Phonological CorpusTools Documentation

Release 1.0.0

PCT

April 28, 2015

Contents:

Introduction

1.1 General Background

Phonological CorpusTools (PCT) is a freely available open-source tool for doing phonological analysis on transcribed corpora. For the latest information, please refer to the PCT website. PCT is intended to be an analysis aid for researchers who are specifically interested in investigating the relationships that may hold between individual sounds in a language. There is an ever-increasing interest in exploring the roles of frequency and usage in understanding phonological phenomena (e.g., [Bybee2001], [Ernestus2011], [Frisch2011]), but many corpora and existing corpus-analysis software tools are focused on dialogue- and sentence-level analysis, and/or the computational skills needed to efficiently handle large corpora can be daunting to learn.

PCT is designed with the phonologist in mind and has an easy-to-use graphical user interface that requires no programming knowledge, though it can also be used with a command-line interface,1 and all of the original code is freely available for those who would like access to the source. It specifically includes the following capabilities:

- Summary descriptions of a corpus, including type and token frequency of individual segments in user-defined environments;

- Calculation of the **phonotactic probability** of a word, given the other words that exist in the corpus (cf. [Vitevitch2004]);

- Calculation of **functional load** of individual pairs of sounds, defined at either the segment or feature level (cf. [Hockett1966]; [Surendran2003]; [Wedel2013]);

- Calculation of the extent to which any pair of sounds is **predictably distributed** given a set of environments that they can occur in, as a measure of phonological contrastiveness (cf. [Hall2009], [Hall2012]; [Hall2013a]);

- Calculation of the **Kullback-Leibler divergence** between the distributions of two sounds, again as a measure of phonological contrastiveness (cf. [Peperkamp2006]);

- Calculation of the extent to which pairs of words are **similar** to each other using either orthographic or phonetic transcription, and calculation of **neighbourhood density** (cf. [Frisch2004], [Khorsi2012]; [Greenberg1964]; [Luce1998]; [Yao2011]);

- Calculation of the **frequency with which two sounds alternate** with each other, given a measure of morphological relatedness (cf. [Silverman 2006]_, [Johnson2010], [Lu2012]);

- Calculation of the **mutual information** between pairs of segments in the corpus (cf. [Brent1999]; [Goldsmith2012]); and

- Calculation of the **acoustic similarity** between sounds or words, derived from sound files, based on alignment of MFCCs (e.g., [Mielke2012]) or of logarithmically spaced amplitude envelopes (cf. [Lewandowski2012]).

The software can make use of pre-existing freely available corpora (e.g., the IPHOD corpus; [IPHOD]), which are included with the system, or a user may upload his or her own corpus in either of two formats. First, lexical lists

with transcription and token frequency information can be directly uploaded; such a list is what is deemed a "corpus" by PCT. Second, raw running text (orthographically and/or phonetically transcribed) can be uploaded and turned into lexical lists in columnar format (corpora) for subsequent analysis. Raw sound files accompanied by Praat TextGrids [PRAAT] may also be uploaded for analyses of acoustic similarity. Orthographic corpora can have their transcriptions "looked up" in a pre-existing transcribed corpus of the same language. Limited support is currently also available for spontaneous speech corpora, where, for example, a user can create a corpus from TextGrids and include information about ngram word combinations.

Phonological analysis can be done using built-in feature charts based on Chomsky & Halle [SPE] or Hayes [Hayes2009], or a user may create his or her own specifications by either modifying these charts or uploading a new chart. Feature specifications can be used to pull out separate "tiers" of segments for analysis (e.g., consonants vs. vowels, all nasal elements, tonal contours, etc.). PCT comes with IPA transcription installed, with characters mapped to the two feature systems mentioned above. Again, users may create their own transcription-to-feature mappings by modifying the existing ones or uploading a new transcription-to-feature mapping file, and several alternative transcription-to-feature mapping files are available for download.

Analysis can be done using type or token frequency, if token frequency is available in the corpus. All analyses are presented both on screen and saved to plain .txt files in user-specfied locations.

The following sections walk through the specifics of downloading, installing, and using the various components of Phonological CorpusTools. We will do our best to keep the software up to date and to answer any questions you might have about it; questions, comments, and suggestions should be sent to Kathleen Currie Hall.

Version 1.0 differs from the original release version (0.15, July 2014) primarily in its user interface; we switched the GUI from TK to QT and tried to reorganize the utility menus to be somewhat more intuitive. For example, the original release version had all segment inventory views in alphabetical order; segments are now arranged as closely as possible to standard IPA chart layouts (based on their featural interpretations). Additionally, we have added greater search and edit functions as well as some additional analysis tools (phonotactic probability, mutual information, neighbourhood density), and a greater ability to work with running text / spontaneous speech corpora.

1.2 Code and interfaces

PCT is written in Python 3.4, and users are welcome to add on other functionality as needed. The software works on any platform that supports Python (Windows, Mac, Linux). All code is available on its SourceForge page; the details for getting access are given in *Downloading and installing*. The GitHub repository for PCT is also public and open source.

There is both a graphical user interface (GUI) and a command-line interface for PCT. In the following sections, we generally discuss interface-independent aspects of some functionality first, and then detail how to implement it in both the GUI and the command line. All functions are available in the GUI; many, but not all, are currently available in the command line due to complications in entering in phonological transcriptions that match a given corpus in a command-line interface.

The command-line interface is accessed using command line scripts that are installed on your machine along with the core PCT GUI.

NOTE: If you did not install PCT on your computer but are instead running the GUI through a binary file (executable), then the command line scripts are not installed on your computer either. In order to run them, you will need to download the PCT source code and then find the scripts within the command_line subdirectory. These can then be run as scripts in Python 3.

The procedure for running command-line analysis scripts is essentially the same for any analysis. First, open a Terminal window (on Mac OS X or Linux) or a CygWin window (on Windows, can be downloaded here). Using the "cd" command, navigate to the directory containing your corpus file. If the analysis you want to perform requires any additional input files, then they must also be in this directory. (Instead of running the script from the relevant file directory, you may also run scripts from any working directory as long as you specify the full path to any files.)

You then type the analysis command into the Terminal and press enter/return to run the analysis. The first (positional) argument after the name of the analysis script is always the name of the corpus file.

Downloading and installing

PCT is currently available in beta form for Mac, PC, and Linux machines. It can be downloaded from SourceForge using the following steps. Note that there are several dependencies that are pre-requisites before PCT can function properly. For Mac and Windows machines, we have created executable files that bundle most of the dependencies and the PCT software itself into a single package. Using these is the easiest / fastest way to get PCT up and running on your machine.

1. Go to the PCT SourceForge page.

2. Click on the "Files" tab.

3. Click on the link with the highest number (= most recent version). As of February 2015, that is 1.0.

2.1 Windows Executable

1. NOTE: This method requires that you are running a 64-bit version of windows. You can check this in Control Panel -> System and Security -> System.

2. Click on "win64."

3. Download the file called "corpustools-1.0.0-amd64.msi" (or similar, for a more recent version), by clicking or right-clicking on the link. This is an installer program.

4. Run the downloaded installer program by double-clicking on it, wherever it has been saved locally.

5. PCT should now be available from your "Start" menu under "Programs."

6. If you run into trouble, try the "Fallback" instructions in below.

2.2 Mac Executable

1. Click on "macosx."

2. Download the file called "pct.zip" by clicking or ctrl-clicking on the link. This is a zipped file containing the PCT application.

3. Unzip the downloaded file from its local location (double-clicking it should automatically open your local software for unzipping files, e.g., StuffIt Expander or Archive Utility).

4. Unzipping the file will give you access to the file "pct.app." Double click on this file to start PCT. You can drag the application to your toolbar like any other application.

5. If you run into trouble, try the "Fallback" instructions in below.

2.3 Linux / Fallback instructions

1. Dependencies: You'll first need to make sure all of the following are installed. The third and fourth ones (NumPy and SciPy) are needed only for the Acoustic Similarity functionality to work.

   1. Python 3.3 or higher

   2. NumPy

   3. SciPy

   4. (NB: If you are on Windows and can't successfully use the acoustic similarity module after installing NumPy and SciPy from the above sources, you may want to try installing them from precompiled binaries.)

2. Get the source code for PCT. Click on either the .zip or the .gz file on the PCT SourceForge page, to download the zipped or tarball version of the code, depending on your preference.

3. After expanding the file, you will find a file called `setup.py` in the top level directory. Run it in one of the following ways:

   1. Double-click it. If this doesn't work, access the file properties and ensure that you have permission to run the file; if not, give them to yourself. In Windows, this may require that you open the file in Administrator mode (also accessible through file properties). If your computer opens the .py file in a text editor rather than running it, you can access the file properties to set Python 3.x as the default program to use with run .py files. If the file is opened in IDLE (a Python editor), you can use the "Run" button in the IDLE interface to run the script instead.

   2. Open a terminal window and run the file. In Linux or Mac OS X, there should be a Terminal application pre-installed. In Windows, you may need to install Cygwin. Once the terminal window is open, nagivate to the top level CorpusTools folder—the one that has setup.py in it. (Use the command 'cd' to navigate your filesystem; Google "terminal change directory" for further instructions.) Once in the correct directory, run this command: `python3 setup.py install`. You may lack proper permissions to run this file, in which case on Linux or Mac OS X you can instead run `sudo python3 setup.py install`. If Python 3.x is the only version of Python on your system, it may be possible or necessary to use the command `python` rather than `python3`.

4. Phonological CorpusTools should now be installed! Run it from a terminal window using the command `pct`. You can also open a "Run" dialogue and use the command `pct` there. In Windows, the Run tool is usually found in All Programs -> Accessories.

Loading in corpora

In order to use the analysis functions in PCT, you'll first need to open up a corpus. There are four possible ways of doing this: first, you can use a built-in corpus (a small, entirely invented sample corpus or the Irvine Phonotactic Online Dictionary of English [IPHOD]; second, you can use a corpus that is independently stored on your local computer; third, you can create a new corpus from text; and fourth, you can import a spontaneous speech corpus (e.g. from Praat TextGrids or from your own local copy of a corpus such as the Buckeye corpus [BUCKEYE] or TIMIT corpus [TIMIT]. Each of these will be discussed in turn. The basic structure of a corpus, however, is a list of words with other possible information about each: e.g., its transcription, its frequency of occurrence, its lexical category, its syllable structure, etc. These are in columnar format; e.g., loaded from a CSV or tab-delimited text file.

3.1 Using a built-in corpus

To use a built-in corpus, simply go to the "Corpus" menu and select "Load corpus..." from the list, which will open the "Load corpus" dialogue box.

The first time you want to use a built-in corpus, you'll need to download it (from a Dropbox link accessed by PCT internally); you must therefore be connected to the internet to complete this step. To do so, click on "Download example corpora" from the right-hand menu. This will allow you to download either the Example corpus and/or the IPHOD corpus [IPHOD]. Note that the version of the IPHOD corpus that is contained here has been altered from the freely downloadable version, in that it (1) does not have the derived columns and (2) has been re-formatted as a .corpus file for easy reading by PCT. It also contains only the following information: word, transcription, and token frequency (from the SUBTLEX corpus [SUBTLEX]). Please note that if you use the IPHOD corpus, you should use the following citation (see more on citing corpora and functions of PCT in *Citing PCT and the algorithms used therein*):
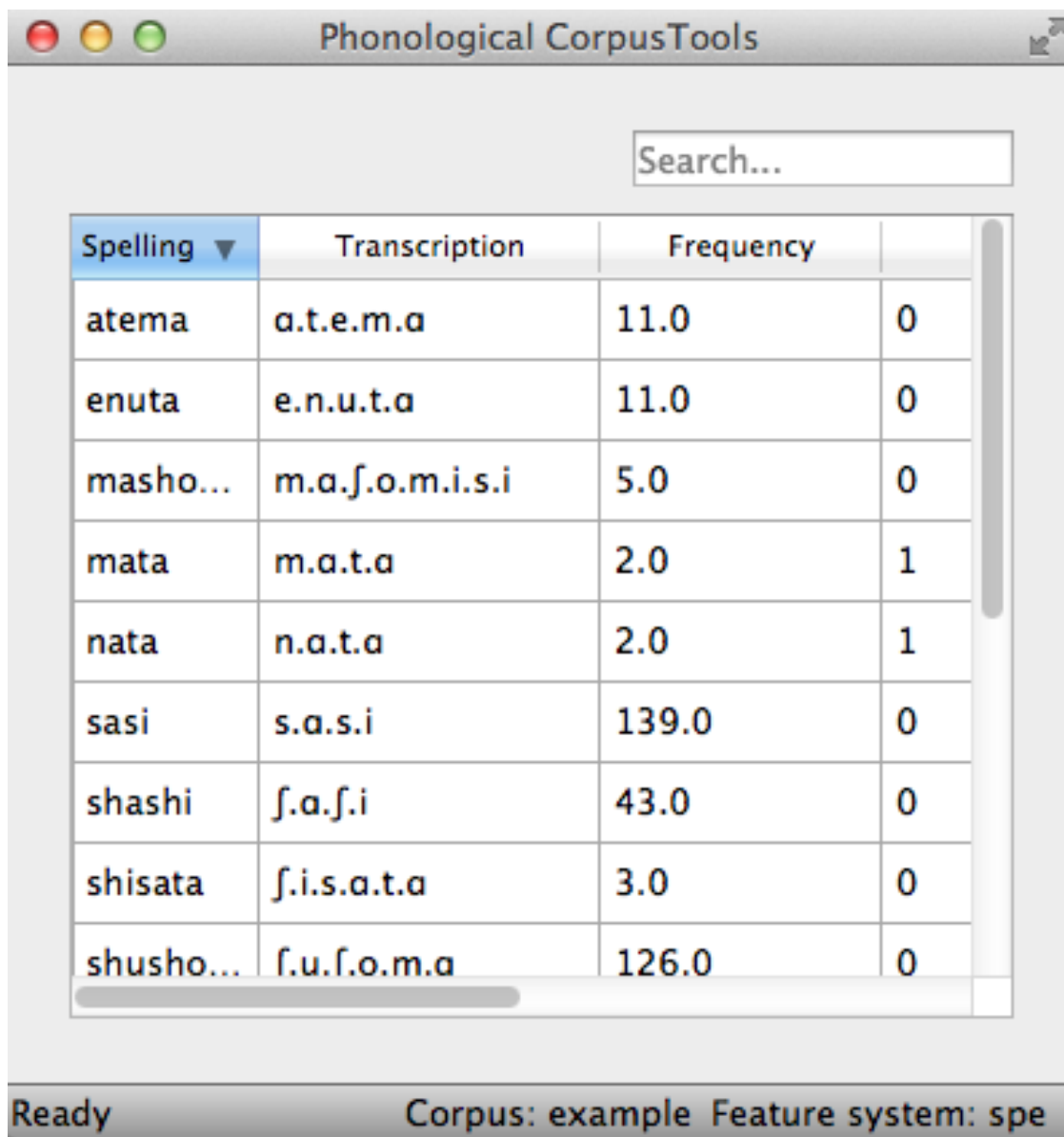
Vaden, K. I., Halpin, H. R., Hickok, G. S. (2009). Irvine Phonotactic Online Dictionary, Version 2.0. [Data file]. Available from http://www.iphod.com/.

After the corpus has been downloaded, it appears in the lefthand side of the "Load Corpus" dialogue box. Simply select the corpus and click on "Load selected corpus" at the bottom of the dialogue box. Once these corpora have been downloaded once, you don't have to do so again; they will be saved automatically to your local system unless and until you delete them. On subsequent loadings of the PCT software, you will still see these corpora listed in the lefthand side of the "Load Corpus" dialogue box, as in the following diagram:
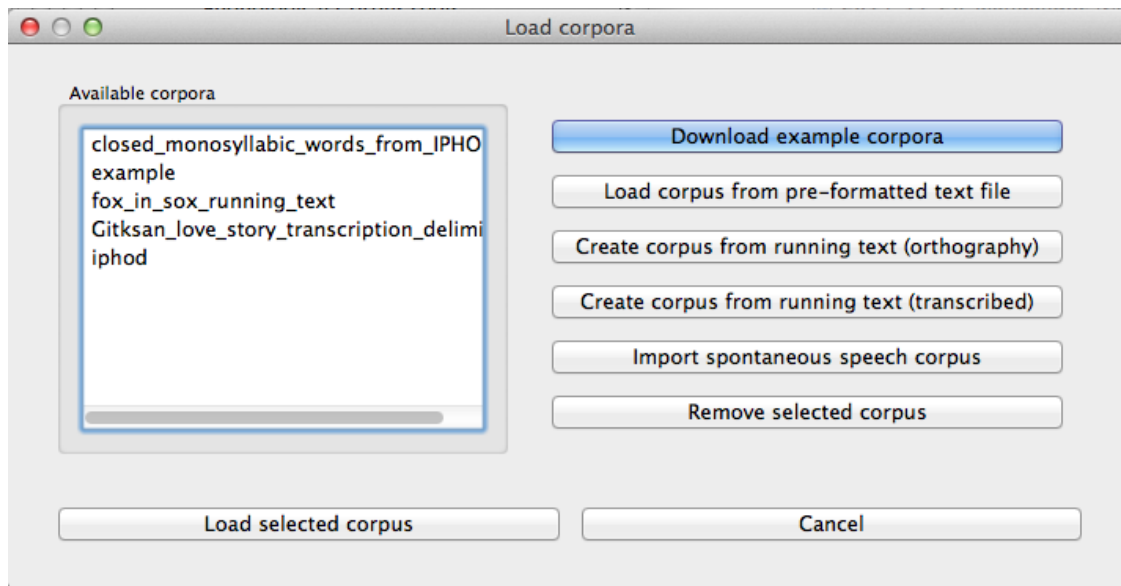
The example corpus and the included version of the IPHOD corpus include phonetic transcriptions in IPA, and are by default interpreted either using the feature system of [Mielke2012], which in turn is based on SPE features [SPE] [this is the default for the example corpus], or using the feature system suggested by [Hayes2009] [this is the default for the IPHOD corpus]. These systems are fully functional for doing subsequent analyses. Note, however, that this is a built-in functionality of these particular corpora, and does not allow you to use SPE or Hayes features with other corpora. To use SPE features with other corpora, or to change the feature system associated with a built-in corpus, you'll need to download the actual feature files, as described in *Working with transcriptions and feature systems*. Features can be used for defining classes of sounds (e.g., creating separate tiers for different types of segments) and for defining environments (e.g., the environments in which segments might occur, for use in calculating their predictability of distribution).

The corpus may take several seconds to load, but will eventually appear; the following is the example corpus:

Note that the name of the corpus and the current feature system are shown at the bottom right-hand corner of the screen for easy reference. *Summary information about a corpus* gives more detail on how to find out summary information about your corpus. Typing a word or part-word in the "search" box takes you to each successive occurrence of that word in the corpus (hit "return" once to see the first instance; hit "return" again to see the second, etc.). Note that the "search" box searches only the "Spelling" column of the corpus. To do a phonological search, please use the "Phonological search" function under the "Corpus" menu (see detailed discussion in *Phonological Search*).

3.2 Using a custom corpus

It is also possible to use a custom corpus, i.e., any corpus that is in the appropriate format (see *Required format of corpus*) and stored independently on your computer. Before doing so, it may be helpful to first load the appropriate feature system into PCT, so that the transcriptions in your corpus can be interpreted; detailed instructions for doing this are given in *Working with transcriptions and feature systems*. It is also possible to load the feature system after you've loaded the corpus.

To use a custom corpus, click on "Corpus" / "Load corpus..." and then choose "Load corpus from pre-formatted text file." Then, enter the path for the corpus or select it using "Choose file..." and navigating to it from a system dialogue box. Enter a name for the corpus and indicate what the delimiter type is; the default is a comma (,); enter *t* if the file is tab-delimited. Any symbol can be used; PCT will simply break elements at that symbol, so whatever symbol is used should be used only to delimit columns within the corpus. Finally, if there is a column in the corpus that shows phonetic transcription, choose which feature system you would like to use. As noted above, in order for there to be feature systems to choose from, you must first have loaded them into PCT (*Working with transcriptions and feature systems*).

Clicking "OK" in the "Load new corpus" dialogue box returns you to the "Load corpus" dialogue box, and you will see that the new corpus has been added to your list of available corpora. Select this new corpus and choose "Load selected corpus" to open it in PCT.

3.2.1 Required format of corpus

In order to use your own corpus, it must have certain properties. First, it should be some plain text file (e.g., .txt, .csv); it cannot, for example, be a .doc or .pdf file. The file should be set up in columns (e.g., imported from a spreadsheet) and be delimited with some uniform character (tab, comma, backslash, etc.). The names of most columns of information can be anything you like, but the column representing common spelling of the word should be called "spelling"; that with transcription should be called "transcription"; and that with token frequency should be called "frequency." All algorithms for doing corpus analysis will assume these column names. If, for example, you were using a corpus that had different frequency columns for total frequency vs. the frequency of occurrence of the word in its lowercase form (cf. the SUBTLEX corpus), then whichever column is to be used for token frequency calculations should simply be labelled "frequency."

3.3 Creating a corpus from running text

It is also possible to have PCT create a corpus for you from running text, either in orthographic or transcribed form. If the text is orthographic, of course, then segmental / phonological analysis won't be possible, but if the text is itself transcribed, then all subsequent analysis functions are available.1 As with pre-existing corpora, it may be helpful to first load the relevant feature system into PCT, so that the transcriptions in your text can be interpreted; detailed instructions for doing this are given in *Working with transcriptions and feature systems* (note that the corpus can be loaded in without featural interpretation, and features added later).

To create a corpus from text, click on "Corpus" / "Load corpus..." and then select either "Create corpus from running text (orthography)" or "Create corpus from running text (transcription)."

1. **File selection**: Select the name of the plain .txt file by entering the path directly or choosing the file using a system dialogue box, by clicking on "Choose file...."

2. **Name of corpus**: Indicate what the name of the corpus should be; PCT will default to the name of the original .txt file.

3. **Word delimiter**: Enter the character used to delimit words in the corpus (e.g., a space).

4. **Punctuation**: If there is punctuation in the text, indicate which elements should be ignored. Ignoring punctuation allows PCT to compile an accurate count of unique words; for example, the words "example" and "example," should be treated as two tokens of the same word, ignoring the comma at the end of the second one. Punctuation can be included, however; this might be desirable in a case where a punctuation symbol is being used within the transcription system (e.g., [!] used for a retroflex click).

5. **Corpus for transcriptions**: If the corpus is an orthographic one, but you have a separate corpus that includes both orthographic and trancribed representations of the words, you can have PCT automatically look up all of the transcribed words in the separate corpus (e.g., you're uploading a new transcribed corpus of English, but want to look up the transcriptions in IPHOD). Select the other corpus from the dropdown menu; if you

haven't loaded the corpus into PCT already, this option is not available. Indicate using the check box whether case (capitalization) should be ignored (e.g., if your corpus contains the name "Bud," PCT will look for an exact match, "Bud," unless case is ignored, in which case, the common noun "bud" will be accepted as a pronunciation source).

6. **Transcription and feature selection**: If the corpus is a transcribed one, you can select the transcription and featural system, if these systems have already been loaded into PCT (see also discussion in *Working with transcriptions and feature systems*). You can also indicate what the delimiter is for units of transcription (if there is such a delimiter). See the note below for details.

7. **A note about complex transcriptions**: There is no way for PCT to know automatically when a single sound is represented by a sequence of multiple characters – e.g., that the digraphs [a], [th], [xw], [p'], [t], and [i] are intended to represent single sounds rather than sequences of two sounds. There are currently three possible ways of ensuring that characters are interpreted correctly:

   (a) **One-to-one transcriptions**: The first way is to use a transcription system with a one-to-one correspondence between sounds and symbols, such as DISC. If you need to create a novel transcription system in order to accomplish this (e.g., using [A] to represent [a] and [2] to represent [th], etc.), you may certainly do so; it is then necessary to create a novel feature file so that PCT can interpret your symbols using known features. See detailed instructions on how to do this in *Downloadable transcription and feature choices*. The word tide in American English might then be transcribed as [2Ad]. This is a relatively easy solution to implement by using find-and-replace in a text editing software, though it does result in less easily human-readable transcriptions.

   (b) **Delimited transcriptions**: The second way is to use a standard transcription system, such as IPA, but to delimit every unitary sound with a consistent mark that is not otherwise used in the transcription system (e.g., a period). Thus the word *tide* in American English might be transcribed in IPA as [.th.a.d.], with periods around every sound that is to be treated as a single unit. When creating the corpus, PCT will give you the option of specifying what the character is. PCT will then read in all elements between delimiting characters as members of a single "segment" object, which can be looked up in a standard feature file (either an included one or a user-defined one; see *Using a custom feature system*). This solution makes it easy to read transcribed words, but can be more labour-intensive to implement without knowledge of more sophisticated searching options (e.g., using regular expressions or other text manipulation coding) to automatically insert delimiters in the appropriate places given a list of complex segments. A first pass can be done using, e.g., commands to find "a" and replace it with ".a." – but delimiters will also have to be added between the remaining single characters, without interrupting the digraphs.

   (c) **Constructed digraphs**: The third option is to tell PCT what the set of digraphs is in your corpus manually, and then to have PCT automatically identify these when it creates the corpus. In the "Create corpus from running text (transcription)" dialogue box, there is an option to "Construct a digraph." Once you have entered the path name of the file you are creating the corpus from, PCT will scan it for single characters and present these to you as options for constructing digraphs from. For example, in the following box, all of the single characters in a Gitksan text file are presented, and can be selected sequentially to create the appropriate digraphs. This method is somewhat more labour-intensive in terms of knowing ahead of time what all the digraphs are and being able to list them, but ensures that all such occurrences are found in the text file. Note, however, that if there's a *distinction* to be made between a sequence of characters and a digraph (e.g., [t] as a sequence in *great ship* vs. as an affricate in *grey chip*), this method will be unable to make that distinction; all instances will be treated as digraphs.
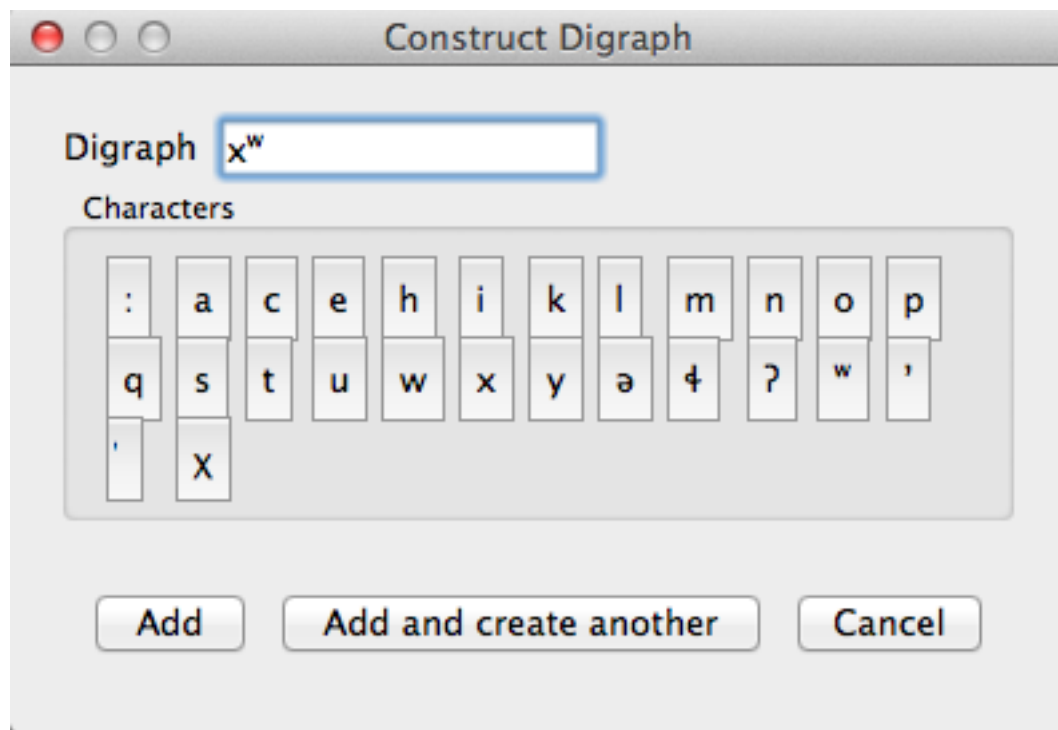
8. **Create corpus**: Once the options have been selected, click on "Create corpus." The columns created are: individual lexical items (words), their raw token frequency in the corpus, and their relative token frequency (raw token frequency / total tokens in the corpus).

9. **Use the corpus**: Once the corpus has been created, it also now appears in your list of corpora in the "Load corpus" dialogue box. Simply select it and choose "Load selected corpus" to open it for use in PCT.

10. **Save the corpus**: The corpus itself will automatically be saved for use on subsequent re-openings of PCT, without needing to be created again. It can be exported as a .txt file and saved to a location of your choosing, however, for use in spreadsheets or with other software. Once the corpus has been created and loaded, simply go to "Corpus" / "Export corpus as text file..." to save it using a system dialogue box.

The following shows an example of a transcribed Gitksan story transformed into a (small!) corpus (with grateful acknowledgement to Barbara Sennott and the UBC Gitksan language research group, headed by Lisa Matthewson & Henry Davis, for granting permission to use this text):

1. The original transcribed story:

2. The transcription delimited with periods to show unitary characters:

3. The dialogue box for creating the corpus from text. Note that hyphens and equal signs, which delimit morphological boundaries in the original, have been ignored during the read-in. A space is entered into the word delimiter box (not visible here). The period is selected as the transcription delimiter. A feature system called gitksan2hayes_delimited, which maps the delimited transcription system used in this example to the features given in [Hayes2009], has already been loaded into PCT (see *Using a custom feature system*), and so is selected here.

