Processing Child Language Samples

Clifton Pye pyersqr at ku dot edu

February 2021

This document describes the steps for producing a basic analysis of a child's language sample. The procedures start with a transcription in ELAN of a recording of a child interacting with other speakers. The analysis procedures export the child's utterances from ELAN to Excel via Unicode text files. The different steps in the process assume that the input files have a standard format. The document describes the use of Perl programs that create a lexical concordance (concord.pl) and a vowel concordance (vowel.pl).

The ELAN transcription

The ELAN transcription tool enables transcribers to link their transcription to audio and video files. These links make it easy to replay a selected portion of the recording in order to check the transcription. ELAN provides horizontal tiers for each speaker. These tiers make it easy to show the overlaps between the utterances of different speakers. The ELAN program can be downloaded from the Max Plank Institute for Psycholinguistics at <https://archive.mpi.nl/tla/elan/download>. Figure 1 shows a screenshot for a typical ELAN transcription.

Figure 1. Screenshot of ELAN transcription
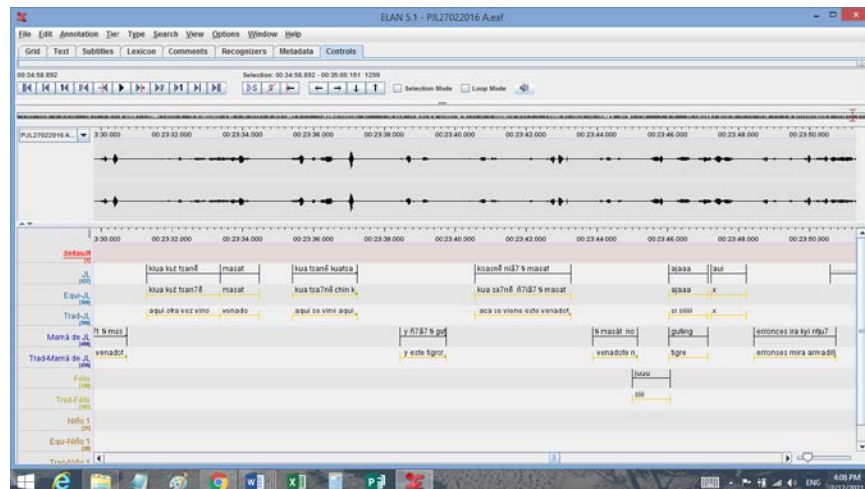
The ELAN transcription shows the standard format for transcriptions of child language samples. The transcription provides one tier for the child's utterance (JL), a following tier for the adult equivalent of the child's utterance, and a third tier for a translation into another language. The analysis procedures assume that children's utterances have these three tiers and no others. The transcription shows that the child's utterances are only divided into words. The following two tiers show the typical transcription format for an adult speaker. The first tier shows the adult's utterance, and the second tier shows the translation. Adult utterances, like the child's are only divided into words. The analysis programs assume that the transcriptions have translation tiers, but do not require them. A blank translation line can be added at a later stage if needed. The

transcription format follows the logic of the minimal transcription method described in Pye (2020).

Transferring ELAN transcriptions to Excel

The analysis programs require text files in the UTF-8 format. The first step in the analysis
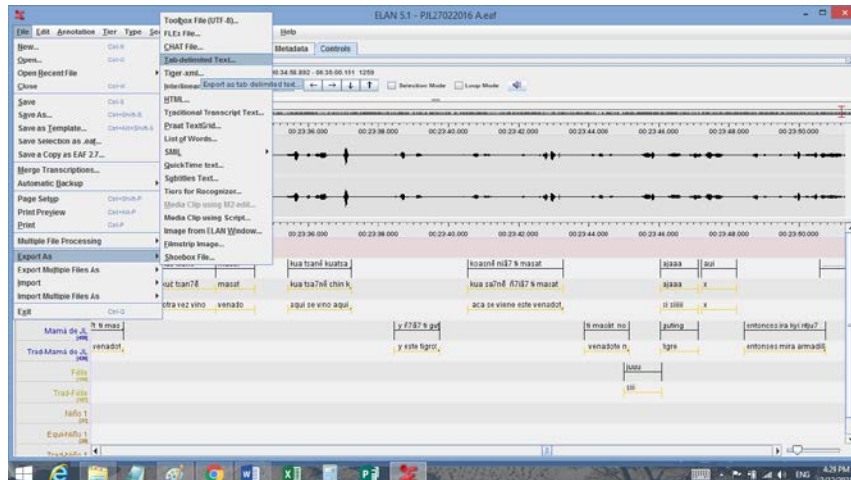

process is to export the ELAN transcription to an Excel file. The Excel file will become the basis for all of the analyses. This helps investigators to keep all of the analyses on the same Excel worksheet. Follow these steps to transfer the ELAN transcription to Excel:

1. Click on the File tab and then scroll down to the Export As tab and click on the Tab-Delimited Text option.

2. This action opens a window for exporting the transcription. Check that all of the tiers have been marked. (There is no need to export the default tier.) Then click ok at the bottom of the window.
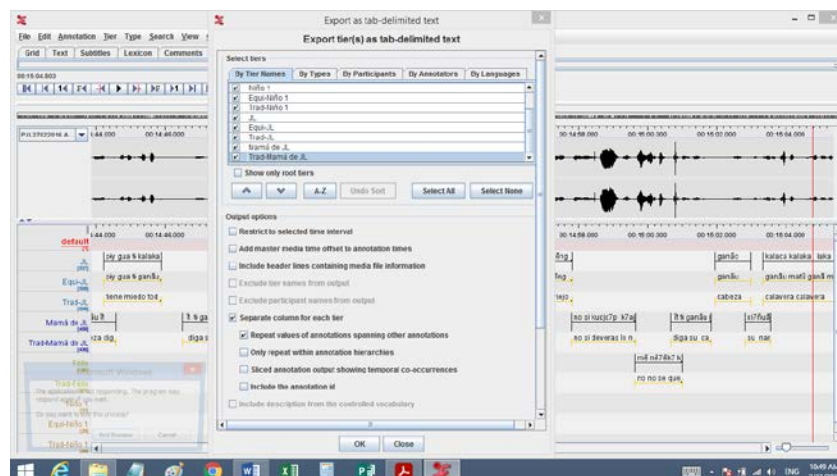
This action opens a third window that allows you to save the text file in the folder and format you wish to use. Be careful to use the UTF-8 encoding format. Select the destination folder where you wish to save the file and click save.
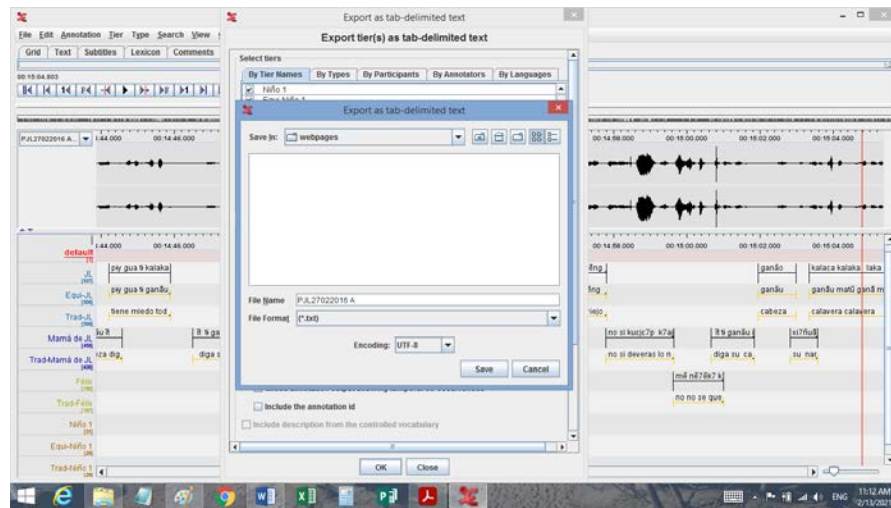

