Analysis Steps for Ultrasound Images of Tongue Curves

Vincent Chanethom & Sean Martin

updated: December 10, 2014

Contents

1 QUICKTIME 7: Extracting wave files from avi files **2**
1.1 Editing video files 2

2 PRAAT: Marking off relevant sound(s) **2**
2.1 Segmenting the relevant sound(s) 2
2.2 Running the praat script to extract boundary values 4
2.2.1 notes for MATLAB version 6

3 Python: Extract frames corresponding to relevant sound(s) **6**
3.1 Extracting frames 7
3.2 Extracting frames with ffmpeg alone 7
3.3 Extracting frames with MATLAB 7
3.3.1 Checking the content of your working folder 7
3.3.2 Extracting the frames 8

4 EDGETRAK: Tracking frames **8**
4.1 Loading the frames 8
4.2 Drawing data points 10
4.3 Saving files 12

5 SSANOVA: Statistical comparisons **12**
5.1 Building up the data file as a csv file 12
5.2 Running SSANOVA in R 14

6 Write up your findings! **17**

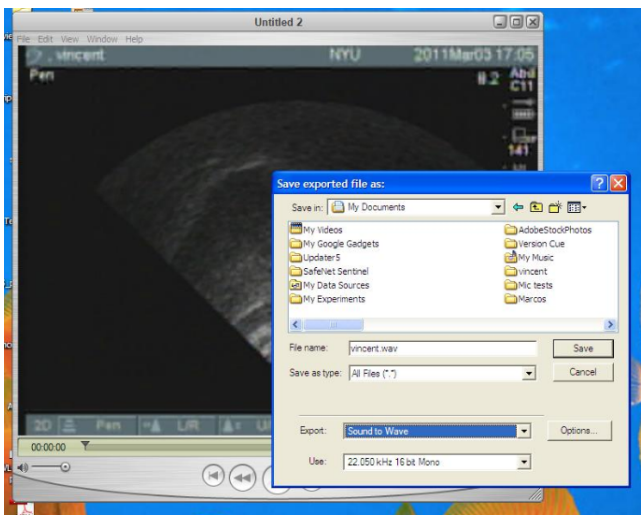
1 QUICKTIME 7: Extracting wave files from avi files

i. Open the avi file on QUICKTIME 7

ii. Select FILE >> EXPORT

iii. Set the following parameters (see picture below):

- Export: "Sound to Wave"
- Use: "16 bit Mono"



1.1 Editing video files

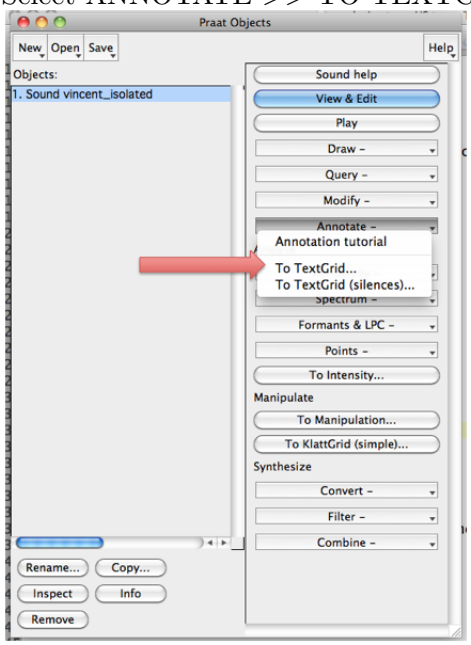
Due to issues with the way Quicktime encodes video files, if you use Quicktime to edit the video and then re-save, the Matlab step below will fail. If you need to edit, do so in Adobe Premier before exporting. Frame extraction using Python and FFMPEG will work normally for edited files but you should check the resolution of the resulting frames as occasional resolution issues have occurred.

