

CogLab: recording data

WEEK 5

recap: Sep 28, 2023

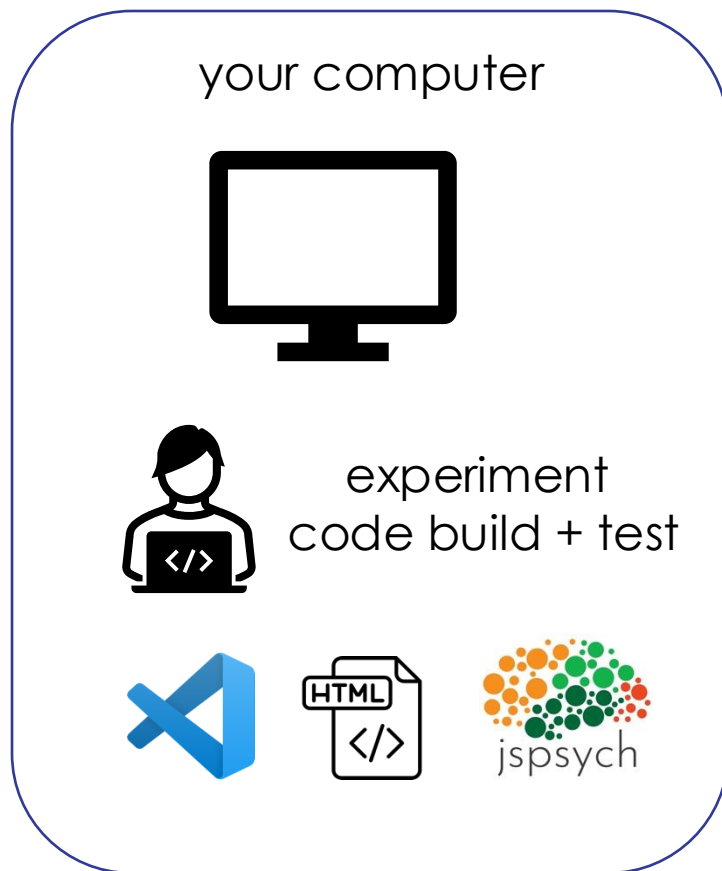
- what we covered:
 - fixing **position & style** of prime/target words
 - adding **instruction** screens
 - **attention** checks & **feedback**
- your to-dos were:
 - **prep:** read about **data storage**
 - **apply:** project milestone #3 (design draft)

going back to our experiment

- open Visual Studio Xcode and open the jsPsych experiment you created last week
- also open the index.html file in your browser to remind yourself of what we did!



github
keeping
track of
changes



Cognition.

cognition.run
going
online

today's agenda

- recording data!

what data do we need for each plugin?

- initial instructions
- training plus association
 - sentences
 - some attention trials
 - association instructions
 - association trials
- priming procedure
 - priming instructions
 - fixation
 - image
 - prime
 - target
 - feedback

```
jsPsych.run([initial_instructions, training_plus_association, priming_instructions, priming_proc]);
```

what data do we need for each plugin?

- initial instructions
- training plus association
 - sentences
 - some attention trials
 - association instructions
 - association trials
- priming procedure
 - priming instructions
 - fixation
 - image
 - prime
 - target
 - feedback

```
jsPsych.run([initial_instructions, training_plus_association, priming_instructions, priming_proc]);
```

what does jsPsych automatically record?

- look at the Data Generated sub-heading for the relevant plugin
 - instructions, sentences
 - attention, association
 - priming procedure
- make note of
 - what is being recorded
 - what else may be needed

viewing the data

- we can add a temporary line of code to **display the data** at the end of the experiment
- change the jsPsych.run() argument to only run the **initial_instructions**

```
const jsPsych = initJsPsych(  
  {  
    on_finish: function(data) {  
      jsPsych.data.displayData();  
    }  
  }  
);
```

```
//jsPsych.run([initial_instructions, training_plus_association, priming_instructions, priming_procl]);  
jsPsych.run([initial_instructions]);
```

```
[  
  {  
    "view_history": [  
      {  
        "page_index": 0,  
        "viewing_time": 1562  
      },  
      {  
        "page_index": 1,  
        "viewing_time": 399  
      },  
      {  
        "page_index": 2,  
        "viewing_time": 480  
      }  
    ],  
    "rt": 2441,  
    "trial_type": "instructions",  
    "trial_index": 0,  
    "time_elapsed": 2442,  
    "internal_node_id": "0.0-0.0"  
  }  
]
```

