Chapter 39

Trouble-Shooting

Computers have an unbounded capacity to generate nonsense. Even when commands appear to execute correctly, there is no guarantee that the results are accurate or meaningful. In this chapter we will identify some of the many things that can go wrong when using the Humdrum Toolkit. We will also present a number of suggestions and tips that will help you avoid potential problems.

Errors can arise from a number of sources including corrupt or error-prone input data, failure to anticipate special circumstances or exceptions, improper processing, software bugs, and incautious interpretations of results.

Encoding Errors

In the first instance, the accuracy of your results will depend on the accuracy of the input data. Humdrum data may originate from a variety of sources. Users may encode their own materials, or use existing data encoded by other individuals or available from institutional sources. Data quality can be highly variable and there may be no easy way to determine the accuracy of a given data set.

It is important to spend time with a data set. Historical musicologists may spend a considerable amount of time becoming familiar with a manuscript, and the same practice is recommended for computational musicologists. Users should look at the data, listen to the data, compare the data to published sources, and generally browse and peruse it. Most data errors are discovered while processing the data — such as finding a suspicious major ninth melodic interval in a simple song. Over time, more and more errors are eventually discovered and corrected. Unfortunately, there is no magic flag that pops up to notify us when all errors have been eliminated from an encoded musical work. Only over time will the user gain confidence (or lose confidence) in a given data set. In working with a file, we are far more apt to discover something is wrong with the data than to learn that the data is a pristine encoding.

Errors can be magnified by the type of processing that is applied. For example, consider the case of an encoded repertory that is known to have a pitch-related error rate of 1 percent. That is, roughly 1 out of every 100 notes has an incorrect pitch representation. If we were to do an inventory of pitches in this repertory, then our results would also exhibit a 1 percent error rate. For many applications, such errors are not a problem.

However, consider what happens when we create an inventory of melodic intervals. One incorrect pitch will falsify *two* melodic intervals, hence the error rates for intervals is now 2 percent. Similarly, if we are looking at four-note chords, a single wrong pitch will falsify an entire chord. So we will have roughly a 4 percent error rate for chord identification. If we are investigating simple chord progressions, a single wrong note will now disrupt the identification of two successive chords. Thus we have an error rate of 8 percent for chord progressions.

There are two general lessons that can be drawn from these observations. The first lesson is obvious. Always try to use the best quality data that is available. When encoding your own data, aim for total accuracy. The second lesson is more subtle. The more data that participates in identifying some pattern, the greater the likelihood that a single data error will cause a problem. Whenever possible try to restrict pattern searches to small or concise patterns.

Searching Tips

Many of the problems in computer-based musicology are evident when searching for some pattern. In general, there are two types of searching errors: *false hits* and *misses*. A *false hit* occurs when the search returns something that is not intended. A *miss* occurs when the search fails to catch an instance that was intended to be a match. Unfortunately, efforts to reduce the number of false hits often tend to increase the number of misses. Similarly, efforts to avoid misses often tend to increase the number of false hits. Precision and caution are necessary.

Search failures can arise from five sources: (1) corrupt or inaccurate data, (2) failure to search all of the intended data, (3) inaccurate or inappropriate definition of the search template, (4) failure to understand how a given search tool or option operates, (5) failure on the part of the user to form a clear idea of what is being sought. Let's deal with each of these problems in turn.

(1) No search can produce accurate results if the data to be searched is inaccurate. You can increase the accuracy of your search by choosing high quality data and preparing the files in an appropriate manner.

Tips:

- Use the **humdrum** command to ensure that the input data conforms to the Humdrum syntax.
- Use the **humdrum -v** command to determine whether the kind of data (signifiers) you are interested in are truly present in all of the files to be searched.
- Use the **proof** command to ensure that any `**kern` data is properly encoded.
- Visually inspect sample passages from the input data. Do not rely solely on the **ms** command; instead, look at the actual ASCII text data using the **more** command.
- Use the **midi** and **perform** commands to listen to sample passages; ensure that the data makes sense.
- If you are uncertain of the quality of the data, try encoding a randomly selected subset and then use the UNIX **diff** command to identify any differences between the original data and the re-encoded data.

- Always read any release notes or README files that accompany the data.
- Use **grep** to search for *warning* and *Nota Bene* reference records (`!!!RWG:` and `!!!ONB:`). These records may contain important editorial notes or warnings.
- Where appropriate, expand files to through-composed versions (using **thru**) before searching. If more than one editorial version is present in a document, select the most appropriate edition before processing.
- Create an inventory of all the types of data tokens present in an input. Inspect the inventory list to determine whether any unexpected data are present.
- If necessary, eliminate certain types of data that might confound or interfere with your search in some way. Use **rid, grep -v**, **extract -i, sed** and/or **humsed** to restrict the data.

