**Eye-tracking Wiki**

**Contents**

1. [Building an experiment](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Building_an_experiment)
   1. [EyeTrack for Windows](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#EyeTrack_for_Windows)
      1. [Create item set and experiment script](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Create_item_set_and_experiment_script)
         1. [Building the item file for EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Building_the_item_file_for_EyeTrack)
            1. [Creating the text file of items](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Creating_the_text_file_of_items)
            2. [Using the eyetrack\_reading.py script](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Using_the_eyetrack_reading.py_script)
            3. [Using randomizer.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Using_randomizer.py)
   2. [EyeTrack for DOS](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#EyeTrack_for_DOS)
   3. [E-Prime](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#E-Prime)
2. [Running the Experiment](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Running_the_Experiment)
   1. [Starting-up the EyeLink host computer](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Starting-up_the_EyeLink_host_computer)
   2. [Initializing EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Initializing_EyeTrack)
   3. [Calibrating the participant](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Calibrating_the_participant)
3. [The Data Stream](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#The_Data_Stream)
4. [Analysis Preparation](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Analysis_Preparation)
   1. [Organizing Your DA1 Files](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Organizing_Your_DA1_Files)
   2. [Marking Regions of Interest (ROIs)](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Marking_Regions_of_Interest_.28ROIs.29)
   3. [Processing Data into R](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Processing_Data_into_R)
5. [Library of Tools](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Library_of_Tools)

This page describes how to build and run an eye-tracking experiment, and analyze the resulting fixation data, using the Eyelink 1000 head-supported eyetracker in the University of Maryland Linguistics Department.

For older (c. 2009) procedures, see [Eyelink: Deprecated Procedures](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink-old).

(See the [Library of Tools](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Library_of_Tools) section below for more resources)

# **Building an experiment**

## EyeTrack for Windows

### Create item set and experiment script

The item format the Eyetrack requires for stimulus presentation is not a very convenient format for when you're designing items. So the better thing to do is to use a script like the 'eyetrack\_reading.py' linked at the bottom of the page to convert to eyetrack format. Check out the comments at the top of that script to figure out how to format your items as you are creating them to make it easy to feed into this Python script. (NOTE: could someone upload an example of a items file in the correct format?) One thing to consider when you start planning your experiment is the length of your items and whether they will be long enough to need two lines. (NOTE: could someone explain here how to mark line breaks so that 'eyetrack\_reading.py' will read it correctly)

#### **Building the items file for EyeTrack**

This section details the use of two scripts used in building [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack) experiments. For a full list of available scripts for download, as well as a few powerpoint tutorials, see the [Library of Tools](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Library_of_Tools) section below. Here are the steps for creating your item file:

1. Create a text file in a particular format.
2. Feed the text file through the eyetrack\_reading.py script to get a .script file that [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack) can read.
3. For each subject, run the randomizer.py script.

##### **Creating the text file of items**

The text file is just a list of all the items in all of their conditions. You can have multiple experiments in the script; they're in separate sections with a title. See the example for reference on the format. Points to note:

1. If you have more than one experiment, all of them must have the same number of conditions. E.g., if you have one experiment with 2 conditions (Experiment A) and one with 3 conditions (Experiment B), you’ll have to make them both 6 conditions by having 3 copies of each item in Experiment A and 2 copies of each item in Experiment B.
2. Condition labels must be one character long.
3. When formatting the text file, do NOT include line breaks within items (or questions). Line breaks that you want to be presented should be marked with "\n". (Again, see example.)

*Example*:

Icestorm  
  
1 a The housemaids encountered the lord in the local park. \nThey laughed at his jokes on the way home.  
? Were the housemaids crying on their way home? N  
  
1 b The housemaids encountered the serf in the local park. \nThey laughed at his jokes on the way home.  
? Were the housemaids crying on their way home? N  
  
  
2 a The girls asked the lord for the latest gossip. \nThey gasped at his sordid stories about the aristocracy.  
? Were the girls curious about the latest gossip? Y  
  
2 b The girls asked the serf for the latest gossip. \nThey gasped at his sordid stories about the aristocracy.  
? Were the girls curious about the latest gossip? Y  
  
...

see [samplelist.txt](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=upload_form&ticket=005034fe2b.f985b9dfd95b8b5bb524d0a507cab15aa6645794&target=samplelist.txt)

##### **Using the eyetrack\_reading.py script**

1. First, make sure that your .txt file and a copy of [eyetrack\_reading.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=eyetrack_reading.py) are in the same directory.
2. Open the command prompt and navigate to the directory containing the scripts, and the stimulus text file.
3. Run the script by typing python eyetrack\_reading.py yourfilename.txt.
4. Note that the file "output.script" has been generated in the same folder. Any existing file by the same name will have been overwritten. Rename the output.script file to reflect your study name.