2 PRAAT: Marking off relevant sound(s)

2.1 Segmenting the relevant sound(s)

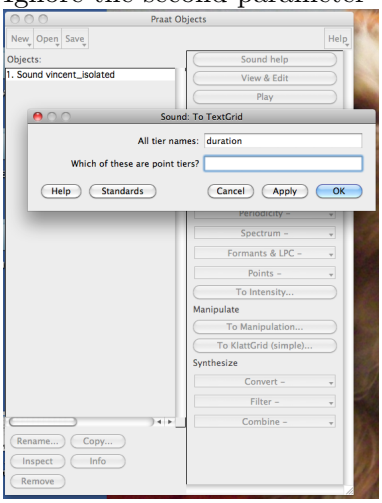
i. Open the extracted wave file on Praat

ii. Select ANNOTATE >> TO TEXTGRID



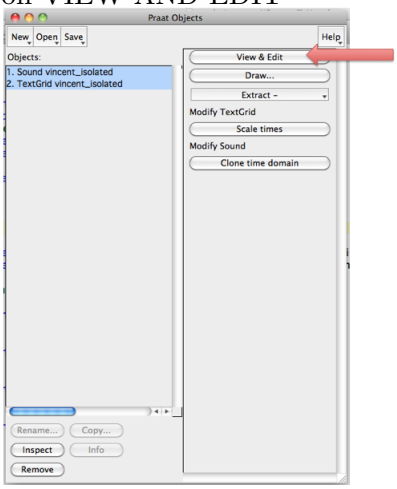
iii. Set the parameters as follows:

- All tier names: "duration" (for instance)
- Ignore the second parameter

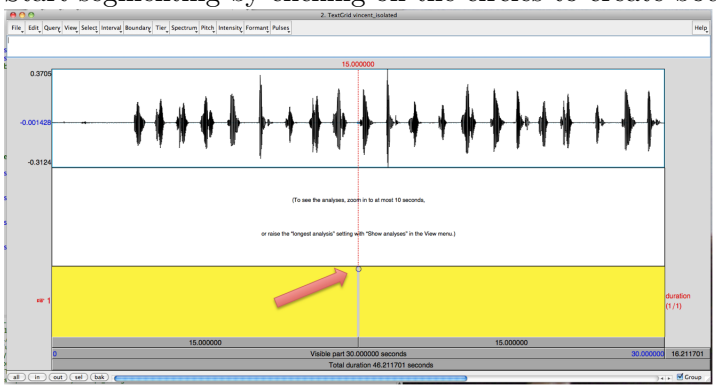


iv. A textgrid file is then created and added into the Praat object window

v. Highlight both the wave file and its corresponding textgrid, then click on VIEW AND EDIT



vi. Start segmenting by clicking on the circles to create boundaries



vii. Save all textgrids in the same folder as the wave files. Make sure the wave files and their corresponding textgrids have the same name (e.g. "vincent.wav" and "vincent.TextGrid")

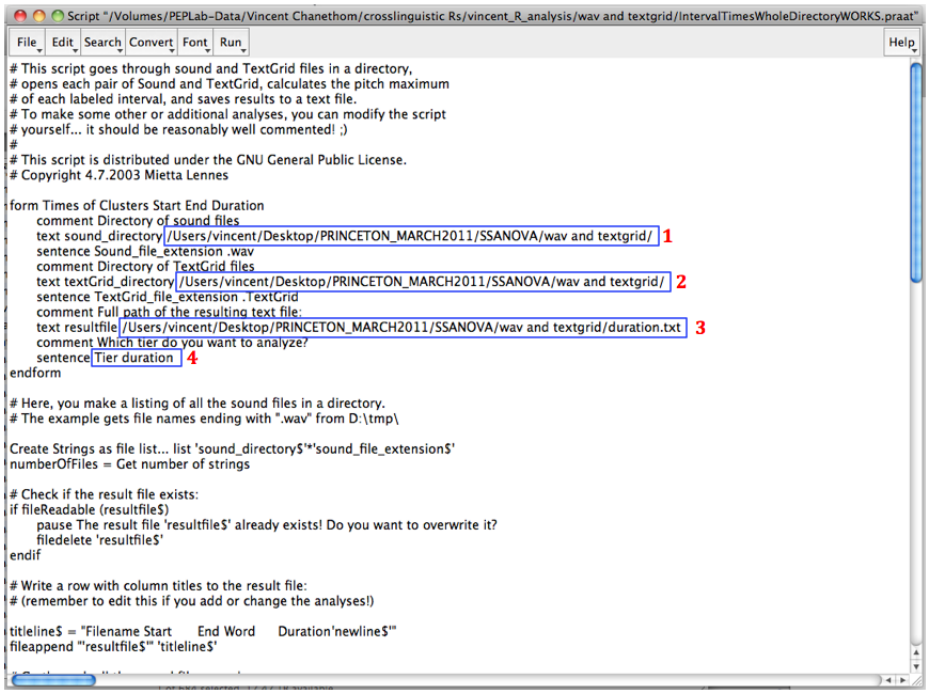
2.2 Running the praat script to extract boundary values

