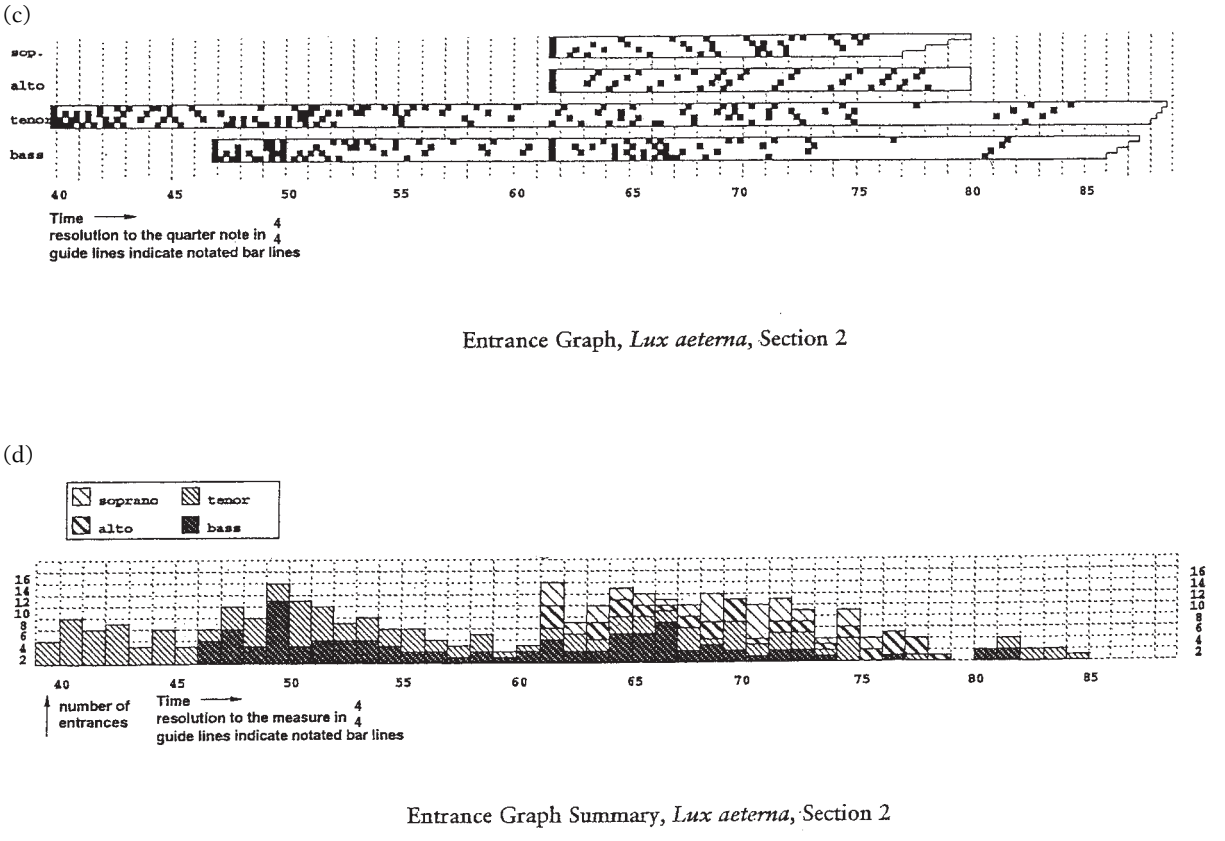


Figure 6.1. Graphic analyses of Ligeti, *Lux aeterna*, section 2 (from Clendenning 1995: 948–49).

Issues of Representation

A picture, as everyone knows, is worth a thousand words. Each of the graphs in Figure 6.1 (from Clendenning 1995: 248–249) represents a different aspect of the same section from Ligeti's *Lux aeterna*: (a) charts pitch against time, giving an immediate impression of the music's registral profile, (b) shows how many different pitches (not pitch classes) are present at any given point, and (c) highlights voice entries, with each separate voice (four each of soprano, alto, tenor, bass) having a separate horizontal line which is shaded whenever the voice enters, while (d) shows essentially the same information as (c), only in terms of the total number of entries at any given point. These graphs are "objective" in the sense that, once you have agreed how a graph is to be laid out, everyone should end up with the same result; the information is all there in the score, so that the analysis becomes, in effect, a matter of reformatting. But the decisions about how to lay them out are not necessarily self-evident. For example, in (a) to (c) the minimum values on the time axis are quavers, whereas in (d) they are whole bars; other values would have been possible, and the decision to use these particular values represents a judgment by the analyst that they will be the most informative in this context. There is also a degree of analytical judgment in the separate identification of the canons in (a) and (b).

That said, however, it is obvious that there is scope for automation in graphs like this—and not just in drawing them up (the graphs in Figure 6.1 were generated by a computer drawing package) but in the processing of the data. The graphs in Figure 6.2 (from Brinkman and Mesiti 1991: 5, 6, 13, 17) were automatically generated from a machine representation of the score.[1] Again, each graph represents a different way of extracting and viewing the information in a score, this time bars 1 to 26 of the first movement from Bartók's String Quartet no. 4: (a) corresponds to Figure 6.1 (a), (b) shows each instrumental part separately with the vertical lines providing an impression of the linear movement from one pitch to another, and (c) charts the first appearance of each pitch, while (d) represents the dynamic level of each instrument (based on whether it is playing and on notated dynamic value) and of the whole (in effect a summation of the dynamic graphs of each instrument).

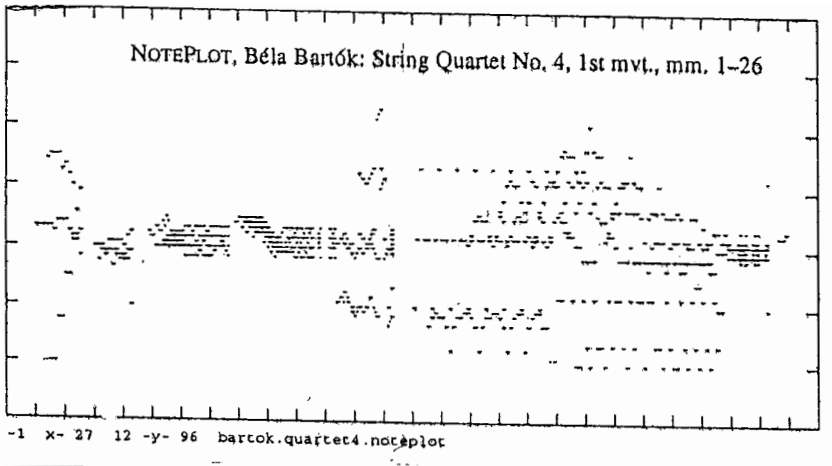
But what are we to make of such representations? What do they enable us to see that we can't hear, and in any case, what is the point of *seeing* music at all? It is a commonplace to describe Ligeti's texture-oriented works as "visual," and Brinkman and Mesiti (who in their article offer a case study of Webern's *Symphonie* Op. 21) emphasize the spatial symmetry of Webern's serial structures, commenting that "the graphs allow us to view all parts together in their spatial environment. [This] is especially helpful since the pitch symmetry that is essential to the musical structure of this work is somewhat obscured by the notated score" (Brinkman and Mesiti 1991: 21). The question then is the extent to which this problem of obscuring applies to other repertories, in other words how valuable this kind of visualization is for music in general. Yet this question is in some ways misguided: there is no such thing as a good or bad representation of music *per se*, there are good and bad—or at least better and worse—representations *for particular purposes*. To say that visual representations are more appropriate for twentieth-century than for other repertories, or for

answering certain kinds of questions about the music than others, is simply to define their usefulness, not to criticize them.

