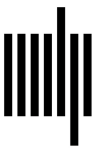
The Open Handbook of Linguistic Data Management

Edited By: Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller, Lauren B. Collister

Citation:

The Open Handbook of Linguistic Data Management
Edited By: Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller, Lauren B. Collister
DOI: 10.7551/mitpress/12200.001.0001
ISBN (electronic): 9780262366076
Publisher: The MIT Press
Published: 2022



The MIT Press

38 Managing Synchronic Corpus Data with the British National Corpus (BNC)

Stefan Th. Gries

1 Introduction

This chapter discusses data management and preparation issues that would arise in a fictitious corpus study using data from the British National Corpus World XML edition (BNC; BNC Consortium 2001); for overview and discussion of corpus linguistics as a field and its relation to notions such as *theory* and *method*, see the 2010 special issue of the *International Journal of Corpus Linguistics* (Pope 2010) and McEnery and Hardie (2011). This corpus consists of approximately 100 million words—4,049 files with 10 million words from spoken and 90 million words from written data—that were compiled to represent British English of the 1980s and is by now downloadable for free from the Oxford Text Archive (http://ota.ox.ac.uk/desc/2554). Specifically, for this chapter, I am discussing a hypothetical study of the so-called dative alternation between a ditransitive construction as in (1a) and the often-available prepositional dative with *to* in (1b); we will restrict our attention to sentences in the active voice.

(1) a. Captain Picard gave Commander Data a new phaser.
    b. Captain Picard gave a new phaser to Commander Data.

To study this kind of alternation, a corpus-linguistic analysis would typically begin from a concordance display that shows instances of each construction in context, as shown in a screenshot in the appendix. This is so that the user can read each example and annotate it for the large number of variables that seem to jointly affect the dative alternation. These include, but are not limited to, morphological, syntactic, semantic, information-structural, psycholinguistic, and other factors and have been identified in a large number of corpus-linguistic and quantitative studies of this alternation (see Gries 2003 for one of the earliest multifactorial studies and Bresnan et al. 2007 for the first one involving mixed-effects regression modeling). However, even if this alternation is fairly well understood by now, this example is still instructive for a variety of reasons:

- The BNC has been one of the most widely used corpora.
- Its XML annotation is fairly comprehensive and, on the morphosyntactic side of things, includes part-of-speech tags, some multi-word annotation, and lemma annotation.
- Its annotation does not include syntactic parse trees.

Given the BNC's annotation scheme and the absence of syntactic parses, retrieving syntactic constructions of the above-mentioned kind from the BNC is typically not possible in a fully automatic way and, therefore, involves the following, quite common corpus-linguistic search process, which will be discussed in what follows.

The user begins by running a query/search that is based on as much existing annotation as possible, here words/lemmas and parts of speech. For a study of the dative alternation, we will imagine that we want to find all instances of the dative alternation (with *to*) that involve one of the following ten verb lemmas that are frequently used in the dative alternation:

- Four verb lemmas that strongly prefer the ditransitive: *tell*, *give*, *show*, and *ask*.
- Three verb lemmas that strongly prefer the prepositional dative: *bring*, *sell*, and *pass*.
- Three verb lemmas that are relatively neutral with regard to the two constructions: *send*, *lend*, and *write*.

These preferences are based on Gries and Stefanowitsch (2004).

If the part-of-speech and lemma annotation is perfect (which would mean that instances of *show* or *shows* used as nouns would not be retrieved), a query/search for these verb lemmas will lead to perfect recall for these

ten verb lemmas with a user-defined context such as, for now, the complete sentence in which they are used. All their uses will be found and thus all their uses in the dative alternation (in corpora other than the BNC such as learner corpora, the user might have to deal with misspellings and other things). However, this result, the concordance lines, will come with fairly bad precision: all the verbs' uses will be found, in other words, also all uses in intransitive or monotransitive constructions or in phrasal verbs, prepositional verbs, and so on. Thus, the second step is to go over the concordance lines and prepare them for two kinds of annotation.

The first kind of annotation serves to identify false positives in the search result: that is, to identify the hits that involve the verbs but not the constructions in question so that we know which search results not to annotate for linguistic/contextual variables. However, given the size of the BNC and the relatively high frequencies of these verbs, we will not want to read all hits returned by the search, but only a subset/sample of them, and I will discuss ways to arrive at such a subset/sample (sections 2 and 3). Once the true positives—uses of forms of the verb lemmas that instantiate one of the two constructions—have been identified, the second kind of annotation is to (also usually manually) annotate each constructional use for the linguistic/contextual variables whose effect on the dative alternation is to be studied and to do that in such a way that facilitates subsequent statistical analysis; this part of the process is often partially outsourced to research assistants, which has some implications for the data management (section 3).

The final step in the process leading up to the actual analysis is to do some final checking and preparatory steps for the following statistical analysis (section 4). However, to make the whole endeavor as precise, consistent, and replicable as possible, I will make a variety of suggestions for this along the way; admittedly, some of these are general best practices for the management of corpus data and do not only apply to studies based on the BNC.