i. The praat script is "IntervalTimesWholeDirectory.praat" (available on the DROBO hardrive under /Ultrasound/Processing scripts/)

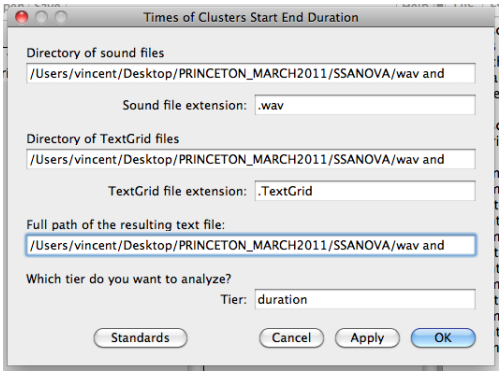
ii. Select PRAAT >> OPEN PRAAT SCRIPT

iii. Change the following parameters (numbers referring to the illustration below)

- #1 Specify the path to the folder where you saved your wave files
- #2 Specify the path to the folder where you saved their corresponding textgrids (this should be the same directory as for #1)
- #3 Specify the location where you want to have the result file (text file) created (usually the same directory as #1 and #2) and create a name for it
- #4 Specify the tier that will be analyzed by the script (we named the tier "duration," see above)



iv. Save the modifications and click on RUN. The following window will appear:



v. Check the parameters one more time then click OK

vi. Go to your folder, the result file should be in there

2.2.1 notes for MATLAB version

If you want to extract frames using MATLAB, you should instead use the script "IntervalTimesWholeDirectory_matlab.praat" (available on the DROBO hardrive in the matlab subfolder of /Processing scripts/) which will generate matlab-friendly formatting.

3 Python: Extract frames corresponding to relevant sound(s)

The following items should be in the same folder:

i. The video files

ii. The csv file generated by the Praat script

iii. The python script 'extractframes.py'

In addition, your computer should have the program ffmpeg installed and available from the command line interface [1].

[1] see http://www.wikihow.com/Install-FFmpeg-on-Windows for Windows-based instructions

3.1 Extracting frames

1. open a command line window (cmd on Windows, terminal for Mac/Linux)

2. change the working directory as appropriate (cd command for all computers)

3. run the following command:
   `python extractframes.py filename.csv target_directory`
   replacing filename.csv and target_directory with the name of the file generated by the Praat script and the directory you would like the extracted frames written to.

4. you only need to give the relative path. It is suggested that you make a folder in the directory with your video files and use the command
   `python extractframes.py filename.csv output`
   (here assuming you've called your output directory 'output')

3.2 Extracting frames with ffmpeg alone

If you only want to extract a single frame rather than processing a large batch of files, ffmpeg can also be used independently. Syntax for single frame extraction is as follows:
`ffmpeg -v 0 -i filename -ss time output_file.jpg`
where `time` is the time corresponding to the frame you want to extract.

Similarly, to extract multiple frames, use
`ffmpeg -v 0 -i filename -t duration -ss time output_file-%3d.jpg`
where `duration` is the length of of the interval beginning at `time` that you would like to extract.

3.3 Extracting frames with MATLAB

3.3.1 Checking the content of your working folder

The following items should be in the same folder:

i. The video files

ii. The data file "data.text" generated by the Praat script[2]

[2]The script is currently set up to look for this file name. You will need to rename the file or alter the script if it doesn't have this name.