view sentence data

- run only the `training_procedure`
- which key contains:
 - the sentence being displayed?
 - the response of the participant?
 - the time taken by the participant?
- what about `attention`?
- what was the plugin type for attention?

```
jsPsych.run([training_procedure]);
```

```
[  
  {  
    "rt": null,  
    "stimulus": "I stayed there for a couple of days, hoping I would find a dodish geck.",  
    "response": null,  
    "trial_type": "html-keyboard-response",  
    "trial_index": 0,  
    "time_elapsed": 106,  
    "internal_node_id": "0.0-0.0-0.0"  
  },  
  {  
    "rt": null,  
    "stimulus": "I would love to see a dodish horse.",  
    "response": null,  
    "trial_type": "html-keyboard-response",  
    "trial_index": 1,  
    "time_elapsed": 207,  
    "internal_node_id": "0.0-0.0-0.1"  
  },  
]
```

view attention data

- Command + F for `survey-text`
- which key contains:
 - the participant response?
 - the time taken by the participant?
- what if we have `another` survey-text plugin in our experiment?

```
jsPsych.run([training_procedure]);
```

```
{  
  "rt": 952,  
  "response": {  
    "Q0": "s"  
  },  
  "trial_type": "survey-text",  
  "trial_index": 14,  
  "time_elapsed": 2444,  
  "internal_node_id": "0.0-0.0-1.13-0.13"  
},
```

view association data

- **association** is also a survey-text trial that has the same format as the attention trial
- this could **cause problems later** in identifying the different types of trials

```
jsPsych.run([association_procedure]);
```

```
[  
  {  
    "rt": 729,  
    "response": {  
      "Q0": "dsa"  
    },  
    "trial_type": "survey-text",  
    "trial_index": 0,  
    "time_elapsed": 731,  
    "internal_node_id": "0.0-0.0-0.0"  
  },  
]
```

view priming data

- the priming procedure consists of four different steps (fixation, image, prime, target + response)
- which trial do we **care** most about?
- how would be identify that trial from the current data being recorded?
- what about **feedback** trials?

```
jsPsych.run([priming_proc]);
```

```
[  
  {  
    "rt": null,  
    "stimulus": "+",  
    "response": null,  
    "trial_type": "html-keyboard-response",  
    "trial_index": 0,  
    "time_elapsed": 507,  
    "internal_node_id": "0.0-0.0-0.0"  
  },  
  {  
    "rt": null,  
    "stimulus": "horseapple.png",  
    "response": null,  
    "trial_type": "image-keyboard-response",  
    "trial_index": 1,  
    "time_elapsed": 1017,  
    "internal_node_id": "0.0-0.0-1.0"  
  },  
  {  
    "rt": null,  
    "stimulus": "horseapple.png",  
    "response": null,  
    "trial_type": "image-keyboard-response",  
    "trial_index": 2,  
    "time_elapsed": 1325,  
    "internal_node_id": "0.0-0.0-2.0"  
  },  
  {  
    "rt": 777,  
    "stimulus": "horseapple.png",  
    "response": "a",  
    "trial_type": "image-keyboard-response",  
    "trial_index": 3,  
    "time_elapsed": 2105,  
    "internal_node_id": "0.0-0.0-3.0"  
  },  
]
```

summary of viewing the data

- we need a better way to identify the different types of trials occurring in the experiment

tagging plugin trials with data

- we can add a **data parameter** to each of the trials that records the specific phase of the experiment
- we start with tagging all instruction trials with a key called **typeoftrial** and value **'instructions'**

```
var initial_instructions = {
  type: jsPsychInstructions,
  pages: [
    'page 1 instructions',
    'page 2 instructions',
    'page 3 instructions.'
  ],
  show_clickable_nav: true,
  data: {
    typeoftrial: 'instructions',
  },
}

var association_instructions = {
  type: jsPsychInstructions,
  pages: [
    'Done with sentences. Association time.'
  ],
  show_clickable_nav: true,
  data: {
    typeoftrial: 'instructions',
  },
}

var priming_instructions = {
  type: jsPsychInstructions,
  pages: [
    'Priming task about to begin.'
  ],
  show_clickable_nav: true,
  data: {
    typeoftrial: 'instructions',
  },
}
```