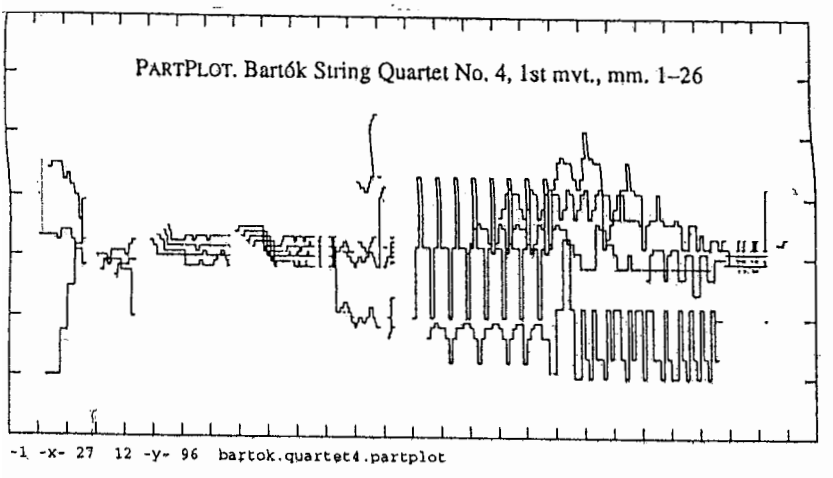
Nevertheless Brinkman and Mesiti produced some interesting results for music that one wouldn't normally think of as particularly "visual," including some strikingly different representations of music by Bach and Mozart as well as by Bartók, Schoenberg, Wolpe, and Berio. At the very least, such comparisons serve to underline the variety of textures found in different composers or styles—and texture is at the same time one of the most important aspects of music in terms of how we experience it, and one of the hardest to say anything useful about. It is not hard to imagine how Brinkman and Mesiti's software could be used within a variety of pedagogical contexts, enabling students to literally and instantaneously "see" different pieces of music in a variety of different ways—and the ability to switch easily and quickly from one representation of music to another is a crucial element of practical musicianship. That this has not happened, and that the potential of the approach has not been fully realized, reflects the cost of translating academic research projects into software products for the real world, and illustrates an all-too-familiar pattern in musicological software: a sustained burst of initial enthusiasm is followed by running out of money, resulting in software that is sometimes less than fully functional, often less than fully documented, rarely properly supported, and usually soon obsolete.

But my principal answer to the question "what are we to make of such representations?" is a different one. Brinkman and Mesiti (1991: 7) emphasize the particular usefulness of their graphs "as a device for comparing different pieces"; Matt Hughes, whose quantitative analysis of Schubert's Op. 94 no. 1 was discussed in chapter 1, similarly claimed of his method that it was "a tool for organizing data so that one may more discernibly view tendencies and interassociations," adding that it was "best utilized when viewing groups of compositions or sections of a composition rather than a single work" (Hughes 1977: 145, 150). And there is a general point to be made here: the more objective an analytical approach is, the more its musicological value is likely to be realized through making comparisons between different pieces—and sometimes large numbers of them. For example, when you carry out a Schenkerian analysis, you are in a sense making a comparison between the piece in question and the norms of common-practice voice leading, as systematized in the Schenkerian background and middleground. But the value of the analysis consists primarily in the lengthy process of making it, deciding which notes go with which, which are more important than others, and so forth; the process is lengthy because it involves a vast number of interpretive judgments, requiring you to weigh up different factors in relation to one another. At the end of it, you have a knowledge of the music—you might call it an intimacy—that you did not have at the outset, and there is a sense in which the final graph is significant mainly as a record of this learning process. With any kind of computational approach, by contrast, all of this happens automatically, and in some cases almost instantaneously; the only output is the graphic or numerical representation of the music that results. And such representations rarely tell you anything very useful by themselves: they may well be vulnerable to the cheap but telling jibe that they either show you what you hear anyhow (in which case they are redundant), or else they don't (in which case there are

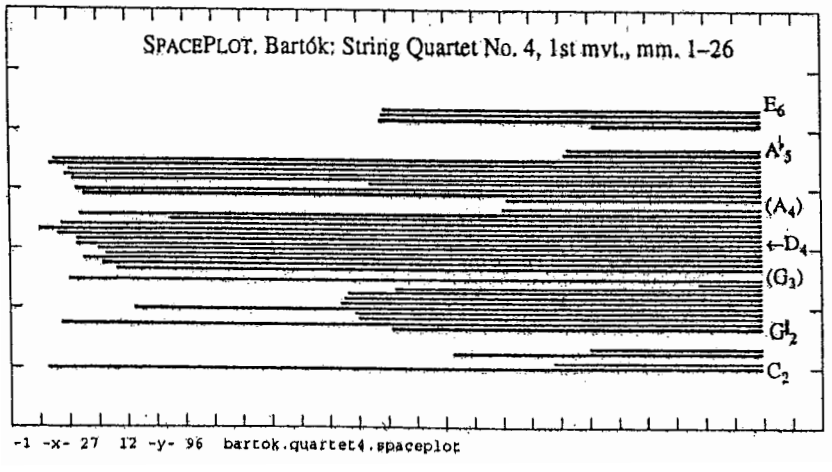
(a)



(b)



(c)



(d)

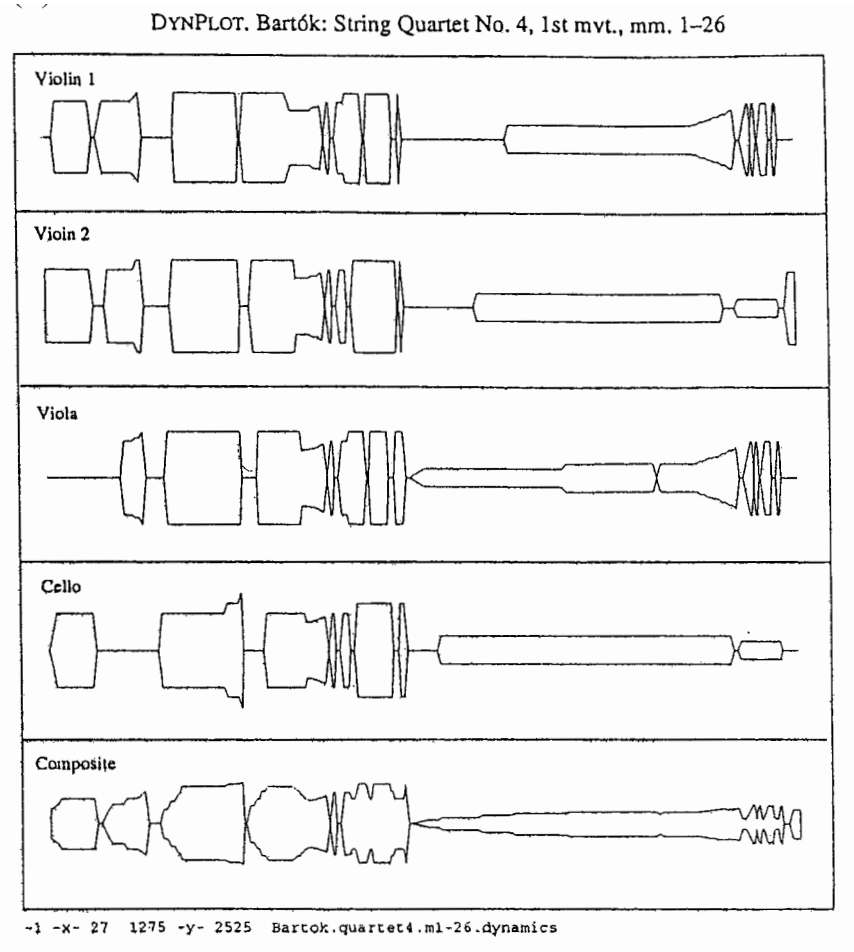


Figure 6.2. Graphic analyses of Bartók, String Quartet No. 4, I, bars 1–26 (from Brinkman and Mesiti 1991: 5, 6, 13, 17).

irrelevant). It is when you compare representations of different pieces that useful information may emerge.