##### **Using randomizer.py**

1. Make sure that your .script file is in the same directory with a copy of [randomizer.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=randomizer.py)
2. Open the command prompt and navigate to the directory. Run the randomizer by typing python randomizer.py studyname.script
3. Respond to the prompts to indicate subject number [n] and verify the counterbalancing condition. This info will be saved in the header of the output.
4. Note that the file "order\_subj[n].script" has been generated.

# **Running the Experiment**

## Starting-up the EyeLink host computer

1. Turn on the [EyeLink](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeLink) host computer
2. Select the DOS option (as opposed to Windows)
3. In the command prompt, type "t" to start [EyeLink](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeLink)

[Note on **Low Disk Space** Error] Since a copy of all the data collected are saved on the DOS partition of the hard drive, you might see a red warning message when the disk space is running low. Don't panic. It can easily be fixed by restarting the computer in Windows (C:/) and moving the .EDF files out of the Eyelink partition (F:/) to the C: drive. You can then restart the computer in DOS and you should no longer see that warning message. This problem is also [discussed](https://web.archive.org/web/20120822154339/https://www.sr-support.com/forums/showthread.php?t=1447&highlight=low+disk+space) in the [EyeLink](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeLink) forum.

## Initializing EyeTrack

1. Open the [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack) program
2. Load the experiment: File -> Open -> "studyname.script"
3. Check for bugs: Experiment -> Validate
4. Run the experiment: Experiment -> Run
5. Choose "Separate File" in the Initialize Experiment window, and browse to "order\_subj[n].script"\*
6. Choose a name for your output file. The file name should not be more than 7 characters or the [EyeLink](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeLink) host computer will fail to save it. We generally name files based on subject number (e.g. 001, 002...).

\*If you're not using the python script to randomize your items, you can have [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack) do it for you. (This is not recommended.) There are four options. Randomize and interleave works the best--this gives you a different order for each subject, and actively works to separate experimental items with fillers so that they don't accidentally appear one after the other. The 'condition number' that you enter should really be the list number. (e.g. if you have 4 conditions per experiment, Latin Square will give you 4 lists of the items and conditions crossed in all possible combinations.

## Calibrating the participant

* Make sure to instruct the subjects to lean their forehead against the head rest first, and then rest their chin on the chinrest - if they only use the chin rest without the head rest, the subject's head can still move and hence the data get noisy.
* It's FINE to re-calibrate! Better to recalibrate than to have crappy data and frustrate the subjects (w/ a bad calibration, they'll have harder time starting trials).
* If it's hard to calibrate, try changing a) angle of the mirror, b) position of the chin (that way, the angle of the eye to the mirror changes), c) height of the chin rest, chair, etc. The machine can track the left eye too, but it seems that in previous eye-tracking studies, people generally only track the right eye.
* Subjects don't have a sense of when the experiment is going to end, and this can be really exhausting/frustrating. Calibrations gradually start to go off usually around after 50 sentences anyway (half the experiment), so feel free to take a break around there, let the subjects sit back and relax a bit, and then re-calibrate.
* If the subject has trouble finding the box, tell them to look towards a corner of the fixation box - that will give you a sense of in what direction the calibration is off. If it seems way off and that they actually need to look at a blank space around the box, then it'd be best to just recalibrate. Of course, calibration just doesn't always work well with everybody, but ideally subjects shouldn't have hard time fixating on the box to start the sentence presentation.

# **The Data Stream**

After data collection there is an unfortunately long intermediary step of preparing your data before you can run any analysis. Below you will find an outline of the data streaming process. For a full explanation of artifact rejection and converting .DA1 files into an R-ready format, see Artifact Rejection in [EyeDoctor](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeDoctor), and [Analysis Preparation](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#AnalysisPreparation) respectively:

* The [EyeLink](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeLink) host computer exports a file in the format .EDF to the display computer. This file format is analyzable in the SR research [DataViewer](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/DataViewer) software, which is located on the display computer. Generally, however, we do not use this software.
* Convert the .EDF file into a .asc file using the “Visual EDF2ASC” program located on the display computer.
* Import the .asc file into [EyeDoctor](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeDoctor) (a free program downloadable from UMass [here](https://web.archive.org/web/20120822154339/http://www.psych.umass.edu/eyelab/software/))
  + Reject artifacts in [EyeDoctor](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeDoctor) and export a .DA1 file
  + Close [EyeDoctor](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeDoctor) and save the experiment as a .EDD file
* Run your .DA1 files through the python scripts described in “Analysis Preparation” section to consolidate your data into a .txt file analyzable in R.

