`ScriptR.r` Manual

Michael Wilson

(Last updated December 10, 2019)

Contents

1. Introduction

The following manual explains how to use `scriptR.r`, a script written in R (R Core Team 2017) to facilitate the preparation of experiment scripts used with UMass EyeTrack (available here). It is indebted to Adrian Staub's `scripter2.pl`. `ScriptR` handles preparation of a script file that is ready to run in EyeTrack with no additional preparation (in most cases). This manual details the system requirements, the structure of (optional) config CSV files that can be used with `scriptR`, and the structure of the stimuli CSV file.

Some useful behavior of `scriptR` includes automatically calculating condition numbers, renumbering items for EyeTrack when there are multiple experiments, supporting questions that depend on an item's condition (including the presence or absence of a question), automatically balancing the presentation order of answers within items in an experiment, automatically reordering conditions for paired items, automatically calculating parameters for gaze control rectangles, automatically calculating display change regions, and options to output a variety of additional useful files. See the following documentation for additional information.

Note: If you have used `scripter2.pl` in the past, make sure to take a look at the stimuli format explanation, since there are noticeable differences in how items should be formatted for `scriptR`.

2. System requirements

`ScriptR` requires R to be installed. It was written in R 3.6.1; it has not been tested on older or newer versions, but it will probably work. It requires the packages `dplyr`, `data.table`, `gplots`, `stringr`, and `tidyr`; if these are not installed, it will automatically install them when it is run (along with any additional required dependencies).

2.1 Usage

`ScriptR` is run through the command prompt (Windows) or Terminal (Mac). To use it, open the command prompt/Terminal, navigate to the directory where it is located, type the following,[1] and hit enter:

- `Rscript scriptR.r`

After running, `scriptR` creates a directory named "`ScriptR (your stimuli file) output`" in the folder where your stimuli file is located, which is where you will find the output. The following are possible output files, where `stimuli.csv` is the name of your stimuli file:

- `stimuli-formatted.script`: the script formatted for use with EyeTrack. This is always output.
- `stimuli-formatted.csv`: the formatted stimuli in a CSV file. This is always output. It is useful because if you have more than one experiment or you use `scriptR` to automatically balance answer order within items or pair items, it will renumber your items. `ScriptR` will also assign your items condition numbers. This CSV will tell you what the condition and item numbers are, which you will need for data analysis later.
- `stimuli_NG.script`: a nogaze version of the script. Useful for running non-native speakers. Optionally output, set by the `create.nogaze` setting.
- `stimuli-testing.script`: a version of the script that replaces all buttons with mouse clicks. Useful for testing the script on computers without a controller hooked up. Optionally output, set by the `create.test` option.
- `stimuli-sentences.txt`: a base for a `sentences.txt` file, which can be used with sideeye during analysis. Optionally output, set by the `create.sentences.txt` option.

[1] You must have your R installation directory added to your system path for this to work. If this is not the case, then you should enter:

- `%RDIRECTORY%/Rscript scriptR.r`

2.2 Specifying regions for a `sentences.txt` *file*

In order to make generating a `sentences.txt` file easier, you may specify analysis regions in sentences in your input CSV file using forward slashes (`/`). You need not include a forward slash at the beginning or end of the sentence, as `scriptR` will generate those automatically. Note that this means that forward slashes will not appear in any sentences displayed in EyeTrack. Practice items should not have analysis regions specified.

3. Configuration

When `scriptR` is run, it will prompt you for the location of a stimuli CSV file, and a configuration CSV file. If these are in the folder from which you are running the script, you just need to provide the CSV file name(s) (with or without the "`.csv`" extension). Otherwise, you can provide the full file paths or file paths relative to the current directory (if you're using relative file paths, don't start them with a slash). If you do not provide a configuration file, it will prompt you for additional required parameters. If you wish to use the default value for a parameter, just hit enter when prompted. Any parameters omitted from the configuration file or left blank in it will use their default values. If you provide an invalid option for a parameter in a configuration file or at the command prompt, you will be re-prompted until you provide a valid response or hit enter (in which case the default value will be used). If you are setting options through the command prompt, you will have the option to output the current settings as a config CSV for future use.

3.1 Config format

A configuration file is a CSV file that has two columns: `option` and `setting`. The `option` column names the parameters to be set, and the `setting` column names the value for the corresponding parameter, as follows:

option	setting
background.color	white
foreground.color	black
…	…

Table 1: Example of config file format

3.2 *Config defaults*

Table 2 shows the possible configuration parameters. The first column provides a name/description of the parameter, the second shows the name that the corresponding option should receive in a config CSV file, and the third shows the default setting.