Alternatively, the same corpus could be read in without being hand-delimited, by constructing digraphs within the load corpus dialogue box, as follows:
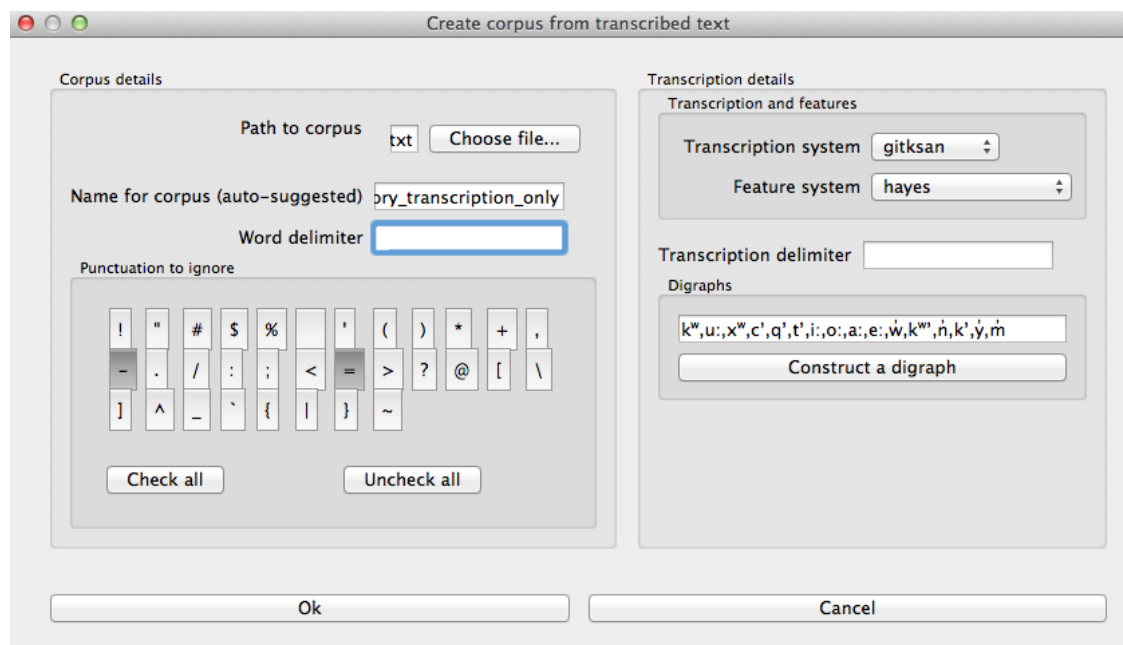
4. The resulting corpus, ready for subsequent analysis:
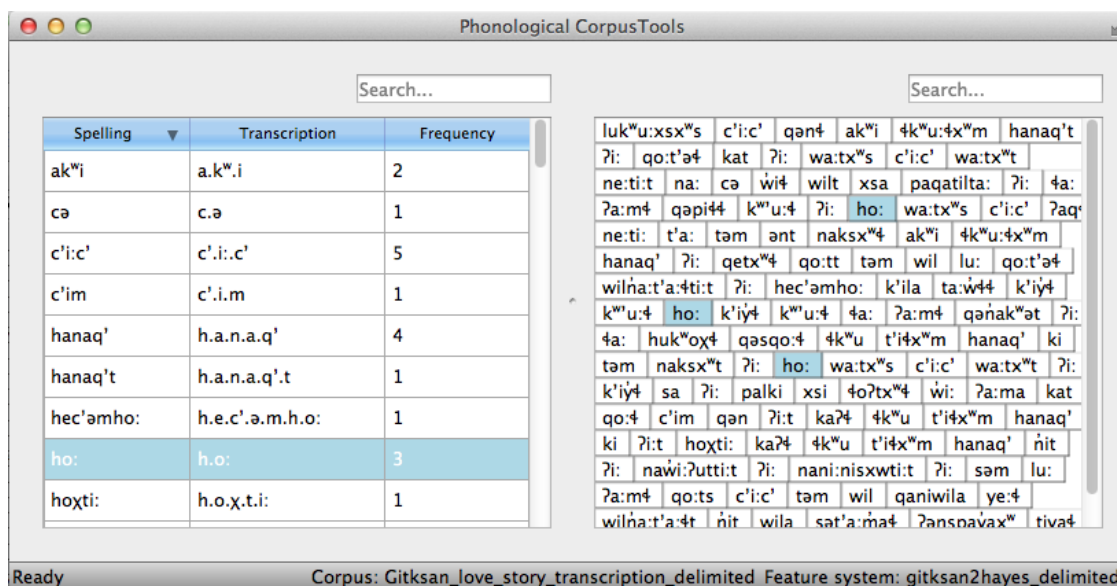
The corpus appears on the left, in the familiar columnar format. The original text of the corpus appears at the right. Right-clicking on a word in the corpus list gives you the option to "Find all tokens" in the running text; these words will be highlighted. Similarly, right-clicking a word in the running text gives you the option to "Look up word," which will highlight the word's entry in the corpus list.

3.4 Creating a spontaneous speech corpus

Currently, only limited functionality is available for spontaneous speech corpora, but this is a top priority for our next version. We provide an interface for importing the TIMIT corpus [TIMIT] or Buckeye corpus [BUCKEYE], if you have independently downloaded their corpus files. We currently provide preliminary capabilities to create a corpus by reading in the text from a set of Praat TextGrids.

3.4.1 Working with your own TextGrids

Textgrids are automatically inspected for two kinds of tiers, words and phones. Word tiers are ones that have some variation of "word" in them, either plural or singular, and with any case (i.e. "Word," "word," and "WORD" would all be detected). Phone tiers are ones that have a variant of "phone," "segment," "transcription," or "seg" in them. All Point Tiers are ignored. All other interval tiers are included as word token attributes (e.g., if there's a tier called "Speech style" and an interval named "casual" extends around the word token, the word token will have an attribute for "Speech style" with the value "casual"). If there are multiple speakers in a text gird, PCT expects word and phone tiers for each speaker, such as "Speaker 1 – word" and "Speaker 2 - word."

To create a spontaneous speech corpus from TextGrids, first ensure that your TextGrids are all located in a single directory and have the above properties. Click on "File" / "Load corpus..." / "Import spontaneous speech corpus." Select the directory where your TextGrids are located, and choose "TextGrid" as the Corpus file set up option.

Here is an example of creating a corpus based on three .TextGrid files from the Corpus of Spontaneous Japanese [CSJ].
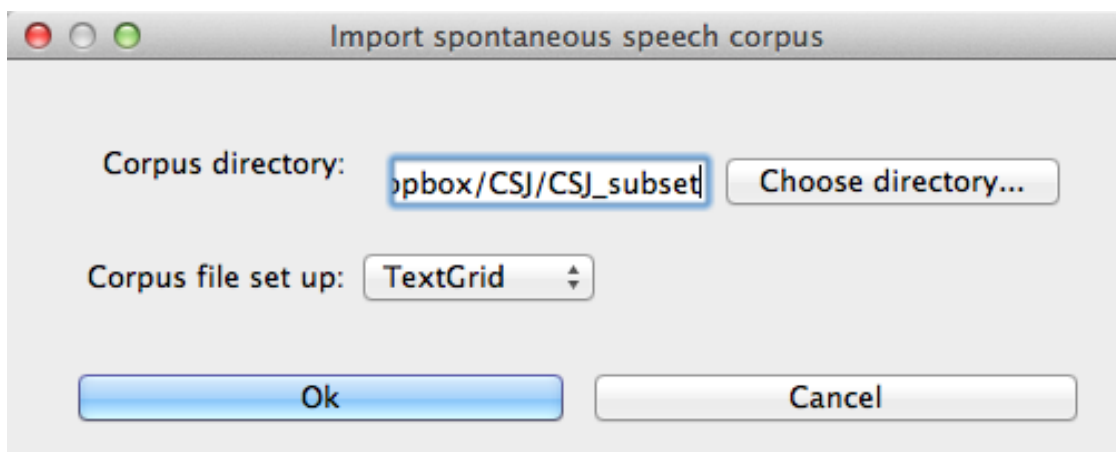
Once the TextGrids have been processed, they appear in a window such as the following. The regular corpus view is in the centre, with frequency counts aggregated over the entire set of speakers / TextGrids. Note that the transcription column may be blank for many words; this is because in spontaneous speech, the citation / spelled words often have multiple different transcribed forms in the corpus itself. To see these various transcriptions, right-click on any word in the corpus and select "List pronunciation variants." A new dialogue box will pop up that shows the individual pronunciation variants that occur in the corpus for that word, along with their token frequencies.
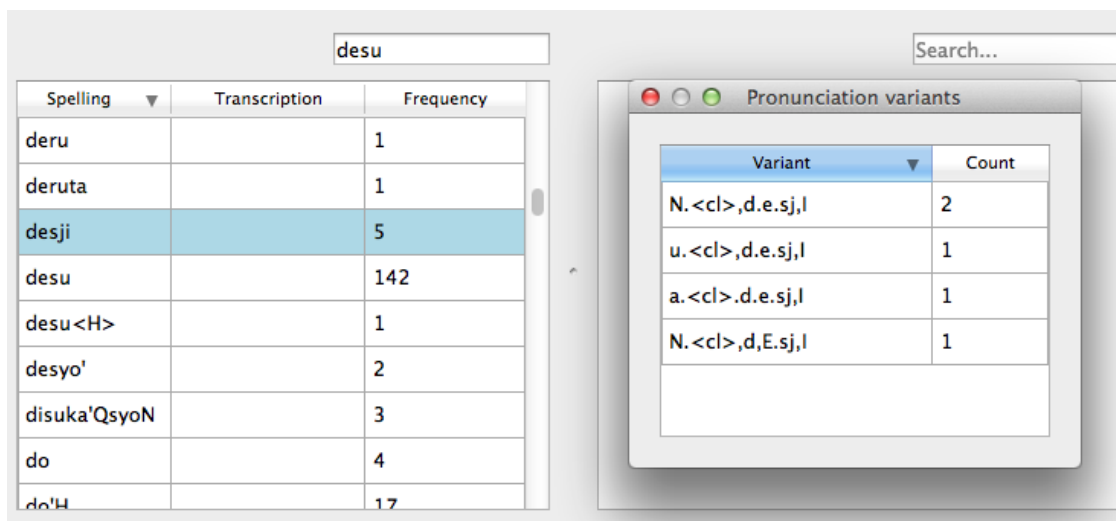
In this example, each TextGrid is interpreted as belonging to a different speaker, and these individual speakers are

listed on the left. Clicking on one of the speakers shows the transcript of that speaker's speech in a box on the right. This is not a corpus, but rather a sequential listing of each word that was extracted, along with the transcription and the timestamp of the beginning of that word in the TextGrid. Right-clicking on a word in this list will give you the option to look up the word's summary entry in the corpus. Right-clicking a word in the overall corpus will give you the option to "Find all tokens" of that word in the transcriptions, where they will simply be highlighted.
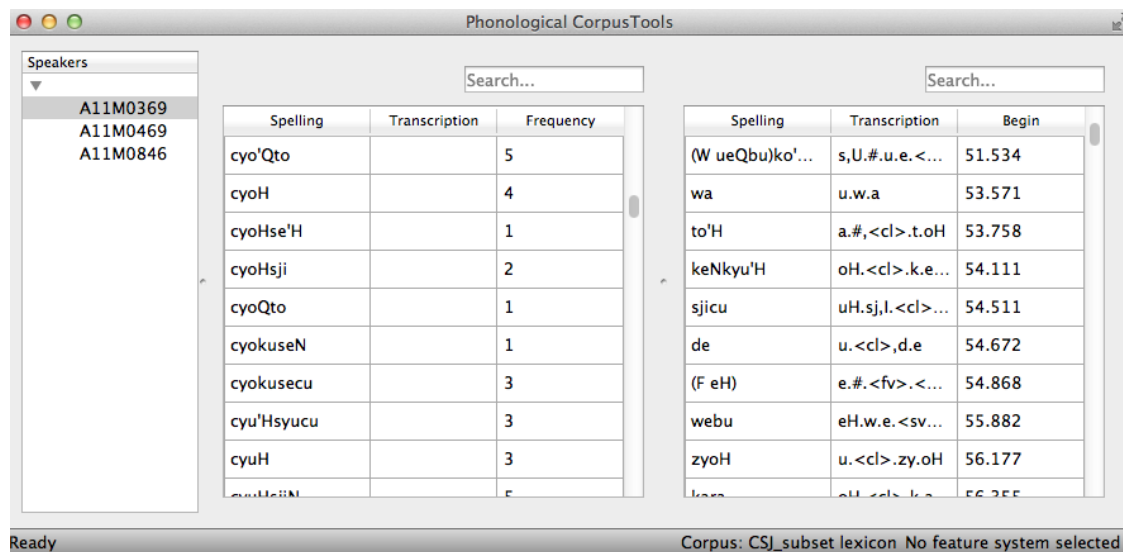
3.5 Creating a corpus file on the command line

In order to create a corpus file on the command line, you must enter a command in the following format into your Terminal:

```
pct_corpus TEXTFILE FEATUREFILE
```

...where TEXTFILE is the name of your input text file and FEATUREFILE is the name of your feature file. You may specify file names using just the file name itself (plus extension) if your current working directory contains the files; alternatively, you can specify the full path to these files. Please do not mix short and full paths. You may also use command line options to change the column delimiter character or segment delimiter character from their defaults ('t' and ''), respectively). Descriptions of these arguments can be viewed by running `pct_corpus -h` or `pct_corpus --help`. The help text from this command is copied below, augmented with specifications of default values:

Positional arguments:

-h
--help
      Show this help message and exit

-d DELIMITER
--delimiter DELIMITER
      Character delimiting columns in input file, defaults to `\t`

-t TRANS_DELIMITER
--trans_delimiter TRANS_DELIMITER
      Character delimiting segments in input file, defaults to the empty string

EXAMPLE:

If your pre-formatted text file is called mytext.txt and your features are hayes.feature, and if mytext.txt uses ; as column delimiters and . as segment delimiters, to create a corpus file, you would need to run the following command:

```
pct_corpus mytext.txt hayes.feature -d ; -t .
```

3.6 Summary information about a corpus

Phonological CorpusTools allows you to get summary information about your corpus at any time. To do so, go to "Corpus" / "Summary."

1. **General information**: At the top of the "Corpus summary" dialogue box, you'll see the name of the corpus, the feature system currently being used, and the number of words (entries) in the corpus.

2. **Inventory**: Under the "Inventory" tab, there will generally be three sections, "Consonants," "Vowels," and "Other." (Note that this assumes there is an interpretable feature system being used; if not, then all elements in the inventory will be shown together.) Clicking the box next to "Consonants" will show you the current set of consonants, roughly arranged according to the IPA chart. Similarly, clicking the box next to "Vowels" will show you the current set of consonants, roughly arranged according to the IPA chart. Any other symbols (e.g., the symbol for a word boundary, #) will be shown under "Other."

    (a) **Segments**: Clicking on any individual segment in the inventory will display its type and token frequency in the corpus, both in terms of the raw number of occurrences and the percentage of occurrences.

3. **Columns**: Under the "Columns" tab, you can get information about each of the columns in your corpus (including any that you have added as tiers or other columns; see *Adding, editing, and removing words, columns, and tiers*). The column labels are listed in the drop-down menu. Selecting any column will show you its type (spelling, tier, numeric, factor) and other available information. Tier columns (based on transcriptions) will indicate which segments are included in the tier. Numeric columns will indicate the range of values contained.

Once you are finished examining the summary information, click "Done" to exit.

3.7 Subsetting a corpus

It is possible to subset a corpus, creating new corpora that have only a portion of the original corpus. For example, one might want to create a subset of a corpus that contains only words with a frequency greater than 1, or contains only words of a particular part of speech or that are spoken by a particular talker (if such information is available). The new subsetted corpus will be saved and made available to open in PCT as simply a new corpus.

To create a subset, click on "File" / "Generate a corpus subset" and follow these steps:

1. **Name**: Enter the name for your new corpus. The default is to use the name of the current corpus, followed by "_subset," but a more informative name (e.g., "Gitksan_nouns") may be useful.

2. **Filters**: Click on "Add filter" to add a filter that will be used to subset the corpus. You can filter based on any numeric or factor tier / column that is part of your corpus. For a numeric column (e.g., frequency), you can specify that you want words that have values that are equal to, greater than, less than, greater than or equal to, less than or equal to, or not equal to any given value. For a factor column (e.g. an abstract CV skeleton tier), you can add as many or as few levels of the factor as you like.

3. **Multiple filters**: After a filter has been created, you can choose to "Add" it or "Add and create another" filter. The filters are cumulative; i.e., having two filters will mean that the subset corpus will contain items that pass through BOTH filters (rather than, say, either filter, or having two subsets, one for each filter).

4. **Create subset**: Once all filters have been selected, click on "Create subset corpus." You will be returned to your current corpus view, but the subsetted corpus is available if you then go to "File" / "Load corpus..." – it will automatically be added to your list of available corpora. Note that the subset corpus will automatically contain any additional tiers that were created in your original corpus before subsetting.

3.8 Saving and exporting a corpus or feature file

If "Auto-save" is on (which is the default; see *Preferences*), most changes to your corpus (adding words, tiers, etc.) will be saved automatically and will be available the next time you load the corpus in PCT. Some changes are not automatically saved (removing or editing word entries), even if Auto-save is on, to prevent inadvertant loss of information. If you have made changes that have not been automatically saved, and then quit PCT, you will receive a warning message indicating that there are unsaved changes. At that point, you may either choose "Don't save" (and therefore lose any such changes), "Save" (to save the changes in its current state, to be used the next time it is loaded into PCT), or "Cancel" (and return to the corpus view). It is also possible to export the corpus as a text file (.txt), which can be opened in other software, by selecting "File" / "Export corpus as text file" and entering the file name and location and the column and transcription delimiters. Similarly, the feature system can also be exported to a .txt file by selecting "File" / "Export feature system as text file" and selecting the file name and location and the column delimiter. See more about the utility of doing so in *Working with transcriptions and feature systems*.

3.9 Setting preferences and options

3.9.1 Preferences

There are several preferences that can be set in PCT. These can be selected by going to "Options" / "Preferences...." The following are available:

1. **Storage**:

   (a) **File location**: By default, PCT will save corpus, feature, and result files to your local "Documents" directory, which should exist under the default settings on most computers. When saving a particular output file, you can generally specify the particular storage location as you are saving. However, it is also possible to change the default storage location by changing the file path in this dialogue box. You may enter the path name directly, or select it from a system window by selecting "Choose directory...".

   (b) **Auto-save**: By default, PCT will automatically save changes to a corpus (e.g., if you have updated a feature system, added a tier, etc.). De-select this option if you prefer to manually save such changes (PCT will prompt you before closing without saving). Changes to word entries (removing or editing a word) are NOT auto-saved and should be saved manually if you want them to be saved; again, PCT will prompt you to save in these cases before exiting. Once Auto-save is deselected, PCT will remember that this is your preference for the next time you open the software - it will not automatically get turned back on.

2. **Display**: By default, PCT will display three decimal places in on-screen results tables (e.g., when calculating predictability of distribution or string similarity, etc.). The number of displayed decimal places can be globally changed here. Note that regardless of the number specified here, PCT will save results to files using all of the decimal places it has calculated.

3. **Processing**: Some calculations consume rather a lot of computational resources and can be made faster by using multiprocessing. To allow PCT to use multiprocessing on multiple cores when that is possible, select this option and indicate how many cores should be used (enter 0 to have PCT automatically use the ¾ of the number of cores available on your machine).

3.9.2 Help and warnings

When using PCT, hovering over a dialogue box within a function will automatically reveal quick ToolTips that give brief information about the various aspects of the function. These can be turned on or off by going to "Options" / "Show tooltips."

PCT will also issue certain warnings if various parameters aren't met. It is possible to turn warning messages off by going to "Options" / "Show warnings."
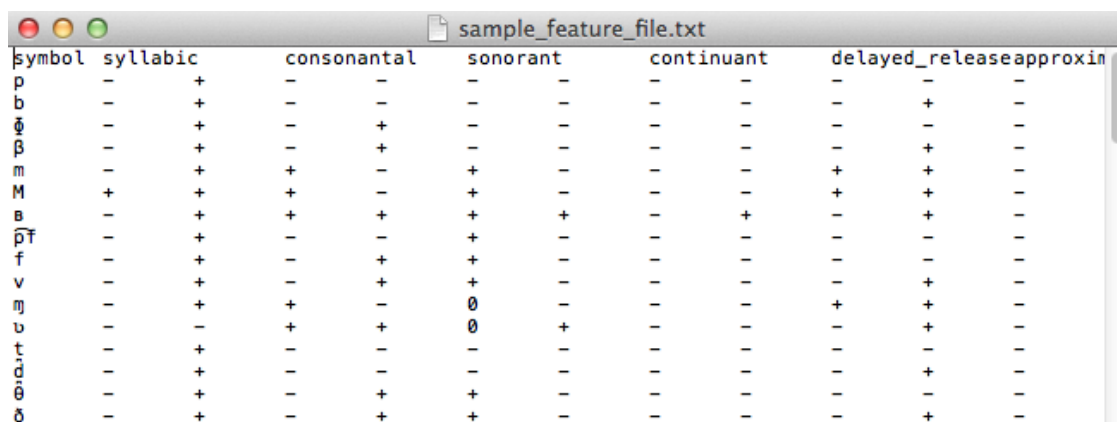
3.9.3 Copying and pasting

It is possible to highlight the cells in any table view (a corpus, a results window, etc.) and copy / paste a tab-delimited string version of the data into another program (e.g., a spreadsheet or text editor) using your standard copy & paste keyboard commands (i.e., Ctrl-C and Ctrl-V on a PC; Command-C and Command-V on a Mac).

Working with transcriptions and feature systems

In order to do phonological analysis, PCT needs to know what segments exist in a corpus, and what features are assigned to each segment. There are a variety of built-in transcription systems and feature systems available, but users can also define their own for custom use. Transcription and feature systems are essentially packaged together as .txt files in the form of a spreadsheet, where particular transcription symbols are mapped to a set of features (described below in *Required format of a feature file*). In general, however, feature systems (i.e., files containing transcriptions and their features) must be explicitly loaded into PCT before they are available for use. Thus, it makes sense to start by loading in at least one such system before attempting to work with corpora.

4.1 Required format of a feature file

As mentioned above, transcription and feature systems are packaged together as .txt files in the form of a delimited spreadsheet. The first column in the file lists all the transcription symbols that are to be recognized. *Note that this column must be labeled with the name "symbol" in order for PCT to correctly read in the file*. Every symbol used in the corpus needs to be in this column. Subsequent columns list individual features that are to be used. Each cell in a row gives either a + or – for each feature, indicating what value the initial symbol in that row has for that feature; a 0 or n can also be used, to indicate that a particular feature is not defined for a given segment (0 is the default in the built-in *hayes* system; n is the default in the built-in *spe* system). The following shows an example; keep in mind that this is in fact a tab-delimited .txt file, but the names in the first row are longer than any of the values in subsequent rows, so the alignment is visually misleading. For example, the first row containing symbols contains the symbol [p], which is designated as [-syllabic], [+consonantal], [-sonorant], [-continuant], etc., although the column names aren't aligned with the feature values visually.

| symbol	syllabic		consonantal	sonorant		continuant		delayed_release	approxim
p	–	+	–	–	–	–	–	–	–	–
b	–	+	–	–	–	–	–	–	–	+	–
ɸ	–	+	–	+	–	–	–	–	–	–	–
β	–	+	–	+	–	–	–	–	–	+	–
m	–	+	+	–	+	–	–	–	+	+	–
M	+	+	+	–	+	–	–	–	+	+	–
ʙ	–	+	+	+	+	+	–	+	–	+	–
p͡ʈ	–	+	–	–	+	–	–	–	–	–	–
f	–	+	–	+	+	–	–	–	–	–	–
v	–	+	–	+	+	–	–	–	–	+	–
ɱ	–	+	+	–	0	–	–	–	+	+	–
ʋ	–	–	+	+	0	+	–	–	–	+	–
t	–	+	–	–	–	–	–	–	–	–	–
d̪	–	+	–	–	–	–	–	–	–	+	–
θ	–	+	–	+	+	–	–	–	–	–	–
ð	–	+	–	+	+	–	–	–	–	+	–

4.2 Downloadable transcription and feature choices

Currently, the built-in transcription systems that are recognized are IPA, ARPABET (used for the [CMU] dictionary), XSAMPA, CELEX, DISC, and Klatt. These transcription systems can be associated with either the features as laid out in [Mielke2012], which in turn are based on [SPE], or as laid out in [Hayes2009] [1]. Each of these transcription-to-feature mappings is laid out as above in a .txt file that can be downloaded from within PCT. The former system is called "spe" for short within PCT, while the latter is called "hayes."

To download one of these systems, click on "Corpus" / "Manage feature systems..." and follow these steps:

1. **Download**: Click on "Download feature systems" to open up the relevant dialogue box.

2. **Transcription**: Select which of the transcription systems you want (IPA, ARPABET, XSAMPA, CELEX, DISC, Klatt).

3. **Feature system**: Select which set of features you would like to map the transcription symbols to (SPE or Hayes).

4. **Saving**: Click "OK" to have PCT load in the selected feature file (you must be connected to the internet to have this functionality). The newly downloaded feature file will now appear in your "Manage feature systems" dialogue box, and is available for all subsequent use of PCT unless and until you delete it (done by selecting the system and clicking "Remove selected feature system"). Click "Done" to return to the regular corpus analysis window.

The example below shows the selection of the CELEX transcription system, interpreted using Hayes features.
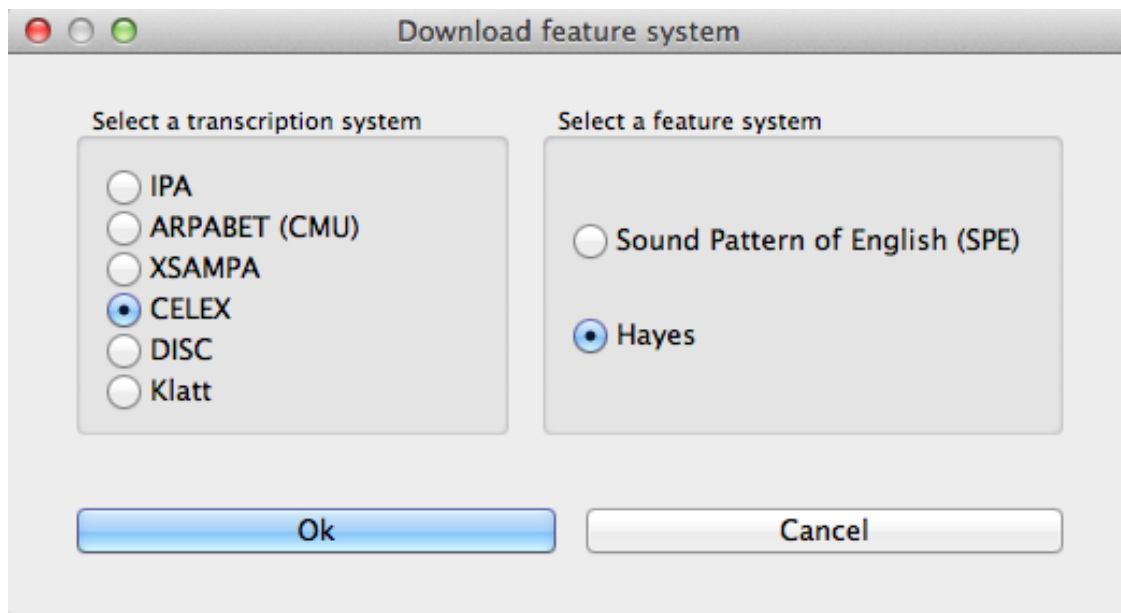
See *Applying / editing feature systems* for more information about applying / editing feature systems in conjunction with corpora.

---

[1] Note that the original [Hayes2009] system does not include diphthongs. We have included featural interpretations for common English diphthongs using two additional features, [diphthong] and [front-diphthong]. The former has a [+] value for all diphthongs, a [-] value for all vowels that are not diphthongs, and a [0] value for non-vowels. The latter references the end point of a diphthong; [a], [e], and [] are [+front-diphthong], [a] and [o] are [-front-diphthong]. All other segments are left unspecified for this feature. Other vowel features for diphthongs are specified based on the first element of the diphthong; e.g., all of [a], [e], [], [a], and [o] are [-high]; of these five, only [a] and [a] are [+low]; only [e] is [+front]; only [o] and [] are [+back]; only [o] and [] are [+round].

4.3 Using a custom feature system

In addition to using one of the built-in feature systems, you can design your own transcription-to-feature mapping, of the format specific in *Required format of a feature file*.

4.3.1 Loading a custom feature system

Once you have a feature file in the required format (see *Required format of a feature file* and *Modifying an existing feature system's text file*), go to "Corpus" / "Manage feature systems..." to load it in. Select "Create feature system from text file" and the "Import feature system" dialogue box will open.

1. **File selection**: Specify the file by entering its directory path or by selecting it using the "Choose file..." button.

2. **Transcription system**: Indicate which transcription system this is a feature file for. (For example, you can create a new feature file for existing IPA transcriptions.) If this is a brand-new system for PCT, i.e., a new transcription system being associated with features, then select "Custom" from the dropdown menu. Then, enter a name for the transcription system in the box.

3. **Feature system**: Indicate which feature system is being used (e.g., is this a case of assigning existing SPE features to a new transcription system?). If this is a brand-new set of features, then select "Custom" from the dropdown menu. Then, enter a name for the feature system in the box.