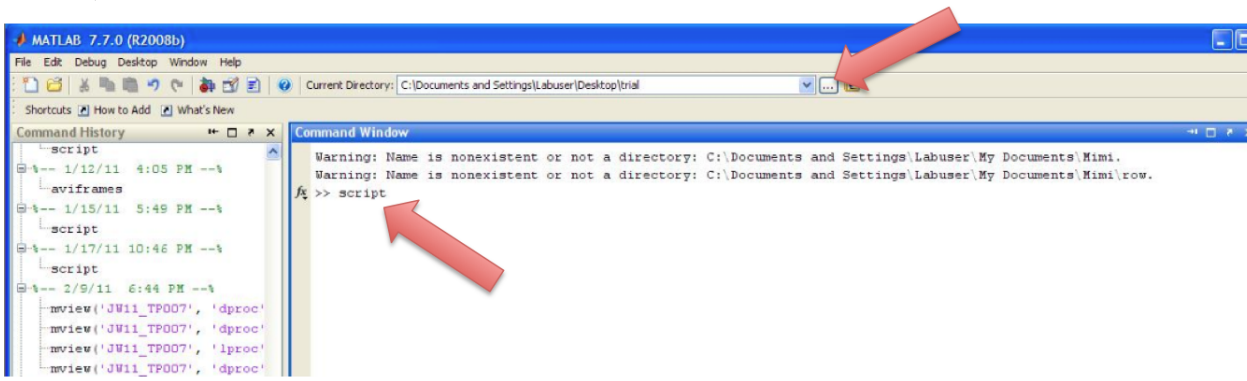
iii. The following 2 Matlab scripts[3]

- "extract_frames.m"
- "aviframes2.m"

3.3.2 Extracting the frames

i. Open Matlab

ii. Specify the working directory by specifying the path to the folder with all the files listed in (i) by clicking on the [...] button (see arrow on picture)



iii. Type "extract_frames" in the command window and press ENTER

iv. Check your folder for the JPEGs corresponding to the frames of the relevant sound(s)

4 EDGETRAK: Tracking frames

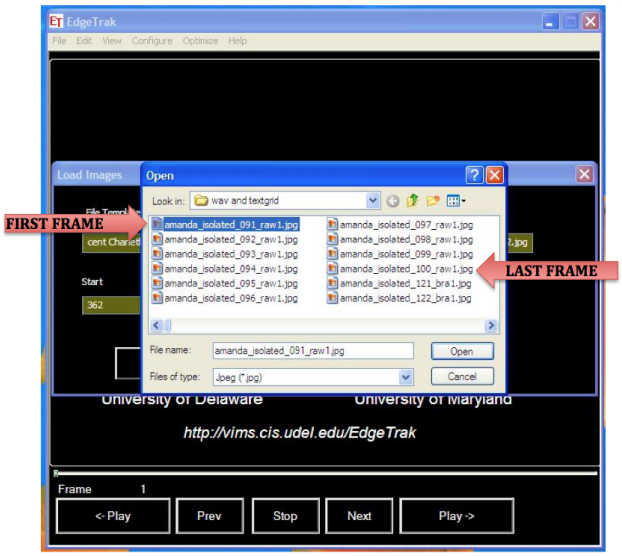
4.1 Loading the frames

i. Select FILE >> LOAD IMAGES

ii. Click on GET SAMPLE

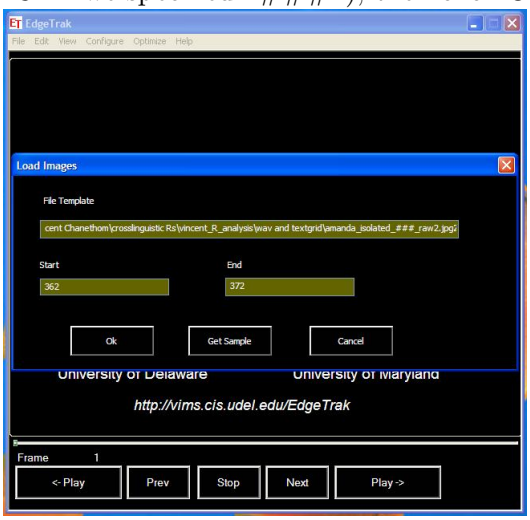
[3]The aviframes2 function was written for Matlab2014. It will work on the PEP lab computer but *may* have problems if you're using it with an older version of Matlab depending on how old your version is due to changes in how Matlab handles video files. Testing suggests that it also has issues with the NYU virtual computer lab.