2 Retrieval

The first step of data management is to extract a first version of the concordance lines from, here, the BNC. The relevant part of the annotation of the corpus is represented in example 38.1 (see Han, chapter 6, this volume for more discussion of annotation) in which <u> is

```
<u who="D8YPS006">
<s n="80"><w c5="CJC" hw="and" pos="CONJ">And
</w><w c5="UNC" hw="erm" pos="UNC">erm </w><pause/><w
c5="DT0" hw="that" pos="ADJ">that </w>
<w c5="VBD" hw="be" pos="VERB">was </w><w c5="VVN"
hw="consider" pos="VERB">considered</w><c
c5="PUN">.</c></s>
</u>
<u who="D8YPS002">
<s n="81"><w c5="ITJ" hw="yes" pos="INTERJ">Yes </w><w
c5="CJS" hw="if" pos="CONJ">if </w><w c5="PNP"
hw="you" pos="PRON">you </w>
<w c5="VVD-VVN" hw="look" pos="VERB">looked </w><w
c5="PRP" hw="after" pos="PREP">after </w><w c5="AT0"
hw="a" pos="ART">a </w>
<w c5="NN1" hw="child" pos="SUBST">child</w><c
c5="PUN">.</c></s>
</u>
```

Example 38.1
Two one-sentence utterances from the BNC World edition, file D8Y.xml.

an utterance tag with a who attribute–value pair marking the speaker; <s> is a sentence number tag; <c> is a punctuation mark tag; and <w> is a word tag with two part-of-speech tags (a fine-grained "c5" version and a coarse-grained "pos" version) and a lemma tag ("hw" for head word). The corpus files are in UTF-8 encoding.

The BNC can be accessed online and with a variety of (free and commercial) corpus-processing tools, but these options restrict the analyst's freedom too much, so the best way to process corpus data is operating on a downloaded version with a programming language; many people are using Python, but I personally find R (R Core Team 2019) to be the altogether better choice (and Gries 2016 provides a detailed book-length introduction to corpus/text processing with R).

Recall, the task is to create a concordance of the ten above-mentioned verb lemmas from all of the BNC. There are two main ways this text processing/retrieval task can be approached in R: one is applying regular expressions to the files on a line-by-line basis, and the other is using packages such as XML (Lang & the CRAN Team 2019) or xml2 (Wickham, Hester, & Ooms 2018) that utilize the complete XML markup tree structure. In many cases, however, we want to retrieve the data in a way that maximally facilitates subsequent data

processing and statistical analysis, which means we want to end up with a file in the so-called case-by-variable, or long, format, which has the following characteristics:

- Every measurement of the dependent variable, every data point, to be studied—here, the constructional choice—gets its own row (in a spreadsheet-like representation).
- Every variable or every feature with regard to which each measurement/data point is annotated gets its own column (see Gries 2021: section 1.4 for more discussion of this format).

Given this secondary goal—getting as close as possible to the case-by-variable format—we will proceed with the regular expression option. This is because the output of the XPath queries in R offered by, say, the XML package do not return two output lines for two instances of the same verb (say, *give*) in the same sentence—at least not straightforwardly. For example, if there was a corpus sentence such as "Picard showed Data a phaser and then Data showed it to Riker," then the case-by-variable format requires that each use of *showed* is in its own row, as shown here in table 38.1, which is not what the XML package would immediately provide.

Without delving into actual R coding too much (again, see Gries 2016), the regular expression route means we could essentially proceed with two loops: one (outer) loop that loads every corpus file (using the right file encoding, which typically is UTF-8) so that its sentences can be searched for the verb lemmas in question, and an inner loop that retrieves every instance of each of the ten verb lemmas from the sentences with, here, the whole sentence as the context; note that often gathering more context can be essential, for example, to annotate discourse-functional/information-structural variables such as givenness/accessibility. Note the arrangement of the loops: because the loop that loads the files from the hard drive is the outer one, our script requires 4,049 hard drive accesses—if we had made the loop that loads the files the inner one, the one nested into the ten verb lemmas, we might need 40,490 file accesses, which would be considerably slower.

Table 38.1
The case-by-variable format for two matches in one sentence

Preceding	Match	Subsequent
Picard	showed	Data a phaser and then Data showed it to Riker
Picard showed Data a phaser and then Data	showed	it to Riker

As we are writing the script to gather this output, we should make sure that the output we are generating is more comprehensive than what table 38.1 suggests. For instance, the following pieces of information are "cheap" to obtain as we are doing the concordancing but could be useful or even vital either for data processing (e.g., sampling, sorting, filtering) or for the statistical analysis later. Thus, in addition to collecting the mere concordance data, there are some other kinds of information that are routinely useful to collect:

- Every concordance line should have a separate case number so that each case (i.e., row in the spreadsheet) can be uniquely identified by that number.
- It is often useful to include information about the circumstances of production of a data point: this could include register information, but it should minimally include the mode, in other words, whether the file contains spoken or written data, which we can extract either from the teiHeader in the first line of each BNC file or from the file's text type tag in the second line.
- We should not just retain the exact verb form found (as in the Match column in table 38.1) but also the verb lemma in a separate column (so that all forms of irregular verbs, such as *be* in a study of subject complementation, can be sorted together).
- It is nearly always useful or even necessary to save not only the file name in which a match was found, but also the line/sentence number. Many linguistic phenomena are subject to priming effects. That means the processes of planning to produce a construction are affected by whether that construction was processed before and how long ago that happened. Retaining the line/sentence numbers shown in figure 38.1 allows us to control for priming effects by computing the distance between two uses of a construction (see Gries 2018 and references discussed therein). (Along the same lines, it can be useful, in the case of spoken/conversational data, to include the speaker from the utterance tag in yet another column, which we will skip here.)
- Finally, it can be useful to immediately clear the output of parts of the annotation that are not going to be required anymore or that would make reading (during the subsequent manual annotation process) harder. In this case, we might delete part-of-speech tags, sentence

number tags (because we have the sentence number in a separate column anyway), and so on, but for some applications it might be useful to retain some tags such as overlap markers, unclear word tags (which might also indicate disfluencies), and others.