Note: For both existing transcription and feature systems, you still need to include both the transcriptions and the features in the .txt file itself; you can simply indicate here in PCT that these transcriptions and / or features are identical to ones that are already extant in the system, so that they can be used / interpreted consistently. The name of the transcription / feature system file in PCT will conventionally be transcription2features (e.g., ipa2hayes for IPA symbols interpreted using Hayes features), so it's useful to be consistent about what the names are.

4. **Delimiter**: Indicate what the column delimiter in the custom file is. The default, tab, is indicated by 't.'

Click "OK," and the feature system should now appear in your "Available feature systems" window. Click "Done." See *Applying / editing feature systems* for more information about applying the feature system to corpora. The image below shows the dialogue box used to load in the custom, tab-delimited feature file for interpreting the custom "gitksan" transcription system using Hayes features.
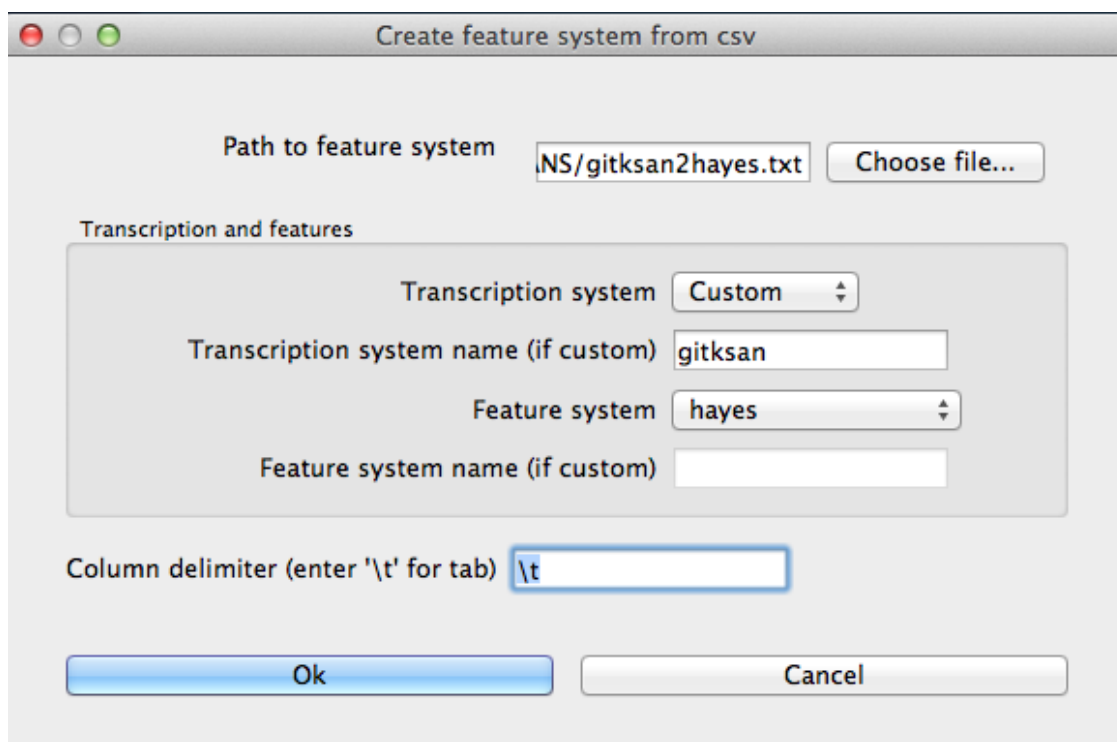
4.3.2 Modifying an existing feature system's text file

A custom system can be created from scratch, following the format described in *Required format of a feature file*. It is probably easier, however, to create a new system by modifying an existing system's file. While this can be done to a certain extent within PCT itself (see *Applying / editing feature systems*), large-scale changes are best done in separate text-editing software. To do so, you'll need to start with an existing file, which can be obtained in one of two ways:

1. Through PCT: Download one of the built-in feature systems (*Downloadable transcription and feature choices*) and apply it to your corpus (*Applying / editing feature systems*). Then go to "File" / "Export feature system as text file..." and save the file locally.

2. Online: You can directly download .txt files with the currently available feature systems from the PCT Source-Forge page, under "Files." One advantage to this method is that there may be additional feature files that are created as .txt files and made available online before they are packaged into the next release of the downloadable software itself.

Once you have the file, open it in whatever software you prefer (e.g., TextEdit, OpenOffice, etc.); it may be easiest to import it into a spreadsheet reader (e.g., Excel, Calc, etc.) in terms of legibility. You can then add new symbols to the first column, feature specifications in the subsequent columns, new feature names, etc., etc. Remember that the name of the first column must always be "symbol" in order for PCT to read the file. Save the new file as a CSV or tab-delimited .txt file, and load it following the instructions in *Loading a custom feature system*.

4.4 Applying / editing feature systems

Once a feature system has been loaded into PCT (*Downloadable transcription and feature choices*, *Using a custom feature system*), it is available for use with a corpus. To do so, first load in a corpus (*Loading in corpora*); if you are using a custom corpus or creating a corpus from text, you can select the feature system you want to use during the loading. Once a corpus has been loaded (with or without a feature system), you can edit the system by clicking on "Features" / "View / change feature system...." The following options are shown:

1. **View system**: At the top of the "Edit feature system" dialogue box, you'll see the current transcription and feature system being used, assuming one has been selected. The first column shows the individual symbols; subsequent columns give the feature specifications for each symbol. Clicking on the name of any column sorts the entire set by the values for that feature; clicking again flips the sort order based on that same column.

2. **Change transcription and feature systems**: If there is no feature system already specified, or if you would like to change the transcription or feature system, use the dropdown menus under "Change feature systems" to select from your currently available systems. If no system is available, or the system you want to use is not available, go back to *Downloadable transcription and feature choices* or *Using a custom feature system* to learn how to load feature systems in to PCT. Be sure to click on "Save changes to this corpus's feature system" after selecting a new feature system in order to actually apply it to the corpus.

3. **Modify the feature system**: You can modify the current feature system directly within PCT. There are three options. *Be sure to click on "Save changes to this corpus's feature system" after adding a new segment or feature, or editing the feature specifications of a segment, in order to actually apply the changes to the corpus*:

   (a) **Add segment**: To add a new segment and its associated feature values to the current feature system, click on "Add segment." A new dialogue box will open up, with a space to input the symbol and dropdown menus for all of the features expected in the current system. You can also specify that all features should be set to a particular value, and then change / edit individual features as needed. Simply fill in all the values and click "OK"; the symbol and features will be added to the feature system.

   (b) **Edit segment**: To change the feature specifications of an existing segment, click on the row containing that segment and then on "Edit Segment." Then use the resulting dialogue box to change the feature specifications.

   (c) **Add feature**: To add an additional feature to the current system, click on "Add feature." Enter the name of the feature in the dialogue box, select the default value that all segments will have for this feature, and hit "OK." The feature will be added to all the segments in the corpus, with the default value. To change the value of the feature for each segment, simply click on the segment and then use the "Edit segment" functionality described above; the new feature will automatically be added to the dialogue box for each segment.

4. **Corpus inventory coverage**: There are two tools built in to help you check the coverage in your corpus.

   (a) **Extraneous symbols**: The built-in feature systems are fairly extensive, and may include symbols for sounds that do not occur in your corpus. Click on "Hide all segments not used by corpus" to remove such segments from the viewing window. (This does NOT remove them from the feature system itself; it just de-clutters your view of the system.) To revert back to the full system, simply click on "Show full feature system."

   (b) **Corpus coverage**: It's possible that there are symbols used in your corpus that are **not** covered in whatever feature system you have selected. To find out, click on "Check corpus inventory coverage." A new window will appear that either confirms that all symbols in the corpus are mapped onto features, or lists the symbols that are currently lacking. If there are symbols that are missing, you'll need to add them before doing phonological analysis on the corpus. You can do so in two ways: (1) adding them within the PCT interface, following the instructions under "Modify the feature system," immediately below; or (2) changing the feature system itself by editing the .txt file and reloading it (more information given in *Modifying an existing feature system's text file*).

5. **Display options**: The standard view (shown below) is to display the segments and features as a matrix. One can also select "tree" view, which allows one to see a list of the segments included in the transcription system, organized by phonetic characteristics (but currently without all of their feature specifications).
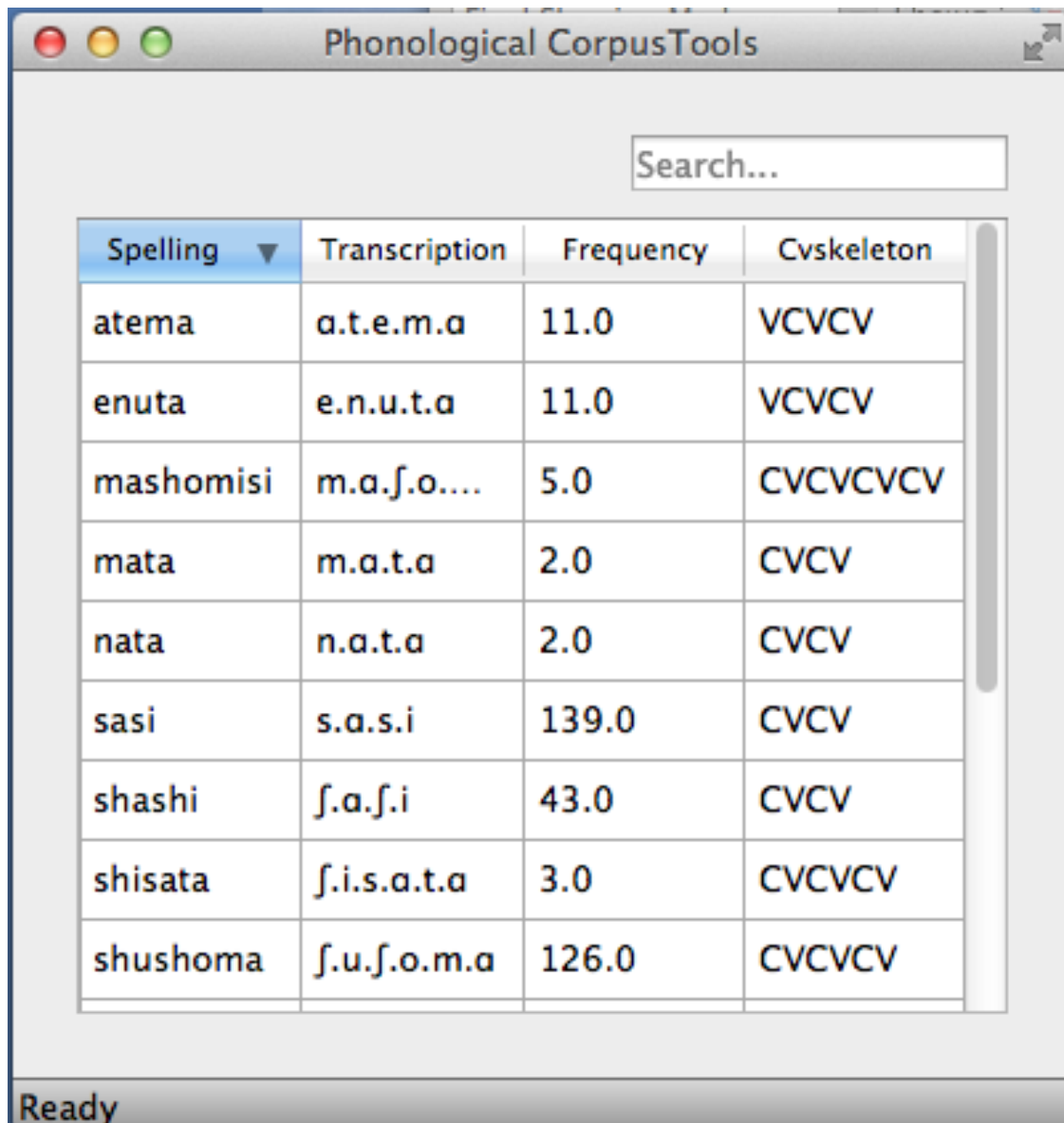
4.5 Creating new tiers in the corpus

It is possible to have PCT extract a tier of segments from the transcribed words in your corpus, based on any segment, feature, or set of features that are defined for your corpus. For example, it is easy to extract separate tiers for consonants and vowels. This extraction is particularly useful if, for example, one is interested in looking at an analysis of predictability of distribution where the conditioning contexts are non-adjacent segments; the extraction of a tier allows otherwise non-adjacent segments to be adjacent to each other on the selected tier. For example, one could examine the possibility of vowel height harmony in language X by extracting the vowels from all words and then calculating the extent to which high / low vowels are predictably distributed in high / low vowel contexts. (See also *Adding a column* for information on how to add a column to a corpus, which contains any kind of user-specified information, and *Adding a "count" column* for information on how to add a count column to a corpus, which contains counts of specific elements within each entry in the corpus.)

To create a new tier for a corpus that is currently open, click on the "Corpus" menu and select either "Add tier..." or "Add abstract tier..."; the "create tier" dialogue box opens. An "abstract" tier is a tier that is not based directly on the transcriptins themselves, but rather abstracts to a higher level. As of January 2015, the only abstract tier available is a CV skeleton tier. Before creating the tier, you can "preview" the tier as in the following example; this shows what segments PCT thinks are consonants and vowels in the current corpus.
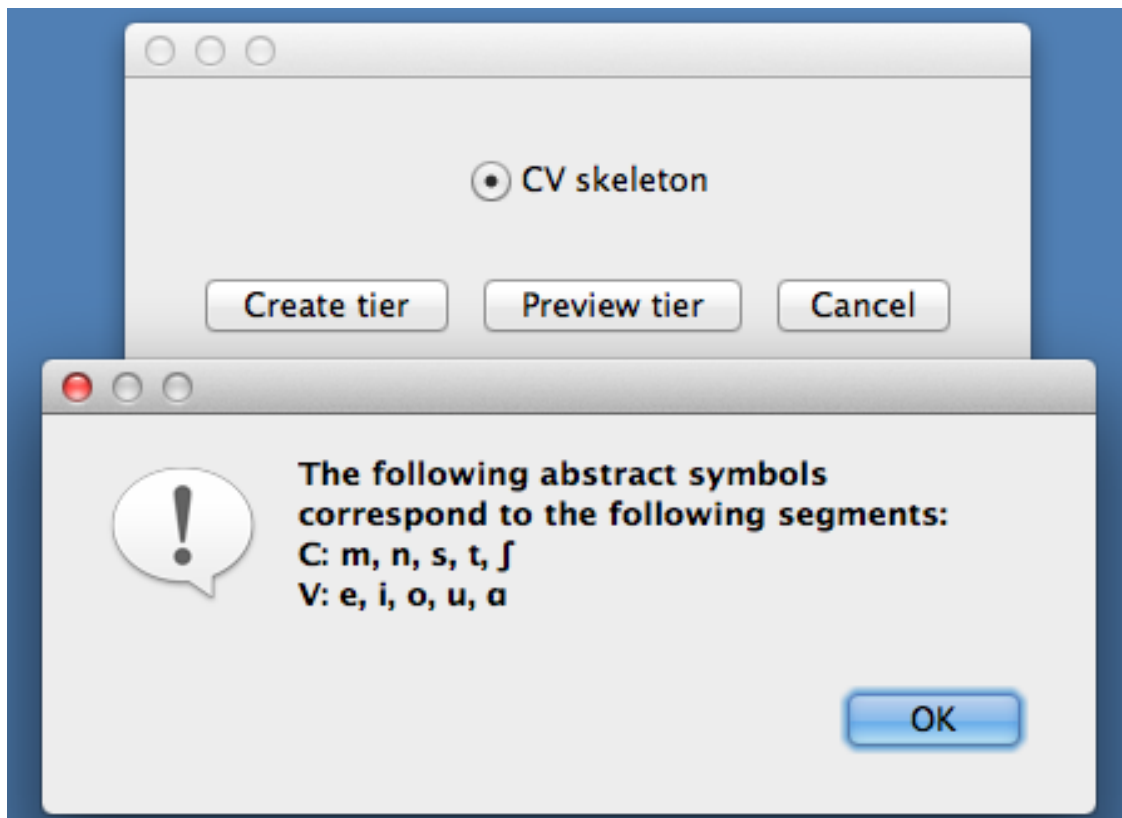
The example corpus after an abstract CV tier has been added:

To create a less abstract tier, i.e., one that is just an extraction of all transcription symbols in the corpus that have some particular feature value, use the following instructions after choosing "Corpus" / "Add tier...":

1. **Name**: Enter a short-hand name for the tier, which will appear as the column header in your corpus. For example, "vowels" or "consonants" or "nasals."

2. **Basis for creating tier**: You can create the tier using natural classes if you base the tier on features; you can also create "unnatural" tiers that are simply extractions of any set of user-defined segments.

3. **Segments**: If creating the tier on segments, you'll see all the segments in your corpus in a preview window arranged roughly along the lines of a standard IPA chart. Select as many of these as you want. The image below shows an example of creating a tier to contain all the non-mid vowels in the example corpus:

Create tier

Name of tier

Non-mid vowels

Basis for creating tier:

Segments ⬍

Segments to define the tier

Consonants

	Labial	Dental	Alveopalatal
Stop		t	
Nasal	m	n	
Fricative		s	ʃ

Vowels

	Front	Near back	Back
Close	i		u
Close mid	e		o
Open		ɑ	
Diphthongs			

Other

#

Create tier   Preview tier   Cancel

4. **Features**: If you're creating the tier based on features, you'll see the list of all available features in your corpus on the left. Select a feature that will be used to define the tier. To define the tier based on positive values for that feature, select "Add [+feature]." To define the tier based on negative values, select "Add [-feature]." For example, to define a vowel tier, one might select the [voc] feature from the SPE feature set, and then add positive values, as shown below. The selected feature will appear in the right-hand menu. To add additional features, repeat this step. For example, one could define a sibilant tier in the SPE system by selecting [+cor] and [+strid]. The features available will be based on whatever feature system has been selected as part of the corpus; see *Downloadable transcription and feature choices* for information on selecting or defining different features for the segments in the corpus.
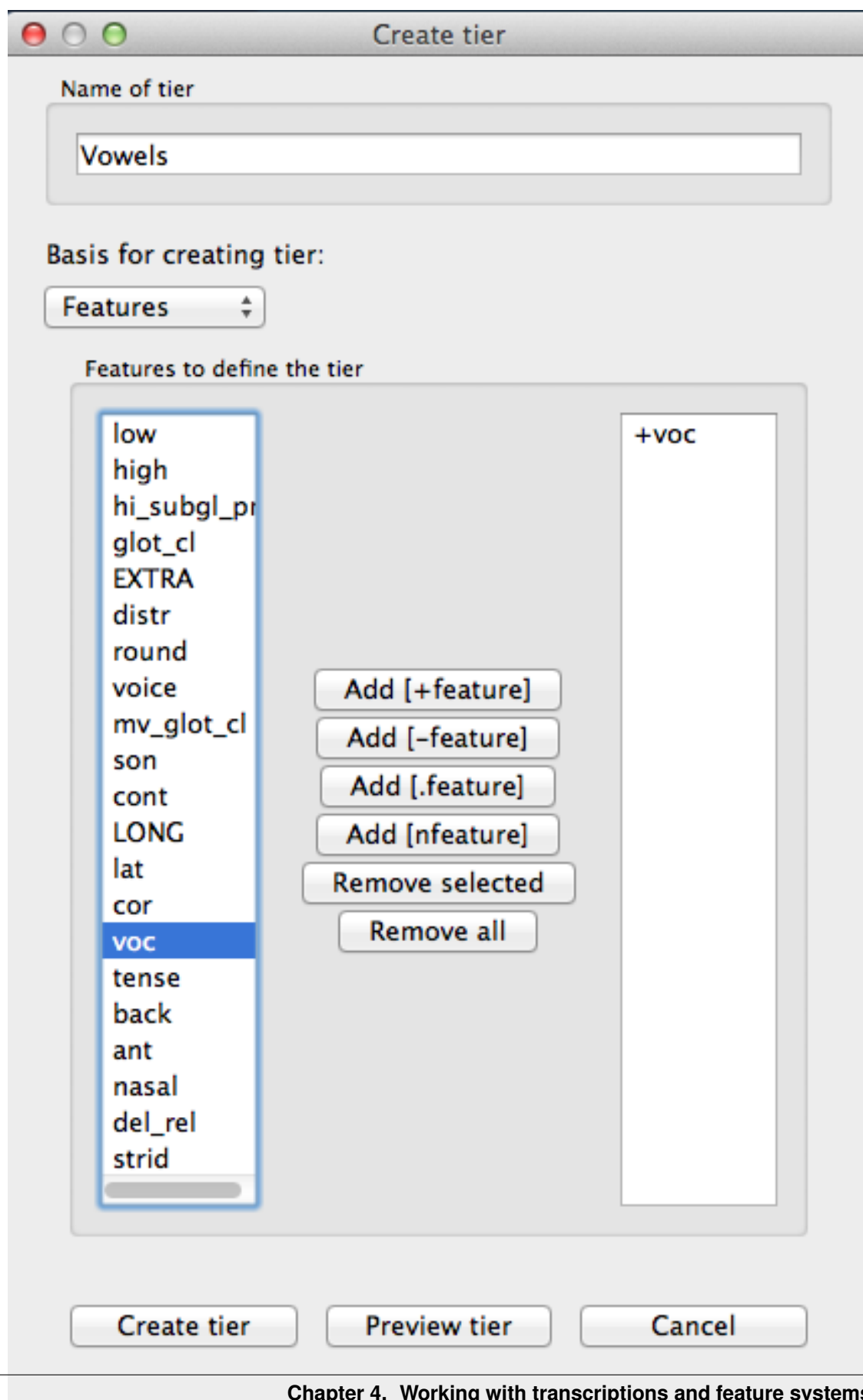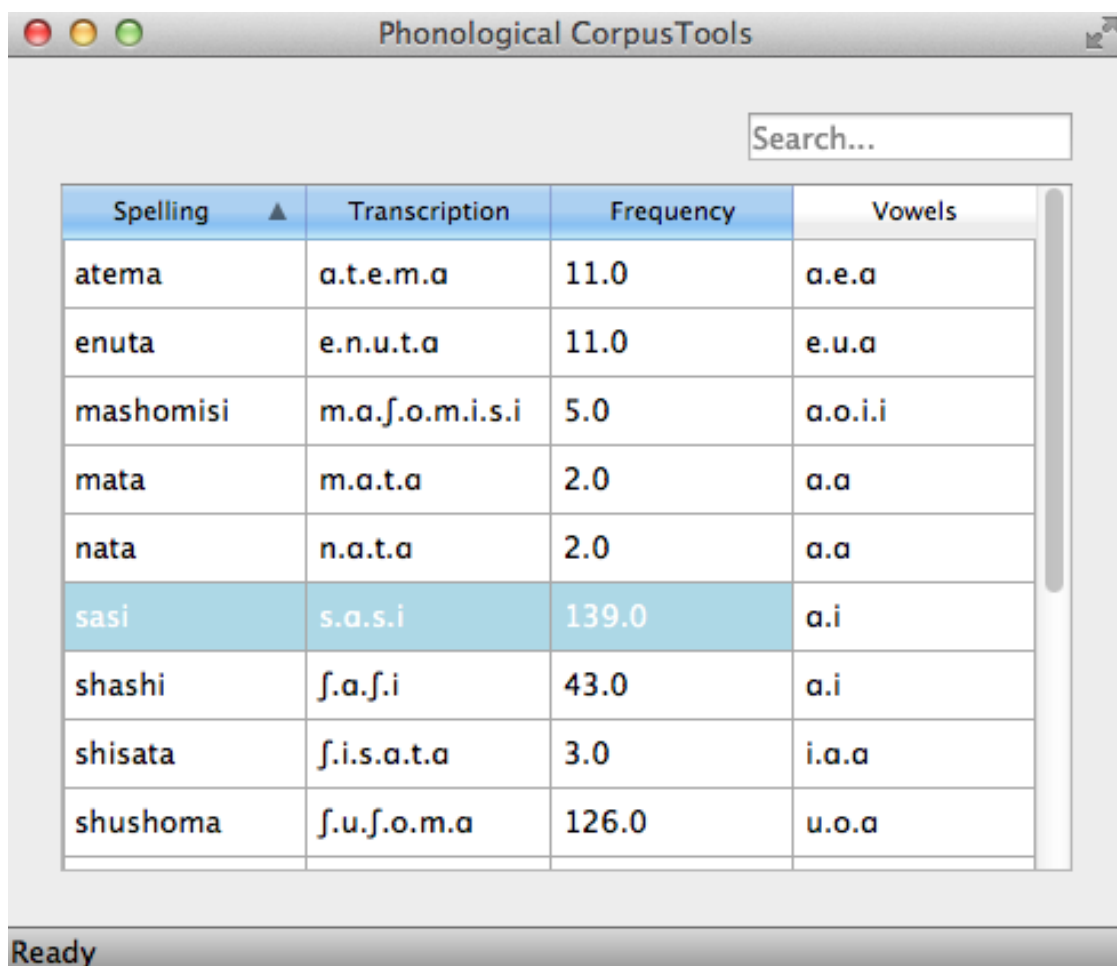
5. Preview: To see which segments the currently selected feature values will extract, select "Preview tier." This will show both the set of segments included by the selected feature values, as well as the segments that are excluded (to help verify that the selection is as expected). A vowel tier based on [+voc] in the example corpus is previewed below:

6. Removing a feature: To remove a selected feature, click on that feature in the "Selected features" menu and then choose "Remove feature."

7. Finalizing the tier: To create the tier and return to the corpus, click on "Create tier." It may take a moment to process the entire corpus, but a new column should be added to the corpus that shows the segments matching these feature selections for every word in the corpus.

8. Saving the tier: The tier can be saved along with the corpus for future use by selecting "Corpus" / "Save corpus" from the menu items (this will be done automatically if auto-save is on; see *Preferences*). It is also possible to export the corpus as a text file (.txt), which can be opened in other software, by selecting "File" / "Export corpus as text file."

9. Removing a tier: To delete a tier that has been created, simply click on "Corpus" / "Remove tier or column..." and select the tier you want to remove; then click "Remove." You can also right-click on the column name and select "Remove column." Note that only tiers that have been added through PCT can be removed; tiers that are inherent in loaded corpora cannot be removed in PCT. You can, of course, export the corpus itself to a text file, remove the column manually, and then re-load the changed corpus. To remove all the added tiers, leaving only the inherent ("essential") tiers in the corpus, select "Remove all non-essential columns." PCT will list which columns are non-essential and verify that you want to remove them before the removal is permanent. The "essential" columns for most corpora are "Spelling," "Transcription," and "Frequency."

The following shows an example of the a vowel tier added to the example corpus using the SPE feature system:

4.6 Adding, editing, and removing words, columns, and tiers

4.6.1 Adding a column

In addition to the ability to add tiers based on information already in the corpus, as described above in *Creating new tiers in the corpus*, it is also possible to add a column containing any other user-specified information to a corpus (see also *Adding a "count" column* to find out how to add a column based on counts of elements within each corpus entry). For example, one could add a "Part of Speech" column and indicate what the lexical category of each entry in the corpus is. Note that as a general proposition, it is probably easier to add such information in a spreadsheet before importing the corpus to PCT, where sorting and batch updates are easier, but we include this functionality in a basic form in case it is useful.

To add a column, go to "Corpus" / "Add column..." and do the following:

1. **Name**: Enter the name of the new column.

2. **Type of column**: Indicate what type of information the column will contain. The choices are "Spelling," "Numeric," and "Factor." A spelling column will have values that are written out as strings of characters, with each entry taken to be a unique string. A numeric column will have numeric values, upon with mathematical operations can be performed. A factor column will have values that can contain characters or numbers, but are limited in number, as in the levels of a categorical variable. This is useful when, for example, the column

encodes categorical information such as part of speech, with each entry in the corpus belonging to one of a limited set of categories such as "Noun," "Verb," and "Preposition."

3. **Default value**: A default value for the column can be entered if desired, such that every entry in the corpus receives that value in the new column. Individual entries can subsequently be edited to reflect its actual value (see *Editing a word*).

Click "Add column" to return to the corpus and see the new column, with its default values.

4.6.2 Adding a "count" column

In addition to adding columns that contain any kind of user-specified information (*Adding a column*), and tiers that contain phonological information based on the entries themselves (*Creating new tiers in the corpus*), one can also add "Count" columns, which contain information about the *number* of occurrences of a feature or segment in each entry in a corpus. For example, one could add a column that lists, for each entry, the number of round vowels that are contained in that entry. To add a count column, go to "Corpus" / "Add count column..." and then do the following:

1. **Name**: Enter the name of the new column.

2. **Tier**: Specify what tier the count column should refer to in order to determine the counts (e.g., transcription or a derived tier such as a vowel tier).

3. **Segment selection**: Choose to count things based on either segments or features. If segments are counted, select the segment or segments from the inventory (click on "Consonants" and / or "Vowels" to reveal which segments are available in the inventory). If features are selected, click on each feature and then the value of that feature that should be used.

Click "Add count column" to return to the corpus and see the new column, with its count values automatically filled in.

4.6.3 Removing a tier or column

To delete a tier or column that has been created, simply click on "Corpus" / "Remove tier or column..." and select the tier you want to remove; then click "Remove." Note that only tiers that have been added through PCT can be removed; tiers that are inherent in loaded corpora cannot be removed in PCT. You can, of course, export the corpus itself to a text file, remove the column manually, and then re-load the changed corpus. To remove all the added tiers, leaving only the inherent ("essential") tiers in the corpus, select "Remove all non-essential columns." PCT will list which columns are non-essential and verify that you want to remove them before the removal is permanent. The "essential" columns for most corpora are "Spelling," "Transcription," and "Frequency."

