Alex: Associative Learning EXperiments

Stefano Ghirlanda Max Temnogorod

Contents

Introduction

Alex is a program for running associative learning experiments described in configuration files. This manual explains how to configure the 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.

Workflow

To build a new experiment you create a dedicated folder, 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 (see the section on text files).
- **Data**: This folder holds the data collected during experiment runs. See Data format.
- **Logs**: This one holds runtime messages recorded for each subject.

The program `alex-init`, run as follows, generates a bare-bones experiment with the files you'll need:

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

This creates folder `<experiment name>` with the above-mentioned subfolders (except Data and Logs, which are created on first run) and skeleton configuration files. It also creates a `RunExperiment` script (`.sh` on Linux and OS X, `.bat` on Windows) that can be double-clicked to run alex in the folder, as an alternative to the command line method described below.

Acknowledgements

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

Starting and stopping alex

From the folder where the Design and Materials folders are, you can just type `alex` in the command line. You can also run experiments in a folder other than the working directory by typing:

```
alex -v <path to folder>
```

The folder is expected to have Design and Materials subfolders with the appropriate files. Data and Logs subfolders will be created 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 `Groups.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. 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 (e.g., that they have the appropriate number of lines). Although this may be inconvenient, 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 (e.g., with an `incomplete-` prefix) and alex will automatically re-run that subject.

To interrupt a running experiment, you can use the interrupt key combination for PEBL: `Ctrl+Alt+Shift+\`.

Configuration files

All configuration files are in the Design folder:

- `Phases.csv` describes the experimental design proper. It defines experimental phases in terms of the stimuli, desired responses, possible outcomes, and number of trials in each.
- `Stimuli.csv` specifies the presentation details of the stimuli mentioned by name in `Phases.csv`.
- `Groups.csv` specifies the number and size of experimental groups, as well as the treatments to which subjects in each group are allocated.
- `Parameters.csv` defines some global parameters such as screen background color, text color, font, and size, the duration of intertrial intervals, and so on. It can also be used to define parameters that are the same for all stimuli, such as a default key used for responses.

The `Phases.csv` file

Suppose we want to teach subjects 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.[1] The file describes an experiment with two phases, with each line defining one type of trial that occurs in a phase. Here, there are two kinds of trials in `Phase 1`, specifying 20 presentations of each of two stimuli (S1s) called `Red` and `White`. `Red` will be rewarded 90% of the time, `White` only 10% (specified by `S2Prob`). On these trials, stimulus `Smiley` will be presented as the reward (`S2`). In `Phase 2`, stimulus `Pink` is presented five times and never rewarded.

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	S1	Trials	S2Prob	S2
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 test responding to Pink. Note that the S2 field can be left empty if probability of S2 presentation is 0.

[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 spreadsheet 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 spreadsheet software uses double quotes by default, but do check in case alex cannot read your CSV files.

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.csv` files in Tables 1 and 2 are equivalent, but the file in Table 3 runs `Phase 2` before `Phase 1`.

Phase	S1	Trials	S2Prob	S2
1	Red	20	0.9	Smiley
2	Pink	5	0	
1	White	20	0.1	Smiley

Table 2: With this `Phases.csv` file, Phase 1 will be run first.

Phase	S1	Trials	S2Prob	S2
2	Pink	5	0	
1	Red	20	0.9	Smiley
1	White	20	0.1	Smiley

Table 3: With this `Phases.csv` file, Phase 2 will be run first.

Stimulus terminology

The above experiment and other examples in this manual refer to reward training in which the Smiley is an unconditioned stimulus (US) that serves to reinforce the conditioned stimuli (CSs) designated as S1s. We maintain the more generic S1/S2 terminology, however, in order to have alex equally applicable to trials where both stimuli are neutral (as in higher-order conditioning), and where S2 would not be considered a reward.

The `Stimuli.csv` file

How does alex know that `Red`, `White`, and `Pink` mentioned in the `Phases.csv` files in Tables 1–3 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	Duration
Red	square	50	red	0	0	1000
White	square	50	white	0	0	1000
Pink	square	50	255-128-128	0	0	1000
Smiley	image	smile-o-white.png		0	-150	1000

Table 4: A `Stimuli.csv` file instructing alex that stimuli Red, White, and Pink are differently colored 50x50 pixel squares, and that Smiley is an image contained in the file `smile-o-white.png`.

