0.1 Introduction

Speriment is software that facilitates the design and running of experiments over the internet. It's built to work with PsiTurk (**?**), a program that runs arbitrary JavaScript experiments on Mechanical Turk and other platforms. PsiTurk takes much of the work out of interfacing with Mechanical Turk and managing participants. Speriment further reduces the workload of the experimenter. Instead of writing a dynamic website in JavaScript, Speriment users can write a description of the structure of their experiment in Python, and the JavaScript will be generated automtically. For instance, instead of implementing the Latin Square algorithm to choose questions to show a given participant, the experimenter simply specifies that the block of questions uses a Latin Square.

0.2 Comparison with Other Frameworks

There are many other frameworks for creating online experiments, all with different features, strengths, and weaknesses.

0.2.1 Ibex

Ibex (**?**) is a program that generates a website from a JavaScript description. Unlike Speriment, the original version of Ibex does not separate the materials from the structure of the experiment. Ibex version 2 will do so, as it makes scripts easier to write and more robust to change in the materials. Ibex experiments can be run on IbexFarm, which is hosted by Drummond, whereas Speriment is hosted by the experimenter. IbexFarm eliminates the need for experimenters to run their own server, but also limits their flexibility. For instance, one of the strenghts of PsiTurk is that it manages participant data with a server-side database, and uses this database to balance participants across conditions. IbexFarm does not do this and to my knowledge, an experimenter would not be able to add this feature to their workflow if they are taking advantage of Drummond's hosting.

Ibex uses units of code called controllers to turn the experimenter's description of the experiment into an actual website. This makes it easy for an experimenter who knows JavaScript to add functionality to the structure of the experiment, as they can simply write a new controller. Not all features can be added via controllers, but Ibex is nevertheless one of the more extensible experimental frameworks.

0.2.2 Experigen

Experigen (**?**) also generates the experiment's website from a description of its structure in JavaScript.

0.2.3 WebExp

WebExp (**?**)

0.2.4 Survey Software

There are several websites and programs available for running online surveys, and some experiments are expressible using these tools. However, the difference in priorities for surveys and experiments means that many experiments are difficult or impossible to create using survey software.

SurveyMonkey (**?**) is one of the most popular survey-making tools, but it does not support some key features of experiments. For instance, users must have a premium account in order to randomize question order, which is nearly always required in experiments.

LimeSurvey (**?**) is an open source project primarily designed to enable the creation of online surveys. It has some of the features crucial to experimental designs and can sometimes be used, but it lacks other key features such as Latin Squares.

SurveyMan (**?**) was the inspiration for Speriment, and the two systems have different strengths. SurveyMan provides survey hosting, saving the experimenter the trouble of running Unlike SurveyMonkey, SurveyMan randomizes question order by default — a good choice for surveys for the same reason it is a good choice in experiments. SurveyMan is a particularly advanced tool for creating surveys because it enables the experimenter to detect which participants appear to have answered at random, so that their data can be excluded from the analysis. However, SurveyMan lacks key experiment-specific features, such as the ability to automatically distribute questions according to a Latin Square. The goal of Speriment is to address these needs.

0.2.5 PsiTurk

PsiTurk can be used without Speriment, and the choice of whether to do so is based on the classic trade-off between ease of use and power. Speriment makes it easier to create an experiment with PsiTurk because the experimenter need only describe what the experiment is in Python, while without it the experimenter must implement most of its functionality in JavaScript, with the help of a few PsiTurk utilities for showing a consent form and the like. Yet Speriment can only express a subset of the experimental designs one could implement in JavaScript.

0.2.6 Speriment

Speriment fills a niche in this ecosystem of experimental frameworks. The philosophy guiding its development is that simple experiments should be simple to write, and an experimenter writing a simple experiment should not have to know about the more complicated options available to him or her. Thus, Speriment will never be able to express all conceivable experiments, the space of

which is vast. However, Speriment will continue to grow to encompass useful experimental designs that do not interfere with the descriptions of simple experiments. For those designs which are beyond the abilities of Speriment, PsiTurk paired with pure JavaScript is likely to be a good option, since JavaScript is a Turing-complete language and PsiTurk limits the possibilities of experiments very little.