The value of objective representations of music, in short, lies principally in the possibility of comparing them and so identifying significant features, and of using computational techniques to carry out such comparisons speedily and accurately. At this point, however, the issue of reduction (already broached in chapter 1, this volume) has to be confronted. This can conveniently be explored in relation to the analysis of folksongs, since a large of proportion of early work in computational analysis was based on such material: there were substantial existing collections of folksongs to work on, and the music was monophonic and scalar, hence could be coded very compactly—a vital consideration in the days when computer memory was a scarce and expensive commodity. But of course, if you code folksongs as a series of pitches, or scale degrees, then you are leaving out all the variables of into-

```
BOEHME
CUT[ DEUTSCHLAND DEUTSCHLAND UEBER ALLES]
REG[ Europa, Mitteleuropa, Deutschland]
KEY[ B0001  16  F 4/4]
MEL[ 1__.2_  3__2__4__3__  2_-7_1__
     6__5__  4__3__2__3_1_  5___
     1__.2_  3__2__4__3__  2_-7_1__
     6__5__  4__3__2__3_1_  5___
     2__3__  2_-7_-5__4__3__  2_-7_-5__
     5__4__  3__.3_4#__4#_5_  5___
     +1__.7_  7_6_5__6__.5_  5_4_3__
     2__.34  5_6_4_2_1__3_2_  1___
     +1__.7_  7_6_5__6__.5_  5_4_3__
     2__.34  5_6_4_2_1__3_2_  1___ //] >>
FCT[ Volks - Hymne, national, politisch, Vaterlands - Lied]
```

Figure 6.3 *EsAC* code for "Deutschland über alles" (Helmut Schaffrath, ***Essen Musical Data Package***)

nation, ornamentation, timbre, and other aspects of singing style, not to mention the original contexts of use and cultural connotations of the songs. And that obviously limits what you can conclude from comparing them.

At this point a concrete example may be helpful, and Figure 6.3 is taken from the Essen Musical Data Package, a series of databases of popular and traditional vocal repertories together with tools for encoding and analysis.[2] Coordinated until his death in 1994 by Helmut Schaffrath, the Essen databases consisted by the following year of some 10,000 songs, mainly European (and largely German), but with some representation of other traditions, particularly Chinese; they are still being added to, though development has been transferred from Essen to Warsaw (Selfridge-Field 1995). Figure 6.3 shows "Deutschland über alles" in the custom code used by the package, *EsAC* (the Essen Associative Code), and several of the data fields are self-explanatory: BOEHME is the name of the database in which this song is located, CUT contains the title, REG is the region from which it comes, and FCT is the functional category of the song, while KEY contains not only its key (F) but also the numbering of the song in the database (B0001), the value of the smallest note value (16, a sixteenth or semi-quaver), and the time signature (4/4). The tune itself is contained in the field MEL, using a simple code based on the scale degree (with # for sharp and b for flat degrees), + and − to indicate shifts to a higher or lower register, respectively, and a rhythmic notation based on the shortest value (semi-quaver), with a dot prolonging the value by 50 percent, a single underline character (_) by 100 percent, and a double underline character (__) by 200 percent. That should suffice for the tune to be read; the only other information contained in the MEL field is // (for the end of the melody), the use of spaces to indicate bar lines (the melody has been notated across the bar lines), and the splitting up of the melody into separate lines, each of which represents a phrase (this represents a judgment on the part of the transcriber, and as such is a rather unusual feature of the *EsAC* code).

What can be done using this information? The distributed version of the Essen

database comes with a simple analytical utility (ANA), which takes a complete database as its input, and generates as its output an annotated version of the file in which up to 12 different kinds of analytical information are added for each song. These include straightforward statistical information such as the distribution of ascending and descending intervals (how many ascending minor seconds?), the distribution of durations (how many crotchets, quavers, semi-quavers?), and distribution of scale degrees (how often does the fifth scale degree appear in the lower register?); there are also some derived measures such as the percentage of ascending steps and leaps. They also include some less obvious information, including the pattern of phrase repetitions in pitch and duration, a coding of the contour of each phrase, the melodic spine (based on the notes at metrically stressed beats), and the final notes of each phrase (this is where the coding of phrases comes in).[3] A limitation of the ANA software is that it provides data for each song separately, whereas the statistical information would be more useful if it were provided across the database as a whole, but there is a reason for this: the package was intended for use with a commercial relational database package, which would facilitate this kind of analysis. Needless to say, the original package is no longer obtainable.[4]

But what can be *done* using this information? Schaffrath himself published a number of studies based on his datasets, in which he used this kind of analytical information to support broad stylistic generalizations, for example as between different traditions: "German folk songs," he concluded from one such study (1992: 107), "generally skip more often up than down; in the Chinese pool the opposite applies. . . . German songs use the interval of a fourth 51% more often than Chinese songs do. This might be explained by the preference for pentatonic scales." But there is also a different way in which such information can be used, which is for purposes of searching, identification, and classification. A database of 10,000 folksongs is not unlike a bibliographic database, and to locate individual items or groups of related items you search for particular features, combining them so as to narrow the search down to a manageable number of hits. In some contexts all you need to search a bibliographic database is the author's name; in other cases you need to combine an author name, two title keywords, and year to locate the article you need. In the same way, you might try to locate a particular song by searching for its first few notes or incipit; this is the way that traditional dictionaries of themes operate (for instance, Barlow and Morgenstern 1983), and it works well with "art" music, where there are fixed scores. But it is not such a good way to search for folksongs, which typically exist in a large number of variants, often involving variation or the interpolation of notes. That is why ANA includes a routine to derive the final notes of each phrase: as Schaffrath (1997: 349) explains, "although variants may contain different numbers of notes or melodic contours, they tend to retain underlying harmonic structures." Or you could combine such a search with a particular melodic spine and/or pattern of phrase repetition. And whereas I have described the use of such features in terms of simply locating items, the same processes can be used to group together songs or repertories in ways that might reflect, for example, historical development, linguistic associations, or conditions of use. In such ways the automated analysis of musical style can become a means of generating or verifying musicological hypotheses.

Or at least that is the aspiration. In practice, *EsAC* and ANA suffer from obvious limitations, ranging from the pragmatic (the output of ANA is only marginally more

user-friendly than machine code) to those of principle. *EsAC* can cope with only a narrow range of music (monophonic, restricted to three octaves, and so on), and is well adapted for only a narrow range of applications. ANA extracts a limited number of features, each of which represents a radical reduction of the music and is intended for a particular purpose (such as statistical generalization or pattern matching); moreover, the usefulness of these reductions depends on a number of quite specific assumptions, such as that the final-note patterns of phrases vary less than incipits. All this makes sense in terms of what the Essen package is designed for; it is, to adopt outdated software jargon, a "turnkey" solution to folksong analysis, or at least to a certain sort of folksong analysis. And it would be possible to imagine trying to develop the code, and the analytical software, in such a way as to be more flexible—to cope with music of any desired degree of complexity and variety, and to extract from it every feature that could be of any possible interest under any possible circumstances. That would be like the approach of business software developers in the 1980s, who aimed at prepackaged solutions in which every eventuality had been foreseen and the appropriate feature bolted on somewhere, if only you could find it.