The final product is then ideally saved into a raw text file (a tab-delimited .csv file ideally still with UTF-8 encoding) that has column headers for all columns (which are separated by tab stops), and that has been cleaned up as well (e.g., no excess spaces anywhere); also, during any such steps, great care needs to be exercised to not compromise the identity and structure of the data. For instance, R, and also spreadsheet software, may need to be told explicitly how to handle single and double quotes, number/pound signs, and so on, which may occur in a corpus file, but must not disrupt R's/the spreadsheet software's parsing of the files column structure; this was also the reason why the columns in the output should be tab-delimited because tabs, unlike commas, are not part of the regular corpus file content and, thus, are no threat to recognizing the structure of the file (see ?read.table and ?write.table in R and check the settings of the Text Import Assistant in spreadsheet software).

Two related brief comments on preparing this output file: First, it is useful to do as much as of this as possible with code in an R script rather than manually or semi-manually (e.g., using database lookup functions) in a spreadsheet. This is because the output usually needs to be fine-tuned over multiple attempts and so it will save time and prevent errors if R does nearly everything with a script rather than when a human has to intervene over and over again manually with the point-and-click interface of a spreadsheet software.

Second, and more generally, it is always tempting to quickly hack together a script that somehow accomplishes the task, but I would encourage you to spend a bit time on thinking and planning this properly. One particularly relevant aspect is that it is often worth the extra five to ten minutes to think about whether (i) the code you're writing scales up to bigger research projects—if not, it is often worth the extra effort to change the code to make it run more efficiently, and (ii), relatedly, the code can be parallelized, that is, it can use the multiple cores or threads that contemporary computer processors now routinely offer.

As for the former, hacking together some code that just about works, but maybe inelegantly so (*elegance* not referring to aesthetics, but computational efficiency), might seem like "it's good enough for now," but it often happens that slight changes or additions need to be made as the scope of a study changes, among other reasons, and then having an elegant script is nearly always a good return on the investment (of the time that went into improving and streamlining the code).

As for the latter, running a script that does all of this on the BNC using R's default of just using a single thread of the computer's processor might take twenty minutes or more (depending on the user's hardware, obviously), but the script that I used for all of this used ten threads on a laptop with a 6-core Intel i5 processor and hyper-threading (using the R packages foreach and doParallel) and finished all the data retrieval and preparation discussed in this chapter within less than three minutes, a speed that even only a few years ago would have been nearly impossible to attain.

Section 3 deals with preparing and performing the annotation processes after the first concordance has been generated and saved.

3 Annotation

The next steps involve (i) preparing to weed out false positives and (ii) adding annotation with regard to the variables that might affect the dative alternation to the true positives. This part of the process is often done by research assistants, so it is useful to be maximally consistent and minimize the risk of errors in data entry, among other things, but, in all honesty, I have applied the same kind of precautions even in cases where I knew I was going to annotate the data myself. In a manner of speaking, I was protecting myself against my own errors, laziness, and such.

With regard to (i), in my experience it is most useful to prepare for the annotation by adding (still in the R script) an additional column to the data that is to the right of the column with the match that is called, say, Construction. That column can contain a placeholder for now, but it will contain the labels *ditrans*, *prepdat*, *other* for the constructions instantiated by the verb uses. Plus, it needs to be able to also contain some other code(s) to be able to, for instance, indicate that the row has been looked at but needs further attention (e.g., to disambiguate).

Also, I always recommend adding a column called Problem, whose only purpose is to (i) be empty if there is no problem whatsoever in any other column of the same row/case, but to (ii) contain the letter of the column that

does contain something problematic requiring further attention to disambiguate or that might lead to the case being discarded. For instance, if a match for one of the verb forms was found but something in the subsequent context column K makes you think that maybe this case should not be included, then the Problem column would contain the letter "K" to indicate that, because of column K, this line merits a second look. A lot of students, but also more senior practitioners, do something like this by changing the font color or the background of the problematic cell, but these are things that cannot easily be sorted by or find/filter in a spreadsheet that has, say two hundred thousand rows, but a code in a separate column is something that can be found/filtered/sorted by. Thus and more generally, if information needs to be added to the data, add it into a column, not with formatting, because only if information made it into a cell of a column can all the data processing power of R/spreadsheet software be applied to it efficiently.