4.6.4 Adding a word

As a general proposition, we don't recommend using PCT as a database manager. It is designed to facilitate analyses of pre-existing corpora rather than to be an interface for creating corpora. That said, it is occasionally useful to be able to add a word to a pre-existing corpus in PCT. Note that this function will actually add the word to the corpus (and, if auto-save is on, the word will be saved automatically in future iterations of the corpus). If you simply need to add a word temporarily, e.g., to calculate the neighbourhood density of a hypothetical word given the current corpus, you can also add a word in the relevant function's dialogue box, without adding the word permanently to the corpus.

To do add the word globally, howveer, go to "Corpus" / "Add new word..." and do the following:

1. **Spelling**: Type in the orthographic representation of the new word.

2. **Transcription**: To add in the phonetic transcription of the new word, it is best to use the provided inventory. While it is possible to type directly in to the transcription box, using the provided inventory will ensure that all characters are understood by PCT to correspond to existing characters in the corpus (with their concomitant

featural interpretation). Click on "Show inventory" and then choose to show "Consonants," "Vowels," and/or other. (If there is no featural interpretation of your inventory, you will simply see a list of all the available segments, but they will not be classifed by major category.) Clicking on the individual segments will add them to the transcription. The selections will remain even when the sub-inventories are hidden; we allow for showing / hiding the inventories to ensure that all relevant buttons on the dialogue box are available, even on small computer screens. Note that you do NOT need to include word boundaries at the beginning and end of the word, even when the boundary symbol is included as a member of the inventory; these will be assumed automatically by PCT.

3. **Frequency**: Enter the token frequency of this word.

4. **Other**: If there are other tiers or columns in your corpus, you can also enter the relevant values for those columns in the dialogue box. For tiers that are defined via features, the values should be automatically populated as you enter the transcription. E.g., if you have a vowel tier, and add the word [pim.ku] to your corpus by selecting the relevant segments from the inventory, the vowel tier should automatically fill in the entry as [i.u].

Once all values are filled in, select "Create word" to return to the corpus with the word added. If auto-save is not on, you can save this new version of the corpus for future use by going to "File" / "Save corpus." If you have added a word and the corpus has NOT been saved (either manually or through Auto-save) afterward, and then try to quit PCT, it will warn you that you have unsaved changes and ask that you verify that you want to quit without saving them.

4.6.5 Removing a word

To remove a word from the corpus, select it in the corpus view and right-click (ctrl-click on a Mac) on it. Choose "Remove word" from the menu. Regardless of whether warnings are turned on or not (see *Help and warnings*), PCT will verify that you want to remove the word before commiting the change. Word removal is not auto-saved with a corpus, even if "Auto-save" is turned on (see *Preferences*); if you want to save the new version of the corpus with the word removed, you should explicitly go to "File" / "Save corpus." If you have removed a word and NOT manually saved the corpus afterward, and then try to quit PCT, it will again warn you that you have unsaved changes and ask that you verify that you want to quit.

4.6.6 Editing a word

To edit a word in the corpus, right-click on the word's entry and chooser "Edit word details," or double-click the word's entry in the corpus. A dialogue box opens that shows the word's spelling, transcription, frequency, and any other information that is included in the corpus. Most of these entries can be edited manually, though a few, such as tiers that are dependent on a word's transcription, cannot themselves be directly edited. To edit such a derived tier, edit the transcription of the word; the derived tier will update automatically as the new transcription is provided.

4.6.7 Hiding / showing non-transcribed items

When working with a corpus, it is possible to hide all entries that do not have a transcription (if any such entries exist). To do this, right-click anywhere in the corpus itself and select "Hide non-transcribed items." To reveal them again, right-click anywhere in the corpus itself and select "Show non-transcribed items."

4.7 Phonological Search

PCT allows you to do searches for various strings, defined by segments or features. The search returns two types of information: one, a general count of the number of entries that fit the search description, and two, the actual list of all the words in the corpus that contain the specified string. To conduct a search, choose "Corpus" / "Phonological search..." and do the following:

1. **Basis for search**: Select either segments or features to search for.

2. **Segment-based search selection**: If segments are chosen as the basis of the search, you will see the segmental inventory of your corpus. Select a segment that you want to search for.

3. **Feature-based search selection**: If features are chosen as the basis of the search, you will see the features associated with your corpus. Select the feature and then the value of that feature that you want to search for; it will automatically be added to the list. Multiple features can be selected (e.g., [-voice, -continuant] if one wants to search for voiceless stops).

4. **Environments**: Click on "Environments" to add one or more environments that you want to search for that contain the specified segment. Both the left- and right-hand side of the environment can be specified, and each can be specified using either segments (automatically populated by your corpus' inventory) or features (automatically populated by the feature system associated with the corpus). After selecting an environment by clicking on the relevant segments or features, click on "Add" if it is the last environment you want to add, or "Add and create another" if you want to add additional environments.

An example of adding environments (in this case, the environment "word-initial, before a vowel"):
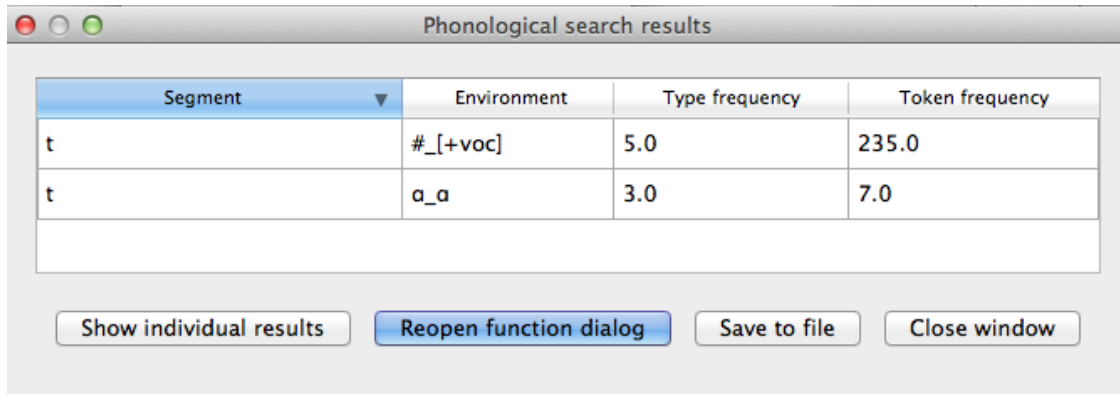
5. Tier: Select the tier on which phonological search should be performed. The default would be the transcription tier, so that phonological environments are defined locally. But, for example, if a vowel tier is selected, then one could search for the occurrence of, e.g., [i] before mid vowels on that tier (hence ignoring intervening consonants).

An example of the phonological search window, set up to search for voiceless stops word-initially before vowels and between [a] vowels, on the transcription tier:
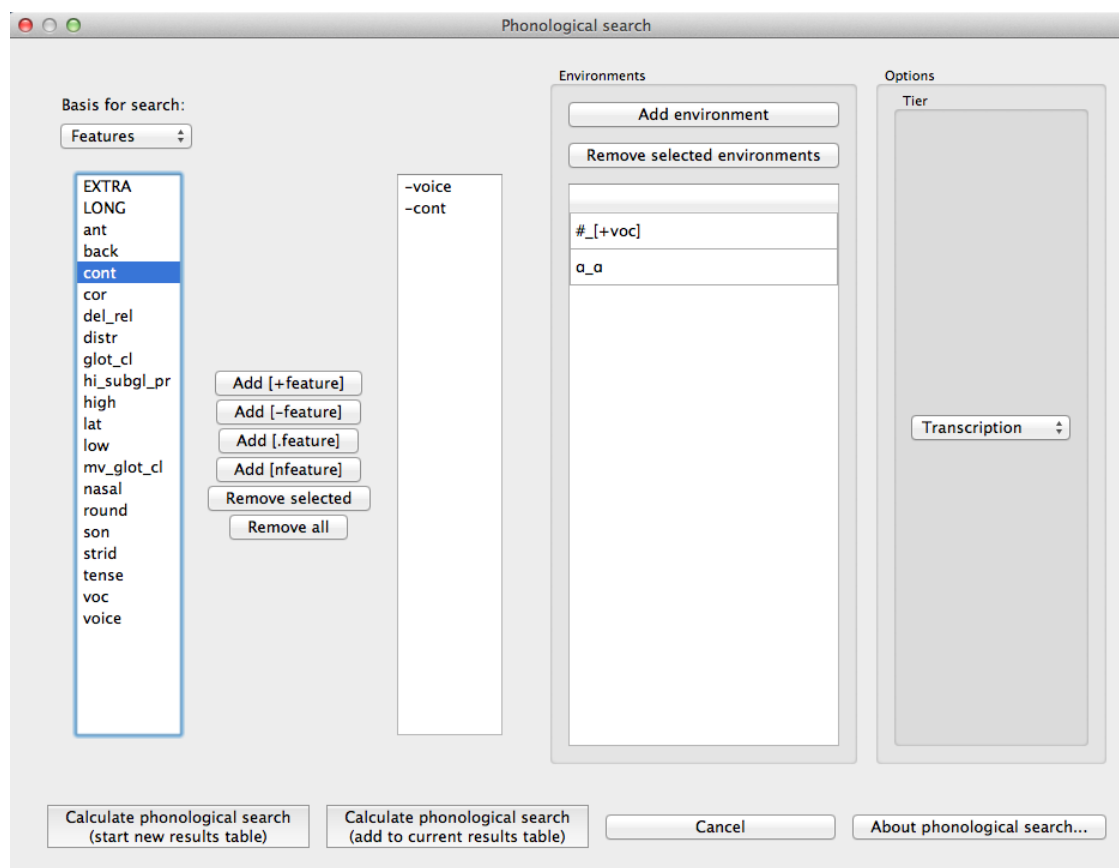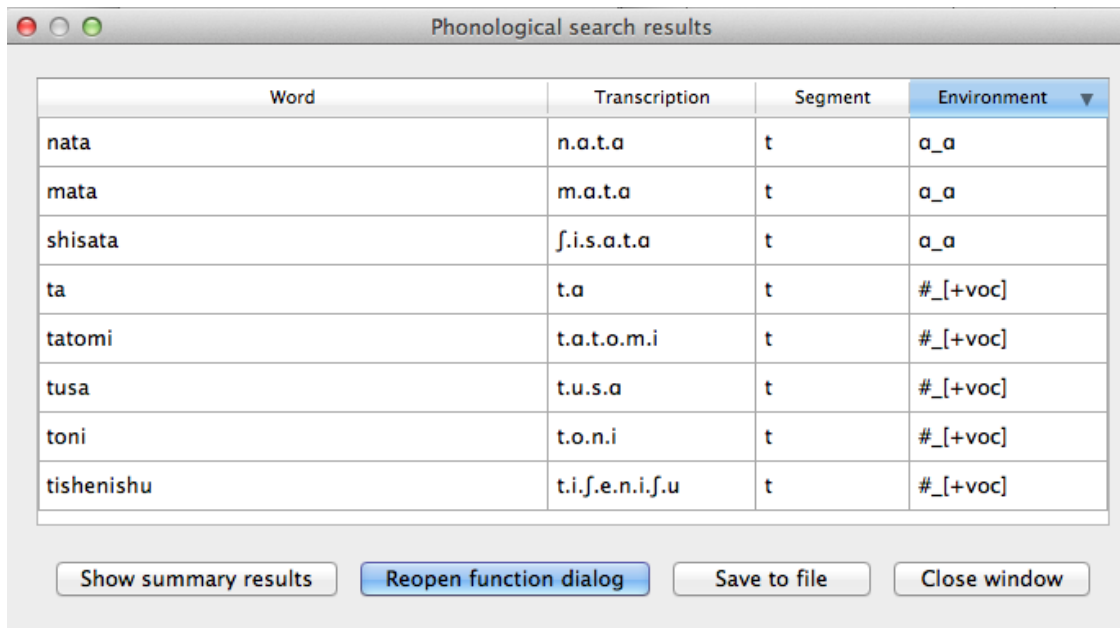
6. Results: Once all selections have been made, click on "Calculate phonological search." If there is not already an existing results table, or you want to start a new once, choose the "Start new results table" option. If you want to add the results to a pre-existing table, choose the "Add to current results table" option. The results appear in a new dialogue box that first shows the summary results, i.e., a list that contains the segment that was searched for, each environment that was searched for, the total count of words that contain that segment in that environment, and the total token frequency for those words (note that these are the frequencies of the WORDS containing the specified environments, so if for example, a particular word contains multiple instances of the same environment, this is NOT reflected in the counts). The individual words in the corpus that match the search criteria can be shown by clicking on "Show individual results" at the bottom of the screen; this opens a new dialogue box in which each word in the corpus that matches the search criteria is listed, including the transcription of the word, the segment that was found that matches the search criteria, and which environment that segment occurred in in that word. Note that the results can be sorted by any of the columns by clicking on that column's name (e.g., to get all the words that contained the [a_a] environment together, simply click on the "Environment" label at the top of that column). To return to the summary results, click on "Show summary results." Each set of results can be saved to a .txt file by clicking "Save to file" at the bottom of the relevant results window. To return to the search selection dialogue box, click on "Reopen function dialogue." Otherwise, when finished, click on "Close window" to return to the corpus.

An example of the summary results window for the above phonological search:

Segment ▼	Environment	Type frequency	Token frequency
t	#_[+voc]	5.0	235.0
t	ɑ_ɑ	3.0	7.0

Show individual results — Reopen function dialog — Save to file — Close window

And the individual results from the same search, sorted by environment:

Word	Transcription	Segment	Environment ▼
nata	n.a.t.ɑ	t	ɑ_ɑ
mata	m.a.t.ɑ	t	ɑ_ɑ
shisata	ʃ.i.s.a.t.ɑ	t	ɑ_ɑ
ta	t.ɑ	t	#_[+voc]
tatomi	t.a.t.o.m.i	t	#_[+voc]
tusa	t.u.s.ɑ	t	#_[+voc]
toni	t.o.n.i	t	#_[+voc]
tishenishu	t.i.ʃ.e.n.i.ʃ.u	t	#_[+voc]

Show summary results — Reopen function dialog — Save to file — Close window

Phonotactic Probability

5.1 About the function

Phonotactic probability refers to the likelihood of a given set of segments occurring in a given order for a given corpus of transcriptions. For instance, *blick* is a phonotactically probable nonword in English, but *bnick* is phonotactically improbable. Words as well as nonwords can be assessed for their phonotactic probability, and this measure has been used in behavioural research ([Vitevitch1999] and others). In particular, the phonotactic probability of words has been correlated with their ability to be segmented, acquired, processed, and produced; see especially the discussion in [Vitevitch2004] for extensive references.

5.2 Method of calculation

One method for computing the phonotactic probability uses average unigram or bigram positional probabilities across a word ([Vitevitch2004]; their online calculator for this function is available here). For a word like *blick* in English, the unigram average would include the probability of /b/ occurring in the first position of a word, the probability of /l/ in the second position, the probability of // occuring in the third position, and the probability of /k/ occurring in the fourth position of a word. Each positional probability is calculated by summing the log token frequency of words containing that segment in that position divided by the sum of the log token frequency of all words that have that position in their transcription. The bigram average is calculated in an equivalent way, except that sequences of two segments and their positions are used instead of single segments. So for *blick* that would be /bl/, /l/, /k/ as the included poisitional probabilities. As with all n-gram based approaches, bigrams are preferable to unigrams. In the example of *blick* versus *bnick*, unigrams wouldn't likely capture the intuitive difference in phonotactic probability, since the probability of /n/ and /l/ in the second position isn't necessarily radically different. Using bigrams, however, would capture that the probability of /bl/ versus /bn/ in the first position is radically different.

There are other ways of calculating phonotactic probability that don't have the strict left-to-right positional assumptions that the Vitevitch & Luce algorithm has, such as the constraint-based method in BLICK by Bruce Hayes (Windows executable available here, Python package available here with source code available at <https://github.com/mmcauliffe/python-BLICK/>_). However, such algorithms require training on a specific language, and the constraints are not computed from transcribed corpora in as straightforward a manner as the probabilities used in the Vitevitch & Luce algorithm. Therefore, PCT currently supports only the Vitevitch & Luce style algorithm.

## 5.3 Implementing the phonotactic probability function in the GUI

To start the analysis, click on "Analysis" / "Calculate phonotactic probability..." in the main menu, and then follow these steps:

1. **Phonotactic probability algorithm**: Currently the only offered algorithm is the Vitevitch & Luce algorithm, described above.

2. **Query type**: Phonotactic probability can be calculated for one of three types of inputs:

   (a) **One word**: The phonotactic probability of a single word can be calculated by entering that word's orthographic representation in the query box.

   (b) **One word/nonword not in the corpus**: The phonotactic probability can be calculated on a word that is not itself in the corpus, but using the probabilities derived from the corpus. These words are distinct from the corpus and won't be added to it, nor will their creation affect any future calculations. See *Adding a word* for information on how to more permanently add a new word to the corpus. Words can be created through the dialogue opened by pressing the button:

      i. **Spelling**: Enter the spelling for your new word / nonword using the regular input keyboard on your computer. The spelling is how the word will be referenced in the results table, but won't affect the calculation of phonotactic probability.

      ii. **Transcription**: To add in the phonetic transcription of the new word, it is best to use the provided inventory. While it is possible to type directly in to the transcription box, using the provided inventory will ensure that all characters are understood by PCT to correspond to existing characters in the corpus (with their concomitant featural interpretation). Click on "Show inventory" and then choose to show "Consonants," "Vowels," and/or other. (If there is no featural interpretation of your inventory, you will simply see a list of all the available segments, but they will not be classifed by major category.) Clicking on the individual segments will add them to the transcription. The selections will remain even when the sub-inventories are hidden; we allow for showing / hiding the inventories to ensure that all relevant buttons on the dialogue box are available, even on small computer screens. Note that you do NOT need to include word boundaries at the beginning and end of the word, even when the boundary symbol is included as a member of the inventory; these will be assumed automatically by PCT.

      iii. **Frequency**: This can be left at the default. Note that entering a value will NOT affect the calculation; there is no particular need to enter anything here (it is an artifact of using the same dialogue box here as in the "Add Word" function described in *Adding a word*).

      iv. **Create word**: To finish and return to the "Phonotactic probability" dialogue box, click on "Create word."

   (c) **List of words**: If there is a specific list of words for which phonotactic probability is to be calculated (e.g., the stimuli list for an experiment), that list can be saved as a .txt file with one word per line and uploaded into PCT for analysis. If words in the list are not in the corpus, you can still calculate their phonotactic probability by entering in the spelling of the word and the transcription of the word in a single line delimited by a tab. The transcription should be delimited by periods.

   (d) **Whole corpus**: Alternatively, the phonotactic probability for every current word in the corpus can be calculated. The phonotactic probability of each word will be added to the corpus itself, as a separate column; in the "query" box, simply enter the name of that column (the default is "Phonotactic probability").

3. **Tier**: Phonotactic probability can be calculated from transcription tiers in a corpus (e.g., transcription or tiers that represent subsets of entries, such as a vowel or consonant tier).

4. **Type vs. token frequency**: Specify whether phonotactic probabilities should be based on word type frequency or token frequency. The original Vitevitch & Luce algorithm uses token frequency. Token frequency will use the log frequency when calculating probabilities.

5. **Probability type**: Specify whether to use biphone positional probabilities or single segment positional probabilities. Defaults to biphone.

6. **Results**: Once all options have been selected, click "Calculate phonotactic probability." If this is not the first calculation, and you want to add the results to a pre-existing results table, select the choice that says "add to

current results table." Otherwise, select "start new results table." A dialogue box will open, showing a table of the results, including the word, its phonotactic probability, the transcription tier from which phonotactic probability was calculated, whether type or token frequency was used, whether the algorithm used unigram or bigram probabilities, and the phonotactic probability algorithm that was used. If the phonotactic probability for all words in the corpus is being calculated, simply click on the "start new results table" option, and you will be returned to your corpus, where a new column has been added automatically.

7. **Saving results**: The results tables can each be saved to tab-delimited .txt files by selecting "Save to file" at the bottom of the window. If all phonotactic probabilities are calculated for a corpus, the corpus itself can be saved by going to "File" / "Export corpus as text file," from where it can be reloaded into PCT for use in future sessions with the phonotactic probabilities included.

An example of the "Phonotactic Probability" dialogue box for calculating the probability of the non-word "pidger" [pd] using unigram position probabilities (using the IPHOD corpus):
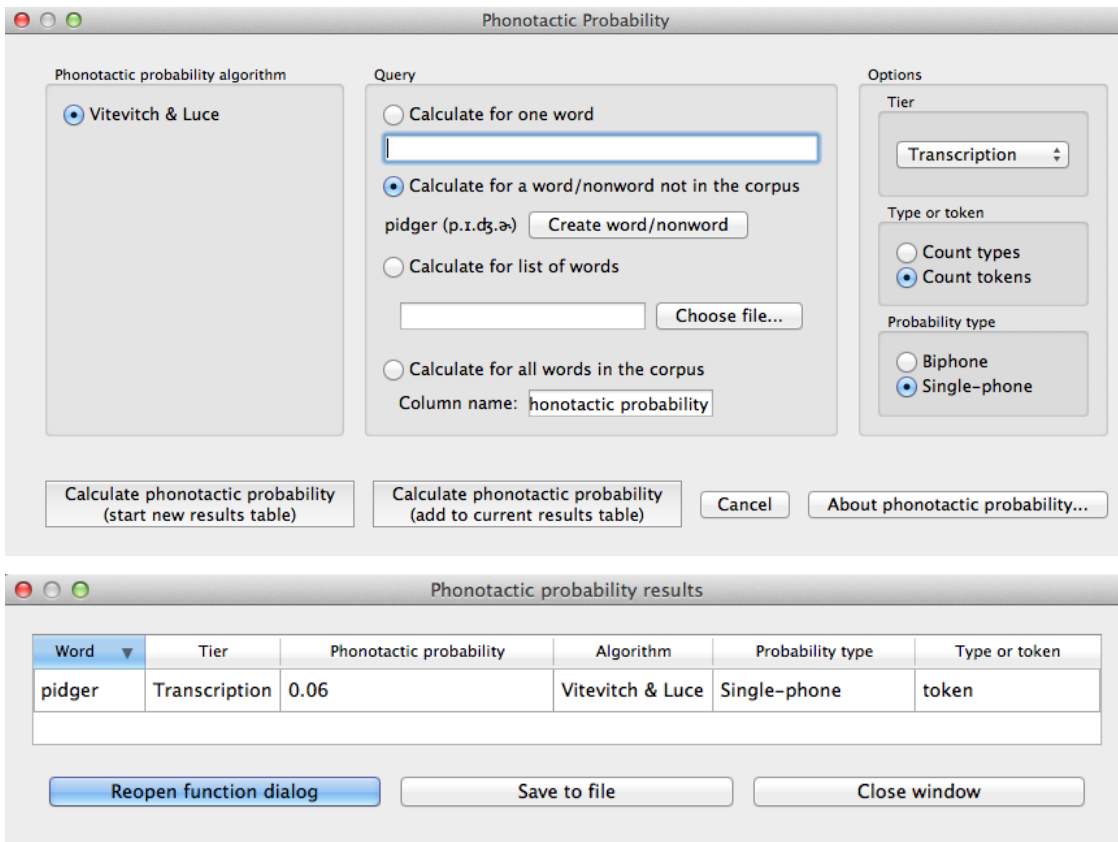
To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

Functional Load

6.1 About the function

Functional load is a measure of the "work" that any particular contrast does in a language, as compared to other contrasts (e.g., [Hockett1955], [Hockett1966]; [Kucera1963]; [King1967]; [Surendran2003]). Two contrasts in a language, such as [d] / [t] vs. [ð] / [θ] in English, may have very different functional loads. The difference between [d] and [t] is used to distinguish between many different lexical items, so it has a high functional load; there are, on the other hand, very few lexical items that hinge on the distinction between [ð] and [θ], so its functional load is much lower. One of the primary claims about functional load is that it is related to sounds' propensity to merge over time, with pairs of sounds that have higher functional loads being less likely to merge than pairs of sounds with lower functional loads (e.g., [Wedel2013], [Todd2012]). The average functional load of a particular sound has also been claimed to affect its likelihood of being used as an epenthetic vowel [Hume2013]. Functional load has also been illustrated to affect the perceived similarity of sounds [Hall2014a].

6.2 Method of calculation

There are two primary ways of calculating functional load that are provided as part of the PCT package. One is based on the change of entropy in a system upon merger of a segment pair or set of segment pairs (cf. [Surendran2003]); the other is based on simply counting up the number of minimal pairs (differing in only the target segment pair or pairs) that occur in the corpus.

6.2.1 Change in entropy

The calculation based on change in entropy is described in detail in [Surendran2003]. Entropy is an Information-Theoretic measure of the amount of uncertainty in a system [Shannon1949], and is calculated using the formula in (1); it will also be used for the calculation of predictability of distribution (see *Method of calculation*). For every symbol i in some inventory (e.g., every phoneme in the phoneme inventory, or every word in the lexicon), one multiplies the probability of i by the log_2 of the probability of i; the entropy is the sum of the products for all symbols in the inventory.

Entropy:

$$H = - \sum_{i \in N} p_i * log_2(p_i)$$

The functional load of any pair of sounds in the system, then, can be calculated by first calculating the entropy of the system at some level of structure (e.g., words, syllables) with all sounds included, then merging the pair of sounds in question and re-calculating the entropy of the new system. That is, the functional load is the amount of uncertainty (entropy) that is lost by the merger. If the pair has a functional load of 0, then nothing has changed when the two are

merged, and H_1 will equal H_2. If the pair has a non-zero functional load, then the total inventory has become smaller through the conflating of pairs of symbols that were distinguished only through the given pair of sounds.

Functional load as change in entropy:

$\Delta H = H_1 - H_2$

Consider a toy example, in which the following corpus is assumed (note that, generally speaking, there is no "type frequency" column in a PCT corpus, as it is assumed that each row in the corpus represents 1 type; it is included here for clarity):

Consider a toy example, in which the following corpus is assumed (note that, generally speaking, there is no "type frequency" column in a PCT corpus, as it is assumed that each row in the corpus represents 1 type; it is included here for clarity):

Word	Original Trans.	Type Freq.	Token Freq.	Under [h] / [■] merger Trans.	Type Freq.	Token Freq.	Under [t] / [d] merger Trans.	Type Freq.	Token Freq.
hot	[ht]	1	2	[Xt]	1	2	[hX]	1	2
song	[s■]	1	4	[sX]	1	4	[s■]	1	4
hat	[hæt]	1	1	[Xæt]	1	1	[hæX]	1	1
sing	[s■]	1	6	[sX]	1	6	[s■]	1	6
tot	[tt]	1	3	[tt]	1	3	[XX]	1	8
dot	[dt]	1	5	[dt]	1	5	[XX]		
hip	[hp]	1	2	[Xp]	1	2	[hp]	1	
2									
hid	[hd]	1	7	[Xd]	1	7	[hX]	1	7
team	[tim]	1	5	[tim]	1	5	[Xim]	1	10
deem	[dim]	1	5	[dim]	1	5	[Xim]		
toot	[tut]	1	9	[tut]	1	9	[XuX]	1	
11									
dude	[dud]	1	2	[dud]	1	2	[XuX]		
hiss	[hs]	1	3	[Xs]	1	3	[hs]	1	
3									
his	[hz]	1	5	[Xz]	1	5	[hz]	1	5
siz-zle	[szl]	1	4	[szl]	1	4	[szl]	1	4
dizzy	[dzi]	1	3	[dzi]	1	3	[Xzi]	1	7
tizzy	[tzi]	1	4	[tzi]	1	4	[Xzi]		
Total		17	70		17	70		13	70

The starting entropy, assuming word types as the relative unit of structure and counting, is:

$H_{1-types} = -[(\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17}))$
$+ (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17}))$
$+ (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17})) + (\frac{1}{17}log_2(\frac{1}{17}))] = 4.087$

The starting entropy, assuming word tokens, is:

$H_{1-tokens} = -[(\frac{2}{70}log_2(\frac{2}{70})) + (\frac{4}{70}log_2(\frac{4}{70})) + (\frac{1}{70}log_2(\frac{1}{70})) + (\frac{6}{70}log_2(\frac{6}{70})) + (\frac{3}{70}log_2(\frac{3}{70}))$
$+ (\frac{5}{70}log_2(\frac{5}{70})) + (\frac{2}{70}log_2(\frac{2}{70})) + (\frac{7}{70}log_2(\frac{7}{70})) + (\frac{5}{70}log_2(\frac{5}{70})) + (\frac{5}{70}log_2(\frac{5}{70})) + (\frac{9}{70}log_2(\frac{9}{70}))$
$+ (\frac{2}{70}log_2(\frac{2}{70})) + (\frac{3}{70}log_2(\frac{3}{70})) + (\frac{5}{70}log_2(\frac{5}{70})) + (\frac{4}{70}log_2(\frac{4}{70})) + (\frac{3}{70}log_2(\frac{3}{70})) + (\frac{4}{70}log_2(\frac{4}{70}))] = 3.924$

Upon merger of [h] and [■], there is no change in the number of unique words; there are still 17 unique words with all their same token frequencies. Thus, the entropy after an [h] / [■] merger will be the same as it was before the merger. The functional load, then would be 0, as the pre-merger and post-merger entropies are identical.

Upon merger of [t] and [d], on the other hand, four pairs of words have been collapsed. E.g., the difference between

team and *deem* no longer exists; there is now just one word, [Xim], where [X] represents the result of the merger. Thus, there are only 13 unique words, and while the total token frequency count remains the same, at 70, those 70 occurrences are divided among only 13 unique words instead of 17.

