Synesthesia Data Analysis Mini-Toolbox for R

Author:

Árni Gunnar Ásgeirsson

[arnigunnarasgeirsson@gmail.com](mailto:arnigunnarasgeirsson@gmail.com)

Leiden University

June 2016

This is a brief manual for using the functions and scripts provided in the *[Synesthesia software] analysis using R.* This is a tiny package of scripts and functions for easy analysis of data from [name of software]. The functions are freely available for re-use and modification by anyone. The software comes with no warranty.

Table of contents:

Install R 1

Standard usage: Analyze and visualize the grapheme-color mappings of synesthetes with the create\_syn\_profile.R script in 8 simple steps. 1

Description and usage information for functions 3

Get grapheme-color consistency and color information. <- SYN\_calculate\_consistency() 4

Extract the time of a session <-SYN\_extract\_time\_of\_screening() 5

Plot a profile of a subject’s grapheme-color synesthesia <- SYN\_plot\_profile() 6

Select the appropriate files to analyze <- SYN\_select\_files() 7

Step 1: Install R

Skip this step if you have a working version of R on your computer.

If you have not yet installed R on your computer, you can download it from here: <https://cran.r-project.org>. Follow the installation instructions that match your operating system.

Step 2: Analyze and visualize the grapheme-color mappings of synesthetes with the create\_syn\_profile script in 8 simple steps.

The create\_syn\_profile script uses all the functions in syn\_functions to return: 1) a visualization of grapheme-color associations and their strength and 2) a file with the consistency measures for all graphemes and detailed color information. The visualization gives a quick overview of a synesthete’s grapheme-color associations, helps identify incorrect (accidental) color-mappings. The visualization is also very useful for comparing the quality of synesthesia over time.

To get started you must first set the working directory to the folder that contains the create\_syn\_profile.R script, and the syn\_functions.R file. To do this, you type:

> setwd(‘[your target directory]’) # step 2.1

Where [target directory] is the path to the directory containing your script. You can make sure that you have reached the correct destination by typing dir() in the console. This gives you a list of all files and folders in the working directory. Example:

> dir()

[1] “create\_syn\_profile.R” “syn\_functions.R”

You run the script by typing:

> source(‘create\_syn\_profile.R’) # step 2.2

You will be prompted to type (or copy) the path to the folder containing your log\_ID\_ data files[[1]](#footnote-1). The prompt looks like this:

# step 2.3

Please provide the path to the folder containing synesthesia data:

and you simply paste (or type) the path after the text prompt. You will now see a list of log\_ID files, and be asked whether you wish to select a subset of the files (y) or all files (n).

[1] "--- These are the data files in your folder:"

[1] "log\_ID\_AU01-1\_Session\_1.csv" "log\_ID\_AU01-2\_Session\_2.csv"

[3] "log\_ID\_AU01-3\_Session\_3.csv" "log\_ID\_IS01\_Session\_1.csv"

[5] "log\_ID\_IS02\_Session\_1.csv" "log\_ID\_IS03\_Session\_1.csv"

[7] "log\_ID\_ku01\_Session\_1.csv" "log\_ID\_KU02\_Session\_1.csv"

[9] "log\_ID\_KU13\_Session\_1.csv" "log\_ID\_KU14\_Session\_1.csv"

[11] "log\_ID\_KU15\_Session\_1.csv" "log\_ID\_KU16\_Session\_1.csv"

[13] "log\_ID\_KU17\_Session\_1.csv" "log\_ID\_KU18\_Session\_1.csv"

[15] "log\_ID\_KU19\_Session\_1.csv" "log\_ID\_KU20\_Session\_1.csv"

[17] "log\_ID\_KU21\_Session\_1.csv" "log\_ID\_SubjectName01\_Session\_1.csv"

[1] "--- end of file list ---"

# step 2.4

Do you wish to select a subset of these files? (n/y). n = all files; y = subset:

You type your response, followed by Enter. If you select n, the script will analyze all files in the list. If you select y, meaning you wish to analyze only a subset of the log\_ID files, you will be prompted for a list of files. The numbers in brackets on the left of the console are the indices to that file in the list of files. Note that, while the example shows an index to every other file, this will vary dependent on the size of your console window.