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

- **Name**: A label for the stimulus, so that it can be referenced in `Phases.csv`. This can be anything that

does not contain the characters " (double quote), + (plus), * (asterisk), : (colon), or , (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 length 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 an image or sound file in the Materials folder. An optional zoom factor, separated from the filename by a + (plus), can be provided to scale an image to a desired size. 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 black 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, you 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 RGB triplets, delimited by hyphens (-). 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` is the foreground and `Color2` the background. If no foreground or background color is given, the defaults set in `Parameters.csv` are 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**: Offset from the center of the screen, in pixels. In Table 4, all stimuli are centered except for `Smiley`, which is displayed 150 pixels above center (negative Y values place stimuli above center, negative X values place them left of center).
- **Duration**: Stimulus duration, in milliseconds.
- **Onset**: Stimulus onset, in milliseconds. An onset of 0 means that the stimulus is diplayed immediately at the beginning of a trial. Positive onsets delay stimulus display. Negative onsets are not allowed. **Note:** By using onsets, you can use stimulus sequences as either S1 or S2.

The `Groups.csv` file

The `Groups.csv` file contains information about the experimental groups. If all subjects undergo the same treatment, you only need to specify one group and its size. The file in Table 5 instructs alex to run a single group of 10 subjects (groups can be numbered or named, as is most convenient to you). Often, however,

subjects need to be divided into different treatment groups. If you want to test, say, two shades of pink, you would extend the `Stimuli.csv` file in Table 6. The special value `*` here indicates that the color of stimulus `Pink` will be looked up, for each subject, in the column `PinkColor` of the `Groups.csv` file (Table 7). This syntax is available for all stimulus properties. For example, to vary the size of the red square across subjects you would use the `Groups.csv` and `Stimuli.csv` files in Tables 9 and 8.

Group	Size
1	10

Table 5: A `Groups.csv` file instructing alex to run 10 subjects.

Name	Type	Parameters	Color	XOffset	YOffset	Duration
Red	square	50	red	0	0	1000
White	square	50	white	0	0	1000
Pink	square	50	*	0	0	1000
Smiley	image	smile-o-white.png		0	-150	500

Table 6: A `Stimuli.csv` file instructing alex to look up the Color of the Pink stimulus in the `Groups.csv` file (see Table 7).

Group	Size	PinkColor
1	10	255-128-128
2	10	255-190-190

Table 7: A `Groups.csv` file instructing alex to run 20 subjects split in two treatment groups with different Color attributes for the Pink stimulus (see Table 6).

Name	Type	Parameters	Color	XOffset	YOffset	Duration
Red	square	*	red	0	0	1000
White	square	50	white	0	0	1000
Pink	square	50	*	0	0	1000
Smiley	image	smile-o-white.png		0	-150	500

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

Group	Size	PinkColor	RedParameters
1	10	255-128-128	25
2	10	255-128-128	50
3	10	255-190-190	50
4	10	255-190-190	75

Table 9: A `Groups.csv` file instructing alex to run 4 experimental groups, each receiving a unique combination of PinkColor and RedParameters (see Table 8).

A `Groups.csv` file can also contain a `PhaseOrder` columns. This mechanism provides the most flexible allocation of phases to groups, include leaving some phases out or rearranging the order of phases for different groups. The syntax is simple: `PhaseOrder` should contain a +-separated list of phases, each of which has been defined in `Phases.csv`. Table 10 provides an example. Note that using the `PhaseOrder` mechanism makes it possible to list phases in `Phases.csv` in any order. Which phases are run, and in which order, is determined entirely by the `PhaseOrder` parameter. (It is possible to leave `PhaseOrder` empty for one or more groups, in which case pahses are run in the order they appear in `Phases.csv`, see above.)

Group	Size	PhaseOrder
1	10	Task1+Task2
2	10	Task2+Task1
3	10	Task1

Table 10: A `Groups.csv` illustrating the `PhaseOrder` parameter. Three experimental groups are defined. Groups 1 and 2 differ in the order in which tasks 1 and 2 are administered; group 3 is administered only task 1. (Phases containing startup or end messages do not appear in the table, but must be included if desired.)

The `Parameters.csv` file

The `Parameters.csv` file contains some parameters that affect the whole experiment. Here is a sample file:

Parameter	Value
S1S2Interval	0
MinITI	1000
MaxITI	3000
Response	<space>
ResponseTimeMin	0
ResponseTimeMax	4000
MaxResponses	100
MaxInvalid	0
BackgroundColor	gray95
ForegroundColor	black
FontName	Vera
FontSize	36
Test	0
Log	1

Table 11: A sample `Parameters.csv` file with default values.

- **S1S2Interval** is the interval between S1 offset and S2 onset. (It can be modified per-S2 by defining

S2's with different onset time.)