In musicology, as perhaps also in the office, this represents a basic misapprehension—in terms not only of the unlimited variety of musical materials, but also of the variety of purposes for which people will want to represent them; as Huron (1992: 11) puts it, "The types of goals to which music representations may serve are legion and unpredictable." Complexity is not the solution. There are, after all, much more complicated codes than *EsAC* which fulfill certain purposes extremely well while being very poorly adapted for others. MIDI code has transformed entire sectors of the music industry, but because it tries to understand all music on the model of keyboard music its handling of microtonal inflection is inelegant, to say the least; it also doesn't distinguish enharmonic equivalents, and has to be drastically reconfigured if useful analytical information is to be extracted from it—even something as simple as locating a C major triad.[5] Again, DARMS[6] is an extremely powerful code designed for purposes of printing and publication (it is the basis of all A-R Edition scores), and accordingly it understands music as a kind of graphics; it doesn't think of an e^1 but of a notehead on the bottom line of a stave that has an F clef on it. As a result, it too needs drastic reconfiguration if analytically useful information is to be extracted from it (Brinkman 1986). And though the problems of multiple codes are alleviated by the existence of interchange utilities (e.g., to turn DARMS into MIDI), there are limitations to what is possible: since MIDI does not distinguish between C♯ and D♭, the information is simply not there to translate into DARMS, which does make the distinction.

As with business software, the solution to this problem of multiple requirements is not the creation of ever more complex and unwieldy integrated solutions, but a modular approach involving an unlimited number of individual software tools designed to serve individual purposes; that way, when you want to do something new, you simply design a new tool to do it. Modular approaches like this, however, need some kind of framework within which to work—in essence, a set of rules for the representation of data in machine-readable files, and for the transfer of information between different software tools. Humdrum is an example of just such a framework.

A Case Study: The Humdrum Toolkit

Developed by David Huron in the early 1990s, Humdrum[7] consists of two distinct elements: in the first place a syntax—the set of rules I referred to—and in the second, a "toolkit" consisting at present of something over 70 separate software routines for manipulating the data in different ways for different purposes. It runs in a UNIX environment,[8] and its modular approach is much more readily understood by those who have experience of UNIX than those who do not. Like Humdrum, UNIX can be thought of as a set of rules plus a set of tools: individual UNIX commands (which have names like *grep*, *sort*, and *cat*) do simple things like searching files for a particular string and extracting, deleting, or altering it, or rearranging data within or between files. The power and flexibility of the system comes not from these individual tools but from the possibility of "pipelining" them so that the output of one tool becomes the input of the next, forming chains of commands of unlimited length. The Humdrum toolkit is simply a set of additional UNIX tools designed specifically for manipulating music-related data. It is this open and modular philosophy, which always allows for an additional tool to be added when required, that explains why—as Huron (1997: 375) puts it—"Humdrum is especially helpful in music research environments where new and unforeseen goals arise." Unfortunately, as we shall see, it also means that the user has to have a detailed understanding of the way the data are encoded and manipulated. Humdrum, in short, has a steep learning curve.

All this may sound very abstract, so Figure 6.4 shows the first two phrases of "Deutschland über alles" in *kern* code, the normal Humdrum format for representing notated music (only two phrases because when printed out—though not in terms of internal storage—*kern* is much less compact than *EsAC*).[9] This is a direct translation into *kern* of Figure 6.3, so the same information may be found in the data fields preceded by !!! (the exclamation marks identify them as comments); only the information concerning Schaffrath, the copyright statement, and the !!!ARL line (which gives latitudinal and longitudinal coordinates) are new. As for the other lines, **kern means that what follows is in *kern* (the double asterisk means that this label defines the data that follows), and the lines preceded by a single asterisk define the instrument category and instrument (in each case "vox" or voice), meter (4/4), key signature (B♭, the "flat" being designated by "−"—the sign for sharp is "+"), and key (F).[10] The tune itself begins on the following line, preceded by a curly bracket (this indicates the phrases, while the lines beginning with "=" are bar lines). "4.f" means that the first note is the F above middle C (if it were an octave lower it would be F, if an octave higher ff, if an octave higher than *that* fff, and so on); the "4" means that it is a quarter note (crotchet), and the " . ", unsurprisingly, that it is dotted. It should now be possible to read the melody quite straightforwardly. (The apparently simplistic reciprocal notation for durations is unexpectedly flexible, coping with most things short of Ferneyhough; triplet quavers, for instance, work out as 6, quintuplet semi-quavers as 20). That explains everything in Figure 6.4, except the " *− ", which marks the end of the record. It should also explain everything in Figure 6.5, which shows the same phrases (with the comments omitted), as set in G major by Haydn in the second movement of his String Quartet Op. 76 no. 3: instead of the single column (or *spine*) of Figure 6.4, there are now four, one for each line of the

```
!!!OTL: DEUTSCHLAND DEUTSCHLAND UEBER ALLES
!!!ARE: Europa, Mitteleuropa, Deutschland
!!!ARD: Europa%Mitteleuropa%Deutschland@
!!!ARL: 51.5/10.5@
!!!SCT: B0001
!!!YEM: Copyright 1995, estate of Helmut Schaffrath.
**kern
*ICvox
*Ivox
*M4/4
*k[ b-]
*F:
{ 4.f
8g
=1
4a
4g
4b-
4a
=2
8g
8e
4f}
{ 4dd
4cc
=3
4b-
4a
4g
8a
8f
=4
2cc}
*-
```

Figure 6.4 *Kern* code for first two phrases of "Deutschland über alles."

music. It is helpful to think of this as being laid out like staff notation, only rotated by 90 degrees.

Kern is not as easy to read as *EsAC* (which, apparently, was designed to be sight read). And it would seem to be better adapted for linear textures than, say keyboard music—though in fact there is no limitation in this respect, partly because you can put more than one note on a single line within any one spine, and also because you can collapse spines into one another or create new ones at any point. It is also more flexible than it looks. In the first place, you don't have to specify everything in Figures 6.4 or 6.5; there is no need to encode phrases, or bar lines, or durations, or pitches (in fact the *only* mandatory lines are "**kern" and "*−"). In the second place, there is a large number of further codes and interpretations built into *kern*, al-

```
**kern      **kern      **kern      **kern
*k[ f#]     *k[ f#]     *k[ f#]     *k[ f#]
*G:         *G:         *G:         *G:
*clefF      *clefF      *clefC3     *clefF4
*M4/4       *M4/4       *M4/4       *M4/4
*tb24       *tb24       *tb24       *tb24
=0-         =0-         =0-         =0-
2r          2r          2r          2r
4.g         4.B         4G          4G
.           .           4r          4r
8a          8d          .           .
=1          =1          =1          =1
4b          4g          2r          2r
4a          4f#         .           .
4cc         4a          4d          4F#
4b          4g          4d          4G
=2          =2          =2          =2
8a          4c          8F#         4D
8f#         .           8A          .
4g          4B          4G          4GG
4ee         4cc         2r          2r
4dd         4b          .           .
=3          =3          =3          =3
4cc         4f#         4A          4D
4b          4g          4B          4G
4a          2e          4e          4C
8b          .           4G          4C#
8g          .           .           .
=4          =4          =4          =4
2dd         2d          2F#         2D
*-          *-          *-          *-
```

Figure 6.5 *Kern* code for Haydn, String Quartet Op. 76, no. 3, II, bars 1–4

lowing you for instance to include symbols for articulation or ornamentation, clefs, stem direction, or the number of staff lines, if any (and whether they should be colored or dotted). But more than that, **kern spines can be mixed with others, including, for example, such predefined representations as **text (for lyrics) or **IPA (lyrics, but now notated using the International Phonetic Alphabet), or new representations defined by the user: you might, for example, create spines labeled **bowing, to show how a given pattern is (or might be) bowed, or **timing, to encode the rhythmic nuances in a particular performance, or **heartbeat, to record listeners' physiological responses to hearing the music. Then again, you might not use **kern at all, but an alternative representation for pitch; predefined representations include **pitch (using the International Standards Organization format), **deg (scale degrees, as in *EsAC*), **solfg (French fixed-do solfège), **freq (frequency

expressed in cycles per second), or the self-explanatory **midi; any of these representations can be automatically translated to any other, provided of course that the relevant information is there to be translated (recall the problem with MIDI and enharmonic equivalents). There is also a particularly well-developed **fret representation, which turns pitch notation into tablature notation based on any specified number of strings and tuning, or of course you might always invent a new representation for some particular repertory or purpose.