# step 2.5

Which files do you want to keep? Use space separated integers to denote the appropriate file indices: 1 3 17

You type the integers that match the files in the list (see file list example). In the example above (# step 2.5), the user has selected files 1, 3 and 17. This returns a new list of files, with a prompt to confirm the correctness of the chosen files.

# step 2.6

[1] "log\_ID\_AU01-1\_Session\_1.csv" "log\_ID\_AU01-3\_Session\_3.csv" "log\_ID\_KU21\_Session\_1.csv"

Are these files correct? (y/n):

If you do not confirm the correctness of the files (by choosing n), you will be returned back to file selection (# step 2.5). When you have confirmed that you have indexed the files you want, you will be asked if you want to save a file of consistency and color information:

# step 2.7

Do you want to save a .txt file of consistency scores and color information (y/n):

If you chose y, then a subfolder called *color-consistency-files* will be created in your data folder (the one you typed/pasted in # step 2.3. The .txt files will be written in this folder. One file will be written for each data file chosen in # step 2.5. Each file will contain information about all valid graphemes, the consistency of mapping of these graphemes, and the colors chosen in each of 3 trials during screening. Finally, an average color is calculated for each grapheme, and this color is what you are most likely to use to generate experimental stimuli, or visualizations of synesthesia. Color information is provided in hexadecimal color codes and as RGB values.

Next, you are asked whether you would like to create PDFs of the visualization of synesthetes’ color mappings:

# step 2.8

Do you want your plots to be printed as .pdf files (y/n):

If you respond y, PDFs will be saved in your data folder. If you respond n, the visualizations will be printed to R’s native graphics device.

When you have made your choice, the console will inform you of which data file is being processed. When the > re-appears, the script has run its course, and you should find your all your analyzed data and graphics in your data folder.

[1] "--- Processing file: log\_ID\_AU01-1\_Session\_1.csv"

[1] "--- Processing file: log\_ID\_AU01-3\_Session\_3.csv"

[1] "--- Processing file: log\_ID\_KU21\_Session\_1.csv"

>

This is the recommended way of using this R mini-toolbox. It requires the least amount of programming skills, and runs without any additional packages. Advanced users are welcome to make their own solutions, and contribute new scripts or functions.

Description and usage information for functions

Advanced users of R may want to use the SYN\_ functions separately and write their own code to tailored to their needs. The first step is to load the functions by running the file containing them: source(‘syn\_functions.R’)

If this is done successfully, the command

> ls(pattern = ‘SYN\_’)

should return the names of the functions:

[1] "SYN\_calculate\_consistency" "SYN\_extract\_time\_of\_screening"

[3] "SYN\_plot\_profile" "SYN\_select\_files"

[5] "SYN\_write\_consistency\_color\_file"

indicating that the functions are loaded to the R-environment.

When the functions are loaded, you can use them on their own, to get extract the information you need. Some of them will, however, be of very limited use outside of the create\_syn\_profile script. What follows is a brief description of the functions, their required input and returned output.

Get grapheme-color consistency and color information. <- SYN\_calculate\_consistency()

grapheme\_consistency\_and\_color\_info = SYN\_calculate\_consistency(data)

This function requires a loaded dataset from the synesthesia screening software [name of software] of type log\_ID\_[Name]\_Session\_[#].csv. If you are using the function on its own (not using a script that pre-loads your data) you must first read the data you want to analyze. You can do this with the read.table() function.

> data = read.table(‘[name of datafile in working dir]’

,skip = 1 # skip the first line of the data file

,header = T # the first read line is a header line

,sep = ‘;’) # the delimiter is a semi-colon

You can check that the data is loaded by typing data in the console. This will print the whole data frame. If you prefer just seeing a few trials of the data, you can also type

> head(data)

Red Green Blue Hue Saturation Value Stimulus Time.since.start..us.