- **MinITI** and **MaxITI** are the minimum and maximum values of the intertrial interval. Each intertrial 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-trial basis, see here.
- **ResponseTimeMin** and **ResponseTimeMax** indicate when, within a trial a response is considered *valid*. Valid responses are processed to possibly trigger the presentation of S2. Invalid responses cannot trigger S2.
- **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 S2 is presented 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 which may result in an S2 being presented. 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.
- **MaxInvalid** is the maximum number of *invalid* responses a subject is allowed to make in one trial. If you set this to 0 or 1, an invalid response terminates the trial (this is the default).
- 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.
- **AskID** determines whether the subject is asked to provide an ID code. (This can be useful to give credit.)
- **AskAge** and **AskSex** do what you think they do.
- **AskRace** asks subjects to check their own 'race or ethnicity' according to U.S. National Insitutes of Health classification. (This is not fully NIH compliament, however, because PEBL does not have a dialog box where you can check multiple items, as NIH would require to list more than one 'race or ethnicity.')
- **Log** is a toggle for the logging feature, which records runtime messages for each subject, in files named the same as corresponding data files except with a `.log` suffix. Disabled by setting to `0`.

More about stimuli

We mentioned above one bit of special notation in the definition of stimuli, namely the value `*` (asterisk). 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 `:` (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 file in Table 6, featuring three squares of the same size as stimuli. The file in Table 12 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	Duration
Red	square	50	red	0	0	1000
White	square	:Red	white	0	0	1000
Pink	square	:Red	*	0	0	1000
Smiley	image	smile-o-white.png		0	-150	1000

Table 12: 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 13 and 14.

Phase	S1	Trials	S2Prob	S2
1	Red	20	0.9	Smiley
1	White	20	0.1	Smiley
2	Red+White	5	0	

Table 13: A `Phases.csv` file with a compound stimulus in Phase 2.

Name	Type	Parameters	Color	XOffset	YOffset	Duration
Red	square	50	red	0	0	1000
White	:Red	:Red	white	60	:Red	1000
Smiley	image	smile-o-white.png		0	150	1000

Table 14: A `Stimuli.csv` file to go with the `Phases.csv` file in Table 13. 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 S2s. This can be used to implement S2s 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 S2s may also be used to present a combination of visual and auditory stimuli.

Superposition of stimuli

Visual stimuli are added to the screen in the order they appear in the `Stimuli.csv` file. This means that, should some stimuli overlap on the screen, those defined *later* will be displayed *on top* of those defined earlier, obscuring them partly or wholly.

Always-present stimuli

It is sometimes desirable to have a stimulus or combination of stimuli present at all times during a phase, including interstimulus intervals, for example as a background on which others are superimposed. A stimulus

whose name starts with `Background` followed by the name of a phase will be displayed for the entire duration of that phase. You can define many such stimuli, e.g., `BackgroundPhase1-1` and `BackgroundPhase1-2`.

Note: The rules for stimulus superposition of always-present stimuli are the same as for other stimuli, see above. This means that if you want to use a stimulus as a backdrop for other stimuli, you have to define the stimulus before all those that are intended to appear on top of it. If the order is incorrect, the intended backdrop will instead obscure the other stimuli.

More about phases

As with stimuli, phase parameters can be set to differ across groups by using `*` notation. For example, let's suppose we want to investigate how discrimination learning varies with reward probability. We could use the `Phases.csv` file in Table 15, which employs `*` notation for the `S2Prob` variable, and the `Groups.csv` file in Table 16, which provides the information that is "starred" in `Phases.csv`.

Phase	S1	Trials	S2Prob	S2
Training	A	50	*	Smiley
Training	B	50	0	

Table 15: A `Phases.csv` using `*` notation to indicate that the value of S2Prob for Training trials with S1 A must be looked up in `Groups.csv` (see Table 16).

Group	Size	TrainingAS2Prob
Rich	20	1
Poor	20	0.5

Table 16: A `Groups.csv` file serving as a companion to the `Phases.csv` file in Table 15.

Note that the name of the corresponding column in `Groups.csv` is `TrainingAS2Prob`, or, more generally, (`phase name`)(`S1 name`)(`parameter`). Thus the column name specifies two things: the phase and the S1 to which the column value refers (in employing the same notation for stimuli, we had to worry only about the stimulus name). This works also to set phase parameters for a compound stimulus. For example, if you want to set the `S2Prob` value for S1 `A+B`, you would use the column `TrainingA+BS2Prob`.

Responses and classical vs. instrumental trials

The default `Response` key for all S1s can be specified in `Parameters.csv`. We can also, however, specify different responses for different S1s by adding a Response column to the `Phases.csv` file. For example, to specify that the left arrow key is the correct response for the S1 `Red` in `Phase 1`, but that the right arrow is correct for `White`, you would write as in Table 17.

Phase	S1	Trials	S2Prob	S2	Response
1	Red	20	1	Smiley	<left>
1	White	20	1	Smiley	<right>

Table 17: A `Phases.csv` specifying different responses for S1s Red and White in Phase 1.