And what does all this enable you to do? For a start, you can replicate the functions of ANA: to extract the melodic spine of Figure 6.4, for instance, you locate the downbeat of each bar (search for lines beginning "=" and then skip to the next line), skip over the duration figures, and extract the pitch symbol(s) to a new file. Or you could just as easily search an entire "BOEHME" database for melodic motions from ^4 to ^5 and establish how frequently they occur by comparison with the reverse motion (you would translate the database into **deg records in order to do this, and search for adjacent lines containing 4 and 5, skipping over irrelevant symbols like durations, and finally counting the number found). Or you could search all of Bach's chorales (the Riemenschneider collection is available in *kern* code) in order to establish how often the melodic leading-notes are approached from beneath, as against from above: to do this, you extract the spines containing the melodies, turn them into **deg records, and count the occurrences of "^7" as against "v7."[11] (This particular task is easier than it sounds, because **deg automatically codes the direction in which a given scale degree was approached, though not the size of the interval involved). Then again, shifting to a different level of complexity, you could analyze the relationship between a particular succession of vowels in song texts and the melodic or rhythmic contexts within which they occur. You could establish how far particular harmonic formations are correlated with specific metrical locations (and analyze the results by composer, period, or geographical origin). Or you could classify different melodic, harmonic, or structural contexts in Chopin's complete mazurkas, and use this as the basis for analyzing timing information extracted from a large number of recordings.

But the best way to see what Humdrum is capable of is to examine published studies that have used it, of which the majority (though by no means all) are by Huron and his coworkers; I shall briefly describe five. An article on the melodic arch in folksongs (Huron 1996) is based on the Essen collection and provides a direct comparison with Schaffrath's own work. Its purpose is to test systematically the frequent, but inadequately supported, claim that "arch" contours are prevalent throughout Western folksongs, and Huron points out the dangers of "unintentional bias" (1996: 3) when such generalizations are supported by a small number of possibly handpicked examples: they can be firmly established only through the analysis of data sets large enough to allow the drawing of statistically significant conclusions. He puts forward two basic ways in which folksongs might be said to exhibit arch-shaped contours—within each phrase, and overall—and systematically analyzes the Essen collection for each. For the first test, individual phrases throughout the collection were sorted according to the number of notes in them, and the average contour for each phrase-length computed; for the second, the average pitch of each phrase of each song was computed, and the resultant contour classified. The overall

conclusion was that there is a strong tendency toward arch-shaped contours on both measures (though a third approach, based on the nesting of arches within one another, was not supported).

A project like this might be described as regulative: it takes an existing musicological claim and tests it rigorously against a large body of relevant data, adopting criteria of significance and certainty that may be unfamiliar in musicology but are normal in data-rich disciplines. Two further studies illustrate the range of musicological claims that can be tested in this manner. A study carried out in collaboration with Paul von Hippel (Hippel and Huron 2000) focused on the "gap-fill" model of melody, according to which listeners expect melodic leaps to be followed by a change of direction; this is an important element of Narmour's "implication-realization" theory, which seeks to explain how listeners experience music on the basis of whether or not implications set up by the music are realized in what follows. What is at issue is not whether or not melodic leaps are usually followed by a change of direction, which is undoubtedly the case; it is whether or not this happens (as Krumhansl has concluded on the basis of experimental studies)[12] because of listener expectations. The alternative is that it is a trivial consequence of the limited registral range of vocal music, a possibility which von Hippel and Huron (2000: 63) explain by comparison with

> a simplified melody that is confined to a range of three adjacent pitches—for example, A, B, and C. In such a melody, the only skip available is between A and C. Upward, this skip must land at the top of the range; downward, it must land at the bottom. After a skip, therefore, there is no way for a melody to continue moving in the same direction. On the contrary, two of the three available pitches can be reached only by a reversal. Although most melodies have a wider range, they are subject to the same basic argument.

If this second possibility is the case, then gap-fill patterns will be no more common in real folksongs than in randomly generated melodies with otherwise comparable characteristics—and, to cut a long story short, this is exactly what von Hippel and Huron found. If their argument (which I have grossly simplified) is accepted, then it provides a striking instance of how studies based on large data sets can raise questions about conclusions derived even from experiments as meticulously controlled as Krumhansl's.[13]

A third study (Huron 2001) makes a similar kind of point in relation to "art" music. It reassesses Allen Forte's (1983) set-theoretical analysis of the first movement of Brahms's String Quartet Op. 51, no. 1—an application to high Romantic music of an approach originally developed for twentieth-century music, bringing with it an appearance of objectivity and rigor in stark contrast to conventional motivic descriptions of such music. (In other words, Forte's article is a late expression of the postwar culture of objectivity described in chapter 1, this volume.) Whereas a "motive" is simply a characteristic melodic figure that recurs prominently throughout a piece, Forte's "alpha" ic (interval class) pattern can be defined much more precisely: it is a particular intervallic configuration that may occur in any of the standard transforms of set theory (prime, inversion, retrograde, retrograde inversion), in any rhythmic configuration, and at any metrical point. So Huron proceeds to search systematically for Forte's "alpha" pattern, not only in the first movement of Op. 51,

no.1 but also in the first movements of Brahms's other quartets—a comparable repertory in which the "alpha" pattern, if it is really characteristic of Op. 51, no. 1, should be significantly less common. First, he demonstrates that, as defined by Forte, the "alpha" pattern is common in Op. 51, no. 1, but no more so than in the other quartets. However, if you restrict the search to instances of the pattern that begin on metrically strong beats and come at the beginning of slurs, then it is more prevalent in Op. 51, no. 1 than in the others. This is particularly the case if you look for the pattern only in its prime form, and even more so if you consider only those cases where the pattern coincides with a long-short-long rhythmic pattern. So the prevalence of Forte's "alpha" pattern is confirmed—but only under precisely those conditions that are captured by the conventional idea of a motive! The conclusion must be that the appearance of rigor in Forte's analysis as against traditional analytical descriptions is just that, an appearance.

Important and indeed salutary as this regulative function may be, it would be wrong to give the impression that computational analysis is good only for testing the validity of existing theoretical or analytical claims. It can also be used to undertake work that could hardly be achieved in any other way. A spectacular example is the correlation of musical features with geography, a project once again based on the Essen database (Aarden and Huron 2001). Recall that the Essen records include information concerning the geographical origins of the songs contained in them—sometimes down to the level of an individual town or village—and that the *kern* version of "Deutschland über alles" added longitudinal and latitudinal coordinates in the !!!ARL field (a new field defined specifically for Aarden and Huron's project). Standard Humdrum commands can be used to extract records from the database and output their coordinates; these are then used as input to a mapping program. By way of example, Figure 6.6 shows the distribution of major- and minor-mode songs within the Essen database; these maps were generated using the GEO-Music site (http://www.music-cog.ohio-state.edu/cgi-bin/Mapping/map.pl, not accessible at publication time), established as part of the project. It is hard to know what conclusions might be drawn from this comparison (other than that the major mode is considerably more widespread than the minor); the southward skewing of the major-mode data as against the minor simply reflects the three occurrences in Italy—hardly a basis for robust generalization—while the concentration in Germany and central Europe evident in both cases reflects the bias of the Essen database. A fully fledged musicological research project would no doubt involve dealing with more features at a greater level of detail. Nevertheless this example does indicate the potential for computational methods to draw together quite diverse kinds of information; it also illustrates the value of graphic presentation of the complex data generated by work in comparative musicology.