Thus, the entropy after a [t] / [d] merger, assuming word types, is:

$$H_{1-types} = -[(\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13}))$$
$$+ (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13}))$$
$$+ (\frac{1}{13}log_2(\frac{1}{13})) + (\frac{1}{13}log_2(\frac{1}{13}))] = 3.700$$

And the entropy after a [t] / [d] merger, assuming word tokens, is:

$$H_{1-tokens} = -[(\frac{2}{70}log_2(\frac{2}{70})) + (\frac{4}{70}log_2(\frac{4}{70})) + (\frac{1}{70}log_2(\frac{1}{70})) + (\frac{6}{70}log_2(\frac{6}{70})) + (\frac{8}{70}log_2(\frac{8}{70}))$$
$$+ (\frac{2}{70}log_2(\frac{2}{70})) + (\frac{7}{70}log_2(\frac{7}{70})) + (\frac{10}{70}log_2(\frac{10}{70})) + (\frac{11}{70}log_2(\frac{11}{70})) + (\frac{3}{70}log_2(\frac{3}{70})) + (\frac{5}{70}log_2(\frac{5}{70}))$$
$$+ (\frac{4}{70}log_2(\frac{4}{70})) + (\frac{7}{70}log_2(\frac{7}{70}))] = 3.466$$

$$\Delta H = H_{1-types} - H_{2-types} = 4.087 \check{~} 3.700 = 0.387$$

And the functional load of [t] / [d] based on word tokens is:

$$\Delta H = H_{1-tokens} - H_{2-tokens} = 3.924 \check{~} 3.466 = 0.458$$

6.2.2 (Relative) Minimal Pair Counts

The second means of calculating functional load that is included in PCT is a straight count of minimal pairs, which can be relativized to the number of words in the corpus that are potential minimal pairs—i.e. the number of words in the corpus with at least one of the target segments.

In the above example, the number of minimal pairs that hinge on [h] vs. [■] is of course 0, so the functional load of [h] / [■] is 0. The number of minimal pairs that hinge on [t] / [d] is 3, and the number of words with either [t] or [d] is 11; the functional load as a relativized minimal pair count would therefore be 3/11 = 0.273. Note that here, a relatively loose definition of minimal pair is used; specifically, two words are considered to be a minimal pair hinging on sounds A and B if, upon merger of A and B into a single symbol X, the words are identical. Thus, *toot* and *dude* are considered a minimal pair on this definition, because they both become [XuX] upon merger of [t] and [d].

The resulting calculations of functional load are thus quite similar between the two measures, but the units are entirely different. Functional load based on change in entropy is measured in *bits*, while functional load based on relativized minimal pair counts is simply a percentage. Also note that functional load based on minimal pairs is only based on type frequency; the frequency of the usage of the words is not used as a weighting factor, the way it can be under the calculation of functional load as change in entropy.

6.3 Implementing the functional load function in the GUI

As with most analysis functions, a corpus must first be loaded (see *Loading in corpora*). Once a corpus is loaded, use the following steps.

1. **Getting started**: Choose "Analysis" / "Calculate functional load..." from the top menu bar.

2. **Sound selection**: First, select which two sounds you want the functional load to be calculated for. Do this by clicking on "Add pair of sounds"; the "Select segment pair" dialogue box will open. The segment choices that are available will automatically correspond to all of the unique transcribed characters in your corpus. The order of the sounds is irrelevant; picking [i] first and [u] second will yield the same results as picking [u] first and [i] second. Once a pair of sounds has been selected, click "Add." They will appear in the "Functional load" dialogue box. Multiple pairs of sounds can be selected and added to the list for calculation simultaneously. To do this without going back to the "Functional Load" dialogue box first, click "Add and create another." When multiple pairs are selected, they can be treated in two different ways, listed under "Options" on the right-hand side of the "Functional Load" dialogue box under "Multiple segment pair behaviour":

(a) **All segment pairs together**: This option allows for the calculation of the functional load of featural contrasts. E.g., if the pairs [e]/[i] and [o]/[u] are chosen, PCT will calculate the functional load from both pairs at the same time. This option is useful for investigating the functional load of featural contrasts: e.g., if the above pairs are the ONLY pairs of sounds in the corpus that differ by exactly the single feature [high], then this option will allow you to calculate the functional load of the [high] contrast. Note that the results table will list "[e], [o]" as "sound 1" and "[i], [u]" as "sound 2" in this scenario, to remind you that you are getting a single functional load value. Note too that this does not collapse all four sounds to a single sound (which would erroneously also neutralize [e]/[o], [e]/[u], [i]/[o], [i]/[u]), but rather collapses each pair of segments and only then checks for any minimal pairs or drop in entropy.

(b) **Each segment pair individually**: This option cycles through the list of pairs and gives the functional load of each pair individually from the corpus. E.g., if the pairs [e]/[i] and [o]/[u] are chosen, you will get results showing first the functional load of [e]/[i] in the corpus and then the functional load of [o]/[u] in the corpus, independently.

3. **Functional load algorithm**: Select which of the two methods of calculation you want to use—i.e., minimal pairs or change in entropy. (See discussion above for details of each.)

4. **Tier**: Select which tier the functional load should be calculated from. The default is the "transcription" tier, i.e., looking at the entire word transcriptions. If another tier has been created (see *Creating new tiers in the corpus*), functional load can be calculated on the basis of that tier. For example, if a vowel tier has been created, then "minimal pairs" will be entries that are identical except for one entry in the vowels only, entirely independently of consonants. Thus, the words [mapotik] and [aefli] would be treated as a minimal pair, given that their vowel-tier representations are [aoi] and [aei].

5. **Minimum frequency**: It is possible to set a minimum token frequency for words in the corpus in order to be included in the calculation. This allows easy exclusion of rare words; for example, if one were calculating the functional load of [s] vs. [] in English and didn't set a minimum frequency, words such as *santy* (vs. *shanty*) might be included, which might not be a particularly accurate reflection of the phonological knowledge of speakers. To include all words in the corpus, regardless of their token frequency, set the the minimum frequency to 0.

6. **Additional parameters for minimal pairs**: If minimal pairs serve as the means of calculation, there are two additional parameters can be set.

(a) **Raw vs. relative count**: First, PCT can report only the raw count of minimal pairs that hinge on the contrast in the corpus, if you just want to know the scope of the contrast. On the other hand, the default is to relativize the raw count to the corpus size, by dividing the raw number by the number of lexical entries that include at least one instance of any of the target segments.

(b) **Include vs. ignore homophones**: Second, PCT can either include homophones or ignore them. For example, if the corpus includes separate entries for the words *sock* (n.), *sock* (v.), *shock* (n.), and *shock* (v.), this would count as four minimal pairs if homophones are included, but only one if homophones are ignored. The default is to ignore homophones.

7. **Additional parameters for change in entropy**: If you are calculating functional load using change in entropy, one additional parameter can be set.

(a) **Type or token frequency**: As described in *Change in entropy*, entropy can be calculated using either type or token frequencies. This option determines which to use.

Here is an example of selecting [m] and [n], with functional load to be calculated on the basis of minimal pairs, only including words with a token frequency of at least 1, from the built-in example corpus:
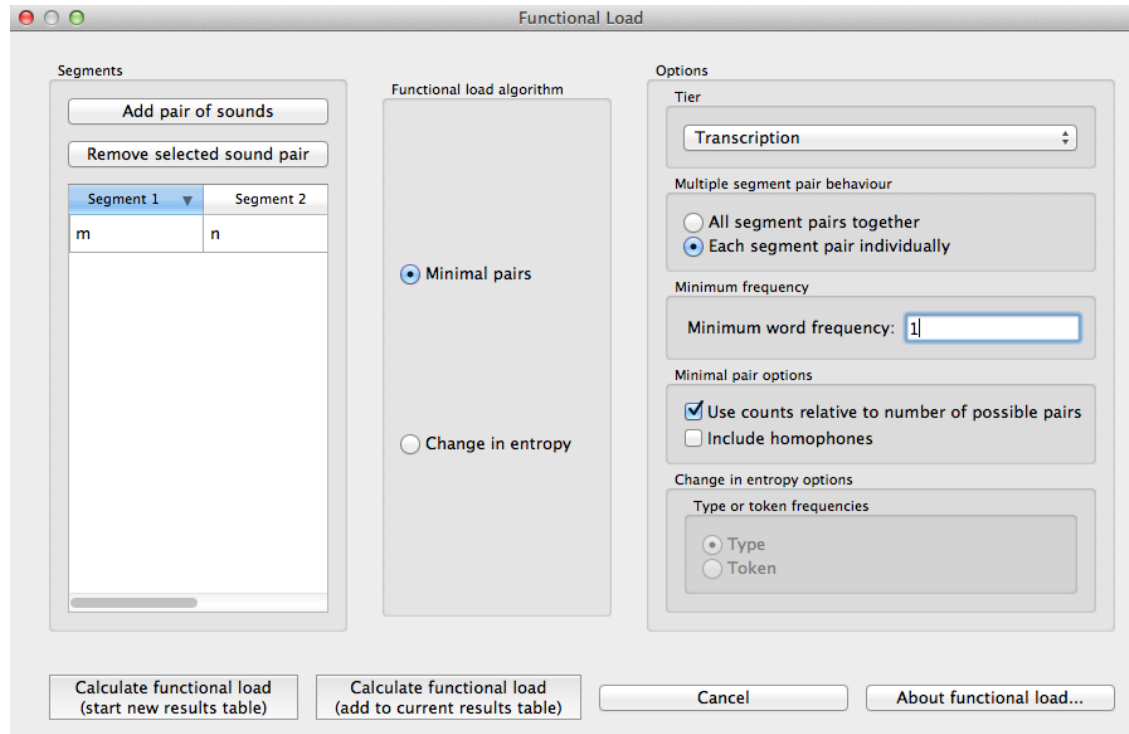
8. Results table: Once all parameters have been set, click one of the two "Calculate functional load" buttons. If this is the first calculation, the option to "start new results table" should be selected. For subsequent calculations, the calculation can be added to the already started table, for direct comparison, or a new table can be started. [Note that if a table is closed, new calculations will not be added to the previously open table; a new table must be started.] Either way, the results table will have the following columns, with one row per calculation: segment 1, segment 2, which tier was used, which measurement method was selected, the resulting functional load, what the minimum frequency was, and for calculations using minimal pairs, whether the count is absolute or relative and whether homophones were ignored or not. (For calculations using change in entropy, "N/A" values are entered into the latter two columns.)

9. Saving results: Once a results table has been generated for at least one pair, the table can be saved by clicking on "Save to file" at the bottom of the table to open a system dialogue box and save the results at a user-designated location.

(Note that in the above screen shot, not all columns are visible; they are visible only by scrolling over to the right, due to constraints on the window size. All columns would be saved to the results file.)

To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

6.4 Implementing the functional load function on the command line

In order to perform this analysis on the command line, you must enter a command in the following format into your Terminal:

`pct_funcload CORPUSFILE ARG2`

...where CORPUSFILE is the name of your *****.corpus file and ARG2 is either the transcription character(s) of a single segment (if calculating relative functional load) or the name of your segment pair(s) file (if calculating a single functional load value). The segment pairs file must list the pairs of segments whose functional load you wish to calculate, with each pair separated by a tab (t) and one pair on each line. You may also use command line options to change various parameters of your functional load calculations. Descriptions of these arguments can be viewed by running `pct_funcload -h` or `pct_funcload --help`. The help text from this command is copied below, augmented with specifications of default values:

Positional arguments:

`corpus_file_name`
> Name of corpus file

`pairs_file_name_or_segment`
> Name of file with segment pairs (or target segment if relative fl is True)

Optional arguments:

`-h`
`--help`
> Show help message and exit

`-a` `ALGORITHM`
`--algorithm` `ALGORITHM`
> Algorithm to use for calculating functional load: "minpair" for minimal pair count or "deltah" for change in entropy. Defaults to minpair.

`-f` `FREQUENCY_CUTOFF`
`--frequency_cutoff` `FREQUENCY_CUTOFF`
> Minimum frequency of words to consider as possible minimal pairs or contributing to lexicon entropy.

`-d` `DISTINGUISH_HOMOPHONES`
`--distinguish_homophones` `DISTINGUISH_HOMOPHONES`
> For minimal pair FL: if False, then you'll count sock~shock (sock=clothing) and sock~shock (sock=punch) as just one minimal pair; but if True, you'll overcount alternative spellings of the same word, e.g. axel~actual and axle~actual. False is the value used by Wedel et al.

`-t` `TYPE_OR_TOKEN`
`--type_or_token` `TYPE_OR_TOKEN`
> For change in entropy FL: specifies whether entropy is based on type or token frequency.

`-e` `RELATIVE_FL`
`--relative_fl` `RELATIVE_FL`
> If True, calculate the relative FL of a single segment by averaging across the functional loads of it and all other segments.

`-s` `SEQUENCE_TYPE`
`--sequence_type` `SEQUENCE_TYPE`
> The attribute of Words to calculate FL over. Normally this will be the transcription, but it can also be the spelling or a user-specified tier.

`-o` `OUTFILE`

--outfile OUTFILE
>   Name of output file

EXAMPLE 1: If your corpus file is example.corpus and you want to calculate the minimal pair functional load of the segments [m] and [n] using defaults for all optional arguments, you first need to create a text file that contains the text "mtn" (where t is a tab; no quotes in the file). Let us call this file pairs.txt. You would then run the following command in your terminal window:

```
pct_funcload example.corpus pairs.txt
```

EXAMPLE 2: Suppose you want to calculate the relative (average) functional load of the segment [m]. Your corpus file is again example.corpus. You want to use the change in entropy measure of functional load rather than the minimal pairs measure, and you also want to use type frequency instead of (the default value of) token frequency. In addition, you want the script to produce an output file called output.txt. You would need to run the following command:

```
pct_funcload example.corpus m -a deltah -t type -o output.txt
```

Predictability of Distribution

7.1 About the function

Predictability of distribution is one of the common methods of determining whether or not two sounds in a language are contrastive or allophonic. The traditional assumption is that two sounds that are predictably distributed (i.e., in complementary distribution) are allophonic, and that any deviation from complete predictability of distribution means that the two sounds are contrastive. [Hall2009], [Hall2012] proposes a way of quantifying predictability of distribution in a gradient fashion, using the information-theoretic quantity of *entropy* (uncertainty), which is also used for calculating functional load (see *Method of calculation*), which can be used to document the *degree* to which sounds are contrastive in a language. This has been shown to be useful in, e.g., documenting sound changes [Hall2013b], understanding the choice of epenthetic vowel in a languages [Hume2013], modeling intra-speaker variability (Thakur 2011), gaining insight into synchronic phonological patterns (Hall & Hall 2013), and understanding the influence of phonological relations on perception ([Hall2009], [Hall2014a]). See also the related measure of Kullback-Leibler divergence (*Kullback-Leibler Divergence*), which is used in [Peperkamp2006] and applied to acquisition; it is also a measure of the degree to which environments overlap, but the method of calculation differs (especially in terms of environment selection).

It should be noted that predictability of distribution and functional load are not the same thing, despite the fact that both give a measure of phonological contrast using entropy. Two sounds could be entirely unpredictably distributed (perfectly contrastive), and still have either a low or high functional load, depending on how often that contrast is actually used in distinguishing lexical items. Indeed, for any degree of predictability of distribution, the functional load may be either high or low, with the exception of the case where both are 0. That is, if two sounds are entirely predictably distributed, and so have an entropy of 0 in terms of distribution, then by definition they cannot be used to distinguish between any words in the language, and so their functional load, measured in terms of change in entropy upon merger, would also be 0.

7.2 Method of calculation

As mentioned above, predictability of distribution is calculated using the same entropy formula as above, repeated here below, but with different inputs.

Entropy:

$$H = - \sum_{i \in N} p_i * log_2(p_i)$$

Because predictability of distribution is determined between exactly two sounds, *i* will have only two values, that is, each of the two sounds. Because of this limitation to two sounds, entropy will range in these situations between 0 and 1. An entropy of 0 means that there is 0 uncertainty about which of the two sounds will occur; i.e., they are perfectly predictably distributed (commonly associated with being allophonic). This will happen when one of the two sounds has a probability of 1 and the other has a probability of 0. On the other hand, an entropy of 1 means that there

is complete uncertainty about which of the two sounds will occur; i.e., they are in perfectly overlapping distribution (what might be termed "perfect" contrast). This will happen when each of the two sounds has a probability of 0.5.

Predictability of distribution can be calculated both within an individual environment and across all environments in the language; these two calculations are discussed in turn.

7.2.1 Predictability of Distribution in a Single Environment

For any particular environment (e.g., word-initially; between vowels; before a [+ATR] vowel with any number of intervening consonants; etc.), one can calculate the probability that each of two sounds can occur. This probability can be calculated using either types or tokens, just as was the case with functional load. Consider the following toy data, which is again repeated from the examples of functional load, though just the original distribution of sounds.

Word	Original Trans.	Type Freq.	Token Freq.
hot	[ht]	1	2
song	[s∎]	1	4
hat	[hæt]	1	1
sing	[s∎]	1	6
tot	[tt]	1	3
dot	[dt]	1	5
hip	[hp]	1	2
hid	[hd]	1	7
team	[tim]	1	5
deem	[dim]	1	5
toot	[tut]	1	9
dude	[dud]	1	2
hiss	[hs]	1	3
his	[hz]	1	5
sizzle	[szl]	1	4
dizzy	[dzi]	1	3
tizzy	[tzi]	1	4
Total		17	70

Consider the distribution of [h] and [∎], word-initially. In this environment, [h] occurs in 6 separate words, with a total token frequency of 20. [∎] occurs in 0 words, with, of course, a token frequency of 0. The probability of [h] occurring in this position as compared to [∎], then, is 6/6 based on types, or 20/20 based on tokens. The entropy of this pair of sounds in this context, then, is:

$$H_{types/tokens} = -[1log_2(1) + 0log_2(0)] = 0$$

Similar results would obtain for [h] and [∎] in word-final position, except of course that it's [∎] and not [h] that can appear in this environment.

For [t] and [d] word-initially, [t] occurs 4 words in this environment, with a total token frequency of 21, and [d] also occurs in 4 words, with a total token frequency of 15. Thus, the probability of [t] in this environment is 4/8, counting types, or 21/36, counting tokens, and the probability of [d] in this environment is 4/8, counting types, or 15/36, counting tokens. The entropy of this pair of sounds is therefore:

$$H_{types} = -[(\tfrac{4}{8}log_2(\tfrac{4}{8})) + (\tfrac{4}{8}log_2(\tfrac{4}{8}))] = 1$$

$$H_{types} = -[(\tfrac{21}{36}log_2(\tfrac{21}{36})) + (\tfrac{15}{36}log_2(\tfrac{15}{36}))] = 0.98$$

In terms of what environment(s) are interesting to examine, that is of course up to individual researchers. As mentioned in the preface to *Predictability of Distribution*, these functions are just tools. It would be just as possible to calculate the entropy of [t] and [d] in word-initial environments before [], separately from word-initial environments before [u]. Or one could calculate the entropy of [t] and [d] that occur anywhere in a word before a bilabial nasal...etc., etc. The choice of environment should be phonologically informed, using all of the resources that have traditionally been used

to identify conditioning environments of interest. See also the caveats in the following section that apply when one is calculating systemic entropy across multiple environments.

7.2.2 Predictability of Distribution across All Environments (Systemic Entropy)

While there are times in which knowing the predictability of distribution within a particular environment is helpful, it is generally the case that phonologists are more interested in the relationship between the two sounds as a whole, across all environments. This is achieved by calculating the weighted average entropy across all environments in which at least one of the two sounds occurs.

As with single environments, of course, the selection of environments for the systemic measure need to be phonologically informed. There are two further caveats that need to be made about environment selection when multiple environments are to be considered, however: (1) **exhaustivity** and (2) **uniqueness**.

With regard to **exhausitivity**: In order to calculate the total predictability of distribution of a pair of sounds in a language, one must be careful to include all possible environments in which at least one of the sounds occurs. That is, the total list of environments needs to encompass all words in the corpus that contain either of the two sounds; otherwise, the measure will obviously be incomplete. For example, one would not want to consider just word-initial and word-medial positions for [h] and [■]; although the answer would in fact be correct (they have 0 entropy across these environments), it would be for the wrong reason—i.e., it ignores what happens in word-final position, where they *could* have had some other distribution.

With regard to **uniqueness**: In order to get an *accurate* calculation of the total predictability of distribution of a pair of sounds, it is important to ensure that the set of environments chosen do not overlap with each other, to ensure that individual tokens of the sounds are not being counted multiple times. For example, one would not want to have both [#__] and [__i] in the environment list for [t]/[d] while calculating systemic entropy, because the words *team* and *deem* would appear in both environments, and the sounds would (in this case) appear to be "more contrastive" (less predictably distributed) than they might otherwise be, because the contrasting nature of these words would be counted twice.

To be sure, one can calculate the entropy in a set of individual environments that are non-exhaustive and/or overlapping, for comparison of the differences in possible generalizations. But, in order to get an accurate measure of the total predictability of distribution, the set of environments must be both exhaustive and non-overlapping. As will be described below, PCT will by default check whether any set of environments you provide does in fact meet these characteristics, and will throw a warning message if it does not.

It is also possible that there are multiple possible ways of developing a set of exhaustive, non-overlapping environments. For example, "word-initial" vs. "non-word-initial" would suffice, but so would "word-initial" vs. "word-medial" vs. "word-final." Again, it is up to individual researchers to determine which set of environments makes the most sense for the particular pheonmenon they are interested in. See [Hall2012] for a comparison of two different sets of possible environments in the description of Canadian Raising.

Once a set of exhaustive and non-overlapping environments has been determined, the entropy in each individual environment is calculated, as described in *Predictability of Distribution in a Single Environment*. The frequency of each environment itself is then calculated by examining how many instances of the two sounds occurred in each environment, as compared to all other environments, and the entropy of each environment is weighted by its frequency. These frequency-weighted entropies are then summed to give the total average entropy of the sounds across the environments. Again, this value will range between 0 (complete predictability; no uncertainty) and 1 (complete unpredictability; maximal uncertainty). This formula is given below; e represents each individual environment in the exhaustive set of non-overlapping environments.

Formula for systemic entropy:

$$H_{total} = -\sum_{e \in E} H(e) * p(e)$$

As an example, consider [t]/[d]. One possible set of exhaustive, non-overlapping environments for this pair of sounds is (1) word-initial and (2) word-final. The relevant words for each environment are shown in the table below, along with the calculation of systemic entropy from these environments.

The calculations for the entropy of word-initial environments were given above; the calculations for word-final environments are analogous.

To calculate the probability of the environments, we simply count up the number of total words (either types or tokens) that occur in each environment, and divide by the total number of words (types or tokens) that occur in all environments.

Calculation of systemic entropy of [t] and [d]:

e	words						Types	Types
words								
	tot, team, toot, tizzy	dot, dude, deem, dizzy	1	(4+4)/ (8+7) =8/15	0.533	0.98	(21+15)/ (36+29) =36/65	0.543
[__#]	hot, hat, tot, dot, toot	hid, dude	0.863	7/15	0.403	0.894	29/65	0.399
				0.533+0.403=0.936			0.543+0.399=0.942	

In this case, [t]/[d] are relatively highly unpredictably distributed (contrastive) in both environments, and both environments contributed approximately equally to the overall measure. Compare this to the example of [s]/[z], shown below.

Calculation of systemic entropy of [s] and [z]:

e	words						Types	Types
words								
	song, sing, sizzle		0	3/8	0	0	14/33	0
[__#]	hiss	his	1	2/8	0.25	0.954	8/33	0.231
[V_V]	sizzle, dizzy, tizzy		0	3/8	0	0	11/33	0
				0.25			0.231	

In this case, there is what would traditionally be called a contrast word finally, with the minimal pair *hiss* vs. *his*; this contrast is neutralized (made predictable) in both word-initial position, where [s] occurs but [z] does not, and intervocalic position, where [z] occurs but [s] does not. The three environments are roughly equally probable, though the environment of contrast is somewhat less frequent than the environments of neutralization. The overall entropy of the pair of sounds is on around 0.25, clearly much closer to perfect predictability (0 entropy) than [t]/[d].

Note, of course, that this is an entirely fictitious example—that is, although these are real English words, one would **not** want to infer anything about the actual relationship between either [t]/[d] or [s]/[z] on the basis of such a small corpus. These examples are simplified for the sake of illustrating the mathematical formulas!

7.2.3 "Predictability of Distribution" Across All Environments (i.e., Frequency-Only Entropy)

Given that the calculation of predictability of distribution is based on probabilities of occurrence across different environments, it is also possible to calculate the overall entropy of two segments using their raw probabilities and ignoring specific environments. Note that this doesn't really reveal anything about predictability of distribution per se; it simply gives the uncertainty of occurrence of two segments that is related to their relative frequencies. This is calculated by simply taking the number of occurrences of each of sound 1 (N1) and sound 2 (N2) in the corpus as a whole, and then applying the following formula:

Formula for frequency-only entropy:

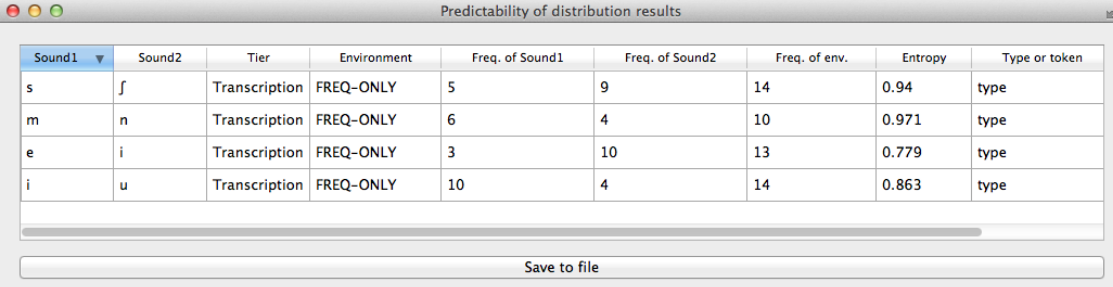$H = (-1) * [(\frac{N1}{N1+N2})log_2(\frac{N1}{N1+N2}) + (\frac{N2}{N1+N2})log_2(\frac{N2}{N1+N2})]$

The entropy will be 0 if one or both of the sounds never occur(s) in the corpus. The entropy will be 1 if the two sounds occur with exactly the same frequency. It will be a number between 0 and 1 if both sounds occur, but not with the same frequency.

Note that an entropy of 1 in this case, which was analogous to perfect contrast in the environment-specific implementation of this function, does *not* align with contrast here. For example, [h] and [■] in English, which are in complementary distribution, could theoretically have an entropy of 1 if environments are ignored and they happened to occur with exactly the same frequency in some corpus. Similarly, two sounds that do in fact occur in the same environments might have a low entropy, close to 0, if one of the sounds is vastly more frequent than the other. That is, this calculation is based ONLY on the frequency of occurrence, and not actually on the distribution of the sounds in the corpus. This function is thus useful only for getting a sense of the frequency balance / imbalance between two sounds. Note that one can also get total frequency counts for any segment in the corpus through the "Summary" information feature (*Summary information about a corpus*).

7.3 Implementing the predictability of distribution function in the GUI

Assuming a corpus has been opened or created, predictability of distribution is calculated using the following steps.

1. **Getting started**: Choose "Analysis" / "Calculate predictability of distribution..." from the top menu bar.

2. **Sound selection**: On the left-hand side of the "Predictability of distribution" dialogue box, select the two sounds of interest by clicking "Add pair of sounds. The order of the sounds is irrelevant; picking [i] first and [u] second will yield the same results as [u] first and [i] second. Currently, PCT only allows entire segments to be selected; the next release will allow a "sound" to be defined as a collection of feature values. The segment choices that are available will automatically correspond to all of the unique transcribed characters in your corpus. You can select more than one pair of sounds to examine in the same environments; each pair of sounds will be treated individually.

3. **Environments**: Click on "Add environment" to add an environment in which to calculate predictability of distribution. The left side of the "Create environment" dialogue box allows left-hand environments to be specified (e.g., [+back]___), while the right side allows right-hand environments to be specified (e.g., __#). Both can be used simultaneously to specify environments on both sides (e.g., [+back]__#).

   (a) **Basis for building environments (segments vs. features)**: Environments can be selected either as entire segments (including #) or as bundles of features. Select from the drop-down menu which you prefer. Each side of an environment can be specified using either type.

   (b) **Segment selection**: To specify an environment using segments, simply click on the segment desired.

   (c) **Feature selection**: To specify an environment using features, select the first feature from the list (e.g., [voice]), and then specify whether you want it to be [+voice] or [-voice] by selecting "Add [+feature]" or "Add [-feature]" as relevant. To add another feature to this same environment, select another feature and again add either the + or – value.

   (d) **No environments**: Note that if NO environments are added, PCT will calculate the overall predictability of distribution of the two sounds based only on their frequency of occurrence. This will simply count the frequency of each sound in the pair and calculate the entropy based on those frequencies (either type or token). See below for an example of calculating environment-free entropy for four different pairs in the sample corpus:

Sound1 ▼	Sound2	Tier	Environment	Freq. of Sound1	Freq. of Sound2	Freq. of env.	Entropy	Type or token
s	ʃ	Transcription	FREQ–ONLY	5	9	14	0.94	type
m	n	Transcription	FREQ–ONLY	6	4	10	0.971	type
e	i	Transcription	FREQ–ONLY	3	10	13	0.779	type
i	u	Transcription	FREQ–ONLY	10	4	14	0.863	type

Save to file
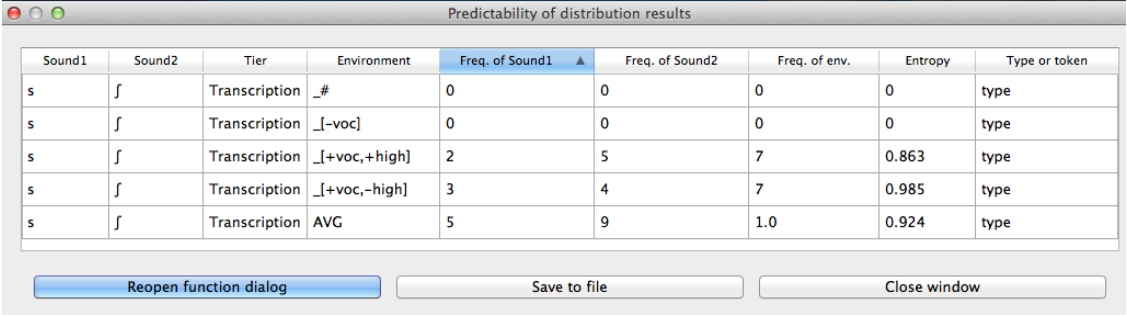
4. **Environment list**: Once all features / segments for a given environment have been selected, for both the left- and right-hand sides, click on "Add"; it will appear back in the "Predictability of Distribution" dialogue box in the environment list. To automatically return to the environment selection window to select another environment, click on "Add and select another" instead. Individual environments from the list can be selected and removed if it is determined that an environment needs to be changed. It is this list that PCT will verify as being both exhaustive and unique; i.e., the default is that the environments on this list will exhaustively cover all instances in your corpus of the selected sounds, but will do so in such a way that each instance is counted exactly once.