Next, we need to select a sample of concordance lines to read to determine whether they actually instantiate one of the two constructions of the dative alternation. The biggest mistake to avoid here is to draw a random sample of, say five thousand concordance lines out of all concordance lines. While this practice is still widespread, it is a really bad idea for two reasons: First, many phenomena are susceptible to priming effects, which means that to analyze an example of a ditransitive in file EFS.xml, it is most likely necessary to see what construction was used last before that and how similar that use was to the current one. But if we sample randomly from all concordance lines, the current line will be separated from all others in the same file, which makes such annotation much harder than necessary. Second, many corpus studies of this type these days are analyzed with mixed-effects models or similar kinds of tools, one selling point of which is that they can control for speaker-, file-, or lexically specific variability in the data. However, if the one ditransitive in file EFS.xml that I am looking at right now is the only one that made it into my random sample (although there are actually many ditransitives and prepositional datives in there, which were just not sampled), then I am not giving my later statistical analysis the chance to determine whether there is something special to this speaker or file.

Thus, what we should do here is make the file the subsetting/sampling unit: choose files (pseudorandomly) to be part of our subset/sample and then look at all concordance lines from that file. Obviously, there are many straightforward ways in which such sampling might be implemented, so I will just mention two of them that provide an additional perspective on this part of the process. The first alternative could involve tabulating all files (in 4,049 rows) with all verb lemmas (in ten columns) to see which verb lemma is attested how often in each file. Then we could decide to only consider files for sampling that contain say, at least eight of the ten lemmas at least, say, twice. Of those files, we could begin with the file with the smallest number of verb lemma tokens and add all concordance lines from files—recall, the sampling unit are files, not lines—with successively more verb lemma tokens until we reach a desired number of concordance lines. The reason for this seemingly convoluted scheme is that (i) it would make sure that the files sampled contain a "decent variety" of relevant verb lemmas and that (ii) we sample a "decent" number of files (which makes sure that no one huge file and its idiosyncrasies could affect our analysis too much).

The second alternative could begin as we did for the first—identifying files with a decent number of matches to begin with—and then randomly sample complete files from those; in such a case, it is absolutely essential to set a random-number seed before any sampling is done (in R: set.seed) so that the sampling is random, but also replicable.

Finally, it is often helpful to sort the complete output in a useful way. This could be sorting by, for instance:

- Whether a file/concordance line is "in the sample" to be studied or not (so that all to-be-annotated items in the sample are together).
- The file name (so that all lines from the same file are together).
- The sentence number (so that all lines from the same file are in order of occurrence in the file).
- The length of the preceding context (so that multiple hits in the same sentence are sorted in order of occurrence in the line).

Ideally, everything so far in this process would be performed with a single fully automated script that, when run on the same data, would give you the same output and would require as little human intervention as possible (in the interest of speed and replicability). This would entail, for instance, that file locations (of the input and output files are hard-coded into the script). Also, the code/script should be extremely heavily commented,

which is useful if ever you want to share the script with others and which is *necessary* to remind your future self what you did and why a year ago (when you submitted the paper to that special issue). A useful check as well as documentation of all your activities during this stage, but also during the later statistical analysis, would be to generate two things: (i) a variety of output files (interim results for fast debugging as well as final results) in useful file formats (such as .rds for everything to be used only within R and tab-delimited .csv for everything that might be loaded into other software), and (ii) an HTML report or an R Markdown document that contains all your code, all your commentary, and all the results that would have been in the console/on the screen in a single shareable HTML file, which ensures proper error checking (because otherwise the report won't compile in the first place) and transparency (to others and your future self).

Then, and only then, do we stop using R for a moment and we can begin to actually annotate (i) whether concordance lines are the right construction(s) and (ii) what their characteristics are that might have affected the speaker's choice. For this part of the process, I recommend using a spreadsheet software such as LibreOffice Calc (which has characteristics such as full-fledged regular expressions and better filtering functionality that, to my mind, make it more useful than competing spreadsheet software; see the appendix for what we are aiming for). Crucially, I recommend using the Data: Validity functionality, which allows a user to limit the number of options that can be entered into a cell. Consider figure 38.1, which shows the first of three tabs of the menu option Data: Validity in LibreOffice Calc. In the Criteria tab, we can list the elements that the user is allowed to enter into the cells of column J, which are then shown in a drop-down selection list for entering with a mouse click; in the Input Help tab, we can enter a title and a help text that is shown when a user clicks on a cell to enter something; and in the Error Alert tab, we can enter what should happen and what feedback should come up when the user tries to enter something they are not supposed to enter.

This feature makes it much easier to avoid data entry errors because, for instance, we can define a list of admissible entries (as shown here), we can only permit numbers or dates, and such, and the pop-up help constantly reminds an annotator of all the possible options that are at their disposal—just make sure there also are options for the annotator to indicate they have a problem and cannot annotate a certain data point decisively yet.

Once all annotation of the relevant variables is complete, we turn to the last data management stage, the final steps before a subsequent statistical analysis.