iii. Select the first frame of the relevant sound, **BUT** memorize the numbers for the first and last frames (see illustration below). Click OPEN



iv. In the "File Template" box, replace the frame number with pound signs (#). e.g. if the first frame is 124 and the last frame 132, replace the "124" in the File Template" box by "1##" or even "###."

v. Type the first frame's number into the "Start" box and the last into the "End" box corresponding to the number of pound signs used in the "File Templat" box (e.g. 24 and 32 if we specified "1##" or 124 and 132 if we specified "###"), then click OK

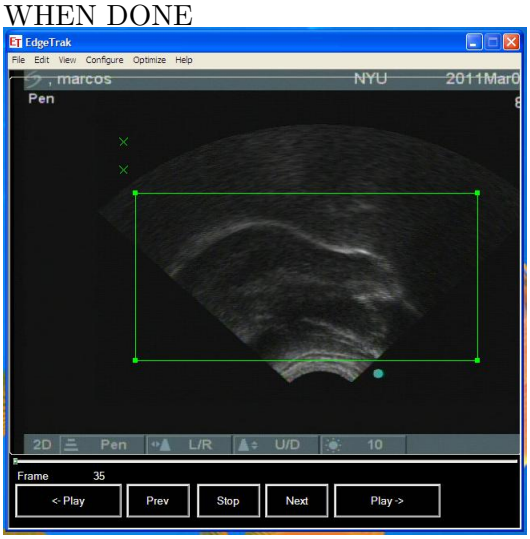


4.2 Drawing data points

i. Select CONFIGURE >> INTERESTING AREA AND SACLING MARKS

ii. Drag the green square onto the image and enlarge to frame the tongue curve (make sure the green box frames the whole tongue for the whole sequence of frames by clicking on "next")

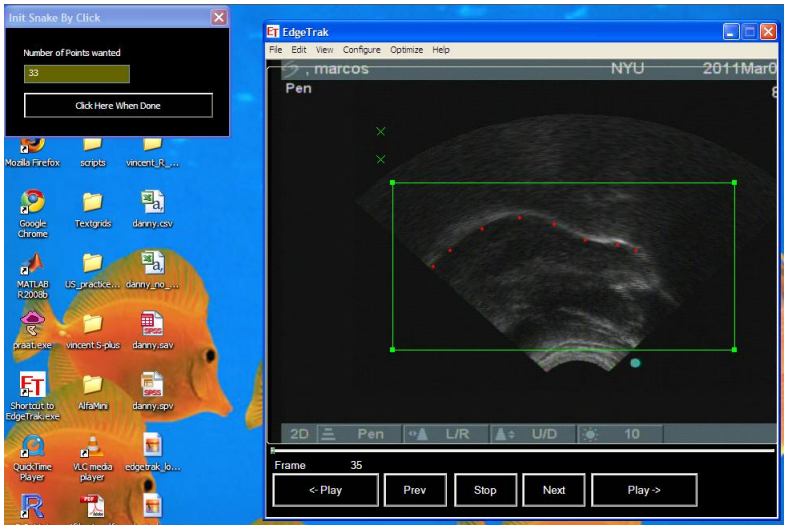
iii. click the black box ADJUST INTERESTING AREA. CLICK HERE WHEN DONE



iv. Choose the relevant frame for your study (e.g. the frame with the most retracted tongue position)

v. Select FILE >> INIT BY CLICK

vi. Add red dots along the tongue curve by clicking on the image (right click to erase a dot)