**Artifact Rejection**

Artifact rejection is the process of examining eye movement data to remove extraneous fixations. The basic idea is that we’re modifying the data file to reflect reading time, rather than say, time spent resting while looking in the corner of the screen. Artifact rejection is also used to account for errors in calibration, allowing us to vertically shift eye movement data so that it falls on the text. The following are meant as guidelines for when to reject fixations, for a detailed explanation of artifact rejection, as well as importing .asc files and exporting .DA1 files, see the wiki page on [EyeDoctor](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeDoctor).

1. Reject/correct artifacts on sentences only--don't bother with the questions. Use the following guidelines to make your rejections/corrections. The overall spirit of the guidelines is to do relatively little to change the data. In order to eliminate a lot of subjective choices in this process, we have to accept a little more noise in the data.
   1. Check to see if there is an overall vertical shift. To correct it, move chunks of fixations, NOT individual fixations. The smallest “chunk” you should use is half a line.
   2. Check for any blinks (marked with a circle), and reject those that occur within the vertical boundaries.
   3. DON’T bother rejecting fixations that fall outside the boundaries. They’ll be dealt with in our data processing script.
   4. DON’T reject any individual fixations that “don’t make sense” (e.g., because of poorly-planned line-change saccades, or recovering from a blink). This is just noise that we’re not going to worry about.

# **Analysis Preparation**

This section outlines the process of merging DA1 files into a single .txt file analyzable in R. There are several scripts available to help with this process, the most recent incarnations of which come from Shevaun Lewis. Downloadable copies of these scripts may be found in the [Library of Tools](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker#Library_of_Tools) section. We have found it preferable to use these rather than the [EyeDry](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeDry) software provided by UMass.

To start, create three folders: *Organize DA1*, *Make Regions*, and *Processing Data*. These folders correspond roughly to the steps you need to get data analyzable in R. Below is an outline of how these scripts are implemented, followed by more detailed descriptions of their use.

* Organize DA1
  + In this folder you should have a folder containing all your DA1 files, and the *organizeDA1.py* and *splitDA1.py* scripts.
  + These two scripts split DA1 files into sentence and question files, and/or into component sub-experiments. After you have run them, move your sorted DA1 files to the Processing Data folder.
* Make Regions
  + In this folder you should have the .script file you used for your experiment, the getQuestions.py script, the getSentences.py script, and the makeRegions.py script.
  + Run the getQuestions.py script to generate the question/answer key for your experiment. Move this file to the “Processing Data” folder
  + Run the getSentences.py script to generate a list of items from your .script file
    - Delimit your regions by inserting / into the item list file generated by getSentences.py. Keep this file in the “Make Regions” folder.
  + Run makeRegions.py to generate a .reg file containing the (x,y) coordinates of the regions from your delimited items list. Move the .reg file to the “Processing Data” folder.
* Processing Data
  + In this folder you should have a folder containing the sentence DA1 files, a folder containing the question DA1 files, the question/answer key and reg file generated in the “Make Regions” step, and the eyeDataToR.py, eyeMeasures.py and readInput.py scripts.
  + Run the eyeDataToR.py script to merge all of your sentence DA1 files into a single data file
  + Read the output of eyeDataToR.py into R and analyze!

## Organizing Your DA1 Files

Once you have finished the process of artifact rejection, you should have a list of DA1 files (labeled ###-studyname.DA1) which contain both sentence and question data. This step of the analysis preparation is designed to split those DA1 files into sentence DA1 files, question DA1 files, and “reject” DA1 files. Reject DA1 files occur when [EyeDoctor](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeDoctor) (for reasons we haven’t yet determined) records extraordinarily long fixation times. This only occurs very rarely, but it’s useful to separate these trials out from the rest of the data.

To split your data files, you’ll need to have in the same directory the organizeDA1.py and splitDA1.py scripts, along with a folder containing your DA1 files. Remember, your DA1 files should all be named ###-studyName.

