Alex: Associative Learning EXperiments

Stefano Ghirlanda

Contents

Introduction

Alex is a program to run associative learning experiments specified through a set of configuration files. This manual describe how to configure experiments. Please refer to the README file that comes with alex for installation instructions. The README also describes in brief what alex can and cannot do.

Acknowledgments: Alex is written using Shane Mueller's Psychology Experiment Building Language (PEBL). Many thanks to Shane for sharing PEBL!

Workflow

To build and run a new experiment you create a dedicated folder, say, MyExperiment, and within it the following subfolders:

- **Design**: This folder contains the files that specify experimental design, such as which stimuli to use, the structure of trials, and different treatments for subjects. See Configuration files.
- **Materials**: Here you have any image, sound, or text files you need for your experiment, including an Instructions.txt file for the initial instructions.
- **Data**: This folder is created by alex if it is not found, and it holds the data collected during experiment runs.

The program alex-init generates a bare-bones experiment so that at least you know what files you need. It is run like this:

```
alex-init -v <experiment name>
```

This creates folder with the above mentioned subfolders and skeleton configuration files.

Running and stopping alex

From the folder where the Design and Materials folders are, you can just type 'alex'. You can also run experiments in other folders using:

```
alex -v <folder>
```

The is then expected to have Design and Materials subfolders with appropriate files. A Data folder will be create if not present.

Alex has been designed so that multiple instances of an experiment can be run simultaneously. This feature is useful when the experiment folder is shared among multiple computers, as it may occur in a lab. All instances of alex will read the same design files, and in particular the same `Subjects.csv` file which describes how to run subjects. Different instances, however, will run different subjects and will not overwrite each other's data files.

The fact that a subject has been run is signaled by the existence of the corresponding data file (see Data format). If the experiment is interrupted before it completes, alex will still consider that subject as having been run. It is up to you to check that data files are complete (you can check that they have the appropriate number of lines, for example). Although this may be inconvenient at times, it is hard to improve upon this situation, because there is no way for alex to decide whether important data would be overwritten by re-running a subject. If you decide a data file is worthless, either remove it or rename it with something like an 'incomplete-' prefix, and alex will automatically re-run that subject.

If you want to interrupt a running experiment, you can use the standard interrupt key comination for PEBL: `Ctrl+Alt+Shift+\`.

Configuration files

All configuration files are in the Design folder:

- `Phases.csv` describe the experimental design proper. I containes one or more experimental phases, each composed of a number of trials in which stimuli are presented, responses recorded, and outcomes delivered.
- `Stimuli.csv` defines the stimuli that are mentioned `Phases.csv`. The latter only mentions stimuli by name, while `Stimuli.csv` informs alex of what the stimuli actually are.
- `Subjects.csv` defines the number of subjects to be run and possibly different the treatments to which subjects are allocated.
- `Parameters.csv` defines some global parameters such as screen background color, text color, font, and size, the duration of inter-trial intervals, and so on. Can also be used to define parameters that are the same for all stimuli, such as which key is used for responses.

In addition, instruction files can be in Materials, see the section on text files.

Example

The `Phases.csv` file

Suppose we want to teach participants to discriminate a red square from a white square. We then want to know how subjects respond to, say, a pink square. Table 1 shows how a suitable `Phases.csv` file might look like.[1] The file describes an experiment with two phases. Each line describes one type of trial that occurs in a phase. There are, for example two kinds of trials in phase 1, specifying 20 presentations of each of two stimuli, called Red and White. Red will be rewarded 90% of the time, White only 10%. On reward trials, stimulus Smiley will be displayed as the reward (US). In phase 2, stimulus Pink is presented five times. When the experiment is run, Red and White trials will be intermixed randomly because they all pertain to phase 1. Pink trials on the other hand, will be performed in phase 2 after all phase 1 trials have been run.

Phase	Stimulus	Trials	Reward	US
1	Red	20	0.9	Smiley
1	White	20	0.1	Smiley
2	Pink	5	0	

Table 1: A simple `Phases.csv` to teach a discrimination between stimuli Red and White, and then testing responding to Pink. Note that the US field can be left empty if the Reward probability is 0.