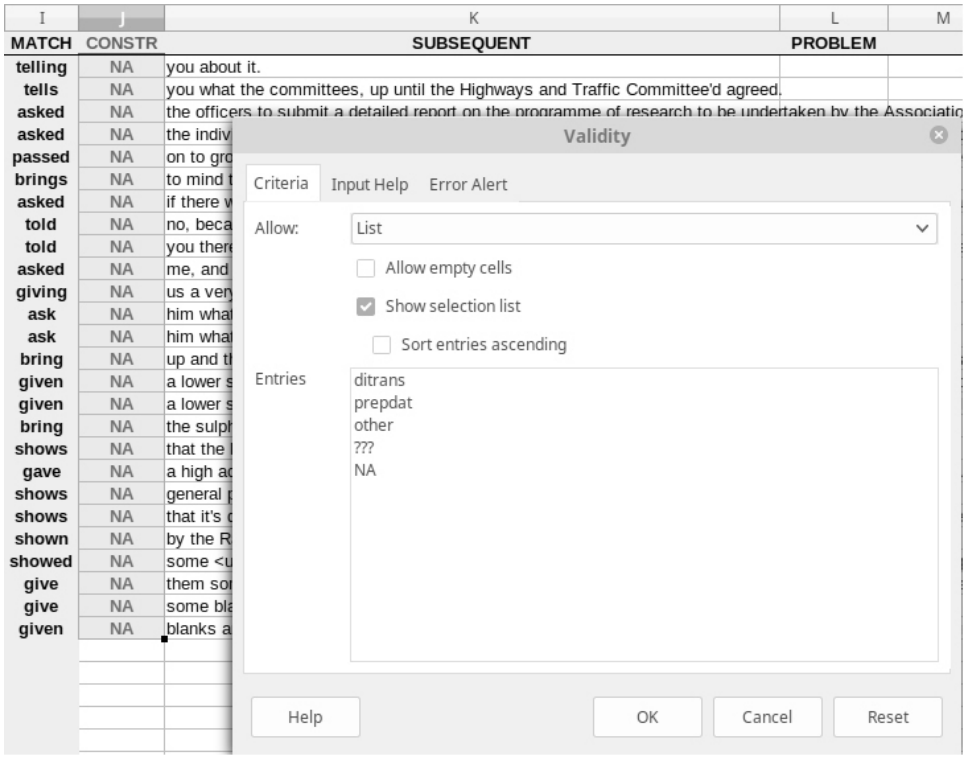


Figure 38.1
The use of the Data: Validity function to guide and constrain data entry.

4 Preparation and documentation

Once the relevant data points—the uses of the two constructions—have been identified and annotated for the factors targeted in the study (e.g., length, animacy, definiteness) there are several final things that need to be done to make sure the data are in good shape for subsequent (statistical) analysis.

The first of these is performing a general sanity check of the data as a whole but specifically the annotation that was entered. This can be done in two ways: in the spreadsheet software (or any more specialized annotation software) with which the annotation was performed and within the software used for statistical analysis (these days, typically, R). As for the former, I recommend using the spreadsheet's filtering function to determine whether a column contains only sensible entries, for example, only the animacy levels you intended to code or only reasonable ages of speakers. This is more important than you might think: I recently met with a student whose spreadsheet (with more than twenty thousand rows) contained a column named Age of Speaker (supposedly measured in years), which upon inspection was found to contain a variety of values exceeding 170. This arose from an unintentional use of dragging down a cell with a numeric value that was not repeated as intended, but incremented instead; obviously, these kinds of things need to be addressed prior to any analysis. Figure 38.2 shows how the Data: Autofilter functionality can be used to check, here, that the Lemma column contains only the lemmas it is supposed to contain.

Similarly, if data entry was not restricted with the Data: Validity tool, annotators often unintentionally add spaces to labels—for example, using the labels "animate" and "animate·"—which will create problems for the later statistical analysis. Microsoft Excel does not even flag this in its filtering function, but in LibreOffice Calc, this would be obvious from the kind of filtering display shown in figure 38.2. Therefore, we want to check for all sorts of other problems including plain typos ("animat"), other implausible values (decimal values in a column that should only contain integers such as length of an NP in words or characters), and many similar problems.

As for the latter kind of sanity check (in R), I recommend loading the .csv file into a data frame in R and explore it, minimally, with the summary function (to get first frequency tables of all categorical variables and numerical summaries of all numeric variables: if you intended to restrict your speakers to younger people and the median age value in the summary output is forty-eight, then you might want to look at your code and your (interim) output files again.