**Splitting Sentences from Questions and Extracting Sub-experiments**

1. Open the command line and navigate to the directory containing the scripts and your DA1 files.
2. Run organizeDA1.py. When starting up, the script should first ask you whether you want to split sentence trials from question files (and separate out reject files).
3. If you answered "y":
   1. you will be asked to input the full path of the directory containing your DA1 files and the name of your study.
   2. When the script is finished processing the files, your original DA1 folder will contain three sub-folders with question, sentence and reject DA1 files respectively.
   3. You will be asked if you have any sub experiments. If you answer "y", you will be prompted for the conditions corresponding to your sub-experiment and the experiment name. If you answer "n", the script will terminate.
4. If you answered "n":
   1. you will be asked if you have any sub-experiments.
   2. If you answer "y":
      1. You will be prompted to input the path of the directory containing your separated DA1 files, the experiment name and the condition numbers corresponding to your experiment (\*).
   3. If you answer "n", The script will terminate.

(\*)

- When splitting your study into separate experiments, the script assumes that you have already separated the study into sentence, question, and reject files, and expects these files to be contained inside three separate folders which are themselves located in a common directory. In other words, a structure like this:

* studyName-sorted
  + studyName-sentences
  + studyName-questions
  + studyName-reject

You should input the directory path for the studyName-sorted folder. Furthermore, your files should be named along the lines of ###-studyName-s.DA1.

- When splitting your study into separate experiments, the experiment name corresponds to the name of your experiment as given in the items file that you provided to EyeTrack. It is common practice in our lab to run several experiments together in the same session, as items from one experiment can function as fillers for another. In these cases, each experiment is identified by a unique name in the items file. In the example given above, the experiment name is “Icestorm”.

- When splitting your study into separate experiments, the organizeDA1.py script asks for condition information by requesting (i) the number of the first condition and (ii) the total number of conditions. Providing the total number of conditions in your experiment should be straightforward if you know the design of your study. Providing the number of the first condition, however, is not. Although conditions in the items file may be named using letters, Eyetrack converts them to numbers. For example, consider a hypothetical items file that contains three experiments, the first (Icestorm) with 2 conditions, the second (Paprika) with 3 and the third (Hurricane) with 4. The condition numbers for Icestorm will be [1,2], [3,4,5] for Paprika and [6,7,8,9] for Hurricane. Therefore, an experimenter wishing to provide the number of the first condition for Icestorm should enter 1, but he should enter 6 for Hurricane.

## Marking Regions of Interest (ROIs)

This step in the process requires that you have your [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack) .script file, and the getQuestions.py, getSentences.py and makeRegions.py script in the same directory. You will use these materials to generate the question/answer key for your data, as well as the regions file used in compiling your data.

**Generating a Question Key**

From the command line, run the getQuestions.py script, using your study.script file as an argument. You will be prompted to input the number of the first item in your experiment(\*), the total number of items, the first condition and the total number of conditions. This should output two files called “allquestions.txt” and "expquestions.txt". The former contains all the questions for every item in the study, and the latter contains only the questions for the experiment conditions you input. If you open these files, you should see three columns.

* The first column is the item number (according to [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack)) of the question.
* The second column contains the number representing the correct answer
* The third column contains the label for the correct button participants were supposed to push. Once you have checked this, move this question key into the folder containing your processing scripts and DA1 files.

(\*) Due to an unresolved bug, Eye Track sometimes re-numbers the experimental items such that the first item is assigned 2. Double-check your experiment.script file, and if that was the case, you should input “2” as the item number for your first item when you run the getQuestions.py and getSentences.py scripts.

**Generating an Items List**

Use getSentences.py to extract the items that [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack) used (with the condition and item numbers it assigned).

1. Make sure you have a copy of getSentences.py in the same directory with the .script file you run in [EyeTrack](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/EyeTrack).
2. Open the command prompt and naviage to that directory. Type python getSentences.py studyname.script
3. Enter the starting condition number for the experiment you're working on within the study, and the total number of conditions in that experiment.
4. You’ll get two .txt files as the output (allsentences.txt with all items from all experiments, expsenteces.txt will the items of the experiment you intended to extract).

**Making a .DEL file**

A .del file is a file delimiting the regions of interest for each sentence. You need to make this by hand, because you're the one who has to decide which parts of the sentence are interesting for your experiment. Once you have your sentence list generated from getSentences.py, add forward slashes to delimit the regions.

* There should be a space at the beginning of each region, if possible. (Obviously this won't be possible for the first region of the item.)
* Include a slash at the end of the last region. (but make sure it does not include any unnecessary line change)
* Save the text file as studyname.del
* The resulting file should look like this:

1 2 /The housemaids/ encountered the/ serf/ in the/ local park. \n/They laughed /at his/ jokes/ on the/ way home./  
2 2 /The housemaids/ encountered the/ lord/ in the/ local park. \n/They laughed /at his/ jokes/ on the/ way home./  
3 2 /The housemaids/ encountered/ Bert/ in the/ local park. \n/They laughed /at his/ jokes/ on the/ way home./  
4 2 /The housemaids/ encountered/ Eric/ in the/ local park. \n/They laughed /at his/ jokes/ on the/ way home./  
1 3 /The girls/ asked the/ serf/ for the/ latest gossip. \n/They gasped /at his/ sordid/ stories/ about the aristocracy./  
2 3 /The girls/ asked the/ lord/ for the/ latest gossip. \n/They gasped /at his/ sordid/ stories/ about the aristocracy./  
...