A final study (Huron and Berec, forthcoming) is perhaps even more suggestive in terms of possible musicological applications. Like Hofstetter's attempt (mentioned in chapter 1) to turn a claim about the "spirit of nationalism" into an empirically testable proposition, Huron's and Berec's starting point is a woolly, common-language concept: idiomaticism. How might you set about defining what is idiomatic on, say, the trumpet in such a way that a computer could make evaluations of just how idiomatic a particular piece of music is? In the first place, Huron and Berec say, idiomaticism is not the same as difficulty: a piece of trumpet music can be difficult but idio-

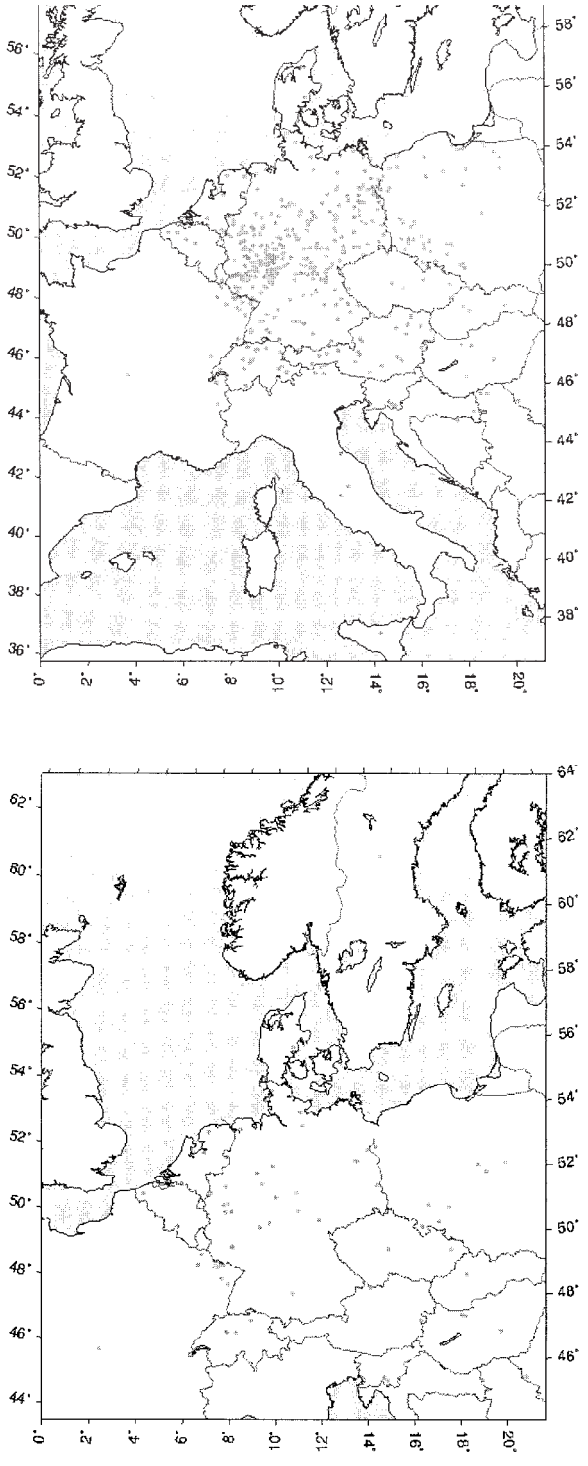


Figure 6.6 Distribution of major- and minor-mode songs (generated using the Geo-Music site)

matic, or easy and unidiomatic. It is rather a matter of achieving the desired musical effect in the easiest possible manner. For example, a piece will be idiomatic if it is significantly easier to play as written than when transposed up or down by say a semitone; this is "transposition idiomaticism." So Huron and Berec evaluated the difficulty of a number of pieces when played in all transpositions from an octave below the original to an octave above, and in several cases the original proved to be significantly easier to play than most or all of the transposed versions. (They also ran a parallel test based on how much more difficult the music was when played at a different tempo from the notated one, with similar results.) But there was also a further finding: the pieces had been selected so as to include examples by both trumpet virtuosi and nontrumpeters, and the conclusion in respect of both transposition and tempo idiomaticism was the same—that expert trumpeters compose more idiomatically than composers who cannot play the instrument.

That conclusion might be considered predictable, not to say obvious (just as in the case of Hofstetter's discovery that there are differences between national styles). And it would have been of limited interest if Huron and Berec had simply asked trumpeters to play the pieces at different transpositions and at different tempos, and to say how difficult they were, an approach that would not have required Humdrum, of course. But instead—and this is the real point of their study—they attempted something much more challenging: what they refer to as "the design and implementation of a computer model of a trumpet performer." Their model takes a *kern* file as its input, and evaluates the difficulty of performance along a variety of different dimensions: in terms of the valve transitions between successive notes; in terms of register, particularly in the case of sustained notes; in terms of dynamics; and in terms of tonguing at different speeds. (The information on which they based the model included performer assessments in the case of valve transitions, tests of performance in the case of tonguing, and physiological data concerning lung capacity and diaphragm support in the case of sustained notes.) The model was tested using a series of trumpet studies set for different grades by the Royal Conservatory of Music of Toronto; there was a generally fairly high correlation between its estimates of their difficulty and the Royal Conservatory's. But of course the main test was the evaluation of idiomaticism, and the fact that the model come up with what everybody knows confirms the extent to which it actually works. That is the real conclusion to be drawn from the study.

Do musicologists really *need* a computer model of a trumpet performer? One answer to this is that the approach could be readily generalized to other instruments: to the recorder (where finger transitions would be of particular importance, owing to the complexity of cross-fingerings), to the piano (the model might propose the best fingering, which could be tested against those specified by editors or used by performers), or to the guitar (in effect a modeling of the dance of the fingers on the fretboard). But the more substantial answer lies behind this. One of the chronic problems with analyzing music is the tendency toward abstraction: the more sophisticated our analytical models, the more divorced the object of analysis seems to become from the experience of music, and especially from the sense of physical engagement in which much if not all music has its source. It is conventional to deplore this, but to do nothing about it. In this context a computer model of the electric gui-

tarist's left hand could become, perhaps paradoxically, a means of thinking the body back into musical analysis: you could, to give just one example, chart the interaction of the dancing fingers with the ear—of the imperatives of the fretboard with those of "purely musical" implication and realization—in rock guitar improvisation. In this way computational musicology holds out the promise not only of providing more robust answers to questions that have already been asked, but also of making it possible to ask new questions.

Conclusion: Brave New World?

Promises, promises . . . and it must be admitted that computational software has a hardly better record of delivering on its promises than dot com companies. Let's stick with Humdrum as the most likely current candidate for a powerful, general-purpose computational aid for musicology, and explore some of the practicalities of doing musicology with it.

Q. Is it easy to get hold of?

A. Yes, at the time of writing you can download it for free, or you can buy it for the cost of the materials from the Center for Computer Assisted Research in the Humanities at Stanford.[14]

Q. Will it actually run on my computer? Don't I have to be using a UNIX machine?

A. You can run it on a PC or a Mac, but you will need to install a UNIX toolkit. You can probably get that for free, too.[15]

Q. But will I be able to find the music I want to work on in *kern* code?

A. That depends what you want to work on. The Essen package is available in *kern*, and so is some of the Densmore collection of Native American music, so if you are interested in folk music you can start work right away. As for Western "art" music, there is quite a lot out there: not only Bach's chorales but also most of his cantatas, as well as the *Well-Tempered Clavier*; substantial amounts of Corelli, Vivaldi, Handel, and Telemann; chamber and orchestral music by Haydn and Mozart; and most of the Beethoven symphonies.[16] The hope, of course, is that as more musicologists use Humdrum, so more music will be encoded, encouraging more musicologists to use Humdrum—and in this way you get a virtuous circle. (The same applies to the development of the tools themselves.)