5. **Analysis tier**: Under "Options," first pick the tier on which you want predictability of distribution to be calculated. The default is for the entire transcription to be used, such that environments are defined on any surrounding segments. If a separate tier has been created as part of the corpus (see *Creating new tiers in the corpus*), however, predictability of distribution can be calculated on this tier. For example, one could extract a separate tier that contains only vowels, and then calculate predictability of distribution based on this tier. This makes it much easier to define non-adjacent contexts. For instance, if one wanted to investigate the extent to which [i] and [u] are predictably distributed before front vs. back vowels, it will be much easier to to specify that the relevant environments are __[+back] and __[-back] on the vowel tier than to try to account for possible intervening segments on the entire transcription tier.

6. **Type vs. Token Frequency**: Next, pick whether you want the calculation to be done on types or tokens, assuming that token frequencies are available in your corpus. If they are not, this option will not be available. (Note: if you think your corpus does include token frequencies, but this option seems to be unavailable, see *Required format of corpus* on the required format for a corpus.)

7. **Exhaustivity & Uniqueness**: The default is for PCT to check for both exhaustivity and uniqueness of environments, as described above in *Predictability of Distribution across All Environments (Systemic Entropy)*. Un-checking this box will turn off this mechanism. For example, if you wanted to compare a series of different possible environments, to see how the entropy calculations differ under different generalizations, uniqueness might not be a concern. Keep in mind that if uniqueness and exhaustivity are not met, however, the calculation of systemic entropy will be inaccurate.

   (a) If you ask PCT to check for exhaustivity, and it is not met, an error message will appear that warns you that the environments you have selected do not exhaustively cover all instances of the symbols in the corpus, as in the following; the "Show details..." option has been clicked to reveal the specific words that occur in the corpus that are not currently covered by your list of environments. Furthermore, a .txt file is automatically created that lists all of the words, so that the environments can be easily adjusted. This file is stored in the ERRORS folder within the working directory that contains the PCT software, and can be accessed directly by clicking "Open errors directory." If exhaustivity is not important, and only the entropy in individual environments matters, then it is safe to not enforce exhaustivity; it should be noted that the weighted average entropy across environments will NOT be accurate in this scenario, because not all words have been included.
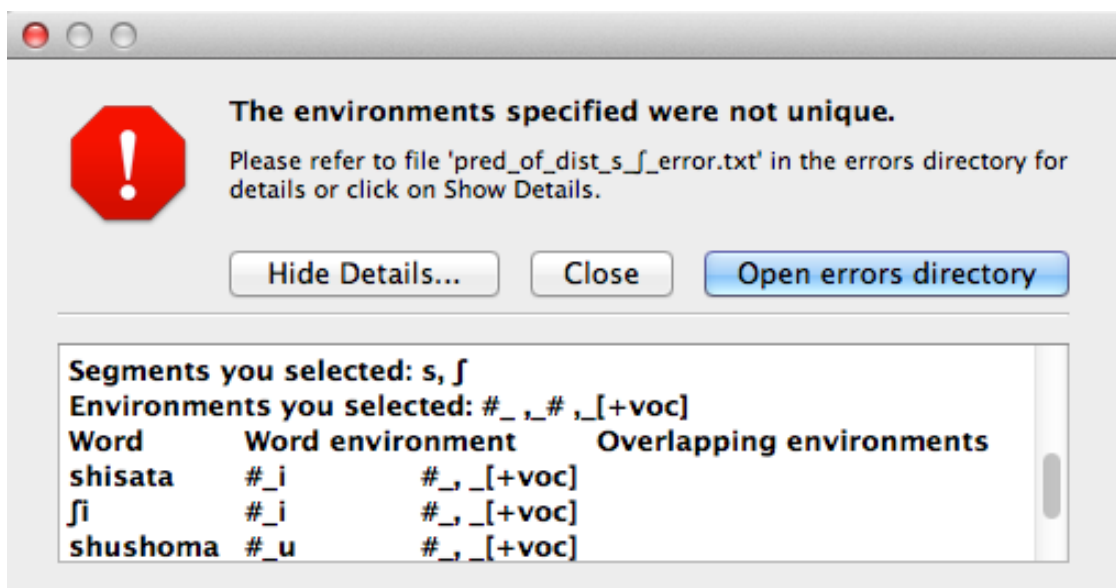
Here's an example of correctly exhaustive and unique selections for calculating the predictability of distribution based on token frequency for [s] and [] in the sample corpus:
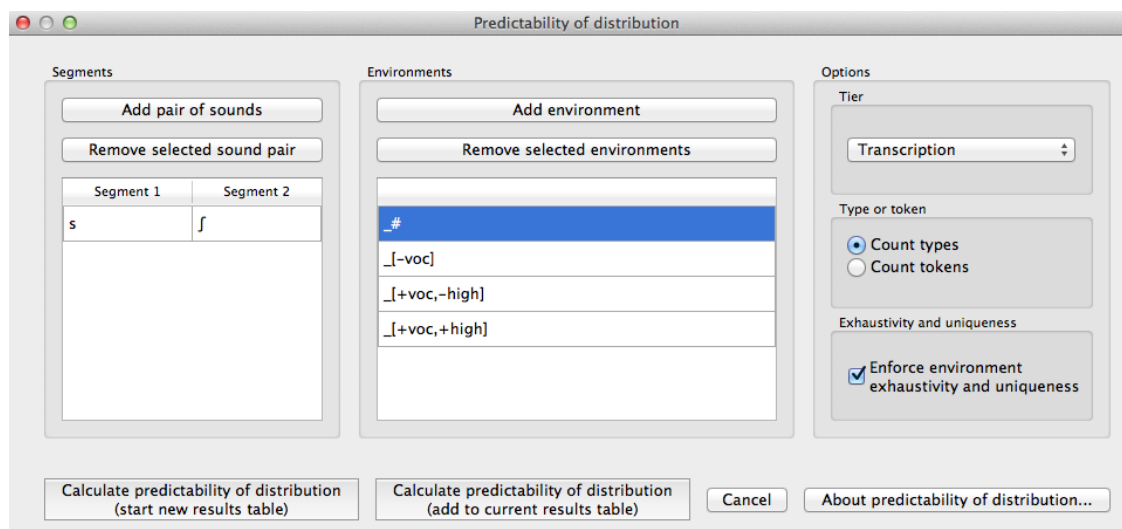
8. **Entropy calculation / results**: Once all environments have been specified, click "Calculate predictability of distribution." If you want to start a new results table, click that button; if you've already done at least one calculation and want to add new calculations to the same table, select the button with "add to current results table." Results will appear in a pop-up window on screen. The last row for each pair gives the weighted average entropy across all selected environments, with the environments being weighted by their own frequency of occurrence. See the following example:

Sound1	Sound2	Tier	Environment	Freq. of Sound1 ▲	Freq. of Sound2	Freq. of env.	Entropy	Type or token
s	ʃ	Transcription	_#	0	0	0	0	type
s	ʃ	Transcription	_[–voc]	0	0	0	0	type
s	ʃ	Transcription	_[+voc,+high]	2	5	7	0.863	type
s	ʃ	Transcription	_[+voc,–high]	3	4	7	0.985	type
s	ʃ	Transcription	AVG	5	9	1.0	0.924	type

Reopen function dialog   Save to file   Close window

9. **Output file / Saving results**: If you want to save the table of results, click on "Save to file" at the bottom of the table. This opens up a system dialogue box where the directory and name can be selected.

To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

Kullback-Leibler Divergence

8.1 About the function

Another way of measuring the distribution of environments as a proxy for phonological relationships is the Kullback-Leibler (KL) measure of the dissimilarity between probability distributions [Kullback1951]. Sounds that are distinct phonemes appear in the same environments, that is, there are minimal or near-minimal, pairs. Allophones, on the other hand, have complementary distribution, and never appear in the same environment. Distributions that are identical have a KL score of 0, and the more dissimilar two distributions, the higher the KL score. Applied to phonology, the idea is to calculate the probability of two sounds across all environments in a corpus, and use KL to measure their dissimilarity. Scores close to 0 suggest that the two sounds are distinct phonemes, since they occur in many of the same environments (or else there is extensive free variation). Higher scores represent higher probabilities that the two sounds are actually allophones. Since KL scores have no upper bound, it is up to the user to decide what counts as "high enough" for two sounds to be allophones (this is unlike the predictability of distribution measure described in *Predictability of Distribution*). See [Peperkamp2006] for a discussion of how to use Z-Scores to make this discrimination.

As with the predictability of distribution measure in *Predictability of Distribution*, spurious allophony is also possible, since many sounds happen to have non-overlapping distributions. As a simple example, vowels and consonants generally have high KL scores, because they occur in such different environments. Individual languages might have cases of accidental complementary distribution too. For example, in English /h/ occurs only initially and [■] only occurs finally. However, it is not usual to analyze them as being in allophones of a single underlying phonemes. Instead, there is a sense that allophones need to be phonetically similar to some degree, and /h/ and /■/ are simply too dissimilar.

To deal with this problem, [Peperkamp2006] suggest two "linguistic filters" that can be applied, which can help identify cases of spurious allophones, such as /h/ and /■/. Their filters do not straightforwardly apply to CorpusTools, since they use 5-dimensional vectors to represent sounds, while in CorpusTools most sounds have only binary features. An alternative filter is used instead, and it is described below.

It is important to note that this function's usefulness depends on the level of analysis in your transcriptions. In many cases, corpora are transcribed at a phonemic level of detail, and KL will not be very informative. For instance, the IPHOD corpus does not distinguish between aspirated and unaspirated voiceless stops, so you cannot measure their KL score.

8.2 Method of calculation

All calculations were adopted from [Peperkamp2006]. The variables involves are as follows: s is a segment, *c* is a context, and *C* is the set of all contexts. The Kullback-Leibler measure of dissimilarity between the distributions of two segments is the sum for all contexts of the entropy of the contexts given the segments:

KL Divergence:

$m_{KL}(s_1, s_2) = \sum_{c \in C} P(c|s_1) log(\frac{P(c|s_1)}{P(c|s_2)}) + P(c|s_2) log(\frac{P(c|s_2)}{P(c|s_1)})$

The notation *P(c|s) means the probability of context c given segment s, and it is calculated as follows:

$P(c|s) = \frac{n(c,s)+1}{n(s)+N}$

...where *n(c,s)* is the number of occurrences of segments *s* in context *c*. [Peperkamp2006] note that this equal to the number of occurrences of the sequence *sc*, which suggests that they are only looking at the right hand environment. This is probably because in their test corpora, they were looking at allophones conditioned by the following segment. PCT provides the option to look only at the left-hand environment, only at the right-hand environment, or at both.

[Peperkamp2006] then compare the average entropy values of each segment, in the pair. The segment with the higher entropy is considered to be a surface representation (SR), i.e. an allophone, while the other is the underlying representation (UR). In a results window in PCT, this is given as "Possible UR." More formally:

$SR = \max_{SR,UR}[\sum_c P(c|s) log \frac{P(c|s)}{P(c)}]$

[Peperkamp2006] give two linguistic filters for getting rid of spurious allophones, which rely on sounds be coded as 5-dimensional vectors. In PCT, this concept as been adopted to deal with binary features. The aim of the filter is the same, however. In a results window the column labeled "spurious allophones" gives the result of applying this filter.

The features of the supposed UR and SR are compared. If they differ by only one feature, they are considered plausibly close enough to be allophones, assuming the KL score is high enough for this to be reasonable (which will depend on the corpus and the user's expectations). In this case, the "spurious allophones?" results will say 'No.'

If they differ by more than 1 feature, PCT checks to see if there any other sounds in the corpus that are closer to the SR than the UR is. For instance, if /p/ and /s/ are compared in the IPHOD corpus, /p/ is considered the UR and /s/ is the SR. The two sounds differ by two features, namely [continuant] and [coronal]. There also exists another sound, /t/, which differs from /s/ by [continuant], but not by [coronal] (or any other feature). In other words, /t/ is more similar to /s/ than /p/ is to /s/. If such an "in-between" sound can be found, then in the "spurious allophones?" column, the results will say 'Yes.'

If the two sounds differ by more than 1 feature, but no in-between sound can be found, then the "spurious allophones?" results will say 'Maybe.'

Note too that a more direct comparison of the acoustic similarity of sounds can also be conducted using the functions in *Acoustic Similarity*.

8.3 Implementing the Kullback-Leibler Divergence function in the GUI

To implement the KL function in the GUI, select "Analysis" / "Calculate Kullback-Leibler..." and then follow these steps:

1. Pair of sounds: Click on "Add pair of sounds" to open the "Select segment pair" dialogue box. The segment choices that are available will automatically correspond to all of the unique transcribed characters in your corpus; click on "Consonants" and/or "Vowels" to see the options. You can select more than one pair of sounds to examine in the same environments; each pair of sounds will be treated individually. Selecting more than two sounds at a time will run the analysis on all possible pairs of those sounds (e.g., selecting [t], [s], and [d] will calculate the KL score for [t]~[s], [s]~[d], and [t]~[d]).

2. Contexts: Using KL requires a notion of "context," and there are three options: left, right, or both. Consider the example word [atema]. If using the "both" option, then this word consists of these environments: [#_t], [a_e], [t_m], [e_a], and [m_#]. If the left-side option is chosen, then only the left-hand side is used, i.e., the word consists of the environments [#_], [a_], [t_], [e_], and [m_]. If the right-side option is chosen, then the environments in the word are [_t], [_e], [_m], [_a], and [_#]. Note that the word boundaries don't count as elements of words, but can count as parts of environments.

3. Results: Once all selections have been made, click "Calculate Kullback-Leibler." If you want to start a new results table, click that button; if you've already done at least one calculation and want to add new calculations to the same table, select the button with "add to current results table." Results will appear in a pop-up window on screen. Each member of the pair is listed, along with which context was selected, the entropy of each segment, the KL score, which of the two members of the pair is more likely to be the UR (as described above), and PCT's judgment as to whether this is a possible case of spurious allophones based on the featural distance.

4. Output file / Saving results: If you want to save the table of results, click on "Save to file" at the bottom of the table. This opens up a system dialogue box where the directory and name can be selected.

To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

An example of calculating the KL scores in the Example corpus, with the sounds [s], [], [t], [n], [m], [e], [u] selected (and therefore all pairwise comparisons thereof calculated), examining only right-hand side contexts:

The "Select segment pair" dialogue box, within the "Kullback-Leibler" dialogue box:

The "Kullback-Leibler" dialogue box, with pairs of sounds and contexts selected:

The resulting table of results:

Segment 1	Segment 2	Context	Segment 1 entropy	Segment 2 entropy	KL	Possible UR	Spurious allophones?
e	m	right	0.093	0.232	0.703	e	Yes
e	n	right	0.093	0.094	0.442	e	Yes
e	s	right	0.093	0.178	0.598	e	Yes
e	t	right	0.093	0.265	0.845	e	Yes
e	u	right	0.093	0.06	0.387	u	Yes
e	ʃ	right	0.093	0.148	0.635	e	Yes
m	n	right	0.232	0.094	0.199	n	No
m	s	right	0.232	0.178	0.01	s	Yes
m	t	right	0.232	0.265	0.172	m	Yes
m	u	right	0.232	0.06	0.693	u	Yes
m	ʃ	right	0.232	0.148	0.414	ʃ	No
n	s	right	0.094	0.178	0.13	n	Yes
n	t	right	0.094	0.265	0.369	n	Yes
n	u	right	0.094	0.06	0.424	u	Yes
n	ʃ	right	0.094	0.148	0.121	n	Yes
s	t	right	0.178	0.265	0.198	s	Yes
s	u	right	0.178	0.06	0.583	u	Yes
s	ʃ	right	0.178	0.148	0.326	ʃ	No
t	u	right	0.265	0.06	0.86	u	Yes
t	ʃ	right	0.265	0.148	0.373	ʃ	Yes

String similarity and neighbourhood density

9.1 About the functions

String similarity is any measure of how similar any two sequences of characters are. These character strings can be strings of letters or phonemes; both of the methods of calculation included in PCT allow for calculations using either type of character. It is, therefore, a basic measure of overall form-based similarity.

String similarity finds more widespread use in areas of linguistics other than phonology; it is, for example, used in Natural Language Processing applications to determine, for example, possible alternative spellings when a word has been mistyped. It is, however, also useful for determining how phonologically close any two words might be.

String similarity could be part of a calculation of morphological relatedness, if used in conjunction with a measure of semantic similarity (see, e.g., [Hall2014b]). In particular, it can be used in conjunction with the Frequency of Alternation function of PCT (see *Frequency of alternation*) as a means of calculating the frequency with which two sounds alternate with each other in a language.

Some measure of string similarity is also used to calculate neighbourhood density (e.g. [Greenberg1964]; [Luce1998]; [Yao2011]), which has been shown to affect phonological processing. A phonological "neighbour" of some word X is a word that is similar in some close way to X. For example, it might differ by maximally one phone (through deletion, addition, or subsitution) from X. X's neighbourhood density, then, is the number of words that fit the criterion for being a neighbour.

9.2 Method of calculation: String similarity

9.2.1 Levenshtein Edit Distance

Edit distance is defined as the minimum number of one-symbol deletions, additions, and substitutions necessary to turn one string into another. For example, *turn* and *burn* would have an edit distance of 1, as the only change necessary is to turn the <t> into a <b>, while the edit distance between *turn* and *surfs* would be 3, with <t> becoming <s>, <n> becoming <f>, and becoming <s> at the end of the word. All such one-symbol changes are treated as equal in Levenshtein edit distance, unlike phonological edit distance, described in the following section. Generally speaking, the neighbourhood density of a particular lexical item is measured by summing the number of lexical items that have an edit distance of 1 from that item [Luce1998].

9.2.2 Phonological Edit Distance

Phonological edit distance is quite similar to Levenshtein edit distance, in that it calculates the number of one-symbol changes between strings, but it differs in that changes are weighted based on featural similarity. For example, depend-

ing on the feature system used, changing <t> to <s> might involve a single feature change (from [-cont] to [+cont]), while changing <t> to <b> might involve two (from [-voice, +cor] to [+voice, -cor]). By default, the formula for calculating the phonological distance between two segments—or between a segment and "silence", i.e. insertion or deletion—is the one used in the Sublexical Learner [Allen2014]. When comparing two segments, the distance between them is equal to the sum of the distances between each of their feature values: the distance between two feature values that are identical is 0, while the distance between two opposing values (+/- or -/+) is 1, and the distance between two feature values in the case that just one of them is 0 (unspecified) is set to by default to 0.25. When comparing a segment to "silence" (insertion/deletion), the silence is given feature values of 0 for all features and then compared to the segment as normal.

9.2.3 Khorsi (2012) Similarity Metric

Khorsi (2012) proposes a particular measure of string similarity based on orthography, which he suggests can be used as a direct measure of morphological relatedness. PCT allows one to calculate this measure, which could be used, as Khorsi describes, on its own, or could be used in conjunction with other measures (e.g., semantic similarity) to create a more nuanced view.

This measure starts with the sum of the log of the inverse of the frequency of occurrence of each of the letters in the longest common shared sequence between two words, and then subtracts the sum of the log of the inverse of the frequency of the letters that are not shared, as shown below.

Formula for string similarity from [Khorsi2012]:

$$\sum_{i=1}^{\|LCS(w_1,w_2)\|} log(\frac{1}{freq(LCS(w_1,w_2)[i])}) - \sum_{i=1}^{\|\overline{LCS(w_1,w_2)}\|} log(\frac{1}{freq(\overline{LCS(w_1,w_2)}[i])})$$

Note: * *w1, w2* are two words whose string similarity is to be measured * *LCS(w1, w2)* represents the Longest Common Shared Sequence of symbols

between the two words

As with other functions, the frequency measure used for each character will be taken from the current corpus. This means that the score will be different for a given pair of words (e.g., *pressed* vs. *pressure*) depending on the frequency of the individual characters in the loaded corpus.

9.3 Implementing the string similarity function in the GUI

To start the analysis, click on "Analysis" / "Calculate string similarity..." in the main menu, and then follow these steps:

1. **String similarity algorithm**: The first step is to choose which of the three methods described above is to be used to calculate string similarity. The options are phonological edit distance, standard (Levenshtein) edit distance, and the algorithm described above and in [Khorsi2012].

2. **Comparison type**: Next, choose what kind of comparison is to be done. One can either take a single word and get its string similarity score to every other word in the corpus (useful, for example, when trying to figure out which words are most / least similar to a given word, as one might for stimuli creation), or can compare individual pairs of words (useful if a limited set of pre-determined words is of interest). For each of these, you can use words that already exist in the corpus or calculate the similarity for words (or non-words) that are not in the corpus. Note that these words will NOT be added to the corpus itself; if you want to globally add the word (and therefore have its own properties affect calculations), please use the instructions in *Adding a word*.

   (a) **One word in the corpus**: To compare the similarity of one word that already exists in the corpus to every other word in the corpus, simply select "Compare one word to entire corpus" and enter the single word into the dialogue box, using its standard orthographic representation. Note that you can choose later which tier string similarity will be calculated on (spelling, transcription, etc.); this simply identifies the word for PCT.

(b) **One word not in the corpus**: Click on "Calculate for a word/nonword not in the corpus" and then select "Create word/nonword" to enter the new word.

    i. **Spelling**: Enter the spelling for your new word / nonword using the regular input keyboard on your computer.

    ii. **Transcription**: To add in the phonetic transcription of the new word, it is best to use the provided inventory. While it is possible to type directly in to the transcription box, using the provided inventory will ensure that all characters are understood by PCT to correspond to existing characters in the corpus (with their concomitant featural interpretation). Click on "Show inventory" and then choose to show "Consonants," "Vowels," and/or other. (If there is no featural interpretation of your inventory, you will simply see a list of all the available segments, but they will not be classified by major category.) Clicking on the individual segments will add them to the transcription. The selections will remain even when the sub-inventories are hidden; we allow for showing / hiding the inventories to ensure that all relevant buttons on the dialogue box are available, even on small computer screens. Note that you do NOT need to include word boundaries at the beginning and end of the word, even when the boundary symbol is included as a member of the inventory; these will be assumed automatically by PCT.

    iii. **Frequency and other columns**: These can be left at the default. Note that entering values will NOT affect the calculation; there is no particular need to enter anything here (it is an artifact of using the same dialogue box here as in the "Add Word" function described in *Adding a word*).

    iv. **Create word**: To finish and return to the "String similarity" dialogue box, click on "Create word."

(c) **Single word pair (in or not in) the corpus**: If the similarity of an individual word pair is to be calculated, one can enter the pair directly into the dialogue box. For each word that **is** in the corpus, simply enter its standard orthographic form. For each word that is **not** in the corpus, you can add it by selecting "Create word/nonword" and following the steps described immediately above in (2b).

(d) **List of word pairs (in the corpus)**: If there is a long list of pairs of words, one can simply create a tab-delimited plain .txt file with one *word pair* per line. In this case, click on "Choose word pairs file" and select the .txt file in the resulting system dialogue box. Note that this option is currently available only for words that already exist in the corpus, and that these pairs should be listed using their standard orthographic representations.

2. **Tier**: The tier from which string similarity is to be calculated can be selected. Generally, one is likely to care most about either spelling or transcription, but other tiers (e.g., a vowel tier) can also be selected; in this case, all information removed from the tier is ignored. Words should always be entered orthographically (e.g., when telling PCT what word pairs to compare). If similarity is to be calculated on the basis of spelling, words that are *entered* are broken into their letter components. If similarity is to be calculated on the basis of transcription, the transcriptions are looked up in the corpus. If a word does not occur in the corpus, its similarity to other words can still be calculated on the basis of spelling, but not transcription (as PCT has no way of inferring the transcription from the spelling).

3. **Frequency type**: If Khorsi similarity is to be calculated, the frequencies of the symbols is relevant, and so will be looked up in the currently loaded corpus. Either type frequency or token frequency can be used for the calculation. This option will not be available for either edit distance algorithm, because frequency isn't taken into account in either one.

4. **Minimum / Maximum similarity**: If one is calculating the similarity of one word to all others in the corpus, an arbitrary minimum and maximum can be set to filter out words that are particularly close or distant. For example, one could require that only words with an edit distance of both at least and at most 1 are returned, to get the members of the standard neighbourhood of a particular lexical item. (Recall that the Khorsi calculation is a measure of similarity, while edit distance and phonological edit distance are measures of difference. Thus, a minimum similarity value is analogous to a maximum distance value. PCT will automatically interpret "minimum" and "maximum" relative to the string-similarity algorithm chosen.

Here's an example for calculating the Khorsi similarity of the pair *mata* (which occurs in the corpus) and *mitoo* [mitu] (which does not), in the sample corpus, using token frequencies and comparing transcriptions:

5. Results: Once all options have been selected, click "Calculate string similarity." If this is not the first calculation, and you want to add the results to a pre-existing results table, select the choice that says "add to current results table." Otherwise, select "start new results table." A dialogue box will open, showing a table of the results, including word 1, word 2, the result (i.e., the similarity score for Khorsi or distance score for either of the edit algorithms), whether type or token frequency was used (if the Khorsi method is selected; otherwise, N/A), and which algorithm was used. Note that the entries in the table will be written in spelling regardless of whether spelling or transcriptions were used. This file can be saved to a desired location by selecting "Save to file" at the bottom of the table.

Here's an example result file for the above selection:

To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

String similarity and neighbourhood density

10.1 About the functions

Some measures of *string similarity* are used to calculate neighbourhood density (e.g. [Greenberg1964]; [Luce1998]; [Yao2011]), which has been shown to affect phonological processing. A phonological "neighbor" of some word X is a word that is similar in some close way to X. For example, it might differ by maximally one phone (through deletion, addition, or subsition) from X. X's neighborhood density, then, is the number of words that fit the criterion for being a neighbour.

10.2 Method of calculation: Neighbourhood density

A word's neighborhood density is equal to the number of other words in the corpus similar to that word (or, if using token frequencies, the sum of those words' counts). The threshold that defines whether two words are considered similar to each other can be calculated using any of the three distance metrics described in *Method of calculation: String similarity*: Levenshtein edit distance, phonological edit distance, or Khorsi (2012) similarity. As implemented in PCT, for a query word, each other word in the corpus is checked for its similarity to the query word and then added to a list of neighbors if sufficiently similar.

For further detail about the available distance/similarity metrics, refer to *Method of calculation: String similarity*.

10.3 Implementing the neighbourhood density function in the GUI

To start the analysis, click on "Analysis" / "Calculate neighbourhood density..." in the main menu, and then follow these steps:

1. **String similarity algorithm**: The first step is to choose which of the three methods of string similarity is to be used to calculate neighbourhood density. Note that the standard way of calculating density is using standard Levenstein edit distance. We include the other two algorithms here as options primarily for the purpose of allowing users to explore whether they might be useful measures; we make no claims that either phonological edit distance or the Khorsi algorithm might be better than edit distance for any reason.

   (a) **Minimal pair counts / Substitution neighbours**: It is also possible to calculate neighbourhood density by using a variation of edit distance that allows for "substitutions only" (not deletions or insertions). This is particularly useful if, for example, you wish to know the number of or identity of all minimal pairs for a given word in the corpus, as minimal pairs are generally assumed to be substitution neighbours with an edit distance of 1. (Note that the substitution neighbours algorithm automatically assumes a threshold of 1; multiple substitutions are not allowed.)

2. **Query type**: Neighbourhood density can be calculated for one of four types of inputs:

   (a) **One word in the corpus**: The neighbourhood density of a single word can be calculated by entering that word's orthographic representation in the query box.

   (b) **One word not in the corpus**: (Note that this will NOT add the word itself to the corpus, and will not affect any subsequent calculations. To globally add a word to the corpus itself, please see the instructions in *Adding a word*.) Select "Calculate for a word/nonword in the corpus," then choose "Create word/nonword" to enter the new word and do the following:

       i. **Spelling**: Enter the spelling for your new word / nonword using the regular input keyboard on your computer.

       ii. **Transcription**: To add in the phonetic transcription of the new word, it is best to use the provided inventory. While it is possible to type directly in to the transcription box, using the provided inventory will ensure that all characters are understood by PCT to correspond to existing characters in the corpus (with their concomitant featural interpretation). Click on "Show inventory" and then choose to show "Consonants," "Vowels," and/or other. (If there is no featural interpretation of your inventory, you will simply see a list of all the available segments, but they will not be classifed by major category.) Clicking on the individual segments will add them to the transcription. The selections will remain even when the sub-inventories are hidden; we allow for showing / hiding the inventories to ensure that all relevant buttons on the dialogue box are available, even on small computer screens. Note that you do NOT need to include word boundaries at the beginning and end of the word, even when the boundary symbol is included as a member of the inventory; these will be assumed automatically by PCT.

       iii. **Frequency and other columns**: These can be left at the default. Note that entering values will NOT affect the calculation; there is no particular need to enter anything here (it is an artifact of using the same dialogue box here as in the "Add Word" function described in *Adding a word*).

       iv. **Create word**: To finish and return to the "String similarity" dialogue box, click on "Create word."