Importantly, you need to be careful about the carriage return encoding, at least if you use a Mac. The python script might not read the .del file correctly if it has Classic Mac CR encoding, and it will give you a .reg file that is incorrectly all on one line. It is easy to check the encoding of the .del file if you use the TextWrangler text editor. Just open the file there and look at the very bottom of the window at the fifth tab from the left. It should read 'Unix LF'. If it reads 'Classic Mac CR' you need to toggle to the 'Unix LF' option and re-save the file.

**Making a. REG file**

A .reg file (formally known as a .CNT file) is a file that contains the x-y coordinates of the regions of interest for each item in each condition. You can create this with the script makeRegions.py.

1. Make sure your .del file is in the same as the copy of make\_reg\_file.py
2. Open the command prompt, and navigate to that directory.
3. To run the script, type python makeRegions.py studyname.del
4. The output will be a file called regions.reg.txt in the same directory. Rename it with your study name: studyname.reg.txt. Move the file to the “Processing Data” folder.

The output file will look like this:

1 2 9 0 0 14 0 30 0 35 0 42 0 0 1 13 1 19 1 25 1 32 1  
2 2 9 0 0 14 0 30 0 35 0 42 0 0 1 13 1 19 1 25 1 32 1  
3 2 9 0 0 14 0 26 0 31 0 38 0 0 1 13 1 19 1 25 1 32 1  
4 2 9 0 0 14 0 26 0 31 0 38 0 0 1 13 1 19 1 25 1 32 1  
1 3 9 0 0 9 0 19 0 24 0 32 0 0 1 12 1 18 1 25 1 33 1  
2 3 9 0 0 9 0 19 0 24 0 32 0 0 1 12 1 18 1 25 1 33 1  
...

## Processing Data into R

This is the final step of data processing prior to reading your data into R and analyzing to your heart’s content. You will need to have all of the following in a single directory: a folder containing your sentence DA1 files, a folder containing your question DA1 files (this can be empty, but it must be present), the .reg file (output.reg.txt from before), the question key file (expquestions.txt from before), and the readInput.py, eyeMeasures.py, and eyeDataToR.py scripts.

**Using eyeDataToR.py**

Run eyeDataToR.py from the command line. The script will prompt you to input the:

1. Name of the .reg file
2. Name of the question key
3. Name of the folder containing your sentence data
4. Name of the folder containing your question data
5. Name of the output file you want to create.

Assuming all of the previous steps have worked, it should display a message “Processing ###” where ### is the current data file in question. If question data for a particular subject isn’t available, the script will print “no question data for ###”. The output of this script will be a file named as in (5), which you can read into R for analysis.

# **Library of Tools**

* Powerpoints:
  + [Running a Reading Experiment.ppt](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=Running+a+Reading+Experiment.ppt)
  + [Reading Studies-Analysis.ppt](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=Reading+Studies-Analysis.ppt)
* Experiment Building:
  + [eyetrack\_reading.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=eyetrack_reading.py)
  + [randomizer.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=randomizer.py)
  + [sample\_list.txt](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=sample_list.txt)
* Data Analysis:
  + Organizing DA1 Files
    - [organizeDA1.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=organizeDA1.py)
    - [splitDA1.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=splitDA1.py)
  + Making Regions
    - [getQuestions.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=getQuestions.py)
    - [getSentences.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=getSentences.py)
    - [makeRegions.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=makeRegions.py)
  + Process Data
    - [eyeDataToR.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=eyeDataToR.py)
    - [eyeMeasures.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=eyeMeasures.py)
    - [readInput.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=readInput.py)
* Old Scripts:
  + [aberlour\_all.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=aberlour_all.py)
  + [aberlour\_all050910.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=aberlour_all050910.py)
  + [eyeMeasures050910.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=eyeMeasures050910.py)
  + [percentExcluded.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=percentExcluded.py)
  + [readInput.py](https://web.archive.org/web/20120822154339/http://languagescience.umd.edu/wiki/Eyelink%201000%20Eyetracker?action=AttachFile&do=view&target=readInput.py)