Option	`config.csv` name	Default
Background color	`background.color`	`white`
Text color	`foreground.color`	`black`
Calibration Type	`calibration.type`	`9`
Font	`font`	`Monaco`
Font size	`font.size`	`12`
Font style	`font.weight`	`normal non-italic`
Font smoothing	`font.smoothing`	`nonantialiased`
Unit	`unit`	`px`
Line spacing	`line.spacing`	`0`
X offset	`x.offset`	`20px` ≈ `15pt`
Y offset	`y.offset`	`357px` ≈ `268pt`
Sentence button	`sentence.button`	`rightTrigger`
Sentence trigger	`sentence.trigger`	`driftandgaze`
Sentence delay	`sentence.delay`	`0`
Sentence output	`sentence.output`	`stream`
Question trigger	`question.trigger`	`nogaze`
Question delay	`question.delay`	`0`
Question output	`question.output`	`nostream`
Calculate gaze control parameters?	`calculate.gc.params`	`FALSE`
Gaze control parameters?	`gc.params`	`0 0 0 0`
Horizontal pixels per character	`h.pixels.per.char`	(none)
Vertical pixels per character	`v.pixels.per.char`	(none)
Display change region padding	`dcr.padding`	`line.spacing`
Display update threshold	`display.update.threshold`	`0`
Revert display changes?	`display.revert`	`FALSE`
Display change delay	`dc.delay`	`0`
Balance answer order experiments?	`balance.answer.order.exps`	`none`
Paired items experiments?	`paired.items.exps`	`none`
Create `sentences.txt`?	`create.sentences.txt`	`FALSE`
Create nogaze version?	`create.nogaze`	`FALSE`
Create test version?	`create.test`	`FALSE`
Create `fix_align start_pts` matrix?	`create.fix.align.start.pts`	`TRUE`
Save settings?	N/A	`FALSE`

Table 2: Options, names, and default settings

3.3 *Option explanations*

The following is an explanation of each option:

- Background color: a color name available to R, or the hex value of a color
- Text color: a color name available to R, or the hex value of a color

- Calibration type: the number of calibration points to use; one of { `3`, `9`, `13` }
- Font: the name of the font to use
- Font size: the font size in pt
- Font style: the name of the font style to use. One of { `normal non-italic`, `normal italic`, `bold non-italic`, `bold italic` }
- Font smoothing: the style of font smoothing to use. One of { `nonantialiased`, `antialiased`, `cleartype` }
- Unit: Which unit to use when setting `line.spacing`, `x.offset`, `y.offset`, and `dcr.padding`. Options are `px` (pixels) or `pt` (points). EyeTrack uses points by default; `scriptR` uses pixels by default. **Note**: regardless of the value of this setting, `font.size` is always set in points, and `h.pixels.per.char` and `v.pixels.per.char` are always set in pixels.
- Line spacing: the spacing between lines
- X offset: the starting x position of text
- Y offset: the starting y position of text
- Sentence button: the button to use to proceed from a sentence trial; using a 7 button controller, one of { `A`, `B`, `X`, `Y`, `toggle`, `leftTrigger`, `rightTrigger`, `mouse` }
- Sentence trigger: the setting for what causes a sentence to display. One of { `driftandgaze`, `driftcorrect`, `gaze`, `nogaze` }
- Sentence delay: how long to wait in ms before displaying the sentence. Set only if sentence trial type is `nogaze`
- Sentence output: the setting for what kind of output to generate for sentence trials. One of { `stream`, `nostream` }
- Question trigger: the setting for what causes a question to display. One of { `driftandgaze`, `driftcorrect`, `gaze`, `nogaze` }
- Question delay: how long to wait in ms before displaying the question. Set only if question trial type is `nogaze`
- Question output: the setting for what kind of output to generate for question trials. One of { `stream`, `nostream` }
- Horizontal pixels per character: a number indicating how many horizontal pixels there are per character; used only if there are display change trials, if automatically setting the parameters of the gaze control rectangle, or if creating a `fix_align start_pts` matrix
- Vertical pixels per character: a number indicating how many vertical pixels there are per character; used only if there are display change trials, if automatically setting the parameters of the gaze control rectangle, or if creating a `fix_align start_pts` matrix
- Calculate gaze control parameters?: a logical value indicating whether the script should automatically calculate the parameters for the gaze control rectangle (if any

trial types use it). Only used if `sentence.trigger` or `question.trigger` is `gaze` or `driftandgaze`. The default size is 4 × `v.pixels.per.char` by 4 × `h.pixels.per.char`.