   (c) **List of words**: If there is a specific list of words for which density is to be calculated (e.g., the stimuli list for an experiment), that list can be saved as a .txt file with one word per line and uploaded into PCT for analysis. Note that in this case, if the words **are** in the corpus, either transcription- or spelling-based neighbourhood density can be calculated; either way, the words on the list should be written in standard orthography (their transcriptions will be looked up in the corpus if needed). If the words are **not** in the corpus, then only spelling-based neighbourhood density can currently be calculated; again, the words should be written in orthographically.

   (d) **Whole corpus**: Alternatively, the neighbourhood density for every word in the corpus can be calculated. This is useful, for example, if one wishes to find words that match a particular neighbourhood density. The density of each word will be added to the corpus itself, as a separate column; in the "query" box, simply enter the name of that column (the default is "Neighborhood Density").

3. **Tier**: Neighbourhood density can be calculated from most of the available tiers in a corpus (e.g., spelling, transcription, or tiers that represent subsets of entries, such as a vowel or consonant tier). (If neighbourhood density is being calculated with phonological edit distance as the similarity metric, spelling cannot be used.) Standard neighbourhood density is calculated using edit distance on transcriptions.

4. **Type vs. token frequency**: If the Khorsi algorithm is selected as the string similarity metric, similarity can be calculated using either type or token frequency, as described in *Khorsi (2012) Similarity Metric*.

5. **Distance / Similarity Threshold**: A specific threshold must be set to determine what counts as a "neighbour." If either of the edit distance metrics is selected, this should be the maximal distance that is allowed – in standard calculations of neighbourhood density, this would be 1, signifying a maximum 1-phone change from the starting word. If the Khorsi algorithm is selected, this should be the minimum similarity score that is required. Because this is not the standard way of calculating neighbourhood density, we have no recommendations for
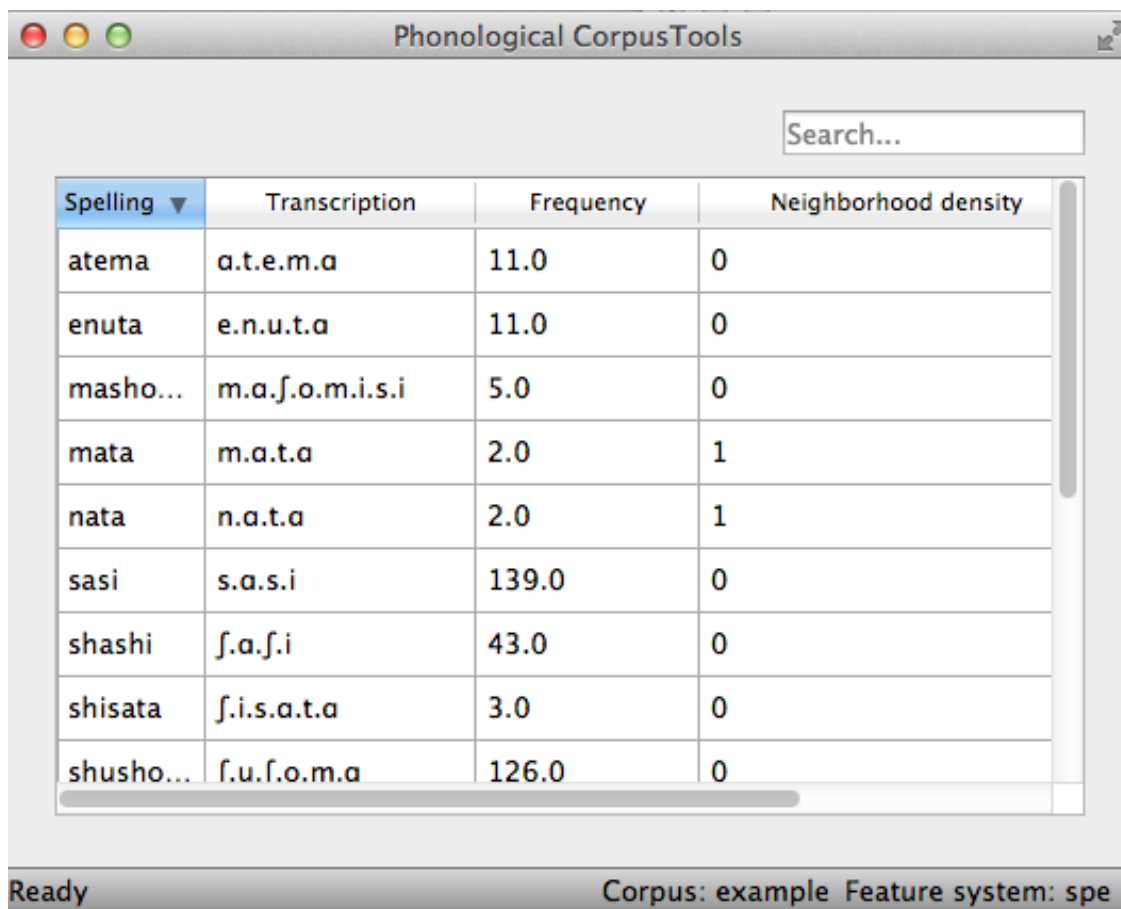
what value(s) might be good defaults here; instead, we recommend experimenting with the string similarity algorithm to determine what kinds of values are common for words that seem to count as neighbours, and working backward from that.

6. **Output file**: If this option is left blank, PCT will simply return the actual neighbourhood density for each word that is calculated (i.e., the number of neighbours of each word). If a file is chosen, then the number will still be returned, but additionally, a file will be created that lists all of the actual neighbours for each word.

7. **Results**: Once all options have been selected, click "Calculate neighborhood density." If this is not the first calculation, and you want to add the results to a pre-existing results table, select the choice that says "add to current results table." Otherwise, select "start new results table." A dialogue box will open, showing a table of the results, including the word, its neighbourhood density, the string type from which neighbourhood density was calculated, whether type or token frequency was used (if applicable), the string similarity algorithm that was used, and the threshold value. If the neighbourhood density for all words in the corpus is being calculated, simply click on the "start new results table" option, and you will be returned to your corpus, where a new column has been added automatically.

8. **Saving results**: The results tables can each be saved to tab-delimited .txt files by selecting "Save to file" at the bottom of the window. Any output files containing actual lists of neighbours are already saved as .txt files in the location specified (see step 6). If all neighbourhood densities are calculated for a corpus, the corpus itself can be saved by going to "File" / "Export corpus as text file," from where it can be reloaded into PCT for use in future sessions with the neighbourhood densities included.

Here's an example of neighbourhood density being calculated on transcriptions for the entire example corpus, using edit distance with a threshold of 1:
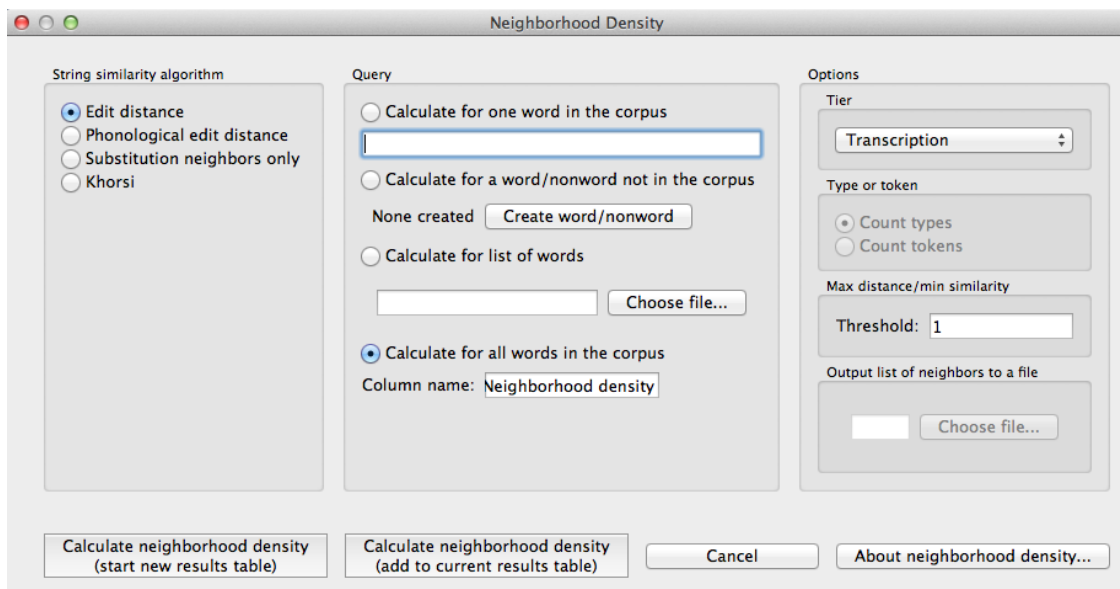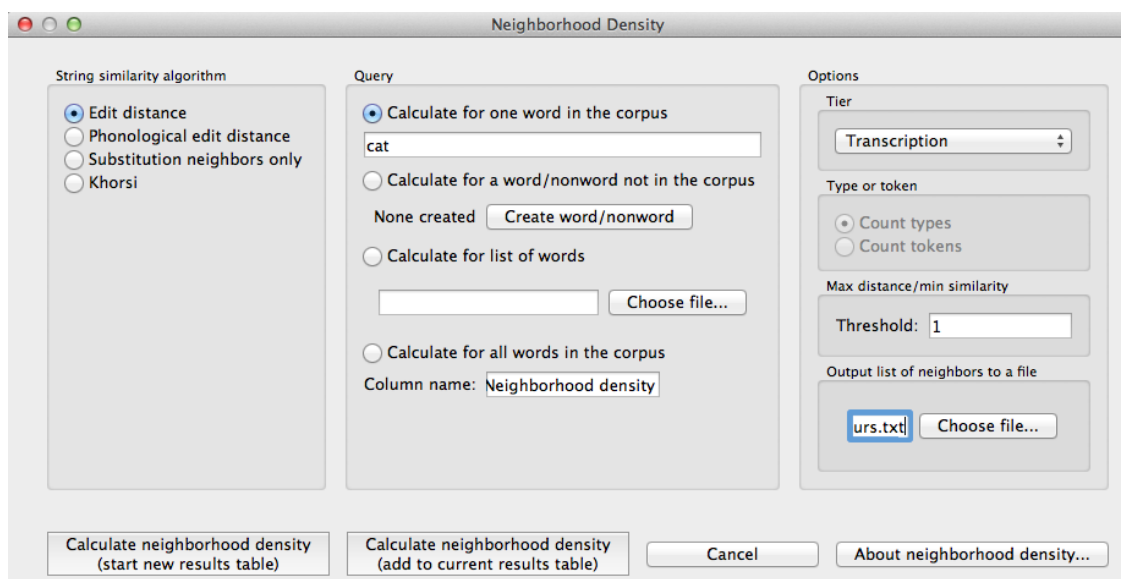
The corpus with all words' densities added:

An example of calculating all the neighbours for a given word in the IPHOD corpus, and saving the resulting list of neighbours to an output file:
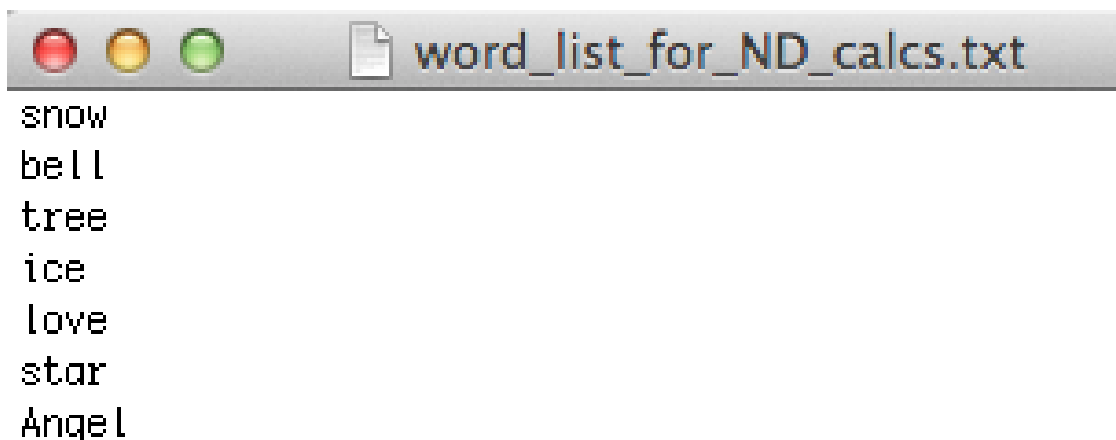

The on-screen results table, which can be saved to a file itself:
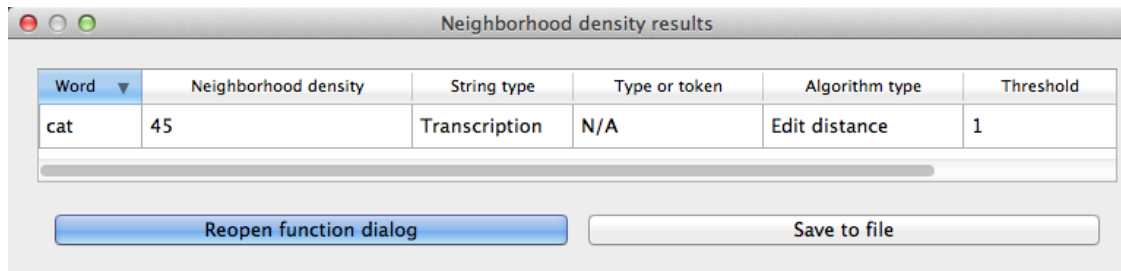
And the saved output file listing all 45 of the neighbours of *cat* in the IPHOD corpus:
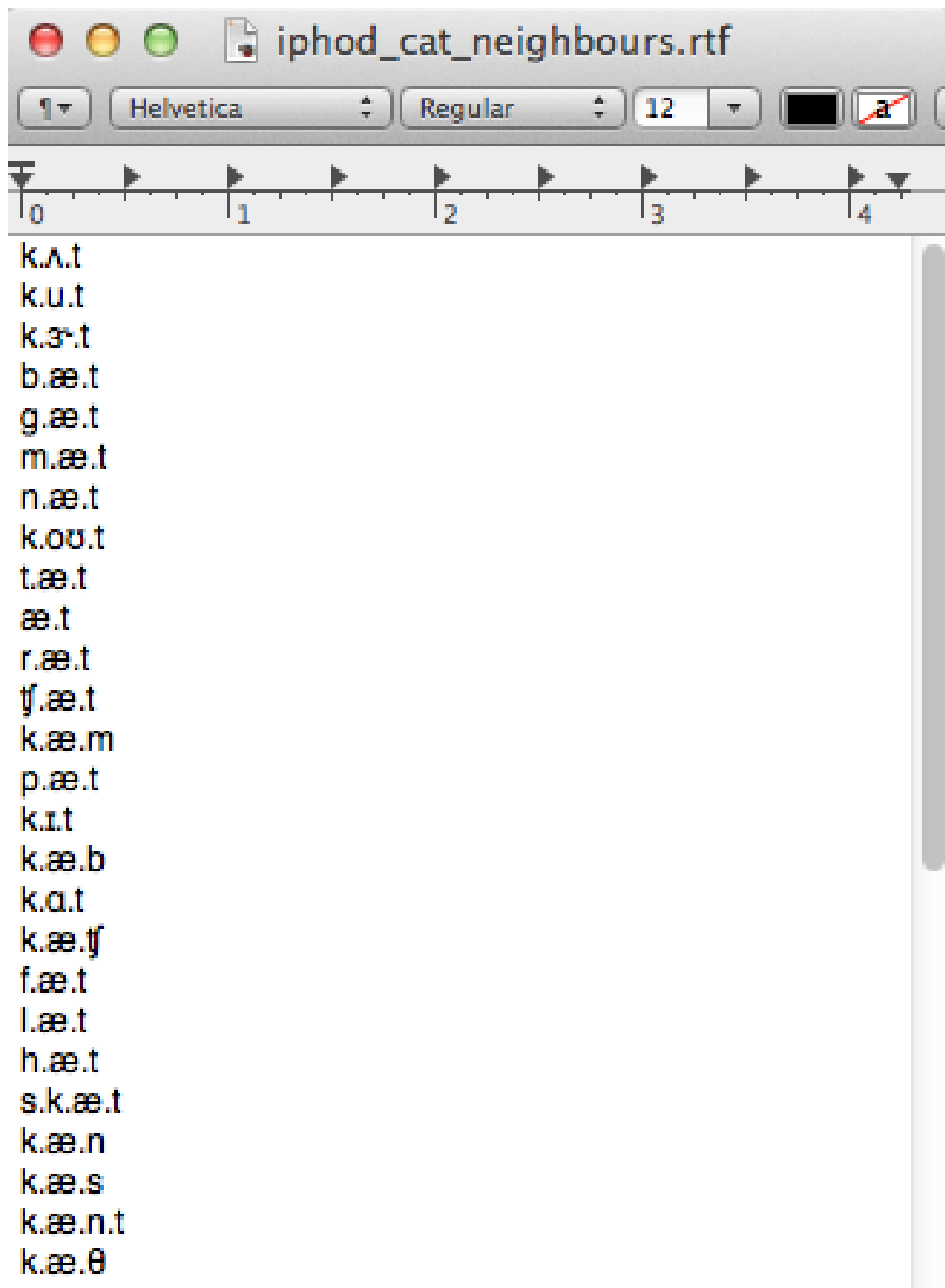
An example .txt file containing one word per line, that can be uploaded into PCT so that the neighbourhood density of each word is calculated:

The resulting table of neighbourhood densities for each word on the list (in the IPHOD corpus, with standard edit distance and a threshold of 1):

To return to the function dialogue box with your most recently used selections after any results table has been created, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

.._neighborhood_density_gui:

10.4 Implementing the neighbourhood density function on the command line

In order to perform this analysis on the command line, you must enter a command in the following format into your Terminal:

```
pct_neighdens CORPUSFILE ARG2
```

...where CORPUSFILE is the name of your *.corpus file and ARG2 is either the word whose neighborhood density you wish to calculate or the name of your word list file (if calculating the neighborhood density of each word). The word list file must contain one word (specified using either spelling or transcription) on each line. You may also use command line options to change various parameters of your neighborhood density calculations. Descriptions of these arguments can be viewed by running `pct_neighdens -h` or `pct_neighdens -help`. The help text from this command is copied below, augmented with specifications of default values:

Positional arguments:

`corpus_file_name`
> Name of corpus file

`query`
> Name of word to query, or name of file including a list of words

Optional arguments:

`-h`
`--help`
> Show this help message and exit

`-a` `ALGORITHM`
`--algorithm` `ALGORITHM`
> The algorithm used to determine distance

`-d` `MAX_DISTANCE`
`--max_distance` `MAX_DISTANCE`
> Maximum edit distance from the queried word to consider a word a neighbor.

`-s` `SEQUENCE_TYPE`
`--sequence_type` `SEQUENCE_TYPE`
> The name of the tier on which to calculate distance

`-w` `COUNT_WHAT`
`--count_what` `COUNT_WHAT`
> If 'type', count neighbors in terms of their type frequency. If 'token', count neighbors in terms of their token frequency.

`-m`
`--find_mutation_minpairs`
> This flag causes the script not to calculate neighborhood density, but rather to find minimal pairs–see documentation.

`-o` `OUTFILE`
`--outfile` `OUTFILE`
> Name of output file.

EXAMPLE 1: If your corpus file is example.corpus and you want to calculate the neighborhood density of the word 'nata' using defaults for all optional arguments, you would run the following command in your terminal window:

```
pct_neighdens example.corpus nata
```

EXAMPLE 2: Suppose you want to calculate the neighborhood distance of a list of words located in the file mywords.txt . Your corpus file is again example.corpus. You want to use the phonological edit distance metric, and you wish to count as a neighbor any word with a distance less than 0.75 from the query word. In addition, you want the script to produce an output file called output.txt . You would need to run the following command:

```
pct_neighdens example.corpus mywords.txt -a phonological_edit_distance -d 0.75 -o output.txt
```

EXAMPLE 3: You wish to find a list of the minimal pairs of the word 'nata'. You would need to run the following command:

```
pct_neighdens example.corpus nata -m
```

Frequency of alternation

11.1 About the function

The occurrence of alternations can be used in assessing whether two phonemes in a language are contrastive or allophonic, with alternations promoting the analysis of allophony (e.g., [Silverman2006], [Johnson2010], [Lu2012]), though it's clear that not all alternating sounds are allophonic (e.g., the [k]~[s] alternation in electric~electricity).

In general, two phonemes are considered to alternate if they occur in corresponding positions in two related words. For example, [s]/[] would be considered an alternation in the words [dps] and [dpn] as they occur in corresponding positions and the words are morphologically related. [Johnson2010] make the point that alternations may be more or less frequent in a language, and imply that this frequency may affect the influence of the alternations on phonological relations. As far as we know, however, there is no literature that directly tests this claim or establishes how frequency of alternation could actually be quantified (though see discussion in [Hall2014b]).

11.2 Method of calculation

In PCT, frequency of alternation [1] is the ratio of the number of words that have an alternation of two phonemes to the total number of words that contain either phoneme, as in:

$$Frequency\ of\ alternation = \frac{Words\ with\ an\ alternation\ of\ s_1\ and\ s_2}{Words\ with\ s_1\ +\ words\ with\ s_2}$$

To determine whether two words have an alternation of the targeted phonemes, one word must contain phoneme 1, the other must contain phoneme 2, and some threshold of "relatedness" must be met. In an ideal world, this would be determined by a combination of orthographic, phonological, and semantic similarity; see discussion in Hall et al. (in submission). Within PCT, however, a much more basic relatedness criterion is used: string similarity. This is indeed what [Khorsi2012] proposes as a measure of morphological relatedness, and though we caution that this is not in particularly close alignment with the standard linguistic interpretation of morphological relatedness, it is a useful stand-in for establishing an objective criterion. If both conditions are met, the two words are considered to have an alternation and are added to the pool of "*words with an alternation of s1 and s2.*"

It is also possible to require a third condition, namely, that the location of phoneme 1 and phoneme 2 be roughly phonologically aligned across the two words (e.g., preceded by the same material). Requiring phonological alignment will make PCT more conservative in terms of what it considers to "count" as alternations. However, the phonological alignment algorithm is based on [Allen2014] and currently only works with English-type morphology, i.e., a heavy reliance on prefixes and suffixes rather than any other kinds of morphological alternations. Thus, it should not be used with non-affixing languages.

Again, we emphasize that we do *not* believe this to currently be a particularly accurate reflection of morphological relatedness, so the resulting calculation of frequency of alternation should be treated with **extreme** caution. We include

it primarily because it is a straightforward function of string similarity that has been claimed to be relevant, not because the current instantiation is thought to be particularly valid.

11.3 Implementing the frequency of alternation function in the GUI

To start the analysis, click on "Analysis" / "Calculate frequency of alternation..." in the main menu, and then follow these steps:

1. **Segments**: The first step is to choose which two sounds you wish to check alternations for. Click on "Add pair of sounds"; PCT will automatically populate a menu of all sounds in the corpus, so all that needs to be done is the selection of the targeted two sounds. Multiple pairs of sounds can be added by selecting "Add and create another" instead "Add" in the selection window.

2. **String similarity algorithm**: Next, choose which distance / similarity metric to use. Refer to *Method of calculation: String similarity* for more details.

3. **Tier**: The tier from which string similarity is to be calculated can be selected. Generally, one is likely to care most about full transcriptions, but other tiers (e.g., a vowel tier) can also be selected; in this case, all information removed from the tier is ignored.

4. **Frequency Type**: Next, select which frequency type to use for your similarity metric, either type or token frequency. This parameter is only available if using the Khorsi similarity metric, which relies on counting the frequency of occurrence of the sounds in the currently selected corpus; neither edit distance metric involves frequency.

5. **Minimal pairs**: Then, select whether you wish to include alternations that occur in minimal pairs. If, for example, the goal is to populate a list containing all instances where two segments potentially alternate, select "include minimal pairs." Alternatively, if one wishes to discard known alternations that are contrastive, select "ignore minimal pairs." (E.g., "bat" and "pat" look like a potential "alternation" of [b] and [p] to PCT, because they are extremely similar except for the sounds in question, which are also phonologically aligned.)

6. **Threshold values**: If the Khorsi algorithm is selected, enter the minimum similarity value required for two words to count as being related. Currently the default is -15; this is an arbitrary (and relatively low / non-conservative) value. We recommend reading [Khorsi2012] and examining the range of values obtained using the string similarity algorithm before selecting an actual value here. Alternatively, if one of the edit distance algorithms is selected, you should instead enter a maximum distance value that is allowed for two words to count as being related. Again, there is a default (6) that is relatively high and non-conservative; an understanding of edit distances is crucial for applying this threshold in a meaningful way.

7. **Phonological alignment**: Choose whether you want to require the phones to be phonologically aligned or not, as per the above explanation.

8. **Corpus size**: Calculating the full set of possible alternations for a pair of sounds may be extremely time-consuming, as all words in the corpus must be compared pairwise. To avoid this problem, a subset of the corpus can be selected (in which case, we recommend running the calculation several times so as to achieve different random subsets for comparison). To do so, enter either (1) the number of words you'd like PCT to extract from the corpus as a subset (e.g., 5000) or (2) a decimal, which will result in that percentage of the corpus being used as a subset (e.g., 0.05 for 5% of the corpus).

9. **Output alternations**: You can choose whether you want PCT to output a list of all the words it considers to be "alternations." This is useful for determining how accurate the calculation is. If you do want the list to be created, enter a file path or select it using the system dialogue box that opens when you click on "Select file location." If you do not want such a list, leave this option blank.

An example of selecting the parameters for frequency of alternation, using the sample corpus:

10. **Results**: Once all options have been selected, click "Calculate frequency of alternation." If this is not the first calculation, and you want to add the results to a pre-existing results table, select the choice that says "add to current results table." Otherwise, select "start new results table." A dialogue box will open, showing a table of the results, including sound 1, sound 2, the total number of words with either sound, and total number of words with an alternation, the frequency of alternation and information about the specified similarity / distance metric and selected threshold values. To save these results to a .txt file, click on "Save to file" at the bottom of the table.

An example of the results table:

To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

Mutual Information

12.1 About the function

Mutual information [1] is a measure of how much dependency there is between two random variables, X and Y. That is, there is a certain amount of information gained by learning that X is present and *also* a certain amount of information gained by learning that Y is present. But knowing that X is present might also tell you something about the likelihood of Y being present, and vice versa. If X and Y always co-occur, then knowing that one is present already tells you that the other must also be present. On the other hand, if X and Y are entirely independent, then knowing that one is present tells you nothing about the likelihood that the other is present.

In phonology, there are two primary ways in which one could interpret X and Y as random variables. In one version, X and Y are equivalent random variables, each varying over "possible speech sounds in some unit" (where the unit could be any level of representation, e.g. a word or even a non-meaningful unit such as a bigram). In this case, one is measuring how much the presence of X anywhere in the defined unit affects the presence of Y in that same unit, regardless of the order in which X and Y occur, such that the mutual information of (X; Y) is the same as the mutual information of (Y; X), and furthermore, the pointwise mutual information of any individual value of each variable (X = a; Y = b) is the same as the pointwise mutual information of (X = b; Y = a). Although his is perhaps the most intuitive version of mutual information, given that it does give a symmetric measure for "how much information does the presence of a provide about the presence of *b*," we are not currently aware of any work that has attempted to use this interpretation of MI for phonological purposes.

The other interpretation of MI assumes that X and Y are different random variables, with X being "possible speech sounds occurring as the first member of a bigram" and Y being "possible speech sounds occurring as the second member of a bigram." This gives a directional interpretation to mutual information, such that, while the mutual information of (X; Y) is the same as the mutual information of (Y; X), the pointwise mutual information of (X = a; Y = b) is NOT the same as the pointwise mutual information of (X = b; Y = a), because the possible values for X and Y are different. (It is still, trivially, the case that the pointwise mutual information of (X = a; Y = b) and (Y = b; X = a) are equal.)

This latter version of mutual information has primarily been used as a measure of co-occurrence restrictions (harmony, phonotactics, etc.). For example, [Goldsmith2012] use pointwise mutual information as a way of examining Finnish vowel harmony; see also discussion in [Goldsmith2002]. Mutual information has also been used instead of transitional probability as a way of finding boundaries between words in running speech, with the idea that bigrams that cross word boundaries will have, on average, lower values of mutual information than bigrams that are within words (see [Brent1999], [Rytting2004]). Note, however, that in order for this latter use of mutual information to be useful, one must be using a corpus based on running text rather than a corpus that is simply a list of individual words and their token frequencies.

[1] The algorithm in PCT calculates what is sometimes referred to as the "pointwise" mutual information of a pair of units X and Y, in contrast to "mutual information," which would be the expected average value of the pointwise mutual information of all possible values of X and Y. We simplify to use "mutual information" throughout.

12.2 Method of calculation

Both of the interpretations of mutual information described above are implemented in PCT. We refer to the first one, in which X and Y are interpreted as equal random variables, varying over "possible speech sounds in a unit," as word-internal co-occurrence pointwise mutual information (pMI), because we specifically use the word as the unit in which to measure pMI. We refer to the second one, in which X and Y are different random variables, over either the first or second members of bigrams, as ordered pair pMI.

The general formula for pointwise mutual information is given below; it is the binary logarithm of the joint probability of X = a and Y = b, divided by the product of the individual probabilities that X = a and Y = b.