0.3 Components and Features

A full user guide for Speriment is available on the GitHub repository at https://github.com/presleyp/Speriment. However, it is useful to discuss the components that make up a description of an experiment in Speriment and the features that each component currently provides.

0.3.1 Pages

Speriment descriptions are made up of pages. A page represents one view of the experiment; whatever information is displayed at one moment in time. These can be questions, instructions, or niceties such as a welcome or thank you page.

The order of pages is randomized by default. If they are assigned conditions, they may also be pseudorandomized so that no two pages of the same condition will appear in a row.

By default, all pages display eventually. However, the experimenter may create groups of pages when not all participants should see all pages. For a given participant, one page will be chosen from each group. This choice may be done randomly or according to a Latin Square.

0.3.2 Options

Pages that pose questions should contain one or more options. Pages take arguments to describe properties of their option sets. Option sets can be exclusive, so that only one of them can be selected, or inclusive. Their order on the page is randomized, and they can be specified to be ordered or unordered. Ordered options, such as a scale of numbers, will not have their order fully randomized, but only kept in place or reversed. Finally, options can be free text, which produces a text box. Exclusive options display as radio buttons and inclusive options display as checkboxes, except if there are more than seven of them, in which case both exclusive and inclusive options are displayed in a drop-down menu. Options can be selected via mouse or keyboard.

0.3.3 Features of Pages and Options

Both pages and options can be associated with tags. Tags do not affect the implementation of the experiment, but serve to simplify the process of analyzing the resulting data. For instance, pages can be tagged with the type of the page,

so that instructional pages can be easily filtered out. Options can be tagged with the aspect of them that is relevant to the analysis. For instance, in an experiment designed to test whether people prefer cats or dogs, "Garfield" and "Stimpy" could have "cat" as the value for their "Species" tag while "Odie" and "Ren" have "dog." This eliminates the need to merge a table associating names with species into the table of results.

Pages and options can also have resources. Resources can be images, audio files, or video files. Page resources will display centered at the top of the page and option resources will display next to the option.

Speriment allows training blocks, which will be explained further below. Pages and options have two features to support the use of training blocks: correctness and feedback. Pages can specify which of their options is correct and options can specify whether they are correct or what regular expression a text answer must match to be correct. Pages can specify a feedback page that should show following the participant's response, or an option can specify a page that should show if it is selected. Correctness is used to determine whether the participant has mastered the task, and feedback can be used to help participants do so.

0.3.4 Sampling

The text, tags, correctness, and resources of pages and options are typically all specified with constants that remain the same across participants. However, they can also be specified via a SampleFrom component. SampleFrom takes the name of a bank containing text or the filenames of resources. These banks are passed to an enclosing block or the entire experiment. The SampleFrom component will sample one string from the bank randomly at runtime, so that the selection varies across participants. By default, sampling is without replacement, so that once a string has been used for a page or option for a given participant it will not be used for a different page or option for that participant. However, there is an option for sampling with replacement. If a more complex relationship among items is needed, experimenters can specify a variable to associate with the sampled string. This variable can be any string or number. When a variable is passed to two SampleFrom components, they will both sample the same string. Conversely, if the same string or number is passed as `variable` in one SampleFrom and `not_variable` in another, they will not sample the same string. To create relationships between different kinds of stimuli, the experimenter can put dictionaries rather than strings in banks. The `variable` and `not_variable` arguments manage the selection of a dictionary from the bank, and an additional argument, `field`, specifies which key to access in the dictionary to arrive at a sampled string.

The SampleFrom feature is one of the more complex and powerful features of Speriment. It enables experimenters to minimize correlations in their data without trying every possible combination of, for instance, page text and resources.