Q. And if I can't find what I want?

A. If it exists in some other code you may be able to translate it into *kern*. Translation routines exist for the *MuseData* code used by the Center for Computer Assisted Research in the Humanities (CCARH), the Plaine and Easie Code used for RISM (*Répertoire international des sources musicales*), and *MUSTRAN*, as well as the code used by Leland Smith's music notation program SCORE, and the "Enigma" code used by Finale. Not all of these translators are readily available or unproblematic, but if you still can't find what you want you can of course encode your own. Rather than doing it by typing in the code, you can use an encoding routine that

enables you to enter the music on a MIDI keyboard—rather like step-time entry in a sequencer package. There are also utilities for playing *kern* code (via MIDI), for displaying music notation on screen, or for outputting it as a PostScript file (though not all of these are available if you are running Humdrum on a PC or a Mac). You can turn *kern* files into *Finale* ones for printing, too.

Unfortunately, none of this addresses the real difficulties of integrating Humdrum into everyday musicological life. The fact is that not everybody is happy with a UNIX command-line environment, or finds it easy to remember the different commands with their multiple (and often complex) options; people who use UNIX (or Humdrum) on a daily basis can operate it much more quickly than a graphic environment (like Windows, with its drop-down menus and mouse control), but if Humdrum is to become part of everyday musicological life then what matters is its accessibility to the *occasional* user. Michael Taylor (1996) has developed a graphical user interface (GUI) for Humdrum, though it is not publicly distributed at the time of writing,[17] and this makes life considerably easier for the occasional user: there are drop-down menus listing the various Humdrum commands and dialog boxes where you set the associated options, and you can even select musical elements using ordinary language ("any bar line," "at least one flat," or "C major chord," for instance).[18] For nonspecialist users such an interface—if generally available and supported—would surely represent a substantial step in the right direction: it not only looks familiar but also constantly reminds you of the commands and options that are available, without significantly compromising Humdrum's functionality and flexibility.

But such an approach can only go so far. This is because, to use Humdrum at all, you need quite a detailed knowledge of its representations (*kern* and the rest); a great deal of Humdrum usage has to do with things like extracting the spine you are interested in, merging spines, or stripping out information that is not wanted for any particular operation—and even ordinary-language definitions cannot protect you from the need to understand what is going on in the code. For this reason Andreas Kornstädt (1996) advocates a different approach: instead of "a *Humdrum* 'command center' with lots of buttons and gadgets with names like `yank` and `MIDI|smf`," as he puts it (1996: 119), he has developed an analytical environment into which different GUI modules can be slotted for different applications. Each module can be thought of as a "browser" that let users view just those musical elements that they are interested in, and no others; for instance, Kornstädt has created such a system for leitmotivic analysis, in which users can define leitmotifs and locate their occurrence in a score on the basis of on-screen notation and intuitive commands. This, again, is not publicly available, but another of Kornstädt's customized applications of Humdrum is: an on-line thematic dictionary, called Themefinder.[19] You type in some pitches from the tune you want to find (or alternatively you can type in the scale degrees, or the intervals between the notes, or even just its contour), and the software finds all the matches in its database and displays them on screen in musical notation. There are no Humdrum commands; you don't even need to know it is Humdrum that is doing the work—and what makes this possible is that Themefinder is designed to do one thing and one thing only. It is hard at the present time

to tell how far Kornstädt's approach—the development of specialized systems for different analytical purposes—will prove to be compatible with the flexibility and unpredictability of real-world musicological research, or whether it represents a particularly sophisticated way of falling into the same traps as 1980s office software. There is also the question of how the substantial costs of developing and updating a comprehensive system of this kind are to be found.

Maybe, in the coming years, the combination of user-friendly interfaces for Humdrum and a developing body of musicological work using it will provide the necessary momentum for it to become more generally used, so that in time it will become just one of those skills that musicologists have to acquire (like using notation software, for example). Or maybe that is just not going to happen; in that case, computational software like Humdrum might still become part of the wider musicological scene, but as a specialist professional service, rather than a skill you acquire for yourself (rather like note processing was a decade or so ago). In other words, if you had a project in which computational approaches could play an important part—in generating or testing hypotheses on the basis of a large body of data, for instance—then you would call in the services of a specialist computational musicologist, who would arrange for any necessary encoding of data, carry out the analysis, and provide advice on the interpretation of the results. What the ordinary musicologist would need to know, then, would be not how to extract or collapse Humdrum spines, but how to formulate a research question in such a way that advantage can be taken of computational methods to answer it more securely than is possible using the methods traditional in data-poor fields. Whether anyone could make a decent living as a specialist computational musicologist is another matter.

Using computers for musicology is like using computers for anything else: you need to have reasonable expectations about what they can do. However open and flexible its design, any musicological software is a tool, and like any tool it is good for some things and not for others. The articles by Huron and his coworkers that I have described in this chapter might be characterized as finding musicological uses to which computational tools can be put; in essence, they illustrate the software's potential, and they do so in a manner that owes more to social-scientific discourse than to traditional humanities writing, with their explicit hypotheses, control groups, quantification of significance, and formulation in what Huron (1999b) has termed the "boilerplate language" of scientific inquiry. Perhaps that is only to underline the distinction between "cognitive" or "systematic" musicology, as such work is sometimes called, and the close attention to context and what might be termed epistemological pluralism of the traditional discipline. But what I am suggesting is that musicology in the broadest sense can take advantage of computational methods and transform itself into a data-rich discipline, without giving up on its humanist values. Understood that way, scientific discourse is unlikely to offer a general model for the musicology of the future: it is up to musicologists to develop such a model. And that does not mean trying to find musicological uses to which computational tools can be put, but the reverse: discovering the tools that are most appropriate to a particular musicological purpose, and taking full advantage of them. At present we hardly have such a thing as a model of how sophisticated computational approaches might be integrated within a sustained, musicologically driven project. Yet the tools are ready and waiting.[20]

Notes

1. The routines used to create these graphs, which took advantage of the NeXT computer's Postscript display, have not to date been disseminated, but the principles underlying them were described in chapter 20 of Brinkman 1990.
2. Helmut Schaffrath, *Essen Musical Data Package* (Menlo Park, Calif.: Center for Computer Assisted Research in the Humanities [CCARH], 1995) [data and analytical software on four floppy discs for MS-DOS]); 6,225 folksongs were included in this release. Both data and software are now available at www.esac-data.org. (All URLs were correct at press time, but Web resources change rapidly. Search engines may provide a better means of access to the resources listed in this chapter.)
3. A full listing of the analytical output for "Deutschland über alles," reformatted and annotated for clarity, may be found in Selfridge-Field 1995; see also Schaffrath 1992, 1997.
4. Since *EsAC* consists entirely of ASCII characters, it is possible to import the records into modern databases. However researchers now generally prefer to access the *kern* version of the databases and process them using Humdrum (see note 7 below).
5. Hence the need for translation programs such as POCO (see chapters 5 and 8, this volume).
6. Digital Alternate Representation of Musical Scores; for details see Selfridge-Field 1997, chapters 11–15.
7. David Huron, *The Humdrum Toolkit: Software for Music Researchers* (Stanford, Calif.: Center for Computer Assisted Research in the Humanities, 1993 [three floppy discs and 16-page installation guide]). A wide range of information may be accessed from the Humdrum Toolkit home page (http://dactyl.som.ohio-state.edu/Humdrum; links to on-line resources are at http://dactyl.som.ohio-state.edu/Humdrum/resources.html).
8. UNIX is an operating system, like MS-DOS or Windows, but includes a large number of general-purpose tools and thus fulfils many of the functions of a programming environment as well. Developed in the late 1960s, it remains the environment of choice for many professional software developers and researchers.
9. For a summary of *kern* see Huron 1997; for detailed descriptions see Huron 1995, 1999a.
10. Lines beginning with asterisks are known as "interpretation records," those with two asterisks being *inclusive* interpretations (something is either in *kern* or it isn't), and those with one asterisk being *tandem* interpretations (of which you can have any number).
11. In a review of the Humdrum Toolkit, Jonathan Wild (1996) includes what is essentially a tutorial on this precise task; assuming that the chorales have been concatenated into one file, the entire analysis consists of seven commands pipelined to another—in other words, it can be executed with a single command line. Further analytical applications of Humdrum are discussed in Huron 2002, which appeared after this chapter had gone into production.
12. See this volume, pp. 7–8.
13. A follow-up study (Hippel 2000) reanalyzed the experimental data on the basis of which Burton Rosner and Leonard Meyer had claimed that gap-fill patterns reflect listener expectations; von Hippel concludes that that the apparent effects of gap-fill expectations were an artifact of experimental design (specifically, effects of training). Of course listeners may have such expectations, but von Hippel's and Huron's point is that these are the result of melodic patterns and not the other way around.

