Introduction to Online Experiments in jsPsych Workshop

Part 2: Complex Blocks, Multiple Trials, ABX Task

Lisa Sullivan (lisa.sullivan@mail.utoronto.ca)

1. Preparation

- Download the materials and upload to the relevant sections
  - Replace `style.css` with the new one
  - Add the csv files to the stimuli folder (replace the old version of `abx.csv`)
  - `code_bank.js` contains some snippets of code that might be useful. Open it in your text/code editor
- Load the following plugins
  - `jspsych-audio-keyboard-response`
  - `jspsych-html-button-response`
- Create 3 variables
  - `abxTxt` (string)
  - `abxTrain` (array)
  - `abxStim` (array)
- Make sure the headphone and sound check blocks are still in your file and add them to the timeline after your background questionnaire
- Add a block for the **ABX instructions**. Use the same method and settings as the consent form except that you should load `abx-inst.txt` into the variable `abxTxt`

2. Set up a Basic ABX Task

- ABX task have 4 parts
  - Play sound A
  - Play sound B
  - Play sound X
  - User Response (is X more like A or B?)
- For each sound part, set up a block with the following specifications
  - `Type: audio-keyboard-response`
  - `Trial should end after audio`
  - `No post_trial_gap since the ISI is included in the sound files`
  - `Choices: jsPsych.NO_KEYS`
  - `Prompt text should be inside a paragraph with class abx-audio`

	A Block	B Block	X Block
Stimulus	sound/joachim_bag.mp3	sound/joachim_beg.mp3	sound/felicia_bag.mp3
Prompt Text	A	B	C

- For the user response part, set up a block with the following specifications:
  - Type: html-button-response
  - Stimulus: "Was the third word (C) the same as the first word (A) or the second word (B)? **[paragraph, class: abx-prompt]**
  - Choices: A, B

3. Combine the 4 Blocks into one ABX block

- Why do this?
  - We only need to add 1 block to the timeline
  - It makes adding multiple trials easier
- Create a new timeline called abx_train using the command: `var abx_train = [];`
- Instead of having each block as a separate variable, we are going to "push" them to the abx_train timeline in the correct order
  - Replace `var BLOCKNAME = … ;` with `abx_train.push(…);`
- We can now add this timeline to the main timeline

4. Looping through Multiple Trials

- Why loop?
  - Otherwise we have to copy paste our block and edit it for every trial
  - Cleaner code
  - Easier to update
- How: Load csv file, randomize the order of trials and create a loop, update the complex block we created to use the data in abx_train_trials to run each trial
- Load csv file
  - `Load abx-training.csv into abxTrain using the function on line 33 of code_bank.js.` This function works similarly to the one we used to import text files, except that it imports csv to an array (table)
- Randomize the order of trials using `jsPsych.randomize.repeat():`
  - `The trials should only be repeated once`
  - `The randomized array should be stored in abx_train_trials`
- Create a for loop using the function `for(var i=0; i<abx_train_trials.length; i++){};`
  - This can be understood to say something like "For a variable I (with the starting value of 0), if the value of I is less than the length of the array abx_train_trials, run through the loop. At the end of the loop, add 1 to the value of I (i++)."

    - Basically what this does is repeats whatever's between {…} abx_train_trials.length times
    - Put the four blocks you pushed to abx_train inside this loop.
    - Note: Do not put var abx_train =[]; into the loop. This needs to come before the loop.
- Update the complex block
    - Open abx-training.csv in Excel to see what data is inside
    - In each of the audio blocks, we need to update the sound that the correct sound files for each trial are retrieved
    - You will need to add 'sound/' to the beginning of each sound file
    - abx_train_trails[i].a retrieves the value in the column labelled 'a' in the ith row of the array
    - In the html-button-response trial, we want to add the data from the csv file to the output file so we know which trial it is
        - Create another parameter called data
        - This parameter adds specified values to columns to the output file (and creates them if the don't yet exist)
        - Use this to add use this to import all the data from the csv file (3 sounds, both speakers, word, dialect, vowel, context) and add a column called phase with the value 'practice'

5. Add the main trials

- Create a timeline called abx_main
- Import abx.csv to abxStim and randomize it, saving the randomized order to abx_trials
- The for loop should be identical except that you should
    - Use the length of abx_trials
    - Retrieve the stimuli and values for data from abx_trials
    - Give the column phase the value 'main'
- Hint: You can copy what we already did and adjust it

6. Test and customize the output file

- If we run this on the server and look at the output fill we generated, we'll see that there are a lot of rows and columns we don't need
- We can clean this up using the filter and ignore functions
- filter([{COLUMN:'VALUE'},{COLUMN:'VALUE'}]) removes rows that don't have specified values in a specified column

- `ignore(['view_history','trial_type'])` removes the specified columns
- `Filter out rows that don't have a value of 'practice' or 'main' in the phase column`
- `Ignore the following columns: 'view_history', 'trial_type', 'time_elapsed', 'internal_node', 'stimulus', 'question_order'`
- Quick notes
    - The response for html-button-response is a number corresponding to the order of the buttons (in our case 0 = A, 1 = B)
    - You can't actually ignore internal_node… it will save anyway