Note: Phases are run in the order they are defined, not in their numerical or alphabetical order (thus you can use descriptive names like Training, Testing, etc). To be more precise, phases are run in the order in which their *first* stimuli are defined. For example, the phases file in Tables 1 and 2 are equivalent, but the file in Table 3 runs phase 2 before phase 1.

Phase	Stimulus	Trials	Reward	US
1	Red	20	0.9	Smiley
2	Pink	5	0	
1	White	20	0.1	Smiley

[1] In this manual, we use tables to display design files in a readable form. These files, however, are actually comma-separated-values (CSV) files. You can edit CSV files in any spreadhseet using the CSV format for saving. Alex wants double quotes (if needed) in CSV files. Single quotes will result in errors. (This comes from the PEBL function that reads CSV files.) Most spreadhseet software uses double quotes by default, but do check in case alex cannot read your CSV files.

Table 2: With this `Phases.csv` file, alex will run phase 1 before phase 2 (cf. Table 3.

Phase	Stimulus	Trials	Reward	US
2	Pink	5	0	
1	Red	20	0.9	Smiley
1	White	20	0.1	Smiley

Table 3: With this `Phases.csv` file, alex will run phase 2 before phase 1 (cf. Table 2.

The `Stimuli.csv` file

In the `Phases.csv` files in Tables 1–3, how does alex know that Red, White, and Pink represent red, white and pink squares, and that Smiley is a smiley face? This information is contained in the `Stimuli.csv` file, see Table 4.

Name	Type	Parameters	Color	XOffset	YOffset
Red	square	50	red	0	0
White	square	50	white	0	0
Pink	square	50	255,128,128	0	0
Smiley	image	smile-o-white.png		0	-150

Table 4: A `Stimuli.csv` file instructing alex that stimuli Red, White, and Pink are colored squares 50 pixels in side, and with different colors, and that Smiley is an image contained in file `smile-o-white.png`.

The fields in Table 4 should be fairly intuitive, but here is a detailed explanation:

- **Name**: An arbitrary label for the stimulus, so that it can be referenced in `Phases.csv`. It can be anything that does not contain the characters " (double quote), + (plus), * (asterisk), : (colon), and , (comma). These characters are reserved for special operations described below.
- **Type**: This can be square, circle, text, textfile, image, or sound.

- **Parameters**: The meaning of parameters varies according to the stimulus type:
- square: side in pixels.
- circle: radius in pixels.
- text: the text to be displayed.
- textfile: name of a file in the Materials folder where the desired text is stored.
- image or sound: name of a file in the Materials folder that contains the image or sound. An optional zoom factor can be provided to scale the image to a desired size. It should be separated from the filename by a `+` sign. The following stylized faces (smileys) come with alex and you can use them without having them in the Materials folder:
  - smile-o-white.png: a happy face, as used above
  - meh-o-white.png: a neutral face
  - frown-o-white.png: a sad face

  These images are drawn in white over a transparent background; equivalent white images are available as smile-o.png, etc. All images have been taken from Font Awesome, via this project. They are 256x256 pixels in size to look OK even on high resolution monitors. If that is too big for you, yu can zoom them as indicated above.
- **Color**: the color of squares, circles, or text. This field is ignored for images and sounds. Colors can either be named or given as an RGB triplet. As the latter are themselves comma-separated lists, they need to be double-quoted in the CSV file (spreadhseet software will do this for you). In the case of text, you can specify the background as well as the foreground color by writing the color in the form Color1+Color2, where Color1 will be foreground and Color2 the background. If no foreground or background color is given, the default set in `Parameters.csv` is used.

The PEBL reference manual lists valid color names, which are many hundreds. If you stick to simple stuff like red, blue, cyan, purple, and so on, you can get by without consulting this file. RGB, of course, enables you to define color shades more precisely.

- **XOffset** and **YOffset**: offsets from the center of the screen, in pixel. In the example, all stimuli are centered but the reward stimulus Smiley, which is displayed 150 pixels above center ("above" is negative Y values).