1 173 173 173 43 0 173 9 0

2 141 141 141 92 0 141 T 93372515

3 10 115 20 127 232 115 6 162727838

4 131 78 32 29 192 131 Ø 226855201

5 148 148 148 109 0 148 D 254540289

6 123 34 10 14 233 123 N 296710614

When the data is loaded into a data frame, you can use the SYN\_calculate\_consistency() function like this:

> my\_results = SYN\_calculate\_consistency(data)

my\_result will be an array of all successfully mapped graphemes with a consistency score and information about color in hex and RGB formats. Color information is available for each trial of grapheme-color mapping, and the average of 3 trials. By typing colnames(my\_result) you can see the names of each column in the array:

> colnames(my\_result)

[1] “Grapheme” “diffScore” “hex1” “hex2” “hex3”

[6] “hexAvg” “R1” “G1” “B1” “R2”

[11] “G2” “B2” “R3” ”G3” “B3”

[16] “R\_avg” “G\_avg” “B\_avg”

where each number-suffix represents the trial number for a particular grapheme, and \_avg is the averaged color value. The diffScore is the value of each graphemes geometric variation in RGB color space over the 3 trials (Eagleman et al., 2007, p. 141, eq. 1).

If you prefer to have keep these data as a data frame, you can do that transformation easily: my\_df = data.frame(my\_result)

After such a transformation, you can refer to each column of the dataset by typing my\_df$[*column name*], e.g. my\_df$Grapheme, to produce a vector of all graphemes. Example:

> my\_df$Grapheme

[1] 1 2 3 4 5 6 7 8 9 A B C […]

Levels: 1 2 3 4 5 6 7 8 9 A B C […]

By averaging the diffScore column, you will get the consistency score suggested by Eagleman and colleagues (2007, p. 141, eq. 2). To refer to this column alone, you can type:

> my\_result[,2]

or

> my\_df$diffScore #if you have transformed to data frame

The R\_avg, G\_avg and B\_avg or the hexAvg, give you the averaged colors that you may want to use for an experiment. Note, however, that you must inspect the data visually beforehand, or set a criterion of maximum within-grapheme variability in color-mappings, before using the average color for test-stimuli. Subjects will sometimes accidentally record an incorrect grapheme-color association, and this will have a large impact on the average color.

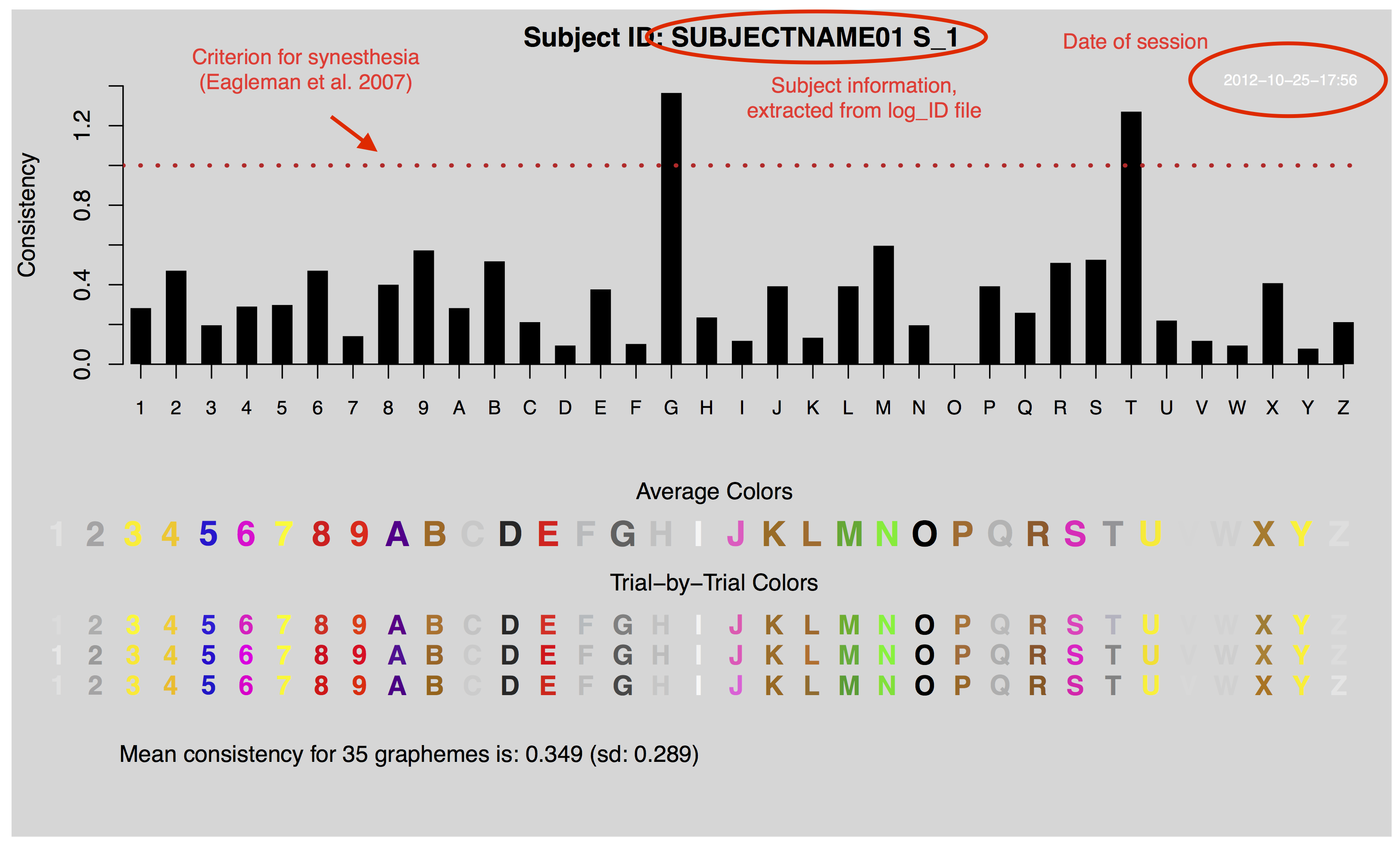
Extract the time of a session <-SYN\_extract\_time\_of\_screening()