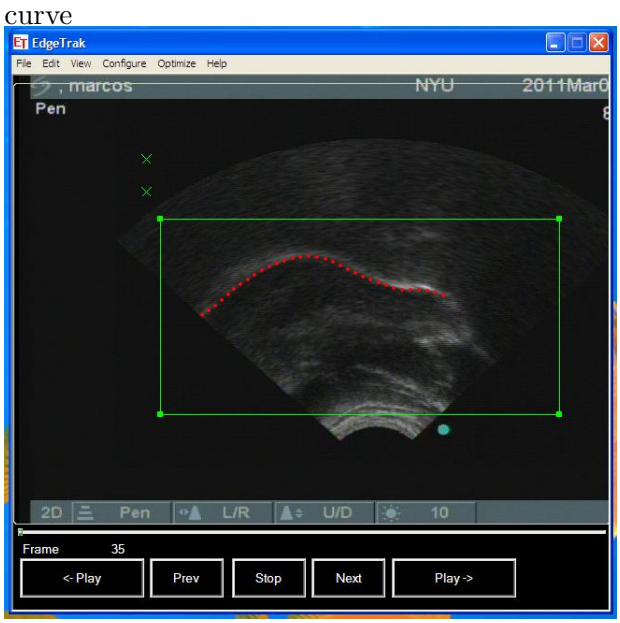


vii. Click the black box CLICK HERE WHEN DONE

viii. Select OPTIMIZE >> IMAGE GRADIENTS

- Click OK on the pop up window

ix. Select OPTIMIZE >> OPTIMIZE ONE FRAME (if only one frame needed) until satisfied with the alignment of the red dots with the tongue curve

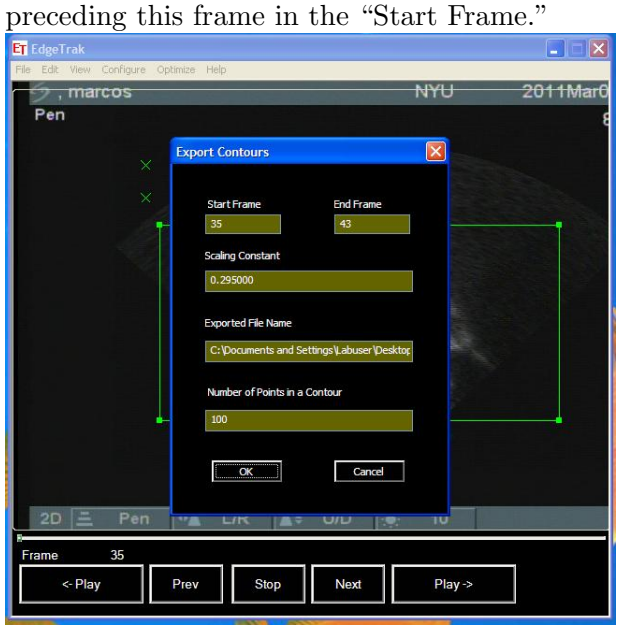


4.3 Saving files

i. Save as a ts file: FILE >> SAVE TS FILE

ii. Save as a con file: FILE >> EXPORT CONTOURS

- After naming the con file, a window will pop up
- Enter the number of the frame that you have selected in the "Start Frame" box and the number of the frame immediately following it (e.g. 35 and 36) in the "End Frame" box. Then, click OK
- If the last frame of the sequence was chosen, enter the number of this last frame into the "End Frame" box and the one immediately preceding this frame in the "Start Frame."



5 SSANOVA: Statistical comparisons

5.1 Building up the data file as a csv file

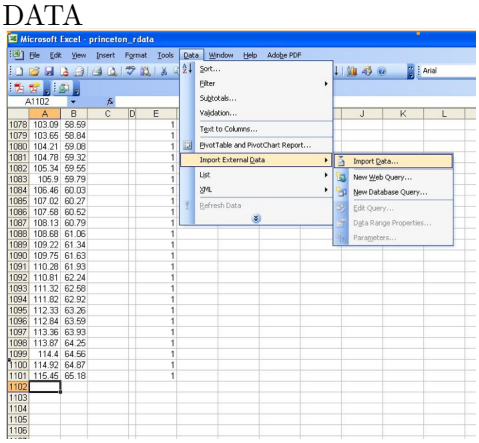
i. Open the first con file using Excel. Two columns of numbers will appear (x/y coordinates): x values and y values (in this order)

ii. Add a header row with labels relevant to your study.