The `Subjects.csv` file

The `Subjects.csv` file contains information about the subjects you want to run. If all subjects undergo the same treatment, as in the present example, you only need to give subject numbers. The file in Table 5, for example, instructs alex to run 6 subjects, all receiving the

same treatment. Often, however, subjects need to be divided in different treatment groups. Any of the fields in the `Stimuli.csv` file can be specified on a per-subject bases. If you want to test two shades of pink, for example, you would extend the `Stimuli.csv` file in Table 6. The special notation *Pink demonstrated there indicates that the color of stimulus Pink will be looked up, for each subject, in the column PinkColor of the `Subjects.csv` file (Table 7). This syntax is available for all stimulus properties. For example, to change the size of Red square across subjects you would use the `Subjects.csv` and `Stimuli.csv` files in Tables 9 and 8.

Subject
1
2
3
4
5
6

Table 5: A `Subjects.csv` file instructing alex to run 6 subjects.

Name	Type	Parameters	Color	XOffset	YOffset
Red	square	50	red	0	0
White	square	50	white	0	0
Pink	square	50	*Pink	0	0
Smiley	image	smiley-o-white.png		0	150

Table 6: A `Stimuli.csv` file instructing alex to look up the Color of the Pink stimulus in the `Subjects.csv` file.

Subject	PinkColor
1	255,128,128
2	255,128,128
3	255,128,128
4	255,190,190

5	255,190,190
6	255,190,190

Table 7: A `Subjects.csv` file instructing alex to run 6 subjects split in two treatment groups with different Color attributes for the Pink stimulus (see Table 6).

Name	Type	Parameters	Color	XOffset	YOffset
Red	square	*Red	red	0	0
White	square	50	white	0	0
Pink	square	50	*Pink	0	0
Smiley	image	smiley-o-white.png		0	150

Table 8: A `Stimuli.csv` file instructing alex to run look up in the `subjects.csv` file both the Color of stimulus Pink and the Parameters of stimulus Red (see Table 9).

Subject	PinkColor	RedParameters
1	255,128,128	25
2	255,128,128	50
3	255,128,128	75
4	255,190,190	25
5	255,190,190	50
6	255,190,190	75

Table 9: A `Subjects.csv` file instructing alex to run 6 participants in 6 experimental groups (of course, in real life you will have more participants per group!). Each group is given by a unique combination of PinkColor and RedParameters (see Table 8).

Special notation for stimuli

We mentioned above one bit of special notation in the definition of stimuli, namely the construction * (star) + stimulus name (see the end of the previous section). There are two more bits of special notation, explained next.

Sometimes we want some stimuli to share characteristics. For example, they should be of the same color. We can express the fact that we want a stimulus characteristic to equal that of another stimulus using a colon (:) followed by the stimulus name (we would have liked to use = rather than :, but unfortunately spreadsheet software stubbornly interprets = as introducing a formula). Consider the example above, with three squares of the same size as stimuli. The file in Table 10 is equivalent but uses colon notation for the Parameters field. This has two advantages: it makes explicit our intention of having three squares of equal size, and it reduces the possibility of typing errors.

Name	Type	Parameters	Color	XOffset	YOffset
Red	square	50	red	0	0
White	square	:Red	white	0	0
Pink	square	:Red	*Pink	0	0
Smiley	image	smiley-o-white.png		0	150

Table 10: A `Stimuli.csv` file demonstrating the * and : special notations for stimuli.

Another bit of special notation is + (plus), which is used to present stimuli together (compound stimuli). Suppose that, after training a discrimination between red and white squares, we want to test the red and white squares together. We would then use the files in Tables 11 and 12.

Phase	Stimulus	Trials	Reward	US
1	Red	20	0.9	Smiley
1	White	20	0.1	Smiley
2	Red+White	5	0	

Table 11: A `Phases.csv` file with a compound stimulus in phase 2.

Name	Type	Parameters	Color	XOffset	YOffset

Red	square	50	red	0	0
White	:Red	:Red	white	60	:Red
Smiley	image	smiley-o-white.png		0	150

Table 12: A `Stimuli.csv` file to go with the `Phases.csv` file in Table 11. Note that we need to offset the white square, otherwise it would overlap with the red one when the two are presented together.

Note: The + notation is also valid for USs. This can be used to implement USs of different "magnitude." For example, one can instruct subjects that each smiley face represents a point earned, and have multiple smileys appear for more valuable stimuli (this requires defining several smiley stimuli offset from each other, so that they do not overlap when displayed simultaneously). Compounding of USs may also be used to present a combination of a visual and auditory US.