You should then find the resulting text file in the destination folder.
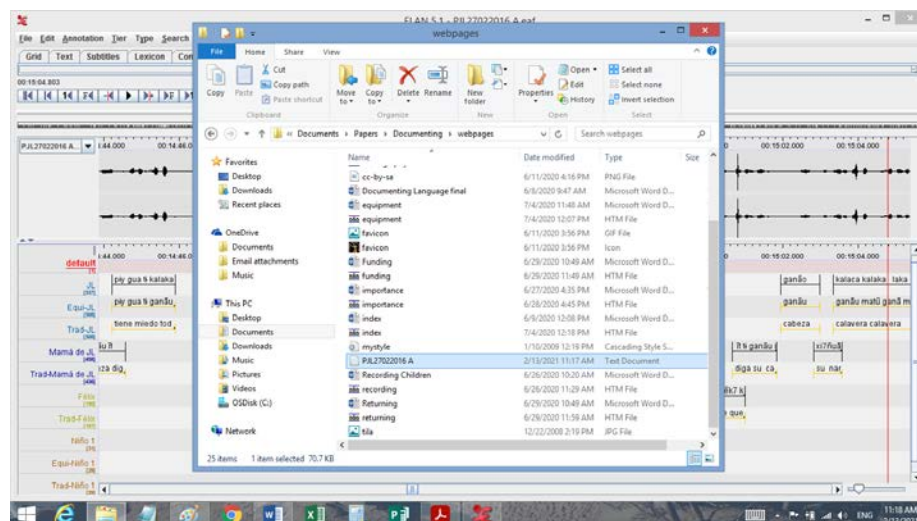
Text files do not have special characters that specify font types and style types so text files are easy to process with computer programs. The text file was exported from ELAN in the UTF-8 format. This format needs to be changed to the Unicode format before importing the file to Excel. Open the text file in notepad. If you do not have the notepad app on your computer you can download the notepad++ app from the internet. Select the Unicode format and save the file.
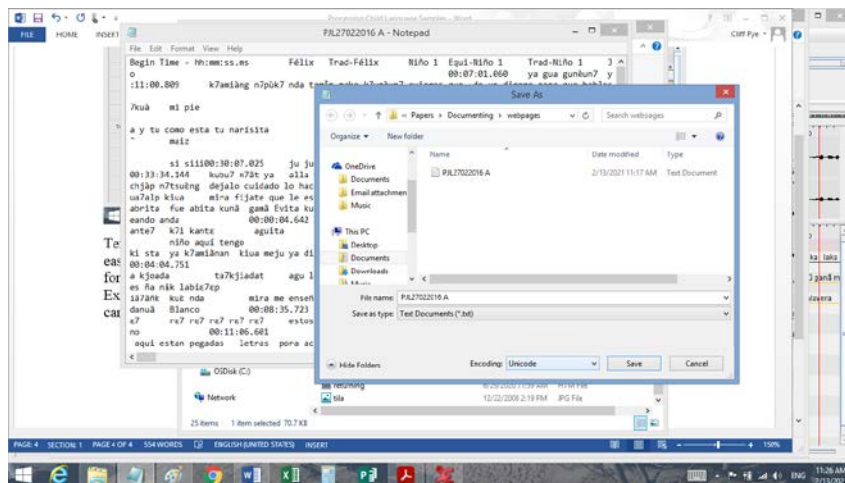
Once the file is saved in the Unicode format, it is ready to import to Excel. Open a blank worksheet in Excel, click on the Data tab at the top of the screen, look for the Get External Data options, and click the From Text option. This action opens a window that lets you find the text file that you exported from ELAN. Click the Import tab at the bottom of the window and follow the instructions.
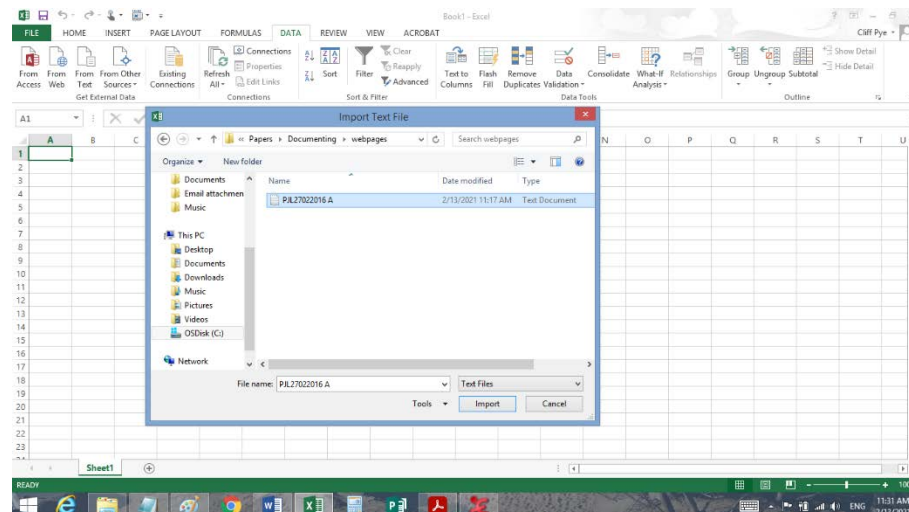
Select the delimited option and click Finish. This action should produce an Excel worksheet with separate columns that correspond to each tier in the ELAN transcription. The tiers for each participant are grouped together. The rows in the worksheet show the beginning time for each utterance by each participant.

Save the worksheet with the same name as the ELAN file in order to identify the source of the data in the Excel file.

The Excel provides sorting and filtering tools that help in data analysis. The worksheet should preserve any special characters that were used in the transcription. The first step in analysis is to create separate worksheets for each participant. Start by selecting the first four columns, which in this example show the Begin Time, JL's utterances, the adult equivalences, and the translations.


Copy these four columns to a new worksheet in the same file. Add an additional column to record which part of the recording the transcription comes from.

The resulting worksheet contains all of the child's utterances with their interpretations and translations. The beginning times make it possible to find each utterance in the ELAN transcription. The analyses of the children's utterances start from this worksheet.