- Gaze control parameters?: a series of four numbers separated by spaces that define a gaze control rectangle as `Y1 X1 Y2 X2` in pixels. The default setting for this, `0 0 0 0`, doesn't draw a gaze control rectangle.
- Display change region padding: a number indicating how much padding in points to add around display change regions; used only if there are display change trials. This is set to `line.spacing` by default, which means display change boxes line up with the vertical line height. Set this to `0` if you would like the display change region height to be equal to the character height.
- Display update threshold: a number indicating the display update threshold. Set only if there are display change trials
- Revert display changes?: a logical value indicating whether to revert display changes. Set only if there are display change trials
- Display change delay: how long to wait in ms before reverting display changes. Set only if display changes are reverted
- Balance answer order experiments?: a string of experiment numbers separated by spaces indicating which experiments to balance the presentation order of answers for. You can also use `all` or `none` to set this for all valid experiments or no experiment without specifying each experiment number manually. **Note**: If you use this option, each answer button must be set to either `leftTrigger` or `rightTrigger` in the stimuli CSV for those experiments that you are balancing the presentation order of answers for. If there is an answer button for such an experiment that isn't one of those values, `scriptR` will not balance the presentation order of answers for that experiment. In addition, note that even if an item has a question in only some of its conditions, all conditions will be repeated twice. E.g., suppose item `1` has 4 conditions, and questions only in conditions 1 and 4. All conditions will be repeated twice if answer order is balanced, even those conditions without questions.
- Paired items experiments?: a string of experiment numbers separated by spaces indicating which experiments to pair items for. You can also use `all` or `none` to set this for all valid experiments or no experiment without specifying each experiment number manually. What this means is that you don't want participants to see odd/even pairs of items in overlapping conditions for that experiment. For instance, consider the following example items in 2 × 2 experiment:

(1)	a.	`X1`	(2)	a.	`X1`
	b.	`X2`		b.	`X2`
	c.	`Y1`		c.	`Y1`
	d.	`Y2`		d.	`Y2`

By default, list 1 in EyeTrack will show participants item 1 in condition `X1`, and item 2 in condition `X2` (assuming this is the first experiment). Suppose we don't want participants to see condition `X` for both items 1 and 2. Putting this experiment in the

`paired.items.exps` list will reverse the order of all but the first condition in all even numbered items, with the following result:

(1)
a. `X1`
b. `X2`
c. `Y1`
d. `Y2`

(2)
a. `X1`
b. `Y2`
c. `Y1`
d. `X2`

Now, list 1 will show item 1 in condition `X1`, and item 2 in condition `Y2`, with items in opposite conditions. The same holds for the rest of the lists (list 2: item 1-`X2`, item 2-`Y1`; list 3: item 1-`Y1`, item 2-`X2`; list 4: item 1-`Y2`, item 2-`X1`).

Note: This is only valid for experiments with an even number of items in an even number of conditions, and will be ignored if you specify an experiment with an odd number of items or conditions per item. Paired items must be odd and even numbered items, starting from the first item in the experiment. That is, items 1 and 2, and items 3 and 4, and so on can be paired; but items 2 and 3 cannot. Items in pairs must have their conditions in the same order in the stimuli CSV file for this to work.

Caution!!

Pairing items will only work if all item-related factors have an even number of levels. If any factors have an odd number of levels, the general algorithm will pair items mostly but not completely correctly. How to do this correctly depends more on the structure of your items if any levels have an odd number of conditions. `ScriptR` isn't able to detect this sort of fine-grained information about item conditions, so use caution if your experiment has an item-related factor with an odd number of conditions. If you have, e.g., 12 conditions per item, `scriptR` can't tell whether they're the result of a 2×6 design or a 3×4 design, and will attempt to pair items using the general algorithm. This algorithm will work for the first design, but not the second, so be careful.

- Create `sentences.txt`?: a logical value indicating whether to output a `sentences.txt` base file to be used with Sideeye. Note that currently the `sentences.txt` file includes only `Sentence`, not `Sentence2` (i.e., it doesn't include any sentences that appear after the initial one in a display change trial).
- Create nogaze version?: a logical value indicating whether to output a version of the script where all triggers are `nogaze`, in addition to the normal version
- Create test version?: a logical value indicating whether to output a version of the script that uses mouse clicks for all triggers for testing purposes
- Create `fix_align start_pts` matrix?: a logical value indicating whether to create a `starts_pts` matrix for use with `fix_align.R`[2] (Cohen 2013) during data processing. If created, the value of the `start_pts` matrix is added to the header of all script files

[2]Currently available here: https://blogs.umass.edu/rdcl/resources/

- Save settings?: a logical value indicating whether to save the current settings as a config CSV file for future use. Only set through the command line.