0.3.5 Blocks

Pages are grouped into blocks. While pages are shuffled by default, blocks display in the order in which they are specified by default. Thus, blocks enable experimenters to specify ordering. However, blocks can be made exchangeable. Exchangeable blocks are those that are allowed to switch places with each other. This feature is more powerful than one that allows shuffling, because it enables the first and third block, for instance, to trade places, while leaving a middle block in place. This can be useful for randomizing the order of test stimuli while leaving instructions and breaks in the sensible spots. The location of an exchangeable block is determined randomly, while the location of a counterbalanced block is deterministic. The difference is that counterbalancing is more likely to give a balanced distribution of orderings across participants.

Blocks can also be assigned to treatments. A given participant will also be assigned to a treatment, and will only see blocks that are not in a treatment, and blocks that are in their treatment, excluding blocks that are in a different treatment. Treatments are determined through the same mechanism as counterbalancing, so if both features are desired, treatments should be paired with exchangeable blocks to avoid unwanted correlations.

Another way to decide if a block should run for a given participant is to base the decision on whether the participant answered a question a certain way. RunIf is an experimental component that specifies the condition that must be met for its enclosing block to run. It specifies a page and the answer that must have been given on that page for the current block to run — either the option that must have been selected or a regular expression that the text answer must match. Conditional running of blocks is useful in two types of situations. In the first, there is a choice between running a block and not running it or anything in its place. For instance, a question about whether the participant enjoys skiing could be followed up with a block of questions about ski experiences, or not. The second case is a choice between multiple blocks. For instance, a question about native language could be followed by a block of questions in whichever language was selected.

0.3.6 Speriment's Repertoire

The options provided for these components make up the repertoire of features available to experimenters using Speriment. Not all conceivable experimental designs are expressible through Speriment, but many of those commonly used in linguistics are. There are plans to add a few more features to cover additional types of linguistic experiments. The most commonly requested additional features are multi-page sequences for use in self-paced reading experiments and increased flexibility of the layout of the page.

0.4 Workflow

The workflow when using Speriment begins the same way as the workflow for using PsiTurk alone. The experimenter downloads and installs PsiTurk, creates a new project, sets up a database, and edits the configuration file and templates provided by PsiTurk.

At this point, Speriment comes in. The experimenter installs Speriment's Python and JavaScript packages, writes a Speriment script in Python, and runs it. The script edits the project's PsiTurk files automatically to use the experiment described in the Python script. A Speriment script usually contains the following parts:

1. An import statement, to make Speriment's Python classes and functions available in the script.

2. One or more lines, possibly using Speriment's `get_rows` or `get_dicts` utility functions, reading in data about the experimental stimuli. It is common to arrange the materials in a csv file with one row per item and one column per piece of information (text, condition, and so on) about the item.

3. A `with` statement creating an ID generator for use within the experimental components. This is an optional utility that can create unique identifiers automatically so that they don't have to be specified in the materials.

4. Lines of code creating pages and options. Frequently, this will be done with a for loop or list comprehension over the rows read in from a csv file. One page and one or more options will be created from each row. However, experimenters are free to create pages however they prefer; the materials do not need to be in any particular format in the csv file.

5. Lines of code creating blocks for the pages.

6. A line creating an experiment for all of the blocks.

7. A line naming and installing the experiment. This final line of the script validates the structure of the experiment, creates a JSON file describing the experiment, and edits PsiTurk files to use both Speriment and the JSON file for this experiment. It is possible to keep more than one script in the project directory for different designs of an experiment. As long as they are assigned different names, their JSON files will not interfere with each other. However, PsiTurk only points to the experiment in the project directory that was most recently installed, so it's good practice to run the script right before launching the experiment.

Once the experiment has been installed, the workflow is again as if PsiTurk were being used alone. The PsiTurk documentation explains how to use the shell to start a server, create a task on Mechanical Turk or elsewhere, and review workers.

Finally, the experimenter can use Speriment's shell command `speriment-output` to download data from the database in a useful format. Instructions are on the GitHub repository. PsiTurk also has a command for accessing results, but Speriment's command is tailored to the way Speriment records data for ease of use.