Lexical Analysis

The first step in lexical analysis is to produce a lexical concordance for the children's words. A concordance provides the utterance contexts for a child's production of each word in the recording. It is best to use the adult targets as the concordance keys in order to group together all of the child's attempts at each adult word. Because it is easier for computer programs to process text files, it is necessary to save the Excel file as a Unicode text file. Go to the File tab in Excel and select the Save As option. Excel will open a window in which you can select a folder for the text file as well as the file format. Select the Unicode Text format and click Save.
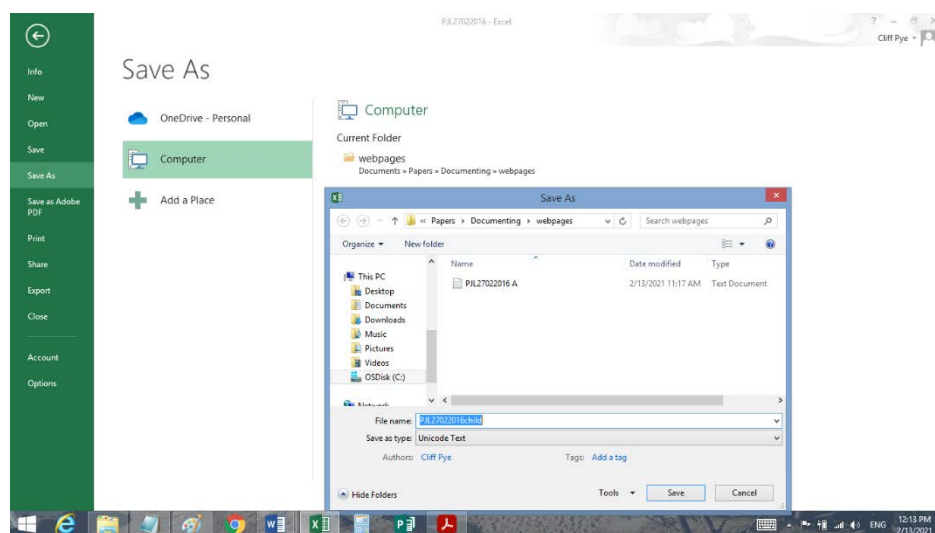
Excel will open a window that says the selected file type does not support multiple sheets. This is not a problem since we are only interested in saving the sheet with the child's data. Click OK.
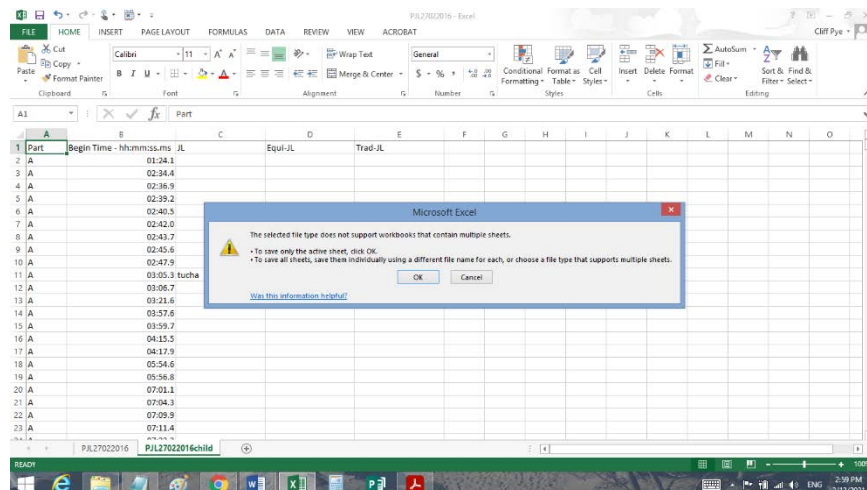
This action should produce the Unicode text file containing the child's utterances. The text file will have five tab-delimited columns that correspond to the columns in the Excel file. The first line in the text file will have the column headers shown in the first row of the Excel file. Save the file in the UTF-8 format in order to use it in the analysis programs.
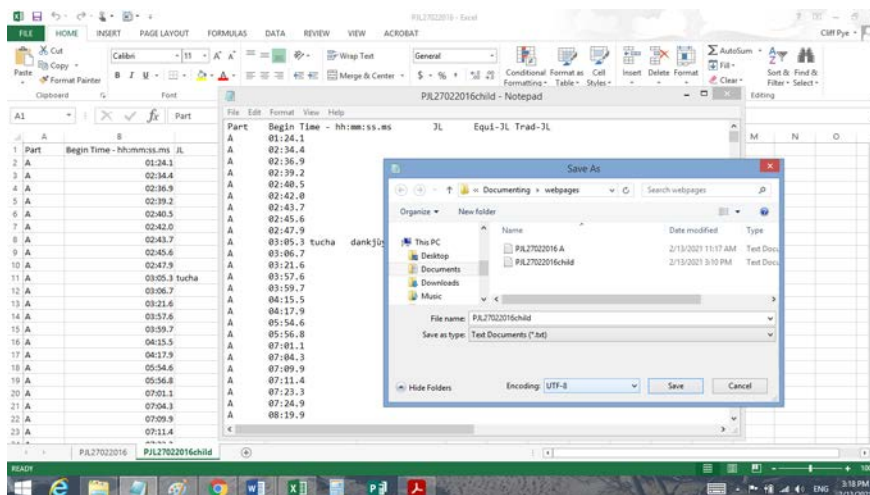
Perl Analysis Programs

It is easy to produce a lexical concordance with programming languages that are specialized for text analysis. Perl is an older one of these languages, while other programming languages like Python will also work. A computer program is a list of instructions that the program interpreter translates into computer instructions. In order to run Perl programs you must first download and install a Perl interpreter. The Strawberry Perl interpreter works on both PCs and Macs so it is a good choice. Strawberry Perl can be downloaded from the webpage <http://strawberryperl.com>.
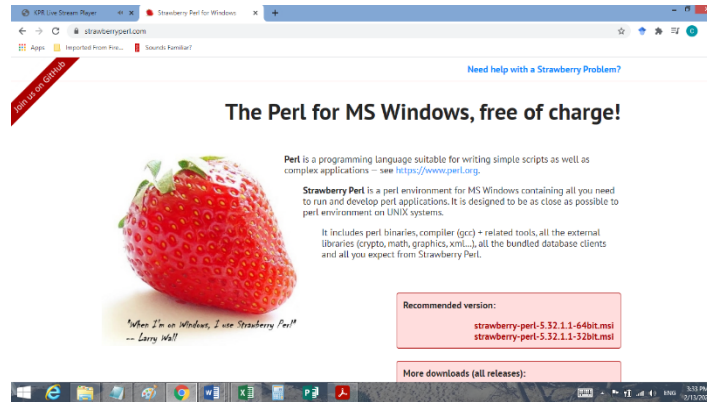
In order for computer programs to work, the programmer has to tell the program where to find the data file. The easiest way to handle this problem is to open a new folder and label it Perl Programs. This folder will serve as the place to use all of the analysis programs. You will also need to copy the child's UTF-8 file to this folder for the analysis. The lexical concordance program concord.pl can be downloaded from the webpage < http://pyersqr/minimal>.

