Michael Becker (michael@linguist.umass.edu) May 28, 2020

`Experigen` basics

1 Intoduction

Examples of what `Experigen` can do:

- Phonotactic wellformedness:
  `http://experigen.phonologist.org/russian/`
- Nonce word tasks:
  `http://experigen.phonologist.org/saloperie/`
- Artificial languages:
  `http://experigen.phonologist.org/affop/`
- Plural morphology:
  https://sandalo.phonologist.org/pesquisa/
- Funny languages:
  `http://experigen.phonologist.org/honaidah/`
  `http://ucl.phonologist.org/tang/chinese/cantomusic/`
- Demo with recording audio from participants:
  https://sdb.phonologist.org/experigen-demo/

Other options:

- Online: IBEX, pcIBEX, turktools, WebExp2, qualtrics...
- Offline: Praat, PsyScope, etc. etc.

2 Before you begin

You will need these things:

1. A designed experiment.
   Have a good idea for the design of your entire experiment, including instructions, practice items, stimuli, etc. If this is your first experiment, look at existing experiments for inspiration.

2. Web space.
   Your university provides some, e.g. for your personal website (not blog). Find out where it is.

3. An SFTP program.
   I like `CyberDuck`. There are many others.

4. A plain text editor.
   For OS X, I like the free `BBEdit`. I am told that `Notepad++` is good for Windows. If you use Linux, you already have your favorite text editor.

5. `Chrome`.
   You'll need it to follow this handout. Install `Firefox` as well if you don't have it.

3 Initial setup

1. Download and unzip the latest version of `Experigen`:

   `https://github.com/tlozoot/experigen`

2. Copy the `web` folder (or just its content) from `Experigen` to your webspace/server, e.g. a folder on your university website.

   `Experigen` doesn't need any special configurations or permissions.

3. You should have a functioning English wug-test on your server at this point. Open a browser and admire.

4 Testing a website

1. The console.
   This is your friend. I like Chrome's best. Go to `View` → `Developer` → `Developer tools`. Activate the `console` tab.

2. The Cache.
   This is your enemy. Click the settings cogwheel in the developer tools, and check the `Disable Cache (while DevTools is open)` box.

3. Other browsers.
   Test your website on various browsers and operating systems, including Windows and phones/tablets, and including Internet Explorer (yes). In principle, `Experigen` should work on a wide range of browsers, but this will always be a struggle. Your best bet is to know your audience and make sure your experiment works with the devices they use.

5 Your first edit

1. Directly on the server — not recommended.
   Use `CyberDuck` to navigate to the `views` folder, and find `intro.ejs`. Click the `edit` icon, and change something in the text.
   No record of this change is left on your computer = bad news.

2. Locally, with manual upload — easiest
   Edit your local `intro.ejs`, then upload to your server, overwriting the server's `intro.ejs`. Refresh your browser to see the result.

3. Locally, with syncing — easy
   Edit your local `intro.ejs`. Then use `rsync` (OS X and Linux, I believe you can rsync on Windows if you install `cygwin`) to update the server. On OS X, you can use the included `sync.command` file. There is also a sync command in `CyberDuck`. To not have to type your password each time, see `http://www.linuxproblem.org/art_9.html`.

4. Locally, with a testing environment — best:
   Follow the instructions in the `local_server` folder.

6 Upload your resources

1. Files

   (a) File names
   When you create files for use with Experigen (sound files, pictures, videos, or any other kind of file that goes on the internet), be conservative with the names you give them. To be on the safe side, only use a–z, 0–9, and _. Uppercase letters, fancy symbols, spaces, etc. may work on some servers, but why risk it.

   (b) Encoding
   All your text files should be UTF-8. Bad things happen otherwise. Do not use UTF-16, as Firefox ignores it silently.
   Line breaks should be LF or CRLF, not CR.

2. `items.txt`
   This is where you list the information that goes into trials.

   (a) Use a spreadsheet program (e.g. `LibreOffice`) to make a file that has one row for each of your items. If you are making a wug-test, for instance, you will have one row for every wug and one row for every filler.

   (b) The first row in the spreadsheet will specify column names. Each column name must be unique and non-empty.

   (c) The first column in the spreadsheet is special; it identifies your items. It must contain unique non-empty values.

   (d) Empty rows are allowed.

   (e) Export your spreadsheet to a text file
   In `LibreOffice`, choose `save as...` and under `file type` choose `Text CSV (.csv)`. Choose `{tab}` as the field delimiter, and an empty string as the text delimiter. Once the file is saved, verify that it's UTF-8, rename it to `items.txt` and upload it into the `resouces` folder.
   Or, copy the content from the spreadsheet and paste in your text file.

