Lecture 7 (20 November 2012)

Assignment 13: read Hayward chapter 7.

You will be tested on your knowledge of spectrograms. You should be able to recognize when there is a stop, or a fricative, or a sonorant consonant, or voicing, and you should be able to identify the b/d/g difference, the a/i/u difference, and the s/ʃ/f/χ difference. If you cannot find all the fricatives in Hayward, produce them yourself in a single recording, and compare their spectrograms (don't have to send me this time).

Assignment 14: programming and the t-test

In assignment 10, you somehow managed to automate duration measurements, to compute basic statistics (average and standard deviation), and to put the measurements into a table. Some of you did not automate this whole process correctly, so here we are going to create a method that is the same for you all.

We will first explain a bit about programming.

Type the names of the eight (or seven) speakers in a script as follows (you can copy it from an earlier script):

```
speaker$ [1] = "F20N"
speaker$ [2] = "F28G"
speaker$ [3] = "F40L"
speaker$ [4] = "F60E"
speaker$ [5] = "M15R"
speaker$ [6] = "M40K"
speaker$ [7] = "M56H"
speaker$ [8] = "M66O"
```

You can now list the names of the speakers with the following *loop*:

```
echo List of speakers:
for i from 1 to 8
   name$ = speaker$ [i]
   printline Speaker 'i' is called 'name$'.
endfor
printline That was the list of speakers.
```

Please try to think what this script will do, then try it out. Answer the following questions:

14a. The loop is called a *for-loop*. The stuff between `for` and `endfor` will be done for all whole values of `i`, starting with 1 and ending with 8. In other words, `i` is first 1, then 2, and so on. What happens if you leave out the `endfor`? (To try this out without typing too much, you can temporarily put a `;` before the `endfor`) Please explain what you see.

14b. The `echo` seems to do the same as `printline`, but it does not. After running the above script, change `echo` into `printline` and run it again. What happens? The cause is that `echo` erases the Info window before writing the text, and `printline` does not. What happens if you replace the last `printline` with `echo`?

14c. There are two kinds of *variables*. There are *numerical variables* like `i`, in which you can put a number. Thus, the line `i = 345` followed by the line `echo 'i'` will write the number 345 into the Info window. There are also *string variables* like `speaker$[5]` and `name$`, in which you can put a text. Thus, the line `speaker$[5] = "M15R"` followed by the line `echo 'speaker$[5]'` will write the text M15R into the Info window. String variables are characterized by a dollar sign at the end. **Question:** why did I make `name$` in the above script a string variable, and why did I make `i` a numeric variable?

14d. The first time through the loop the variable `i` has the value 1, so that the line `name$ = speaker$ [i]` puts the value of the variable `speaker$[1]` (namely "F20N") into the variable `name$`.

The next line is `printline Speaker 'i' is called 'name$'`. The single quotes do *variable substitution*, i.e. they turn variables into texts. Tell me what this line becomes after variable substitution.

At this point, you should understand the above script. For more information, see the **Scripting Tutorial** and the **Formulas Tutorial**.
Now we return to the case of the vowel durations. In order to be able to compute the standard deviation between the eight speakers, one has to have computed and stored the mean for each speaker. Take the script of assignment 10c and add the following just after the line with `printline`, i.e. just before the `endfor`:

```
#
# Store the durations of this speaker.
#
ie [speaker] = ie
i [speaker] = i
aa [speaker] = aa
```

You hereby stored the durations into the 24 variables `ie1`, `ie2`, `ie3`, `ie4`, `ie5`, `ie6`, `ie7`, `ie8`, `i1`, `i2`, `i3`, `i4`, `i5`, `i6`, `i7`, `i8`, `aa1`, `aa2`, `aa3`, `aa4`, `aa5`, `aa6`, `aa7`, and `aa8`.

After modifying the loop like this, you can compute the average duration of each vowel, following assignment 2:

```
#
# Compute the mean durations.
#
sum_ie = 0
sum_i = 0
for speaker to numberOfSpeakers
   sum_ie += ie [speaker]
   sum_i += i [speaker]
endfor
mean_ie = sum_ie / numberOfSpeakers
mean_i = sum_i / numberOfSpeakers
```

Of course this only works if you assigned the correct value to the variable `numberOfSpeakers`, probably at the top of the script. And you have to add the stuff for *aa*.

After computing the means, you can compute the standard deviations, following assignment 4:

```
#
# Compute the standard deviations of the durations:
#
sum2_ie = 0
sum2_i = 0
for speaker to numberOfSpeakers
   sum2_ie += (ie [speaker] - mean_ie) ^ 2
   sum2_i += (i [speaker] - mean_i) ^ 2
endfor
stdev_ie = sqrt (sum2_ie / (numberOfSpeakers - 1))
stdev_i = sqrt (sum2_i / (numberOfSpeakers - 1))
```

You can then report the basic statistics:

```
printline ie: mean duration 'mean_ie:6' s, stdev 'stdev_ie:6' s.
printline i: mean duration 'mean_i:6' s, stdev 'stdev_i:6' s.
printline aa: mean duration 'mean_aa:6' s, stdev 'stdev_aa:6' s.
```

14e. What does the "6" mean?

In the above script, all the means and standard deviations have to be computed for the differences ie-a and aa-ie as well. Do this (perhaps analogously to the second script of assignment 8). You use lines like

```
   diff_ie_i [speaker] = ie - i
...
mean_diff_ie_i = sum_diff_ie_i / numberOfSpeakers
...
stdev_diff_ie_i = sqrt (sum2_diff_ie_i / (numberOfSpeakers - 1))
```

in the appropriate locations. After thus computing the basic statistics of the differences, we can compute the standard errors of these differences:

```
#
# Compute the standard errors of the differences of the durations:
#
stderror_diff_ie_i = stdev_diff_ie_i / sqrt (numberOfSpeakers)
stderror_diff_aa_ie = stdev_diff_aa_ie / sqrt (numberOfSpeakers)
```

So now we know how accurately the mean differences are known.

14f. Make your script report these two standard errors with an informative text.

You now compute the lower bound of the confidence interval of each difference:

```
#
# Compute the confidence intervals of the duration differences:
#
lowerBound_diff_ie_i = mean_diff_ie_i - stderror_diff_ie_i *
...  invStudentQ (0.025, numberOfSpeakers - 1)
```

(the three dots are a way to make long lines in a Praat script)

14g. Also compute the upper bound of each difference, and make your script report the confidence intervals with the usual formulation.

We now compute the *t*-values for each difference:

```
#
# Compute Student's t:
#
t_diff_ie_i = mean_diff_ie_i / stderror_diff_ie_i
```

14h. Modify your script so that it reports *t*.

Finally, we can compute the *p* value for the reliability that each difference is not zero. Here is the formula for the two-tailed test:

```
#
# Compute the significance from zero:
#
p_diff_ie_i = 2 * studentQ (t_diff_ie_i, numberOfSpeakers - 1)
```

14i. Make your script report the two *p* values.

14j. Compare the results with those of the table (assignment 12). They should be exactly identical.