With the concord.pl file and the data file, e.g. PJL27021016child saved in the Perl Programs folder, start the program by double clicking on the concord.pl file. A program window will open and ask you to type in the name of the data file, in this case PJL27021016child. Type in the filename and click the Enter key on the computer's keyboard.
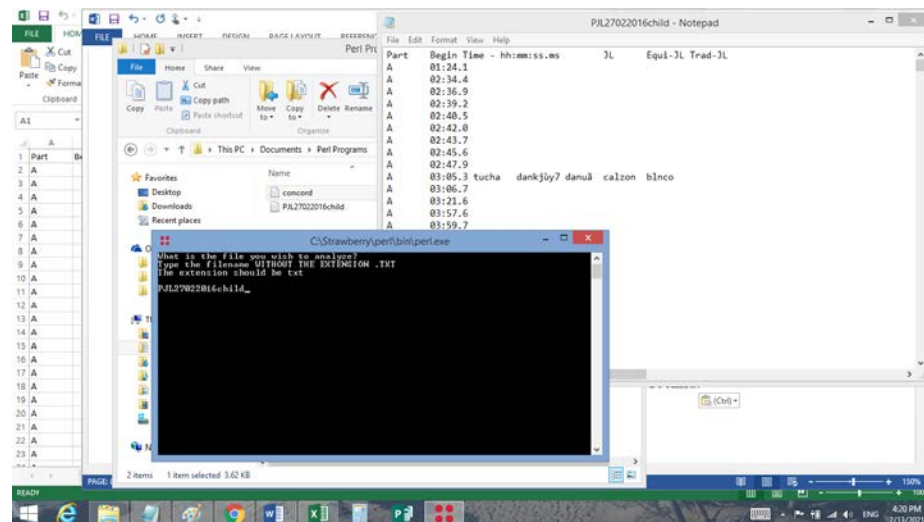
When the program runs successfully, the program window will close and the program will output the concordance file in the Perl Programs folder with name of the datafile, but with a new extension '.con'. The concordance file contains the results in tab-delimited columns that can be imported into the Excel worksheet. In order to preserve any special characters, open the concordance file in notepad and save it in the Unicode format.
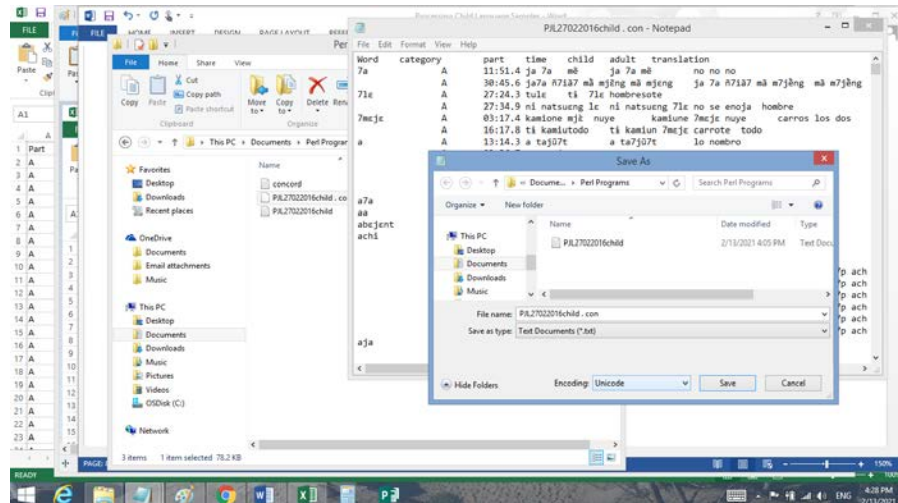
The lexical concordance is now ready to be imported into the Excel file. Open a new worksheet in the Excel file, rename it lexical, and click on the data tab in order to import the text file. Look for your Perl Programs folder and change the format at the bottom of the window from txt files to all files. You should then see the concordance file that the program produced. Click Import.
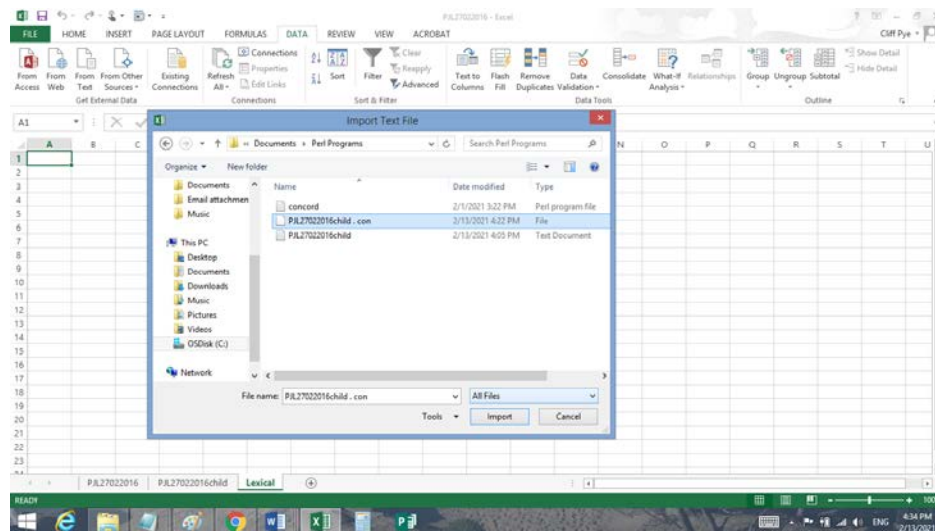
The Excel program will ask again if the file is delimited. Click the Finish tab, and the concordance will be imported to the new worksheet.