14. See the Humdrum download page at http://dactyl.som.ohio-state.edu/HumdrumDownload/downloading.html. Additional tools developed by Craig Sapp can be downloaded from http://www.ccarh.org/software/humdrum/museinfo.
15. UWIN (free to educational/research users) is accessible through the Humdrum download page (see n. 14). Commercial alternatives are available from Morton Kern Systems (MKS Toolkit, like UWIN for Windows) or from Tenon Intersystems (for Mac). There is a further alternative: a number of Humdrum commands can be run online at http://musedata.stanford.edu/software/humdrum/online; you select the command you want, supply the input data (by pasting it into a window, uploading a file, or supplying its URL), set the options, and receive the output as plain text or in HTML. Twenty of the 70 or more Humdrum commands are currently available in this way, with an emphasis on data translation and conversion. (Not accessible at publication time.)
16. There are two principal sources for such material (from both of which they may be obtained free): the KernScores site at http://kern.humdrum.net, and CCARH's MuseData site (http://www.musedata.org).
17. "Humdrum Toolkit GUI" currently exists only in a 16-bit version, but further development (and eventual distribution) is planned by the Sonic Arts Centre at Queen's University, Belfast.
18. Only a few of these "regular expressions," as they are known in UNIX jargon, are built into the interface, but you can add your own, along with new interpretations; this is where you would define **timing or **heartbeat.
19. Accessible at http://www.themefinder.org; for a brief description see Kornstädt 1998.
20. My grateful thanks to David Huron, who has influenced this chapter in many ways, not least through making it possible for me to work with him in 2000 as a visiting scholar at the Cognitive Musicology Laboratory, Ohio State University, Columbus.

References

Aarden, B., and Huron, D. (2001). "Mapping European folksong: Geographical localization of musical features," in Walter B. Hewlett and Eleanor Selfridge Field (eds.), *The Virnal Score: Representation, Retrieval, Restoration* (Computing in Musicology 12). Cambridge, Mass.: MIT Press, 169–183.

Barlow, H., and Morgenstern, S. (1983). *A Dictionary of Musical Themes*, rev. ed. London: Faber.

Brinkman, A. R. (1986). "Representing musical scores for computer analysis." *Journal of Music Theory* 30: 225–275.

Brinkman, A. R. (1990). *Pascal Programming for Music Research*. Chicago: Chicago University Press.

Brinkman, A. R., and Mesiti, M. R. (1991). "Graphic modeling of musical structure." *Computers in Music Research* 3: 1–42.

Clendenning, J. P. (1995). "Structural factors in the microcanonic compositions of György Ligeti," in E. W. Marvin and R. Hermann (eds.), *Concert Music, Rock, and Jazz since 1945: Essays and Analytical Studies*. Rochester, N.Y.: University of Rochester Press, 229–256.

Forte, A. (1983). "Motivic design and structural level in the First Movement of Brahms's String Quartet in C Minor." *Musical Quarterly* 69: 471–502.

Hippel, P. von (2000). "Questioning a melodic archetype: Do listeners use gap–fill to classify melodies?" *Music Perception* 18: 139–153.

Hippel, P. von, and Huron, D. (2000). "Why do skips precede melodic reversals? The effect of tessitura on melodic structure." *Music Perception* 18: 59–85.

Hughes, M. (1977). "[Schubert, Op. 94 No. 1:] a quantitative approach," in M. Yeston (ed.), *Readings in Schenker Analysis and Other Approaches*. New Haven: Yale University Press, 144–164.

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, 5–39.

Huron, D. (1995). *The Humdrum Toolkit: Reference Manual*. Menlo Park, Calif.: Center for Computer Assisted Research in the Humanities [accessible online at http://dactyl.som.ohio-state.edu/Humdrum/commands.toc.html].

Huron, D. (1996). "The melodic arch in Western folksongs." *Computing in Musicology* 10: 3–23.

Huron, D. (1997). "*Humdrum* and *Kern*: Selective feature encoding," in E. Selfridge-Field (ed.), *Beyond MIDI: The Handbook of Musical Codes*. Cambridge, Mass.: MIT Press, 375–401.

Huron, D. (1999a), *Music Research Using Humdrum: A User's Guide*. Stanford, Calif.: Center for Computer Assisted Research in the Humanities [accessible online at http://dactyl.som.ohio-state.edu/Humdrum/guide.toc.html].

Huron, D. (1999b). "Methodology: On Finding Field-Appropriate Methodologies at the Intersection of the Humanities and the Social Sciences." 1999 Ernest Bloch lectures, University of California at Berkeley, #3 [accessible online at http://dactyl.som.ohio-state.edu/Music220/Bloch.lectures/3.Methodology.html].

Huron, D. (2001). "What is a musical feature? Forte's analysis of Brahms's Opus 51, No. 1, Revisited." *Music Theory Online* 7.4.

Huron, D. (2002). "Music information processing using the Humdrum Toolkit: Concepts, examples, and lessons." *Computer Music Journal* 26/2: 11–26.

Huron, D., and J. Berec (forthcoming). "Characterizing idiomatic organization in music: A theory and case study." *Journal of New Music Research*.

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." in Walter B. Hewlett and Eleanor Selfridge-Field (eds.), *Melodic Similarity: Concepts, Procedures, and Applications* (Computing in Musicology, 11). Cambridge, Mass.: MIT Press, 231–236.

Schaffrath, H. (1992). "The retrieval of monophonic melodies and their variants: Concepts and strategies for computer-aided analysis," in A. Marsden and A. Pople (eds.), *Computer Representations and Models in Music*. London: Academic Press, 95–109.

Schaffrath, H. (1997). "The *Essen Associative Code*," in Eleanor Selfridge-Field (ed.), *Beyond MIDI: The Handbook of Musical Codes*. Cambridge, Mass.: MIT Press, 343–361.

Selfridge-Field, E. (1995). *Essen Musical Data Package* (CCARH Technical Report No. 1), Menlo Park, Calif.: Center for Computer Assisted Research in the Humanities [distributed with the *Essen Musical Data Package*].

Selfridge-Field, E. (1997). *Beyond MIDI: The Handbook of Musical Codes*. Cambridge, Mass.: MIT Press.

Taylor, W. M. (1966). "*Humdrum* Graphical User Interface." M.A. dissertation., Music Technology program, Queen's University of Belfast.

Wild, J. (1996). "A review of the Humdrum Toolkit: UNIX tools for music research, created by David Huron." *Music Theory Online* 2.7.