Global parameters

The file `Design/Parameters.csv` contains some parameters that affect the whole experiment. Here is a sample file (as above, the file is in CSV format, displayed here as a table for legibility):

Parameter	Value
CSDuration	4000
CSUSInterval	0
USDuration	400
Response	space
ResponseTimeMin	0
ResponseTimeMax	4000
MinITI	1000
MaxITI	3000
MaxResponses	100
BackgroundColor	gray95
ForegroundColor	black
FontName	Vera

FontSize	36
Test	0

Table 13: Sample `Parameters.csv` file with default values for parameters.

CSDuration is the default duration of all the non-US stimuli, while **USDuration** is the default duration of all US stimuli. All durations are in milliseconds. Note that you can set different durations for different stimuli by including a Duration column in the `Stimuli.csv` file. When using compound stimuli, all components must have the same duration.

CSUSInterval and **USDuration** should be self-explanatory.

ReactionTimeMin and **ReactionTimeMax** define at what times within a trial subjects can respond. Responses outside this time window are registered with a special code (see Data format) no USs are delivered. If not specified, ResponseTimeMin is set to 0 and ResponseTimeMax to CSDuration, thus allowing responses anywhere in the trial.

MinITI and **MaxITI** are the minimum and maximum values of the inter-trial interval. Each inter-trial interval will be drawn between these values with uniform distribution.

Response is the key subjects are instructed to press if they want to respond. Note that this can also be set on a per-stimulus basis, see here.

MaxResponses is the maximum number of response a subject is allowed to make in one trial. There are essentially two useful settings. If you set this to 1 the trial ends with the first response (the US is delivered if appropriate, of course). If you set it to an unrealistically large value, say 1000, you can record any number of responses per trial. Each of these may result in the US being delivered, as described above. Note that you can set MaxReponses to a different value for different trial types, by including a MaxResponses column in `Phases.csv` (see the section on text files for an example). If a MaxResponses column exists, but the value is empty for some stimuli, the MaxResponses value in `Parameters.csv` will be looked up. If MaxResponses is not set there, it is given a default value of 1.

The next few parameters control the screen background color while the experiment is running and the color, font, and size of text used for instructions and other messages.

The **Test** parameter, if set to 1, skips instructions and acquisition of demographic information. It is meant to quickly start the experiment during development.

Responses and classical vs. instrumental trials

If we wish to record only one kind of response, e.g., space bar presses, the Response key can be specified in the `Parameters.csv` file. We can also, however, specify different responses for different stimuli by adding a Response column to the `Phases.csv` file. For example, to

specify that the left arrow key is the correct response for stimulus Red, but the right arrow is correct for White, you would write as in Table 14.

Phase	Stimulus	Trials	Reward	US	Response
1	Red	20	1	Smiley	
1	White	20	1	Smiley	

Table 14: A `Phases.csv` specifying different responses for stimuli Red and White.

Here `<left>` and `<right>` are special codes that denote the left and right arrow key. You can look up the codes for different special keys in the "Keyboard Entry" section of the PEBL manual. If you only want to use letter and number keys, you simply can write the letter or number as a Response.

There are two special response codes. One is `space`, indicating a space bar press. We made this special because the space would be hard to see when editing the CSV file.

The other special response code is obtained by prefixing the response with a `*` (asterisk). This means that the US will be displayed *only* at the end of the trial (with the appropriate Reward probability) *regardless* of what the subject does during the trial, as in classical conditioning or causal rating studies. Thus an entry like:

Phase	Stimulus	Trials	Reward	US	Response
1	Red	20	.9	Smiley	*space

Table 15: Sample `Phases.csv` file.

Specifies that Red is to be rewarded 90% of the time at the end of a trial, *regardless* of whether the subject responds or not. Note that subject responses are still recorded, and if they exceed the allowed maximum the trial terminates without reward. This last feature makes it possible to implement omission training, i.e., reward subjects only when they abstain from responding. This is controlled by the MaxResponses parameter. The default value is 1, which corresponds precisley to omission training. If you don't want the trial to ever terminate before the allotted time, you can use a value of MaxResponses so high that it cannot be possibly reached, such as 1000.