3.4 Determining pixels per character

You can determine the number of pixels per character for display change trials or automatically calculating gaze control rectangle parameters by setting up an EyeTrack experiment with line spacing set to `0`. Create an item with your desired font settings. Preview the item by double-clicking on it in the list of items, then under "Display Properties" click "Change." Take a screenshot and open it in an image editing program. The measures `scriptR` assumes for these parameters are as shown in the following image: the horizontal pixels per character measures from one blue pixel before the character to one blue pixel after the character, while the vertical pixels per character measures from the top to the bottom of the blue rectangle that the character appears in. See the following screenshot:

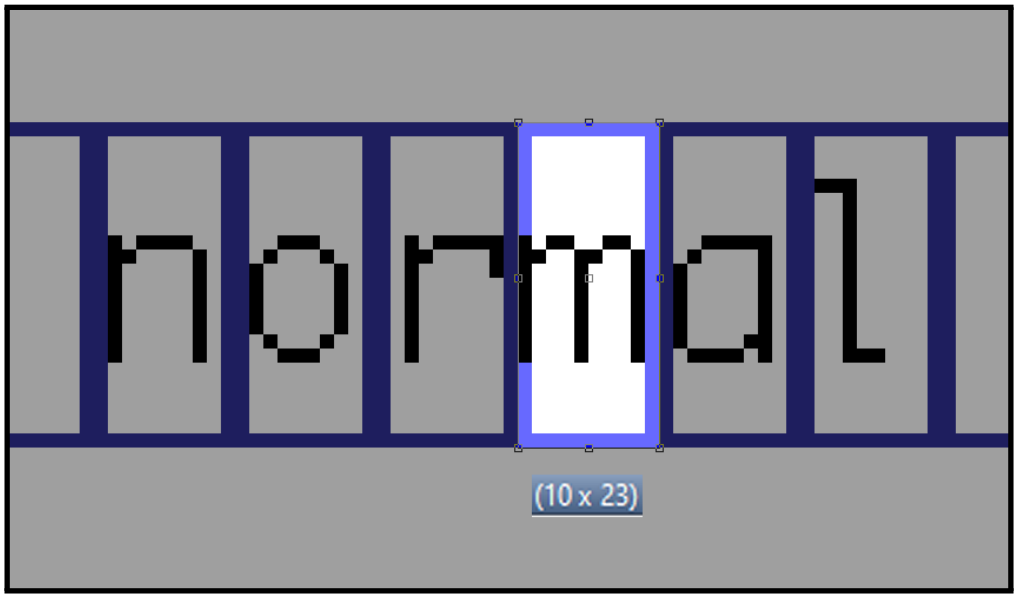

Figure 1: How to measure character specifications

Thus, in this case, you would set `h.pixels.per.char` to `10`, and `v.pixels.per.char` to `23`. If you would like display change region boxes to extend past the actual height of the characters, you can set `dcr.padding` to the number of pixels or points you would like the region box to extend past the character. Setting `dcr.padding` to the value of `line.spacing` makes region boxes line up with the character boxes; `dcr.padding` is set to `line.spacing` by default.

3.5 Note on number formats

`ScriptR` handles conversion from pixels to points or vice versa depending on the unit you set. However, note that by default, EyeTrack interprets line spacing, x offset, and y offset in points; while it interprets gaze control boxes and display change region boxes in pixels. In case this matters for you, for ease of use, the conversion between the two is the following:

- Points $\times 1\frac{1}{3}$ = Pixels
- Pixels $\times 0.75$ = Points

Important note: EyeTrack appears to use a slightly simplified conversion for line spacing, which is the following:

- Line spacing points $\times 1.32$ = Line spacing pixels

In order to facilitate lining up display change regions with line regions, `scriptR` uses this same conversion for `dcr.padding`. Note also that EyeTrack does appear to use $1\frac{1}{3}$ for converting from points to pixels for `x.offset` and `y.offset`, in contrast to what it does for line spacing; be careful in case this might make a difference for your experiment, though `scriptR` is designed to be able to handle the conversion in a way that you shouldn't have to worry about it.

4. Stimuli format

`ScriptR` requires you to provide a CSV file containing your stimuli. The format of the stimuli CSV file should be as follows. Note that the order of the columns can be whatever you want, but column names must be exact. You may also include additional columns, and the script will match these up correctly in the `stimuli-formatted.csv` file, which may be useful for data analysis (for instance, if you want to include non-numeric condition information).