The concordance sheet contains 8 columns, in this case labeled: Adult Word, Child Word, category, part, time, child utterances, adult utterances and translation. The Adult Word column contains all of the words that the program found in the adult utterances column and shows all of the utterances that contained that word. In most cases, the child's line will contain the child's attempt at producing the adult word, but in some cases, the child might have omitted the adult word. The concordance program extracts a word from the child's utterances every time it extracts a word from the adult utterance. When there is not a one-to-one match between the number of words in child's utterances and the adult's utterances the program extracts a mismatch. Examples of these mismatches can be seen on rows 3 and 4 in the example. In row 3 the program matches the adult word 7a with the child's word ñ7iã7. This mismatch occurred because the adult utterance begins with two words—ja 7a that correspond to the single word ja7a in the child's utterance. Such mismatches could be corrected in the original transcription, but this would take time. It is much easier to fix such mismatches at the analysis stage because they are easier to spot.

The variation between the child's production tokens for each adult target contains information about the stability of the child's productions. This information can be used to address the research question of what factors affect the child's productions. Children produce some words more faithfully than others, which reflect the child's ability to produce specific sounds, syllables and words. Comparisons between different children will reveal which factors the children have in common and which productions are specific to individual children.

Selecting a Representative Phonetic Type

We assume that children store a single representation for each word in their lexicon. David Ingram (1989) published a method for selecting a phonetic token from the child's productions for each word as a representative phonetic type. The phonetic type is chosen on the basis of the features that it shares with the other tokens. In most cases, the phonetic type is the token that the child produce more frequently. If frequency does not yield a phonetic type, then the token that shares the most phonetic features is chosen as the phonetic type.

It is easy to use the lexical concordance to select the child's representative type for each word. The first step in this process, as in all subsequent analyses, is to copy the child's lexicon to a new worksheet. The first rule of analysis is to preserve a copy of each step in the process so that it will be possible to recover an earlier step in the process if a mistake is discovered in the analysis. Copy the child's lexicon to a new sheet and label it Types.

As well as identifying a representative phonetic type, it is also helpful to identify the word's part of speech. It will then be possible to how frequent each syntactic category appears in the child's utterances. I add an abbreviation in the category column that identifies the word's lexical category. I move the adult word and the category symbol to identify the representative phonetic type. I use the following list of category symbols, but others can be added as needed.

Lexical category symbols

A adjectives, adverbs   O onomatopoeia

C common nouns   P pronouns

D demonstratives   Q wh-words, yes, no

E exclamations   T transitive verbs

I intransitive verbs   X uninterpreted

N names

The following example shows the worksheet with the representative types and lexical category information. The selected cells in this example show that I moved the adult word 7lɛ to the fifth row in order to select the child's token lɛ as the representative type for this word. The lexical category for the word is a common noun so I added a C in the category column.

Once the representative phonetic types are selected, the custom sorting option in Excel can be used to group together the lexical types that belong to each lexical category. The sorting should be done by copying the sheet to a new worksheet and sorting the words in the new sheet as shown below. The selected cells in this example show the adult word 7lɛ 'man' is the first common noun in the worksheet.

The number of lexical types that belong to different lexical categories such as intransitive and transitive verbs, etc. are easily counted in Excel. Excel also makes it possible to rapidly calculate the percentage of lexical types in each category. For examples of research on children's lexical categories see Gentner & Boroditsky (2001); Pye, Berthiaume & Pfeiler (2020).

Vowel Analysis

The representative types provide the data for a basic phonological analysis following Ingram's method for analyzing child phonology (1989). I wrote the perl program vowel.pl to extract the vowels from a child's phonetic types as the preliminary step to analyzing the accuracy of the child's vowel production as well as the substitution patterns that occurred. As always, the first

step in a new analysis is to copy the child's lexical types to a new worksheet. This step is necessary when the analysis will not include all of the words that the child produced. The idea is to only include the child's words that correspond to identifiable adult words. This step involves deleting the child's uninterpretable words as well as such categories as exclamations and onomatopoeia form. The example shows the sheet with these categories omitted.