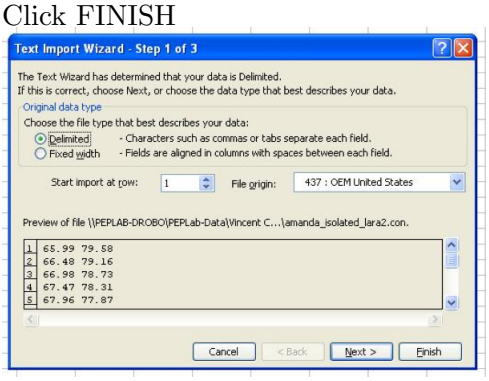
- e.g. / X / Y / WORD / REPETITION / VOICING / etc...

iii. Import the rest of the data

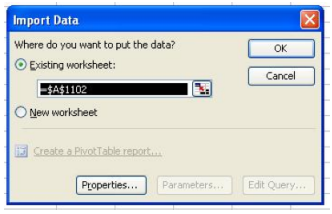
- Select DATA >> IMPORT EXTERNAL DATA >> IMPORT DATA



- Locate the next con file to upload (make sure "all files" is specified in the "File of Type" box
- In the popup window:
  - Select DELIMITED
  - Click FINISH



- A second popup window asks you where to put the data
  - Select EXISTING WORKSHEET
  - Make sure the relevant cell is highlighted on your spreadsheet
  - Click OK



- Fill in the rest of the information (WORD, REPETITION, VOICING, etc)
- Repeat these steps for the remainder of the con files

iv. Saving the document:

- select FILE >> SAVE AS >> and select file type "CSV"

v. NOTE: if the last frame for a relevant sound was selected during the EDGETRAK stage (see section 5.3[ii]), importing this file will misalign the x/y coordinate from the rest of the data:

- / ... / ... / X / Y / instead of / X / Y / ... / ... /
- Just rectify by cutting and pasting

5.2 Running SSANOVA in R

i. Put the following files in the same folder:

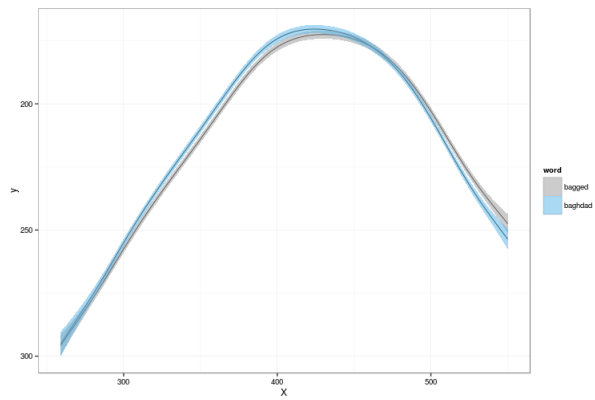
- The data file (csv file just created: e.g. "vincent_rdata.csv")
- The R files (available in the DROBO hardrive):
  - "ssanova.R" (detailed script that goes through stuff step-by-step)
  - "ssaplot.R" (script that loads an R function which performs comparisons

ii. Open R and follow these steps (using ssaplot.R):

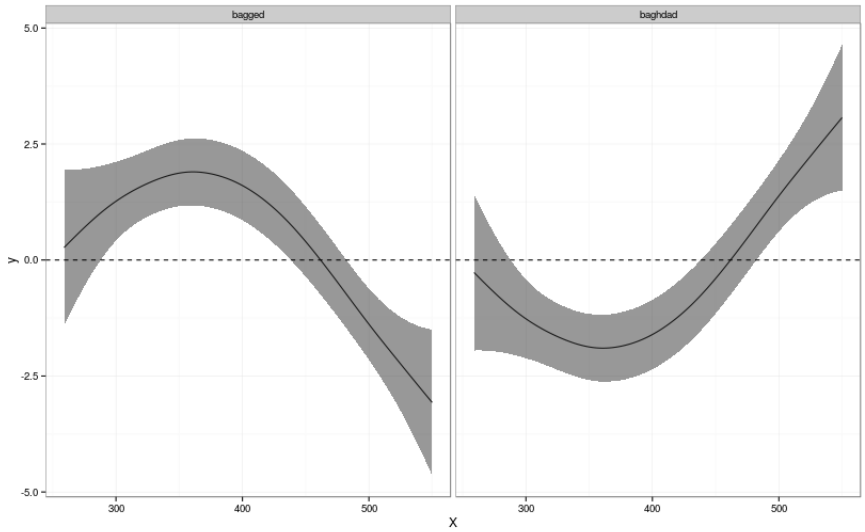
- Load the functions and the relevant R packages:
  - `install.packages(c("gss","ggplot2")`[4]
  - `source('/path/to/file/ssaplot.R')`