Here <left> and <right> are special codes that denote the left and right arrow keys. The following is a comprehensive list of valid key codes that can be used to specify correct responses:

- Characters: a–z, 0–9, all standard punctuation (except braces, pipes, tildes, and percent signs)
- Edit keys: <space>, <backspace>, <tab>, <clear>, <kp_enter>, <return>, <insert>, <delete>
- Modkeys: <lshift>, <rshift>, <lctrl>, <rctrl>, <lalt>, <ralt>, <lmeta>, <rmeta>, <numlock>, <capslock>, <scrollock>
- Navigation, function keys: <up>, <down>, <left>, <right>, <home>, <end>, <pageup>, <pagedown>, <esc>, <f1>–<f15>

Another special response code is <classical>. Specifying a response as such, either for a particular phase or as the default for all S1s, effectively means that the S2 will be displayed *only* at the end of the trial (with the appropriate S2Prob) *regardless* of what the subject does during the trial, as in classical conditioning or causal rating studies. Thus the Phases.csv file in Table 18 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 S2 presentation. This last feature makes it possible to implement omission training, i.e., to reward subjects only when they abstain from responding. This is controlled by the MaxResponses parameter. The default value is 1, which corresponds precisely 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.

Phase	S1	Trials	S2Prob	S2	Response
1	Red	20	.9	Smiley	<classical>

Table 18: A Phases.csv file using the Response notation <classical> to indicate a classical conditioning trial in which the S2 is presented at the end of the trial regardless of subject behavior.

Note also that on classical trials, the ResponseTimeMin/Max features are disabled (see Global parameters). The S2 (if any) is presented only once at the end of the trial, so it is irrelevant when a subject 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 Phases.csv and Stimuli.csv file like those in Tables 19 and 20 to include the presentation of text files that are displayed until the subject responds once. As you see in these tables, 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, by default, is 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 only has to press the space bar (the default response) once to move on, even if a larger number of responses is allowed for actual experimental trials.

Phase	S1	Trials	MaxResponses
Start	StartText	1	1
End	EndText	1	1

Table 19: A `Phases.csv` file for displaying to subjects instructions and a final message (see also Table 20).

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

Table 20: A `Stimuli.csv` file for displaying to subjects instructions and a final message (see also Table 19).

Data format

When you run an experiment with alex, data are saved in the Data folder (which alex creates if it is not found) in CSV files named with group names and subject numbers, e.g., `Data/Training-1.csv` for the first subject of group `Training`. These files have a header followed by one data line per response. This is so that each line identifies all variables it pertains to (so called "long format" in statistical software) and can be loaded easily into statistical software without having to manually add data.

The first few columns of each data line consist of the hostname, followed by the group, subject number, and pertinent treatments as specified in the `Groups.csv` line for the particular subject. The remaining columns are as follows:

- **Sex**: Subject's sex (collected by alex at the start of experiments, otherwise `NA`).
- **Age**: Subject's age (ditto).
- **Time**: Time when response was made (since start of experiment, in ms).
- **Phase**: Experimental phase the trial belongs to.
- **Trial**: Trial number within the phase.
- **S1**: Designated S1 for this trial, or `ITI` for responses registered between trials.
- **S1Duration**: Duration of S1 (or intertrial interval).
- **S1On**: Was S1 present when the response was made? (`T` for true, `F` for false.)
- **S2**: Designated S2 for this trial (or `NA` if not specified in `Phases.csv` and during ITIs; `S2Duration` and `S2Prob` are also `NA` in these cases).
- **S2Duration**: Duration of S2.
- **S2On**: see `S1On`.
- **S2Prob**: Probability of S2 presentation, given a correct response (both specified in `Phases.csv`).
- **Response**: Key designated as the correct response (`NA` during ITIs), or the code `<classical>` if the trial was a "classical conditioning" one (see here).
- **RT**: Response time since start of trial (`NA` if trial timed out).
- **S2Pres**: Did this response result in S2 presentation? (For classical trials: was an S2 scheduled for the end of this trial?) (`T` or `F`).
- **Key**: Subject's actual response. This can be the correct key, any other key the subject may have pressed, or `<timeout>` in the case of no responses within a trial (the goal is to have a faithful record of everything the subject does).

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

Troubleshooting

Errors may arise if design files have incorrect or incomplete information. 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. When running alex through the PEBL launcher, these messages will appear in files `stdout.txt` and `stderr.txt`, which PEBL creates in the folder where alex is run.

In addition to double quotes in CSV design files (see the footnote below Table 1), alex requires final line endings in these files. If a design file lacks a newline at the very end, alex will exclude the last row when reading in this file, resulting in obvious errors. Most spreadsheet software saves CSV files with this final newline, but be sure to check for this if you encounter any problems with running your experiment.

There is another error that will appear mysterious to the uninitiated: the screen remains black and alex hangs forever. The reason is that alex uses a lock system on the `Groups.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 `Groups.csv.lck` in the Design folder.

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 below. Also do contact us if you think that your design files are correct but the experiment does not run as you expect.

Contacts

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