The vowel analysis program compares each adult word with the child's representative phonetic type. Once again, it is necessary to save this worksheet as a Unicode text document for the perl program. Open the Unicode text document in notepad and change its format to UTF-8. Copy this file to the Perl program folder and run the program vowel.pl by double clicking on the file. The program will produce a UTF-8 text file with all of the child's and adult words that contain each vowel.

The UTF-8 file needs to be converted back to the Unicode format before importing the results to excel. I added a new worksheet labeled vowels in Excel and imported the vowel data into this sheet.

As shown in this example, the vowel program outputs the results in three columns. The first column contains the target vowel, the second column contains the adult target word, while the third column shows the child's word. As shown in the first row, this is an analysis of the vowels that were in the child's words. In this analysis, the program found all of the child's words that had the target vowel. Note that the program outputs the child's word for each vowel so if the

word contains two or more instances of the same vowel, the word will be shown that number of times. In this example the child produced two words (tostosing, ogon) with two instances of /o/ so the program shows these words twice. This output allows the investigator to count how many phonetic types the child produced with each target vowel. This child produce /o/ 15 times in 13 phonetic types.

The program does not separate vowel strings in order to make it easier to separate monophthongs from diphthongs. The example shows the child's words with the vowel sequence [iɨ] at the top. The investigator can analyze such sequences as either successive monophthongs or diphthongs depending on the goal of the analysis.

The program outputs an analysis of the vowels that were in the adult target words separately from the analysis of the vowels in the child's words. There are often discrepancies between the vowels in adult words and the vowels that children produce in their own words. An analysis of the number of times the child produced the vowels in the adult words provides an indication of the accuracy of the child's vowel productions. A final determination of a child's vowel productivity will be based on the frequency of use and accuracy for each vowel.

Analyses of vowels in children's words are still rare. Examples of such analyses can be found in Bernhardt & Stemberger (1998), Fee (1991) and Levelt (1994).

References

Bernhardt, B. H., & Stemberger, J. P. (1998). *Handbook of phonological development: From the perspective of constraint-based nonlinear phonology*. New York: Academic Press.

Gentner, D. & Boroditsky, L. (2001). Individuation, relativity, and early word learning. In M. Bowerman & S. Levinson (Eds.), *Language acquisition and conceptual development* (pp. 215-256). Cambridge: Cambridge University Press.

Fee, E. J. (1991). Underspecification, parameters, and the acquisition of vowels. Unpublished doctoral dissertation: University of British Columbia.

Ingram, D. (1981). *Procedures for the analysis of children's language*. Baltimore, MD: University Park Press.

Ingram, D. (1989). *First language acquisition: Method, description and explanation*. Cambridge University Press.

Levelt, C. (1994). On the acquisition of Place. Doctoral dissertation. HIL Dissertations in Linguistics. Dordrecht: ICP Printing.

Pye, C. 2020. Documenting the acquisition of indigenous languages. *Journal of Child Language* 1-26. [https://doi.org/10.1017/S0305000920000318].

Pye, C., Berthiaume, S., & Pfeiler, B. (2020). Northern Pame-Spanish language acquisition in the context of incipient language loss. *International Journal of Bilingualism*, 24(2), 246–265.

Perl Programs

Concord.pl

```perl
#!/usr/local/bin/perl
#
# concord.pl
# Clifton Pye
# This program produces a lexical concordance for a unicode text file exported from Elan via
# Excel.
# The exported text file should have a header row and be saved in the UTF-8 format with the .txt
# extension
# The header row should be: "part    time    child    adult    translation"
# The program does not process blank lines


%words = ();
$extension = ".txt";
$concord = "con";
print "What is the file you wish to analyze? \n";
print "Type the filename WITHOUT THE EXTENSION .TXT \n";
print "The extension should be txt \n\n";


$textfile = <>;                          # Read the filename from the keyboard input
chop $textfile;                          # Remove line return
$file = $textfile . $extension;
print "Now analyzing $textfile\n\n";
open text_in, "< $file" or die( "Could not open $file" );
$line = <text_in>;                       # Read header row
chomp($line);                            # Remove line returns
@tiers = split ("\t", $line );           # Count the number of columns
```

```perl
open text_out, "> $textfile . $concord" or die( "Could not open the output file" );

if ( scalar( @tiers ) >= 5 ) {  #Process files with 5 column

  print text_out "Adult Word  Child Word    category       part    time    child utterances
        adult utterances        translation\n"; # Print header

  while ($line = <text_in>) {

    $line_no = $line_no + 1;            # Count the line number

    chomp($line);                       # Remove line returns

    @tiers = split ("\t", $line );          # Split line into tiers

        $child = $tiers[ 2 ];

          $child = lc($child);                      # Change to lower case

        $adult = $tiers[ 3 ];

          $adult = lc($adult);                      # Change to lower case

    $adult = ' ' . $adult . ' ';             # Add spaces

    $child =~ s/["(),;:.!?]/ /g;         # Remove punctuation

    $adult =~ s/["(),;:.!?]/ /g;         # Remove punctuation

    @chwords = split (" ", $child );             # Put words into an array

    @revchild = reverse( @chwords );

    @adwords = split (" ", $adult );             # Put words into an array

      foreach $adword (@adwords) {

        $chword = pop( @revchild );

        $words{$adword} = $words{$adword} . $chword . '              '. $tiers[ 0 ] . ' ' .
$tiers[ 1 ] . '    '. $tiers[ 2 ] . ' ' . $tiers[ 3 ] . ' ' . $tiers[ 4 ] . '#';          # Put lines in hash

      } #end foreach word

  } #end while

} #end if


else {  #Process files with 4 columns

  print text_out "Word category       part    time    adult   translation";     # Print header
```

```perl
  while ($line = <text_in>) {

    $line_no = $line_no + 1;            # Count the line number

    chomp($line);                       # Remove line returns

    @tiers = split ("\t", $line );       # Split line into tiers

        $adult = $tiers[ 1 ];

          $adult = lc($adult);                  # Change to lower case

    $adult = ' ' . $adult . ' ';             # Add spaces

    $adult =~ s/["(),;:.!?]/ /g;          # Remove punctuation

    @words = split (" ", $adult );          # Put words into an array

      foreach $word (@words) {

        $words{$word} = $words{$word} . $tiers[ 0 ] . '    ' . $tiers[ 1 ] . ' ' . $tiers[ 2 ]  . " .
$tiers[ 3 ] . '#';            # Put lines in hash

      } #end foreach word


  } #end while
} #end else
close text_in;


# Display the results
foreach $word (sort keys %words) {              # Sort the word hash
  @lines = split ("#", $words{$word} );         # Put lines into an array
  print text_out "$word";                       # Print word
    foreach $line (@lines) {
      print text_out "   $line\n";            # Print line
    } #end foreach line
} #end foreach word
```