[4]The R functions need both of these. If you already have them installed you don't need to run this. If you get an error that says 'Error: could not find function "name"', you may need to install something.

- Load the data file:
  - Template: `data_name <- read.csv('/path/to/file/datafile name')`
    - `jasa_data <- read.csv('~/R/ssanova/sample_JASA2006_SSANOVAdata.csv')`
- plot comparison (in this example, the words "bagged" and "baghdad":
  - Template: `ssaplot(c("word1","word2"..."wordN"), data_name)`
    - `ssaplot(c("bagged", "baghdad"),jasa_data)`

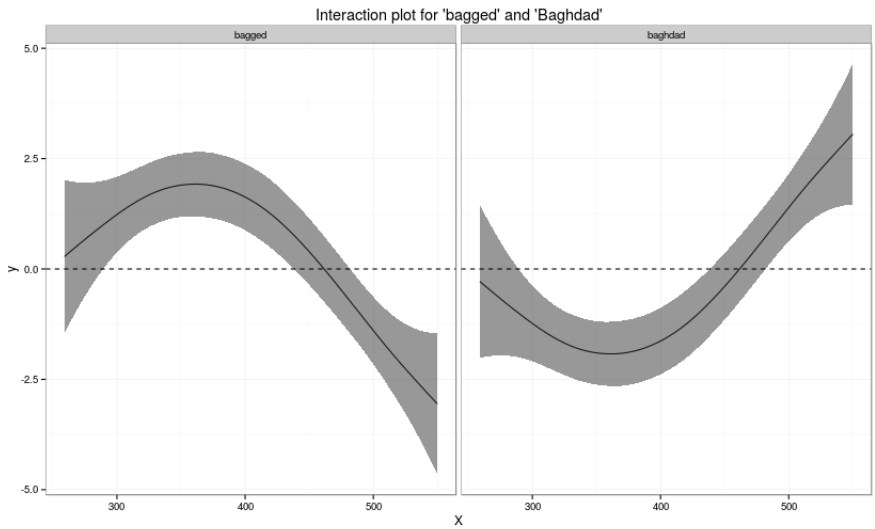


    - Here, we can determine which regions of the tongue shapes are significantly different by seeing whether the confidence intervals for the two curves overlap.
- Because it is not always easy to see where the two curves are different from the main effects comparison plot, it can be helpful to plot the interaction of the two words.
  - `ssaplot(c("bagged", "baghdad"),jasa_data, plot="interaction")`



  - In this case, we can determine whether the tongue shapes are significantly different at some point by seeing whether the confidence interval for the interaction includes 0.

- It might also be helpful to add a title to the plots you're generating (or otherwise alter them using the many options in ggplot. To do this, we first assign the plot to an object, then make modifications. For example, adding a title:
  - `inter <- ssaplot(c("bagged", "baghdad"),jasa_data, plot="interaction")`
  - `inter <- inter + ggtitle("Interaction plot for 'bagged' and 'Baghdad'")`



- ssaplot also takes the following options:
  - `color` with values `TRUE` (default) or `FALSE`. If `FALSE`, the plot will be greyscale.
  - `points` with values `TRUE` or `FALSE` (default). If `TRUE`, the plot will have individual data points from all repetitions of the word plotted.
- For more control over the plots (or if you're doing a more complicated analysis) see the file ssanova.R which goes through the analysis step by step and has comments describing what the steps are doing. The file ssaplot.R is just wrapping all of these steps into a function for convenience.

6 Write up your findings!