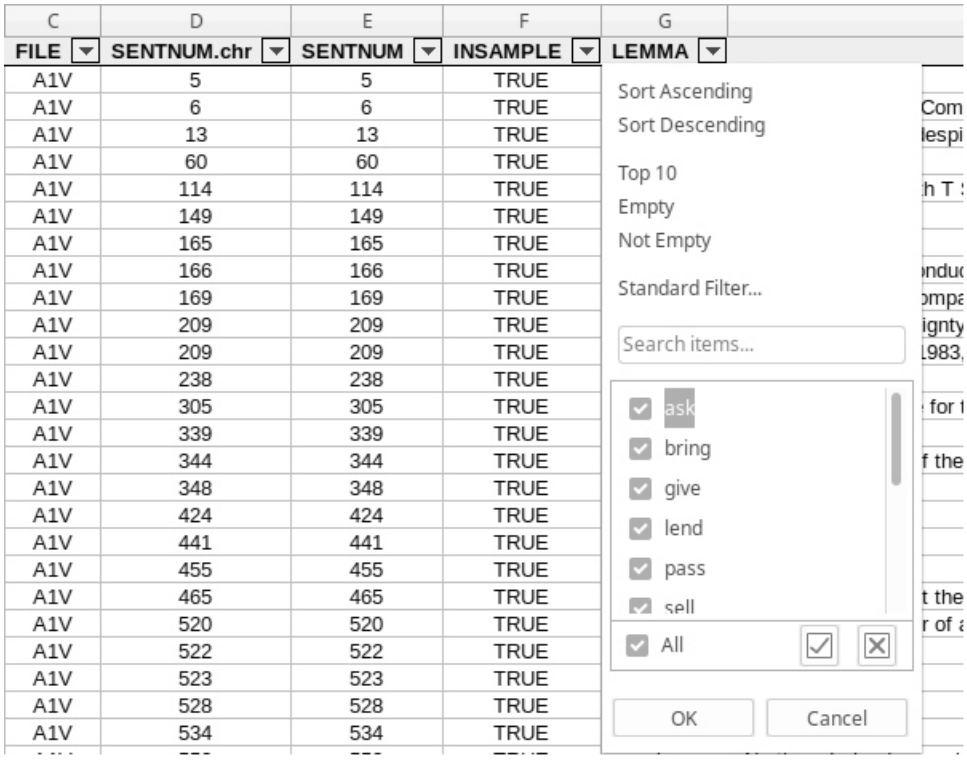


Figure 38.2
The use of Data: Autofilter to check data entry.

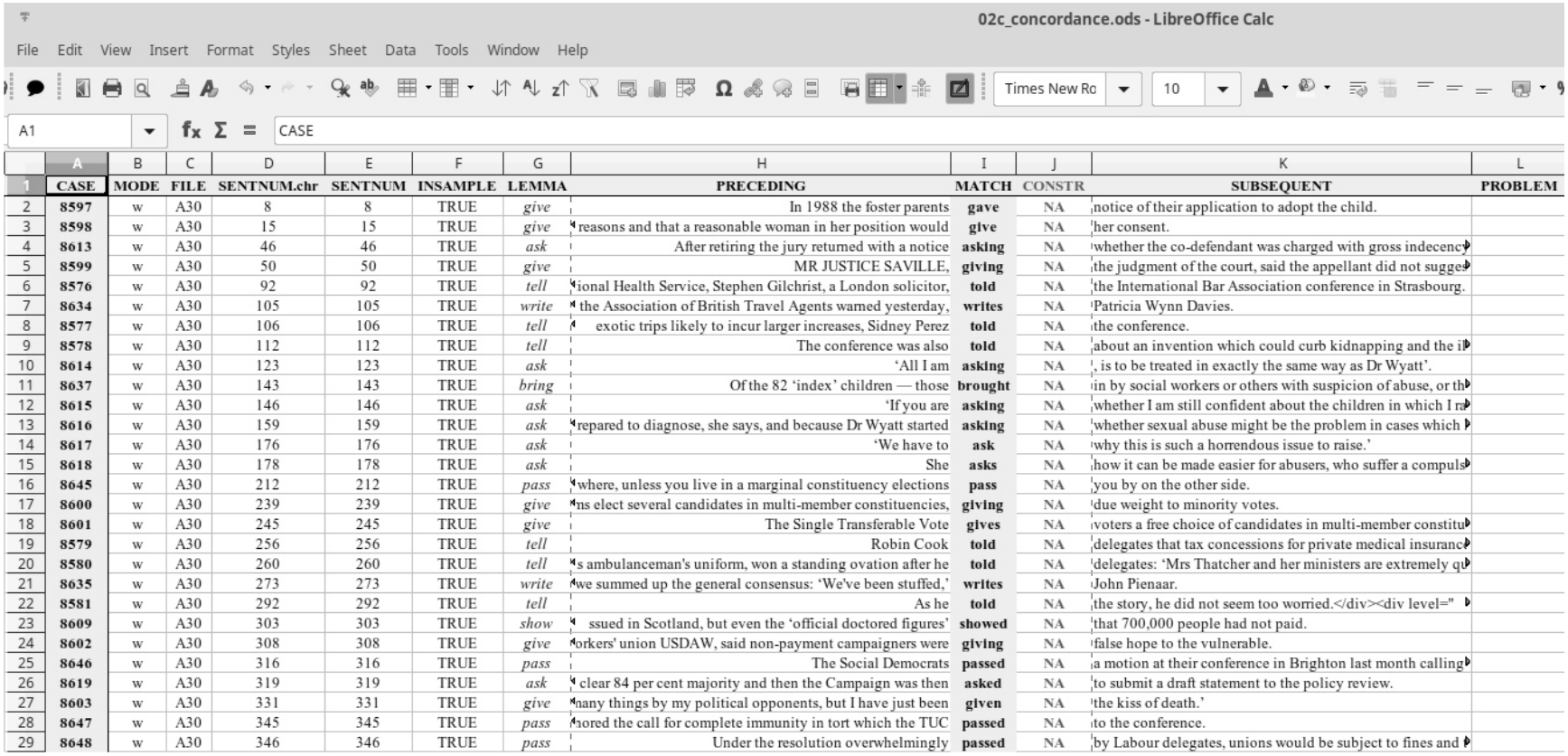
Finally, and this is beginning to move away from the core processing of corpus data per se and toward initial statistical processing, there are a few exploratory steps that have implications on the data structure you're working with. For instance, it is straightforward, but very useful, in R to quickly check for each column of your data how many unique types it contains and what their frequency distribution is and looks like when visualized in a statistical plot. This kind of information is useful because it often offers suggestions with regard to (i) which column (especially of categorical variables) might contain more unique values than it should, (ii) which column (especially of categorical variables) might contain more super-rare values than is useful for a subsequent statistical analysis, and (iii) which column (especially of numerical variables) has distributional characteristics that are problematic for whatever subsequent analysis was planned and thus needs to be transformed. If such exploration leads to new columns—because levels of a categorical variable are conflated or because numerical variables are log-transformed—it is usually a good idea to leave the old columns in the data frame in case you unexpectedly need to go back and use the original values again. However, if you have been doing what I've recommended—diligently documenting all steps in a code file and creating many interim results files—revisiting an earlier variable state should be unproblematic anyway. Note that sometimes you might even have to combine multiple columns into one to disambiguate information for the later analysis: Imagine a case where speaker IDs are simply numbers from 1 to *n* in each corpus file. This would mean that concordance line one thousand might be from a speaker labeled as "1," as might be concordance line two thousand—but the two concordance lines might be from different files! Thus, you would need to create a new column that conflates the file name and the speaker name into one string so that the former becomes, say, "D8Y.xml_1" and the latter becomes "EFS.xml_1." The BNC speaker codes include the file name, which avoids this problem, but it is useful to be on guard for such or similar situations; for instance, this careful separation of speaker codes is also important for experimental data.