Experiment	item_id	Timeout	QTimeout	Sentence	Sentence2	//

Question	LeftAnswer	RightAnswer	AnswerButton

Table 3: Stimuli CSV columns

Each line contains one item. You may also omit most columns to use default values. See §4.3 for details on default values.

Important: You should not have a column named `item_condition` in your stimuli CSV. If you do, `scriptR` will overwrite it in the output files.

4.1 Stimuli format explanation

The following explains what goes in each of the columns:

- `Experiment`: a number indicating which experiment the item is associated with. Stimuli files can contain multiple experiments. Each experiment must have the same number of conditions per item.
- `item_id`: a number indicating which item the row belongs to. **Note**: This is set per experiment (i.e., each experiment starts at item `1`). When there are multiple experiments, `scriptR` handles adjusting this for EyeTrack automatically.
- `Timeout`: the length of time in ms to display the sentence before timing out.
- `QTimeout`: the length of time in ms to display the question before timing out.

- `Sentence`: the sentence to display. **Note**: You do not need to put `\n` at the end of each sentence, though you should specify any line breaks within a sentence at the appropriate place.
- `Sentence2`: the second sentence to display (for display change items). Display change regions must be marked with "%" signs on either side. There must be at least one display change region marked for each `Sentence2`. You may have multiple regions marked per sentence and per line, and regions may occur at the beginning of lines. **Note**: You do not need to put `\n` at the end of each sentence, though you should specify any line breaks **within** a sentence at the appropriate place manually. If you have no display change trials, you may omit this column entirely. Display change trials are not currently supported for `Message` items.
- `Question`: the question associated with the sentence. If there is no question for a particular item/condition, leave this blank. This is set up per sentence in order to handle cases when questions are different depending on the condition, and necessitates additional processing during analysis. Currently, `scriptR` does not offer an alternative way of doing this. The dependent number assigned to the question in the EyeTrack script is the item's condition number + 100. **Note**: You do not need to put `\n` at the end of each question.
- `LeftAnswer`: the first answer. Displayed on the left. **Note**: `scriptR` cannot currently handle questions with more than two possible answers correctly. Answers are automatically separated by five spaces.
- `RightAnswer`: the second answer. Display on the right. **Note**: `scriptR` cannot currently handle questions with more than two possible answers correctly. Answers are automatically separated by five spaces.
- `AnswerButton`: the button used for the correct answer. **Note**: If you are using the `balance.answer.order.exps` option for an experiment, this must be either `leftTrigger` or `rightTrigger` for all items in that experiment.

4.2 Practice items

You can also put practice items in the stimuli CSV. Practice items are signalled by a special value for Experiment, which is "`P`." Practice items are put into a sequence whose order is the same order they appear in the stimuli file. You can also specify instructions here, using "`Message`" as the value for `item_id`. Except when `item_id` is `Message`, all `item_id`s should be `1` for practice trials. Line breaks are automatically inserted into practice messages at the next nearest word break every 67 characters following any manually specified line break. Paired items are not supported for practice trials. Answer orders are not balanced within items for practice trials, even when the balance answer order option is set to true. Practice trials are not included in the `stimuli-sentences.txt` or `stimuli-formatted.csv` files.

4.3 Default values

You may omit certain columns from your stimuli file, in which case `scriptR` will fill them out with default values, which are the following:

Column	Default value
`Experiment`	For any `Message` item(s) and all items between the first and last `Message` item(s), `P`. For other items, a sequence of integers starting with `1`, set based on `item_id`, under the assumption that contiguous items with increasing item numbers that have the same number of conditions belong to the same experiment.
`item_id`	Must be set manually.
`Timeout`	`30000`
`QTimeout`	`30000`
`Sentence`	
`Sentence2`	
`Question`	
`LeftAnswer`	
`RightAnswer`	
`AnswerButton`	`leftTrigger`

Table 4: Default values for omitted/blank columns in stimuli CSV

For most of these, all and only missing values will be replaced with defaults. Any values outside the permissible possible values will also be replaced with defaults. For `Experiment`, if any value is missing, all values, including those already present, will be recalculated automatically following the above description. You will see a warning if `Sentence` is empty for any item.

5. Contact

If you have any questions or suggestions regarding `scriptR`, you may contact the author at `mawilson@linguist.umass.edu`.

References

Cohen, A. (2013). Software for the automatic correction of recorded eye fixation locations in reading experiments. *Behavior Research Methods*, *45*(3), 679–683.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. Available at `http://www.R-project.org`