The function SYN\_extract\_time\_of\_screening() deciphers the time of the grapheme-color mapping session from a rather obscure string at the beginning of log\_ID\_[subj]\_Session\_[#].csvfiles. This information is registered in the data files in a rather inaccessible manner, so this function simply gives you a nicely formatted YYYY-MM-DD HH:MM data and time of the session from which data is being analyzed. If you have set the working directory to the directory that includes your file of interest, simply type my\_data = SYN\_extract\_time\_of\_screening(‘myFile.csv’)

In any directory, you may also provide the full path to the file to obtain the same result.

Plot a profile of a subject’s grapheme-color synesthesia <- SYN\_plot\_profile()

This function visualizes the data of any synesthete. It doesn’t work well on its own, because it requires many input parameters. The best way to use this function is to run the create\_syn\_profile script. A typical profile will look something like Figure 1.



**Figure 1. A typical visualized “profile” of a synesthete, created from a single session of grapheme-color mapping with the [name of software] software.**

*Input parameters:*

data <- a data frame that contains the imported data from a log\_ID file.

subj <- a string that contains the subject identity

omittedGraphms <- is a vector of graphemes to omit from the visualization, due to missing data in, because it has been registered as having “no color”, or because you don’t want it visualized and remove omit it manually.

BGcontrEnhancement <- is a Boolean (True or False) value that determines whether you enhance the background of low color-contrast letters (similar to the gray background) with a darker background. If the value is F, FALSE or 0, then the background of all letters in the visualization will be the same (default). If it is set to T, TRUE or a positive numeric value, the background will be darkened to sharpen the contrast of some letters. Figure 2 shows an example of this:



**Figure 2. An example of enhanced background contrast for letters that have a very low color contrast.**

Select the appropriate files to analyze <- SYN\_select\_files()

The SYN\_select\_files() function lets the user pick a subset of a list of files. This is explained in the create\_syn\_profile tutorial.

Reference

Eagleman, D. M., Kagan, A. D., Nelson, S. S., Sagaram, D., & Sarma, A. K. (2007). A standardized test battery for the study of synesthesia. *Journal of Neuroscience Methods*, *159*, 139–145. doi:10.1016/j.jneumeth.2006.07.012

1. This is not the most elegant solution, and advanced R users may want to replace part of the script with a tcltk user interface solution. However, the current scripts and functions were written to run on an out-of-the-box version of R, without the need to install and load any additional packages. [↑](#footnote-ref-1)