How `items.txt` gets converted to an array of objects:

- Say you start with this `items.txt` file:

stem	type	audio
ballon	iamb	ballon42.mp3
raccoon	iamb	racc543.mp3
possum	trochee	possur.mp3

- It will end up like this:

{ | stem | balloon | | type | iamb | | audio | ballon42.mp3 | , | stem | raccoon | | type | iamb | | audio | racc53.mp3 | , | stem | possum | | type | trochee | | audio | possur.mp3 | }

- Use a `.pairWith` command to add fields, e.g.

```
items = items.pairWith("view","stim.ejs")
```

{ | stem | balloon | | type | iamb | | audio | ballon42.mp3 | | view | stim.ejs | , | stem | raccoon | | type | iamb | | audio | racc53.mp3 | | view | stim.ejs | , | stem | possum | | type | trochee | | audio | possur.mp3 | | view | stim.ejs | }

- Pairing with a short list recycles the values, e.g.

```
items = items.pairWith("order",[1,2])
```

{ | stem | balloon | | type | iamb | | audio | ballon42.mp3 | | view | stim.ejs | | order | 1 | , | stem | raccoon | | type | iamb | | audio | racc53.mp3 | | view | stim.ejs | | order | 2 | , | stem | possum | | type | trochee | | audio | possur.mp3 | | view | stim.ejs | | order | 1 | }

3. Sound files

   (a) If you have them, save them in mp3 format. To convert a bunch of files in one go, use `Adapter` (OS X).

   (b) Optional: empty out the `sounds` folder (inside the `resources` folder).

   (c) Upload your sound files into the `sounds` folder.

4. Pictures and frame sentences

   (a) Pictures on the web are usually in `jpg` or `png` format. It's usually a good idea to make all your pictures the same size.

   (b) Optional: empty out the `pictures` folder (inside the `resources` folder).

   (c) Upload your pictures into the `pictures` folder.

   (d) If each item is always paired with the same picture(s)/frame sentence(s), put the file names of the pictures or your frame sentences in your `items.txt` file.

   (e) If items are randomly paired with pictures/frame sentences, create a `pictures.txt`/`frames.txt` file, following the same procedure you did for `items.txt`. It may be as simple as a single column with file names/frames in it, with a column name in the first row.

7 Setup

1. Use `footer.html` (in the `views` folder) to change the text that appears in the footer. You probably want to keep the footer free of links; you don't want participants to click on any links while they are doing the experiment.

   This file is written in standard HTML; use any HTML tutorial to learn more.

2. Use `style.css` (in the `setup` folder) to specify colors, fonts, etc. For right-to-left languages, this is where you specify `direction: rtl;`.
   This file is written in standard CSS. Use any CSS tutorial to learn more.

3. Use `settings.js` (in the `setup` folder) to specify the title of the browser tab/window, various default messages, and the width/visibilty of the progress bar.

   This file is written in standard Javascript, and specifies the properties of a Javascript object. You should probably change it minimally, i.e. just change the strings in quotes. You cannot use quotes inside quotes; type a single quote twice if needed.

   The `experimentName` field is important. The experimental results will be stored under this name in the database. You can use this, for instance, to

build the experiment with one `experimentName`, e.g. "pilot", then change to a different `experimentName` to start with a clean slate in the database. You can only use A–Z, a–z, 0–9 in this field.

8 Static pages

These are pages that contain instructions, a demographics form, a thank you note, etc.

1. Each page must have a button that leads to the following screen (a sound button or a continue button).

2. In your demographics form, each field must have a unique name.

3. These pages are written in EJS, which is HTML that is augmented with two special tags, `<%= %>` and `<% %>`. The contents of these tags is written in Javascript. Here are some examples of useful things:

   - `<%= continueButton() %>` makes a button.
     The text on the button is specified in your `settings.js` file.
   - `<%= continueButton("→") %>` makes a button with an arrow on it (old browser might not show it).
   - `<%= makeSoundButton("pharon.mp3") %>` will make a sound button. `Experigen` will expect to find the file in whatever you specified as your sounds folder, which by default is `resources/sounds`.

4. If you want to display a page piecemeal, put each piece inside a `div` with a `class="trialpartWrapper"` specification. Be sure to close these `div`'s, like so:

   ```
   <div class="trialpartWrapper">
        <%= continueButton() %>
   </div>
   ```

   These `div`s will appear in order, triggered by the press of a continue button or a response button. You can override the default order by specifying an `id` for each div, named `part1`, `part2`, etc.

9 Dynamic pages

These display a row from your `items.txt` file.