Vowel.pl

```perl
#!/usr/local/bin/perl
# vowel.pl
# Created by Clifton Pye 2021


# This program compares the vowels in a child's tokens with their adult targets
# It analyzes data from a text file with two columns: adult   child


$extension = ".txt";
$vowel = ".pon";
print "What is the file you wish to analyze? \n";
print "Type the filename WITHOUT THE EXTENSION .TXT \n";
print "The extension should be txt \n\n";


# Read in the data file


$textfile = <>;                          # Read the filename from the keyboard input
chop $textfile;                          # Remove line return
$file = $textfile . $extension;
print "Now analyzing $textfile\n\n";
open text_in, "< $file" or die( "Could not open $file" );
$line = <text_in>;
open text_out, "> $textfile $vowel" or die( "Could not open the output file" );


# Read datafile
while ($line = <text_in>) {
  chop($line);
  @words = split ("\t", $line );         # Split line into words
```

```perl
  $adult = $words[0];

  $child = $words[1];

  $child =~ s/[bdgptkmnlrwyzsfxch]+/ /g;   # Delete consonants

  $adult =~ s/[bdgptkmnlrwyzsfxch]+/ /g;   # Delete consonants

  @vowel = split (" ", $child );          # Split vowels by spaces

  while ( @vowel ) {

    $vow = pop( @vowel );

    if ( $vow != 7 && $vow ne 'j' ) {

    $vow =~ s/^7//;   # Delete initial 7

    $vow =~ s/7$//;   # Delete final 7

    $vow =~ s/^j//;   # Delete initial j

    $vow =~ s/j$//;   # Delete final j

    $childphon{$vow} = $childphon{$vow} . $words[0] . "" . $words[1] . " ";   #add word to
sound hash

    }

  }

  @vowel = split (" ", $adult );          # Split vowels by spaces

  while ( @vowel ) {

    $vow = pop( @vowel );

    if ( $vow != 7 && $vow ne 'j' ) {

    $vow =~ s/^7//;   # Delete initial 7

    $vow =~ s/7$//;   # Delete final 7

    $vow =~ s/^j//;   # Delete initial j

    $vow =~ s/j$//;   # Delete final j

    $adultphon{$vow} = $adultphon{$vow} . $words[0] . "" . $words[1] . " ";   #add word to
sound hash

    }

  }

}
```

```
close text_in;

# Display the results
print text_out "Child's vowels Adult word     Child word\n";
@keys = keys( %childphon );
@sortkeys = sort @keys;
@reversed = reverse( @sortkeys );
while ( $key = pop( @reversed ) ) {
  @words1 = split( / /, $childphon{$key} );
  @sorted1 = sort @words1;
  print text_out "\n$key";
  for ($i = 0; $i <= $#sorted1; $i = $i + 1 ) {
    print text_out "     $sorted1[$i]\n";
  }
} #end child

print text_out "\nAdult's vowels      Adult word     Child word\n";
@keys = keys( %adultphon );
@sortkeys = sort @keys;
@reversed = reverse( @sortkeys );

while ( $key = pop( @reversed ) ) {
  @words1 = split( / /, $adultphon{$key} );
  @sorted1 = sort @words1;
  print text_out "\n$key";
  for ($i = 0; $i <= $#sorted1; $i = $i + 1 ) {
    print text_out "     $sorted1[$i]\n";
  }
```

} #end adult