A final comment that is more general than the specific scope of this chapter—data management with the BNC—but still so important that I feel compelled to make this point anyway: I strongly recommend the adoption of a rigorous practice of naming files and documenting workflow across files. Too often, I see students as well as senior practitioners presenting me with files called somescript.r, newscript.r, betterversion.r, and ten different output files whose names make no sense even to the users themselves anymore once even only a moderate amount of time has passed (see https://xkcd.com/1459/). My recommendations for a corpus study of the type I've discussed, with the BNC or any other corpus, are the following:

- Create a new folder for each project.
- Name nearly all files in the folder so they are numbered in order of creation and indicate what they produce.
- Name all files having to do with the paper or slides you produce from the data 01a_paper.odt, for example, with major successive revisions being called 01b_paper.odt, 01c_paper.odt, and so on. This makes sure they are shown at the top of the folder.
- Number all files involved in the analysis as follows: The script that generates the first concordance from the corpus is called 02a_concordance.r, which might generate an output file called 02b_concordance.csv. That file is then also saved as an .ods file 02b_concordance.ods, which is used for annotating the data and which, when the annotation is complete, is saved as 03_annotated.ods and, for R, as a tab-delimited version called 03_annotated.csv, and so forth.
- Note that the R script that reads this file and contains the statistical analysis of 03_annotated.csv is called 04a_eval.r and might produce output files 04b_. . . . csv via 04e_. . . . rds to 04h_. . . . png and so on.
- Create the first file in the folder and name it 00_overview.txt. This file contains your notes that state for every file in the folder (i) its name, (ii) what it does (and what its input files are), and (iii) its output file.

As obsessive as this may seem, corpus studies of large corpora often lead to huge amounts of results, not to mention the possibility that multiple slightly different attempts to come to grips with the data may have been made, which quickly can lead to an explosion of files. (I have seen many times, during office hours, the inability of a student to even locate "the current analysis.") In the interest of proper data management and, hopefully, the increase in precision, transparency, and replicability of our corpus studies, these kinds of scenarios—see http://phdcomics.com/comics/archive.php?comicid=1323—need to be avoided. If you follow the guidelines in this chapter, they will be; of course, you can also use git or similar tools.