$pMI = log_2(\frac{p(X=a\&Y=b)}{p(X=a)*p(Y=b)})$

Word-internal co-occurrence pMI: In this version, the joint probability that X = a and Y = b is equal to the probability that some unit (here, a word) contains both a and b (in any order). Therefore, the pointwise mutual information of the sounds a and b is equal to the binary logarithm of the probability of some word containing both a and b, divided by the product of the individual probabilities of a word containing a and a word containing b.

Pointwise mutual information for individual segments:

$pMI_{word-internal} = log_2(\frac{p(a\in W \& b\in W)}{p(a\in W)*p(b\in W)})$

Ordered pair pMI: In this version, the joint probability that X = a and Y = b is equal to the probability of occurrence of the sequence ab. Therefore, the pointwise mutual information of a bigram (e.g., ab) is equal to the binary logarithm of the probability of the bigram divided by the product of the individual segment probabilities, as shown in the formula below.

Pointwise mutual information for bigrams:

$pMI_{ordered-pair} = log_2(\frac{p(ab)}{p(a)*p(b)})$

For example, given the bigram [a, b], its pointwise mutual information is the binary logarithm of the probability of the sequence [ab] in the corpus divided by a quantity equal to the probability of [a] times the probability of [b]. Bigram probabilities are calculated by dividing counts by the total number of bigrams, and unigram probabilities are calculated equivalently.

Note that pMI can also be expressed in terms of the information content of each of the members of the bigram. Information is measured as the negative log of the probability of a unit ($I(a) = -log_2 * p(a)$), so the pMI of a bigram ab is also equal to $I(a) + I(b)\check{}I(ab)$.

Note that in PCT, calculations are not rounded until the final stage, whereas in [Goldsmith2012], rounding was done at some intermediate stages as well, which may result in slightly different final pMI values being calculated.

12.3 Implementing the mutual information function in the GUI

To start the analysis, click on "Analysis" / "Calculate mutual information..." in the main menu, and then follow these steps:

1. **Bigram**: Click on the "Add bigram" button in the "Mutual Information" dialogue box. A new window will open with a phonetic inventory of all the segments that occur in your corpus. Select the bigram by clicking on one segment from the "left-hand side" and one segment from the "right-hand side." To add more than one bigram, click "Add and create another" to be automatically returned to the selection window. Once the last bigram has been selected, simply click "Add" to return to the Mutual Information dialogue box. All the selected bigrams will appear in a list. To remove one, click on it and select "Remove selected bigram."

2. **Tier**: Mutual information can be calculated on any available tier. The default is transcription. If a vowel tier has been created, for example, one could calculate the mutual information between vowels on that tier, ignoring intervening consonants, to examine harmony effects.

3. **Domain**: Choosing "set domain to word" will change the calculation so that the calculation is for word-internal co-occurrence pMI. In this case, the order and adjacency of the bigram does not matter; it is simply treated as a pair of segments that could occur anywhere in a word.

4. **Word boundary count**: A standard word object in PCT contains word boundaries on both sides of it (e.g., [#kæt#] 'cat'). If words were concatenated in real running speech, however, one would expect to see only one word boundary between each pair of words (e.g., [#mai#kæt#] 'my cat' instead of [#mai##kæt#]). To reproduce this effect and assume that word boundaries occur only once between words (as is assumed in [Goldsmith2012], choose "halve word boundary count." Note that this technically divides the number of boundaries in half and then adds one, to compensate for the extra "final" boundary at the end of an utterance. (It will make a difference only for calculations that include a boundary as one member of the pair.)

5. **Results**: Once all options have been selected, click "Calculate mutual information." If this is not the first calculation, and you want to add the results to a pre-existing results table, select the choice that says "add to current results table." Otherwise, select "start new results table." A dialogue box will open, showing a table of the results, including sound 1, sound 2, the tier used, and the mutual information value. To save these results to a .txt file, click on "Save to file" at the bottom of the table.

The following image shows the inventory window used for selecting bigrams in the sample corpus:

The selected bigrams appear in the list in the "Mutual Information" dialogue box:

The resulting mutual information results table:

To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

12.4 Implementing the mutual information function on the command line

In order to perform this analysis on the command line, you must enter a command in the following format into your Terminal:

```
pct_mutualinfo CORPUSFILE ARG2
```

...where CORPUSFILE is the name of your *.corpus file and ARG2 is the bigram whose mutual information you wish to calculate. The bigram must be in the format 's1,s2' where s1 and s2 are the first and second segments in the bigram. You may also use command line options to change the sequency type to use for your calculations, or to specify an output file name. Descriptions of these arguments can be viewed by running `pct_mutualinfo -h` or `pct_mutualinfo --help`. The help text from this command is copied below, augmented with specifications of default values:

Positional arguments:

corpus_file_name
> Name of corpus file

query
> Bigram, as str separated by comma

Optional arguments:

-h
--help
> Show help message and exit

-s SEQUENCE_TYPE
--sequence_type SEQUENCE_TYPE
> The attribute of Words to calculate FL over. Normally, this will be the transcription, but it can also be the spelling or a user-specified tier.

-o OUTFILE
--outfile OUTFILE
> Name of output file

EXAMPLE 1: If your corpus file is example.corpus and you want to calculate the mutual information of the bigram 'si' using defaults for all optional arguments, you would run the following command in your terminal window:

```
pct_mutualinfo example.corpus s,i
```

EXAMPLE 2: Suppose you want to calculate the mutual information of the bigram 'si' on the spelling tier. In addition, you want the script to produce an output file called output.txt. You would need to run the following command:

```
pct_mutualinfo example.corpus s,i -s spelling -o output.txt
```

Acoustic Similarity

13.1 About the function

Acoustic similarity analyses quantify the degree to which waveforms of linguistic objects (such as sounds or words) are similar to each other. The acoustic similarity measures provided here have primarily been used in the study of phonetic convergence between interacting speakers; convergence is measured as a function of increasing similarity. These measures are also commonly used in automatic speech and voice recognition systems, where incoming speech is compared to stored representations. Phonologically, acoustic similarity also has a number of applications. For example, it has been claimed that sounds that are acoustically distant from each other cannot be allophonically related, even if they are in complementary distribution (e.g. [Pike1947]; [Janda1999]).

Acoustic similarity alogorithms work on an aggregate scale, quantifying, on average, how similar one group of waveforms is to another. Representations have traditionally been in terms of mel-frequency cepstrum coefficents (MFCCs; [Delvaux2007]; [Mielke2012]), which is used widely for automatic speech recognition, but one recent introduction is multiple band amplitude envelopes [Lewandowski2012]. Both MFCCs and amplitude envelopes will be described in more detail in the following sections, and both are available as part of PCT.

The second dimension to consider is the algorithm used to match representations. The most common one is dynamic time warping (DTW), which uses dynamic programming to calculate the optimal path through a distance matrix [Sakoe1971], and gives the best alignment of two time series. Because one frame in one series can align to multiple frames in another series without a significant cost, DTW provides a distance independent of time. The other algorithm that is used is cross-correlation (see discussion in [Lewandowski2012], which aligns two time series at variable lags. Taking the max value of the alignment gives a similarity value for the two time series, with higher values corresponding to higher similarity.

13.2 Method of calculation

13.2.1 Preprocessing

Prior to conversion to MFCCs or amplitude envelopes, the waveform is pre-emphasized to give a flatter spectrum and correct for the higher drop off in amplitude of higher frequencies due to distance from the mouth.

13.2.2 MFCCs

The calculation of MFCCs in PCT's function follows the Rastamat [Ellis2005]'s implementation of HTK-style MFCCs [HTK] in [Matlab]. Generating MFCCs involves windowing the acoustic waveform and transforming the windowed signal to the linear frequency domain through a Fourier transform. Following that, a filterbank of triangular filters is constructed in the mel domain, which gives greater resolution to lower frequencies than higher frequencies. Once the

filterbank is applied to the spectrum from the Fourier transform, the spectrum is represented as the log of the power in each of the mel filters. Using this mel spectrum, the mel frequency cepstrum is computed by performing a discrete cosine transform. This transform returns orthogonal coefficients describing the shape of the spectrum, with the first coeffcent as the average value, the second as the slope of the spectrum, the third as the curvature, and so on, with each coefficient representing higher order deviations. The first coeffcent is discarded, and the next X coeffcents are taken, where X is the number of coeffcents specified when calling the function. The number of coeffcents must be one less than the number of filters, as the number of coeffcents returned by the discrete cosine transform is equal to the number of filters in the mel filterbank.

13.2.3 Amplitude envelopes

The calculation of amplitude envelopes follows the Matlab implementation found in [Lewandowski2012]. First, the signal is filtered into X number of logarithmically spaced bands, where X is specified in the function call, using 4th order Butterworth filters. For each band, the amplitude envelope is calculated by converting the signal to its analytic signal through a Hilbert transform. Each envelope is downsampled to 120 Hz.

13.2.4 Dynamic time warping (DTW)

PCT implements a standard DTW algorithm [SakoeChiba, 1971]_ and gives similar results as the dtw package [Giorgino2009)]_ in [R]. Given two representations, a 2D matrix is constructed where the dimensions are equal to the number of frames in each representation. The initial values for each cell of the matrix is the Euclidean distance between the two feature vectors of those frames. The cells are updated so that they equal the local distance plus the minimum distance of the possible previous cells. At the end, the final cell contains the summed distance of the best path through the matrix, and this is the minimum distance between two representations.

13.2.5 Cross-correlation

Cross-correlation seeks to align two time series based on corresponding peaks and valleys. From each representation a time series is extracted for each frame's feature and this time series is cross-correlated with the respective time series in the other representation. For instance, the time series for an amplitude envelope's representation corresponds to each frequency band, and each frequency band of the first representation is cross-correlated with each respective frequency band of the second representation. The time series are normalized so that they sum to 1, and so matching signals receive a cross-correlation value of 1 and completely opposite signals receive a cross-correlation value of 0. The overall distance between two representations is the inverse of the average cross-correlation values for each band.

13.2.6 Similarity across directories

The algorithm for assessing the similarity of two directories (corresponding to segments) averages the similarity of each .wav file in the first directory to each .wav file in the second directory.

13.3 Implementing the acoustic similarity function in the GUI

To start the analysis, click on the "Calculate acoustic similarity..." in the Analysis menu and provide the following parameters. Note that unlike the other functions, acoustic similarity is not tied directly to any corpus that is loaded into PCT; sound files are accessed directly through directories on your computer.

1. **Comparison type**: There are three kinds of comparisons that can be done in PCT: single-directory, two-directory, or pairwise.

(a) **Single directory**: If a single directory is selected (using the "Choose directory..." dialogue box), two types of results will be returned: (1) each of the pairwise comparisons and (2) an average of all these comparisons (i.e., a single value).

(b) **Two directories**: Choose two directories, each corresponding to a set of sounds to be compared. For example, if one were interested in the similarity of [s] and [] in Hungarian, one directory would contain .wav files of individual [s] tokens, and the other directory would contain .wav files of individual [] tokens. Every sound file in the first directory will be compared to every sound file in the second directory, and the acoustic similarity measures that are returned will again be (1) all the pairwise comparisons and (2) an average of all these comparisons (i.e., a single value).

(c) **Pairwise**: One can also use a tab-delimied.txt file that lists all of the pairwise comparisons of individual sound files by listing their full path names. As with a single directory, each pairwise comparison will be returned separately.

2. **Representation**: Select whether the sound files should be represented as MFCCs or amplitude envelopes (described in more detail above).

3. **Distance algorithm**: Select whether comparison of sound files should be done using dynamic time warping or cross-correlation (described in more detail above).

4. **Frequency limits**: Select a minimum frequency and a maximum frequency to use when generating representations. The human voice typically doesn't go below 80 Hz, so that is the default cut off to avoid low-frequency noise. The maximum frequency has a hard bound of the Nyquist frequency of the sound files, that is, half their sampling rate. The lowest sampling rate that is typically used for speech is 16,000 Hz, so a cutoff near the Nyquist (8,000 Hz) is used as the default. The range of human hearing is 20 Hz to 20 kHz, but most energy in speech tends to fall off after 10 kHz.

5. **Frequency resolution**: Select the number of filters to be used to divide up the frequency range specified above. The default for MFCCs is for 26 filters to be constructed, and for amplitude envelopes, 8 filters.

6. **Number of coefficients (MFCC only)**: Select the number of coefficients to be used in MFCC representations. The default is 12 coefficients, as that is standard in the field. If the number of coefficients is more than the number of filters minus one, the number of coefficients will be set to the number of filters minus one.

7. **Output**: Select whether to return results as similarity (inverse distance) or to us ethe default, distance (inverse similarity). Dynamic time warping natively returns a distance measure which gets inverted to similarity and cross-correlation natively returns a similarity value which gets inverted to distance.

8. **Multiprocessing**: As the generation and comparison of representations can be time-intensive, using multiprocessing on parts that can be run in parallel can speed the process up overall. In order to make this option available, the python-acoustic-similarity module must be installed; multiprocessing itself can be enabled by going to "Options" / "Preferences" / "Processing" (see also §3.9.1).

Here's an example of the parameter-selection box:
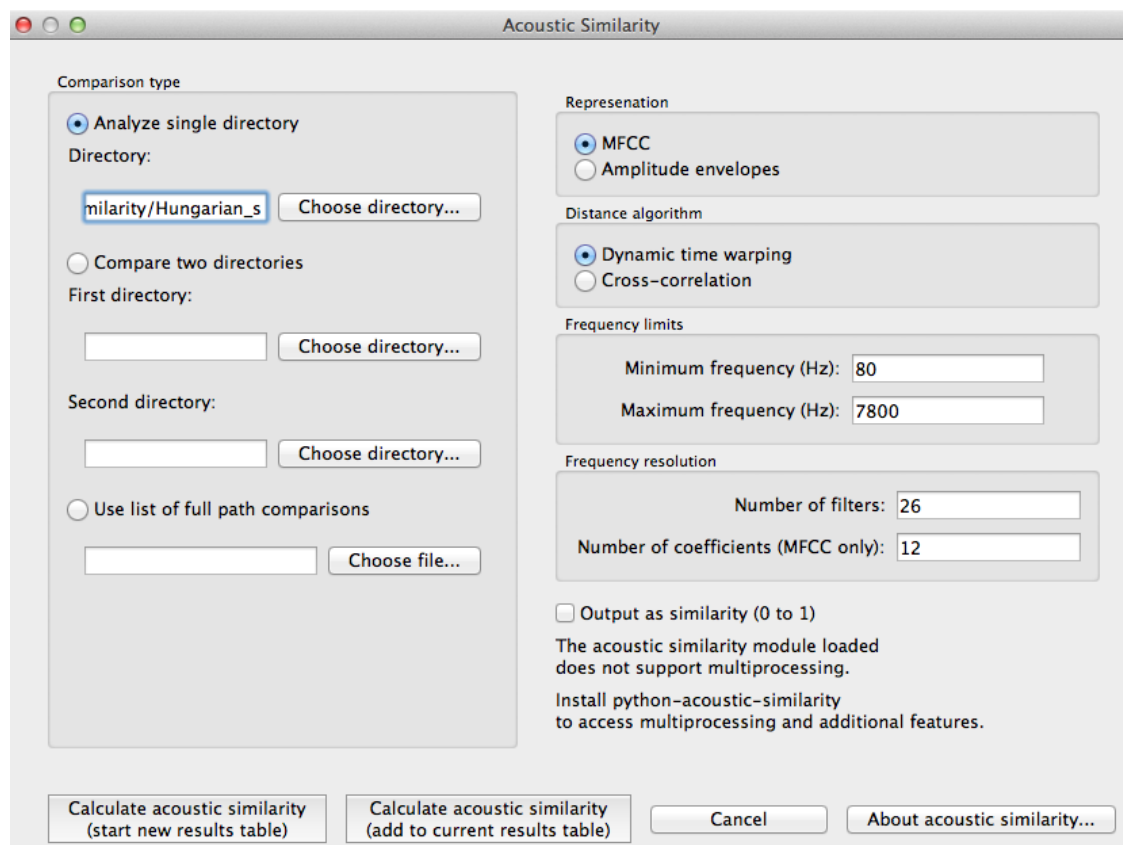
9. **Calculating and saving results**: The first time an analysis is run, the option to "Calculate acoustic similarity (start new results table)" should be selected. This will output the results to a pop-up window that lists the directories, the representation choice, the matching function, the minimum and maximum frequencies, the number of filters, the number of coefficients, the raw result, and whether the result is similarity (1) or distance (0). Subsequent analyses can either be added to the current table (as long as it hasn't been closed between analyses) or put into a new table. Once a table has been created, click on "Save to file" at the bottom of the table window in order to open a system dialogue box and choose a directory; the table will be saved as a tab-delimited .txt file.

Here's an example of the results file:

To return to the function dialogue box with your most recently used selections, click on "Reopen function dialog." Otherwise, the results table can be closed and you will be returned to your corpus view.

Citing PCT and the algorithms used therein

Please cite PCT as the following (all authors after K. C. Hall are listed alphabetically):

Hall, Kathleen Currie, Blake Allen, Michael Fry, Scott Mackie, and Michael McAuliffe. (2015). Phonological CorpusTools, Version 1.0. [Computer program]. Available from PCT SourceForge page.

If you need to cite a more traditional academic source rather than the software itself, please use:

Mackie, Scott, Kathleen Currie Hall, Blake Allen, Michael McAuliffe, Michael Fry. (2014). Phonological CorpusTools: A free, open-source tool for phonological analysis. Presented at the 14th Conference for Laboratory Phonology, Tokyo, Japan.

If you are using the IPHOD corpus as distributed with PCT, please also be sure to cite:

Vaden, K. I., Halpin, H. R., Hickok, G. S. (2009). Irvine Phonotactic Online Dictionary, Version 2.0. [Data file]. Available from http://www.iphod.com.

and if you are making use of the SUBTLEX token frequencies as part of the IPHOD corpus, you should cite:

Brysbaert, Marc, & Boris New. (2009). Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. Behavior Research Methods 41(4): 977-990.

More generally, the algorithms that are implemented in PCT are taken from published sources. As mentioned in the introduction, we highly encourage users of PCT to cite the original sources of the algorithms rather than, for example, saying that "functional load was calculated using PCT" and just citing PCT itself. First, there are multiple parameters within PCT that can be selected for any given calculation, and these should themselves be specified for maximum clarity and replicability. Second, credit for the original creation or application of the algorithms should obviously be given to the proper sources. We have attempted to make this as easy as possible by both giving these sources here in the user's manual and also embedding them in each function in the "About" option for each. Furthermore, if you are the author of a function that is currently implemented in PCT and you disagree with the way in which it has been implemented, please contact us to let us know! We have done our best to faithfully replicate published descriptions, but it is obviously possible that we have made errors. Finally, if you are the author of a function that you would like to see implemented in PCT, please contact us to discuss the possibility.

References

Indices and tables

- *genindex*
- *modindex*
- *search*

[Allen2014] Allen, Blake & Michael Becker (2014). Learning alternations from surface forms with sublexical phonology. Ms. University of British Columbia and Stony Brook University. See also http://sublexical.phonologist.org/.

[Archangeli2013] Archangeli, Diana & Douglas Pulleyblank. 2013. The role of UG in phonology. Proceedings of the West Coast Conference on Formal Linguistics 31. Somerville, MA: Cascadilla Press.

[PRAAT] Boersma, Paul & Weenink, David (2014). Praat: doing phonetics by computer [Computer program]. Available from http://www.praat.org/

[Brent1999] Brent, Michael R. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. Machine Learning 34.71-105.

[SUBTLEX] Brysbaert, Marc, & Boris New. 2009. Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. Behavior Research Methods 41(4): 977-990.

[Bybee2001] Bybee, Joan L. 2001. Phonology and language use. Cambridge: Cambridge UP.

[SPE] Chomsky, Noam & Morris Halle. 1968. The sound pattern of English. New York: Harper & Row.

[Delvaux2007] Delvaux, V., Soquet, A., 2007. The influence of ambient speech on adult speech productions through unintentional imitation. Phonetica 64 (2-3), 145–173.

[Ellis2005] Ellis, D. P. W. (2005). PLP and RASTA (and MFCC), and inversion) in Matlab. Online web resource. http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/

[Ernestus2011] Ernestus, Mirjam. 2011. Gradience and categoricality in phonological theory. In The Blackwell Companion to Phonology, ed. by M. van Oostendorp, C.J. Ewen, E. Hume & K. Rice, 2115-36. Oxford: Wiley-Blackwell.

[HTK] Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., ... & Woodland, P. (1997). The HTK book (Vol. 2). Cambridge: Entropic Cambridge Research Laboratory.

[Frisch2011] Frisch, Stefan A. 2011. Frequency effects. In The Blackwell Companion to Phonology, ed. by M. van Oostendorp, C.J. Ewen, E. Hume & K. Rice, 2137-63. Oxford: Wiley-Blackwell.

[Frisch2004] Frisch, Stefan, Janet B. Pierrehumbert & Michael B. Broe. 2004. Similarity avoidance and the OCP. Natural Language and Linguistic Theory 22.179-228.

[TIMIT] Garofolo, John, et al. 1993. TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium.

[Giorgino2009] Giorgino, T. (2009). Computing and visualizing dynamic time warping alignments in R: the dtw package. Journal of statistical Software, 31(7), 1-24.

[Goldsmith2002] Goldsmith, John. 2002. Probabilistic models of grammar: phonology as information minimization. Phonological Studies 5.21-46.

[Goldsmith2012] Goldsmith, John & Jason Riggle. 2012. Information theoretic approaches to phonological structure: the case of Finnish vowel harmony. Natural Language and Linguistic Theory 30.859-96.

[Greenberg1964] Greenberg, J.H. & J. Jenkins. 1964. Studies in the psychological correlated of the sound system of American English. Word 20.157-77.

[Hall2013a] Hall, Daniel Currie & Kathleen Currie Hall. 2013. Marginal contrasts and the Contrastivist Hypothesis. Paper presented to the Linguistics Association of Great Britain, London, 2013.

[Hall2009] Hall, Kathleen Currie. 2009. A probabilistic model of phonological relationships from contrast to allophony. Columbus, OH: The Ohio State University Doctoral dissertation.

[Hall2012] Hall, Kathleen Currie. 2012. Phonological relationships: A probabilistic model. McGill Working Papers in Linguistics 22.

[Hall2013b] Hall, Kathleen Currie. 2013. Documenting phonological change: A comparison of two Japanese phonemic splits. In: Luo, S. (Ed.), Proceedings of the 2013 Annual Meeting of the Canadian Linguistic Association. Canadian Linguistic Association, Toronto, published online at http://homes.chass.utoronto.ca/~cla-acl/actes2013/actes2013.html.

[Hall2014a] Hall, Kathleen Currie, and Elizabeth Hume. 2014. Modeling Perceptual Similarity: Phonetic, Phonological and Other Influences on the Perception of French Vowels. Ms., University of British Columbia & University of Canterbury.

[Hall2014b] Hall, Kathleen Currie, Claire Allen, Tess Fairburn, Kevin McMullin, Michael Fry, & Masaki Noguchi. 2014. Measuring perceived morphological relatedness. Paper presented at the Canadian Linguistics Association annual meeting.

[Hayes2009] Hayes, Bruce. 2009. Introductory Phonology. Malden, MA: Blackwell - Wiley.

[Hockett1955] Hockett, Charles F. (1955). A manual of phonology. International Journal of American Linguistics, 21(4).

[Hockett1966] Hockett, Charles F. 1966. The quantification of functional load: A linguistic problem. U.S. Air Force Memorandum RM-5168-PR.

[Hume2015] Hume, Elizabeth, Kathleen Currie Hall & Andrew Wedel. to appear. Strategic responses to uncertainty: Strong and weak sound patterns. Proceedings of the 5th International Conference on Phonology and Morphology. Korea.

[Hume2013] Hume, Elizabeth, Hall, Kathleen Currie, Wedel, Andrew, Ussishkin, Adam, Adda-Decker, Martine, & Gendrot, Cédric. (2013). Anti-markedness patterns in French epenthesis: An information-theoretic approach. In C. Cathcart, I.-H. Chen, G. Finley, S. Kang, C. S. Sandy & E. Stickles (Eds.), Proceedings of the Thirty-Seventh Annual Meeting of the Berkeley Linguistics Society (pp. 104-123). Berkeley: Berkeley Linguistics Society.

[Janda1999] Janda, Richard D. (1999). Accounts of phonemic split have been greatly exaggerated – but not enough. Proceedings of the 14th International Congress of Phonetic Sciences, 329-332.

[Johnson2010] Johnson, Keith, & Molly Babel. 2010. On the perceptual basis of distinctive features: Evidence from the perception of fricatives by Dutch and English speakers. Journal of Phonetics 38: 127-136.

[Khorsi2012] Khorsi, Ahmed. 2012. On morphological relatedness. Natural Language Engineering.1-19.

[King1967] King, Robert D. (1967). Functional load and sound change. Language, 43(4), 831-852.

[Kucera1963] Kučera, Henry. (1963). Entropy, redundancy, and functional load in Russian and Czech. American contributions to the Fifth International Conference of Slavists (Sofia), 191-219.

[Lewandowski2012] Lewandowski, Natalie. 2012. Talent in nonnative phonetic convergence: Universität Stuttgart Doctoral dissertation.

[Lu2012] Lu, Yu-an. 2012. The role of alternation in phonological relationships: Stony Brook University Doctoral dissertation.

[Luce1998] Luce, Paul A. & David B. Pisoni. 1998. Recognizing spoken words: The neighborhood activation model. Ear Hear 19.1-36.

[Maekawa2003] Maekawa, Kikuo. 2003. Corpus of Spontaneous Japanese: Its Design and Evaluation. Proceedings of ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR2003).7-12.

[CSJ] Maekawa, Kikuo. 2004. Design, compilation, and some preliminary analyses of the Corpus of Spontaneous Japanese. Spontaneous speech: Data and analysis, ed. by K. Maekawa & K. Yoneyama, 87-108. Tokyo: The National Institute of Japanese Language.

[MATLAB] The MathWorks Inc. (2014). MATLAB, Version R2014a.

[Mielke2008] Mielke, Jeff. 2008. The emergence of distinctive features. Oxford: Oxford UP.

[Mielke2012] Mielke, J. 2012. A phonetically based metric of sound similarity. Lingua, 122(2), 145-163.

[Peperkamp2003] Peperkamp, Sharon, Michèle Pettinato & Emmanuel Dupoux. 2003. Allophonic variation and the acquisition of phoneme categories. Proceedings of the 27th Annual Boston University Conference on Language Development, 650-61. Somerville, MA: Cascadilla Press.

[Peperkamp2006] Peperkamp, Sharon, Le Calvez, Rozenn, Nadal, Jean-Pierre, & Dupoux, Emmanuel. (2006). The acquisition of allophonic rules: Statistical learning with linguistic constraints. Cognition, 101, B31-B41.

[Pike1947] Pike, Kenneth L. (1947). Phonemics. Ann Arbor: The University of Michigan Press.

[BUCKEYE] Pitt, M.A., Dilley, L., Johnson, K., Kiesling, S., Raymond, W., Hume, E. and Fosler-Lussier, E. (2007) Buckeye Corpus of Conversational Speech (2nd release) [www.buckeyecorpus.osu.edu] Columbus, OH: Department of Psychology, Ohio State University (Distributor).

[R] R Core Team (2014). R: A Language and Environment for Statistical Computing, Version 3.1.0. http://www.R-project.org/

[Rytting2004] Rytting, C. Anton. 2004. Segment predictability as a cue in word segmentation: Application to Modern Greek. Proceedings of the Workshop of the ACL Special Interest Group on Computational Phonology (SIGPHON).

[Sakoe1971] Sakoe, H., & Chiba, S. (1971). A dynamic programming approach to continuous speech recognition. In Proceedings of the seventh international congress on acoustics (Vol. 3, pp. 65-69).

[Shannon1949] Shannon, Claude E., & Weaver, Warren. (1949). The Mathematical Theory of Communication (1998 ed.). Urbana-Champaign: University of Illinois Press.

[Silverman2006] Silverman, Daniel. 2006. A critical introduction to phonology: Of sound, mind, and body. London/New York: Continuum.

[Surendran2003] Surendran, Dinoj & Partha Niyogi. 2003. Measuring the functional load of phonological contrasts. In Tech. Rep. No. TR-2003-12. Chicago.

[Thakur2011] Thakur, Purnima (2011). Sibilants in Gujarati phonology. Paper presented at Information-theoretic approaches to linguistics, University of Colorado - Boulder.

[Todd2012] Todd, Simon. 2012. Functional load and length-based Mori vowel contrast. Poster presented at the Annual Meeting of the New Zealand Linguistic Society. Auckland, Dec. 2012.

[IPHOD] Vaden, K. I., H. R. Halpin & G. S. Hickok. 2009. Irvine Phonotactic Online Dictionary, Version 2.0. [Data file.] Available from: http://www.iphod.com.

[Vitevitch1999] Vitevitch, M.S. and Luce, P.A. (1999). Probabilistic phonotactics and neighborhood activation in spoken word recognition. Journal of Memory & Language, 40, 374-408.

[Vitevitch2004] Vitevitch, M.S. & Luce, P.A. (2004). A web-based interface to calculate phonotactic probability for words and nonwords in English. Behavior Research Methods, Instruments, and Computers, 36, 481-487.

[Wedel2013] Wedel, Andrew, Abby Kaplan & Scott Jackson. (2013). High functional load inhibits phonological contrast loss: A corpus study. Cognition 128.179-86.

[CMU] Weide, Robert L. (1994). CMU Pronouncing Dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[Yao2011] Yao, Yao. (2011). The effects of phonological neighborhoods on pronunciation variation in conversational speech. Berkeley: University of California, Berkeley Doctoral dissertation.

Symbols

P

Q