(2) Ensure that you are searching all of the intended data:

- Use **grep** to search for titles, or composers, or opus numbers, etc., in order to ensure that the file or files you are searching are the ones you want. Also check to ensure that you are not searching materials that don't belong in the input.
- Be wary of searching duplicated materials. Create inventories of titles, opus numbers, etc., and use **uniq -d** to identify unwanted duplicate copies of works or files.
- Use the **ls -l** or **wc** commands to determine the size of the search data. Does the amount of input data seem unduly small or unduly large?
- Use the **find** command to search the system for other files that ought to be included in the search task.

(3) One of the most common problems in searching arises from inaccurate or inappropriate search templates.

Tips:

- Be careful when formulating regular expressions. Read aloud the meaning of the regular expression.
- Do not use *extended* regular expressions with the **grep** command. Use **egrep** instead.
- Ensure that you know which characters in your search template are meta-characters.
- Execute your command from a shell script file so that you don't inadvertently make a typing error when entering the command.
- Maintain a command history file so that you have a permanent record of what you did. Depending on the system settings, the UNIX **history** command will display the past 100 (or more) commands you have executed. Place this information in a permanent `record` file as follows:

  ```
  history > record
  ```

  In addition, keep records of the precise regular expressions used for a given project. These records will help you determine later whether you made a mistake. For added security, print-out these files and glue them into a lab book.

- Create a test file containing different patterns, and test the ability of your regular expressions to catch all cases. Included "lures" in your test — i.e., patterns that are close to what you want, but should be rejected.
- Use extra caution when using "not" logic. For example, the **grep** expression "not-A" (i.e. `[^A]`) will still match records containing the letter `A` as long as one non-A letter is present. The commands

```
        grep [^A]
and     grep -v A
```

  are *not* the same.
- Compare outputs from a search that you know ought to increase the number of false hits. Compare outputs from a search that you know ought to miss some sought patterns.
- Translate the data to another representation and repeat the search using a different pattern tailored to the new representation. The results should be identical.
- Maintain a file containing regular expressions you have tested so you can re-use them in later projects.
- Visually inspect the ASCII output to ensure that the results are correct. Remember that visual inspection will only help you identify *false hits.* Visual inspection of the output will not help you identify *misses.*
- Use the **midi** and **perform** commands to proof-listen to your output. Again remember that aural inspection will only help you identify *false hits.*
- Ask whether the output makes sense. Given the amount of music searched, does it make sense to find the number of occurrences found?
- Try making a slight modification to your pattern template — a modification that you know should produce a different result.
- Look for converging evidence. Try two or three contrasting approaches to ensure that the same answer arises for each approach. For example, try searching each part individually using the **extract** command.

(4) Ensure that you understand how a given search tool or option operates.

Tips:

- Remember that *extended* regular expressions require the use of **egrep** rather than **grep**.
- Re-read the documentation to ensure that each software tool does what you think it does.
- Refer to the examples given in the *Humdrum Reference Manual* in order to ensure that you understand what a given option does.
- Compare outputs using different options. Ensure that your selected option(s) is matching the correct pattern.
- Use the **humver** command to determine which version of the Humdrum Toolkit you are using. Ensure that the documentation pertains to the correct version.

- Read the "Release Notes" for the software you use. Known software bugs are often reported in such notes or in the documentation.
- Report discovered bugs to the software's author. Even if the software is not revised, other users should be informed of the problem.

(5) Perhaps the most onerous problems in pattern searching arises when the user fails to have a clear understanding of what is being sought:

Tips:

- Think carefully about the search problem. What precisely are you looking for?
- Inspect the input to familiarize yourself with various contexts and possible variants.
- Check your search by carrying out a manual search of a random subset of the data.

Compared with manual research, computer searches are impressively fast. However, don't let yourself be caught-up by the speed of interaction. Take your time and reflect on the problem being addressed. Formulate a search strategy away from the computer so that you have time to consider possible confounds.

Pipeline Tips

Apart from searching tasks, most Humdrum processing involves two or more software tools linked in a pipeline. Pipelines can obscure all sorts of processing errors.

Tips:

- Slowly assemble your pipeline by adding one software tool at a time. Visually inspect the output following the addition of each process.
- Start with a small volume of input data. Once you have some confidence in your pipeline use a *different* sample of input data. Again add one software tool at a time while inspecting the results at each stage.
- Use the UNIX **tee** command to generate files at intermediate points in the processing. Use the **assemble** command to align inputs and outputs at various stages in the processing.
- Execute your finalized pipeline from a shell script in order to avoid undetected typing errors.

Reprise

In research-oriented activities, it is essential to exercise care when relying on computer-based methods. Computers have an unbounded capacity to generate false results. Unfortunately, computer outputs often seem deceptively authoritative. Take your time and develop a coherent strategy for solving a particular problem. Test your materials and processes, and maintain good records of what you have done. For critical tasks, always use two or more independent methods to ensure that the results agree. In general, cultivate a skeptical attitude; wise users are wary users.