Appendix



	CASE	MODE	FILE	SENTNUM.chr	SENTNUM	INSAMPLE	LEMMA	PRECEDING	MATCH	CONSTR	SUBSEQUENT	PROBLEM
2	8597	w	A30	8	8	TRUE	*give*	In 1988 the foster parents	gave	NA	notice of their application to adopt the child.	
3	8598	w	A30	15	15	TRUE	*give*	reasons and that a reasonable woman in her position would	give	NA	her consent.	
4	8613	w	A30	46	46	TRUE	*ask*	After retiring the jury returned with a notice	asking	NA	whether the co-defendant was charged with gross indecency	
5	8599	w	A30	50	50	TRUE	*give*	MR JUSTICE SAVILLE,	giving	NA	the judgment of the court, said the appellant did not sugge	
6	8576	w	A30	92	92	TRUE	*tell*	ional Health Service, Stephen Gilchrist, a London solicitor,	told	NA	the International Bar Association conference in Strasbourg.	
7	8634	w	A30	105	105	TRUE	*write*	the Association of British Travel Agents warned yesterday,	writes	NA	Patricia Wynn Davies.	
8	8577	w	A30	106	106	TRUE	*tell*	exotic trips likely to incur larger increases, Sidney Perez	told	NA	the conference.	
9	8578	w	A30	112	112	TRUE	*tell*	The conference was also	told	NA	about an invention which could curb kidnapping and the il	
10	8614	w	A30	123	123	TRUE	*ask*	'All I am	asking	NA	, is to be treated in exactly the same way as Dr Wyatt'.	
11	8637	w	A30	143	143	TRUE	*bring*	Of the 82 'index' children — those	brought	NA	in by social workers or others with suspicion of abuse, or th	
12	8615	w	A30	146	146	TRUE	*ask*	'If you are	asking	NA	whether I am still confident about the children in which I ra	
13	8616	w	A30	159	159	TRUE	*ask*	repared to diagnose, she says, and because Dr Wyatt started	asking	NA	whether sexual abuse might be the problem in cases which	
14	8617	w	A30	176	176	TRUE	*ask*	'We have to	ask	NA	why this is such a horrendous issue to raise.'	
15	8618	w	A30	178	178	TRUE	*ask*	She	asks	NA	how it can be made easier for abusers, who suffer a compuls	
16	8645	w	A30	212	212	TRUE	*pass*	where, unless you live in a marginal constituency elections	pass	NA	you by on the other side.	
17	8600	w	A30	239	239	TRUE	*give*	ns elect several candidates in multi-member constituencies,	giving	NA	due weight to minority votes.	
18	8601	w	A30	245	245	TRUE	*give*	The Single Transferable Vote	gives	NA	voters a free choice of candidates in multi-member constitu	
19	8579	w	A30	256	256	TRUE	*tell*	Robin Cook	told	NA	delegates that tax concessions for private medical insuranc	
20	8580	w	A30	260	260	TRUE	*tell*	s ambulanceman's uniform, won a standing ovation after he	told	NA	delegates: 'Mrs Thatcher and her ministers are extremely qu	
21	8635	w	A30	273	273	TRUE	*write*	we summed up the general consensus: 'We've been stuffed,'	writes	NA	John Pienaar.	
22	8581	w	A30	292	292	TRUE	*tell*	As he	told	NA	the story, he did not seem too worried. <div level="	
23	8609	w	A30	303	303	TRUE	*show*	ssued in Scotland, but even the 'official doctored figures'	showed	NA	that 700,000 people had not paid.	
24	8602	w	A30	308	308	TRUE	*give*	orkers' union USDAW, said non-payment campaigners were	giving	NA	false hope to the vulnerable.	
25	8646	w	A30	316	316	TRUE	*pass*	The Social Democrats	passed	NA	a motion at their conference in Brighton last month calling	
26	8619	w	A30	319	319	TRUE	*ask*	clear 84 per cent majority and then the Campaign was then	asked	NA	to submit a draft statement to the policy review.	
27	8603	w	A30	331	331	TRUE	*give*	nany things by my political opponents, but I have just been	given	NA	the kiss of death.'	
28	8647	w	A30	345	345	TRUE	*pass*	nored the call for complete immunity in tort which the TUC	passed	NA	to the conference.	
29	8648	w	A30	346	346	TRUE	*pass*	Under the resolution overwhelmingly	passed	NA	by Labour delegates, unions would be subject to fines and	

Figure 38.3
Concordance display.

References

BNC Consortium. 2001. *The British National Corpus, Version 2 (BNC World)*. Distributed by Oxford University Computing Services on behalf of the BNC Consortium. http://www.natcorp.ox.ac.uk/.

Bresnan, Joan, Anna Cueni, Tatiana Nikitina, and R. Harald Baayen. 2007. Predicting the dative alternation. In *Cognitive Foundations of Interpretation*, ed. Gerlof Bouma, Irene Kraemer, and Joost Zwarts, 9–94. Amsterdam: Royal Netherlands Academy of Science.

Gries, Stefan Th. 2003. Towards a corpus-based identification of prototypical instances of constructions. *Annual Review of Cognitive Linguistics* 1 (1): 1–27.

Gries, Stefan Th. 2021. *Statistics for Linguistics with R*. 3rd rev. and extended ed. Berlin: De Gruyter Mouton.

Gries, Stefan Th. 2016. *Quantitative Corpus Linguistics with R*. 2nd rev. and extended ed. London: Routledge, Taylor & Francis Group.

Gries, Stefan Th. 2018. Syntactic alternation research: Taking stock and some suggestions for the future. *Belgian Journal of Linguistics* 31 (1): 8–29.

Gries, Stefan Th., and Anatol Stefanowitsch. 2004. Extending collostructional analysis: A corpus-based perspective on "alternations." *International Journal of Corpus Linguistics* 9 (1): 97–129.

Lang, Duncan Temple, and the CRAN Team. 2019. XML: Tools for Parsing and Generating XML within R and S-Plus. R package version 3.98–1.20. https://CRAN.R-project.org/package=XML.

McEnery, Tony, and Andrew Hardie. 2011. *Corpus Linguistics: Method, Theory and Practice*. Cambridge: Cambridge University Press.

Pope, Caty Worlock, ed. 2010. The bootcamp discourse and beyond. Special issue, *International Journal of Corpus Linguistics* 15 (3).

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing. https://www.R-project.org/.

Wickham, Hadley, James Hester, and Jeroen Ooms. 2018. xml2: Parse XML. R package version 1.2.0. https://CRAN.R-project.org/package=xml2.