1. Column names can usually be referenced directly, e.g. `<%= singular %>` will display the content of the column "singular" of the current item.

2. Alternatively, use `this`, e.g. `<%= this["singular"] %>`.
   You will have to use it if your column name is a Javascript reserved word.

3. If you want to compose a column name, you have to use the `this` syntax, e.g. `<%= this["plural" + "1"] %>`.

4. Many experiments only need one dynamic page (`stimulus.ejs`), but you can have as many as you'd like.

10 Structure

The structure of your experiment is mostly in `design.js`.

1. This file is written in standard `Javascript`, with a few extra methods for arrays, such as `.shuffle()`, `.chooseRandom()`, and others.

2. `.pairWith()` essentially adds a field/column to your items. You must add a `view` field/column to every item, so minimally, you will have:
   `items = items.pairWith("view","stimulus.ejs")`.
   Equivalently, and more cumbersomely, you can add a column called `view` to your `items.txt`, and enter `stimulus.ejs` in every row.

3. Usually, the first part of `design.js` defines the trial block(s), e.g. item selection and randomization.

4. The second part includes `addStaticScreen` and `addBlock` commands.

To discuss: example of a `design.js` for a wug-test (the default)
To discuss: example of an artificial language experiment: `design_art.txt`

11 Read the results

The database:

1. There is a database somewhere in the cloud at `https://sdb.phonologist.org/experigen1/`. I don't know where it is physically. You are welcome to use it if you'd like. Sometime it goes offline for a while. If so, wait until it goes back online again. I don't control it.
2. If you want to set up your own database server, use the cgi scripts in the `dbserver` folder.
3. The database has no security to speak of. If this level of security is not sufficient for your purposes, `Experigen` is not for you.
4. Experigen is provided as is and with no guarantee of any kind.

Read the results through R, even if you want to use a spreadsheet program to look at them.

1. In `R`, open the `getresults.R` file.
2. Update the `experimentName` field, which should match the one you specified in your `settings.css` file.
3. Update the `sourceurl` field. This is the URL of your experiment, but without the initial `http://` or `https://`, without the final slash, without any query strings, with other slashes changed to periods, and hyphens or tildes removed.
4. Run the `R` script.
5. The results are written to wherever `R`'s working folder is. By default, it's your home folder.
6. You can open the resulting `csv` files with your spreadsheet program.

What gets written to the server? For each page the participant sees, the server writes the following fields:

- The experiment name (which you specify in `settings.css`).
- The experiment's URL, simplified as above.
- A user code (randomly generated by the server).
- The value from the first column of your `items.txt` file.
- The values of variables you associated with your items using the `pairWith` function.

Any other values from your `items.txt`, and any other variables you create, will not be written.
Since your first column contains unique values, you can use `R`'s `merge` command to add the extra fields from your `items.txt` file.

12 The browser

1. Some people won't be able to participate because they have an old browser, painfully slow connection, javascript is turned off, etc. Not a big problem in my experience - you will rarely know they had a problem (one or two Mechanical Turkers will complain).
2. Browsers are different from each other. You will never be able to make your experiment look exactly the same on all browsers. Your goal should be to just make your experiment work in the browsers your audience is using.
3. Experigen's default CSS file adapts to small screens by reducing margins and fonts, and various other things. Feel free to play with it.
4. The core functions of `Experigen` were designed to work on all browsers, but of course your actual compatibility depends on your materials.
5. Recording audio don't work all that well, and doesn't work at all with iPhone's Chrome.

6. To maximize your experiment's compatibility, prefer standard, widely-implemented HTML, CSS, and Javascript.
7. It's wise to test with a variety of browser/operating system combinations, including a Windows machine and mobile devices.
8. Many people will have crappy speakers/headphones and a noisy environment. If good listening conditions are important (are you expecting people to hear the place of final stops?), use control questions to identify poor listeners.
9. Where appropriate, present materials both visually and auditorily.

13 Check list

Before your experiment goes live, check the following things:

1. Make sure that all sound files and/or pictures (dis)play correctly.

   Missing sound files will cause the experiment to halt.
2. Check that the server is writing everything you want it to write, and in particular, the participants' responses.
3. Check that all the fields in your demographics form are written to the server.

14 Publish

It's nicer than perishing. When you write your paper, please acknowledge:

- Becker, Michael & Jonathan Levine (2020). `Experigen` – an online experiment platform. Available at github.com/tlozoot/experigen.

15 How to get help

1. Turn to your friends and colleagues. If you write to me, I might answer if I have the time/energy (I rarely do).
2. `Experigen` is provided as is and with no guarantee of any kind.