Note also that on `*` trials, the ResponseTimeMin and ResponseTimeMax features are disabled (see Global parameters). Because the US (if any), is delivered only once at the end of the trial, it is irrelevant when subjects responds.

Instructions and other text displays

Instructions or other longish text can be displayed with the textfile stimulus type. For example, to include both a start and an end message (say a 'thank you' or similar) you can use something like:

Phase	Stimulus	Trials	MaxResponses
Start	StartText	1	1
End	EndText	1	1

Table 16: Sample `Phases.csv` file.

Name	Type	Parameters	Color	XOffset	YOffset	Duration
StartText	textfile	Start.txt				600000
EndText	textfile	End.txt				600000

Table 17: Sample `Stimuli.csv` file.

As you see, the display of instructions is construed simply as a stimulus that stays on for a long time (here 10 minutes), unless the subject performs the required response (which, recall, is by default the space bar). The Start.txt and End.txt files will be looked for in the Materials folder of the experiment. Note the column MaxResponses in `Phases.csv`, which makes sure the user has to press the space bar (the default response) only once to move on, even if a larger number of responses is allowed for actual experimental trials.

Data Format

When you run an experiment with alex, data are saved in the Data folder (which alex creates if it does not find) in CSV files named with subject numbers, i.e., Data/1.dat and so on. These files have a header followed by one data line per trial. The first columns replicate the `Subjects.csv` line for the particular subject. This is so that each line identifies all independent variables it pertains to (so called "long format" in statistical software). Another reason for including this information is that in this way you don't have to load it into your statistical software from other files and then merge the data.

The other columns of the data files are as follows:

- **Sex**: subject sex (collected by alex at the start of experiments).

- **Age**: subject sex (ditto).
- **Phase**: experimental phase the trial belongs to.
- **Trial**: trial number within the phase.
- **Stimulus**: stimulus presented in the trial (one of those defined in the `Stimuli.csv` design file).
- **RewardPr**: reward probability assigned to the stimulus (from the `Phases.csv` design file).
- **Response**: which key was monitored on that trial. Recall that `space` is a special code for the space bar and that the key may be prepended by `*` (asterisk) if the trial was a "classical conditioning" one (see here).
- **Responses**: The number of times the subject responded to the stimulus. This includes *all* responses, even those that may have bee registered at invalid times (see ReactionTimeMin and ReactionTimeMax above). See *Rewards* below for how to obtaine detailed information about valid and invalid responses.
- **RTs**: Reaction times for *all* registered responses (see *Responses*). This is a comma-separated list.
- **Rewards**: A comma-separated list of rewards received for each response. Each element has three possible values:
- 1: The response was rewarded (the US was presented)
- 0: The response was not rewarded (no US presented)
- -1: The response was invalid, i.e., it fell outside of the window delimited by ReactionTimeMin and ReactionTimeMax, see above. No US is presented on such responses.

We think this information characterizes subject behavior competely, but please do let us know if you think details could be added.

Troubleshooting

Errors may arise if Design files have incorrect or incomplete information. Presently, alex performs some checks at startup, but some errors are caught only as they occur while running the experiment. We advise to always run the experiment a few times before putting it into production. If you think errors are due to bugs in alex, please write us at the address in Contacts. Also do contact us if you think that your design files are correct but the experiment does not run as you expect.

With a few exceptions, all errors print a hopefully informative message both on the standard console output (terminal) and on screen. A few errors that may occur before the screen is

set up, such as not finding necessary files, are reported only on the standard output. On Windows, these messages will appear in files `stdout.txt` and `stderr.txt`, which PEBL creates in the folder where alex is run.

There is one error that appears mysterious to the uninitiated: the screen remains black and alex hangs forever. The reason is that alex uses a lock system on the `Subjects.csv` file to prevent concurrent instances of alex from running the same subject. The lock is held for as little as possible, but if you interrupt alex at a critical time, or if alex crashes for any reason before the lock is released, subsequent instances of alex will wait forever for the lock to be released. In these cases, you can simply delete the lock file, which is `Subjects.csv.lck` in the Design folder.

Contacts

Please send suggestions to improve alex or this manual to Stefano Ghirlanda, drghirlanda@gmail.com.