tagging more trials

- tag the sentence, attention, and association trials accordingly

```
var sentence = {
  type: jsPsychHtmlKeyboardResponse,
  stimulus: jsPsych.timelineVariable('sentence'),
  choices: [''],
  trial_duration: 100,
  on_finish: function(data) {
    last_trial_index = (data.trial_index + 1) % 40;
    console.log("last_trial_index= " + last_trial_index);
  },
  data: {
    typeoftrial: 'sentence',
  },
};
```

```
var attention = {
  type: jsPsychSurveyText,
  questions: [{prompt: "Type any ONE novel word from the previous sentence:"}],
  data: {
    typeoftrial: 'attention',
  },
};
```

```
var association = {
  type: jsPsychSurveyText,
  questions: [{prompt: jsPsych.timelineVariable('cue')}],
  trial_duration: 10,
  data: {
    typeoftrial: 'association',
  },
};
```


check data

- run the **training_plus_association** procedure in the browser and view the data
- Command + F for “sentence”, “attention”, and “association”
- **attention** should occur 9 times (3 times per block)
- **association** should occur 36 times (4 words x 3 times x 3 blocks)

```
jsPsych.run([training_plus_association]);
```

```
{  
  "rt": 575,  
  "response": {  
    "Q0": "das"  
  },  
  "typeoftrial": "association",  
  "trial_type": "survey-text",  
  "trial_index": 156,  
  "time_elapsed": 35206,  
  "internal_node_id": "0.0-0.0-2.2-0.24"  
},
```

recoding association responses

- we want to record the association cues and responses in a reasonable manner

```
var association = {  
  type: jsPsychSurveyText,  
  questions: [{prompt: jsPsych.timelineVariable('cue')}],  
  trial_duration: 10,  
  data: {  
    typeoftrial: 'association',  
    cue: jsPsych.timelineVariable('cue'),  
  },  
  on_finish: function(data){  
    data.response = data.response.Q0  
  }  
};
```

before

```
{  
  "rt": 575,  
  "response": {  
    "Q0": "das"  
  },  
  "typeoftrial": "association",  
  "trial_type": "survey-text",  
  "trial_index": 156,  
  "time_elapsed": 35206,  
  "internal_node_id": "0.0-0.0-2.2-0.24"  
},
```

after

```
{  
  "rt": 674,  
  "response": "dsa",  
  "typeoftrial": "association",  
  "cue": "geck",  
  "trial_type": "survey-text",  
  "trial_index": 46,  
  "time_elapsed": 10424,  
  "internal_node_id": "0.0-2.0-2.0-0.0",  
  "ID": 127962287  
},
```

tag the priming procedure trials

- clearly distinguish between fixation, image, prime, target
- run the priming procedure
- view the data
- what about feedback?

```
var fixation = {  
  type: jsPsychHtmlKeyboardResponse,  
  stimulus: "+",  
  choices: "NO_KEYS",  
  trial_duration: 500,  
  data: {  
    typeoftrial: 'fixation',  
  },  
};
```

```
jsPsych.run([priming_proc]);
```

```
{  
  "rt": 344,  
  "stimulus": "applehorse.png",  
  "response": "a",  
  "typeoftrial": "target",  
  "trial_type": "image-keyboard-response",  
  "trial_index": 3,  
  "time_elapsed": 1674,  
  "internal_node_id": "0.0-0.0-3.0"  
},
```

recording accuracy

- we would like to record:
 - whether the keys they press on target trials (A/L) are **correct**
 - whether participants are typing the **novel words** on **attention** trials
- can we use the information in priming.js to help us out?

```
var practice_stimuli = [  
  {  
    "block_number": "practice",  
    "triad": 1,  
    "target_image_pair": "apple-horse",  
    "part": "priming",  
    "prime_word": "boff",  
    "target_word": "apple",  
    "type": "novel",  
    "relatedness": "novel",  
    "correct_response": 1,  
    "image_path": "applehorse.png",  
    "correct_key": "A"  
  },  
]
```

tagging target trials

- we add more data to the critical target trials using the key-value pairs from priming.js
- save, reload, and view the data

```
var practice_stimuli = [
  {
    "block_number": "practice",
    "triad": 1,
    "target_image_pair": "apple-horse",
    "part": "priming",
    "prime_word": "boff",
    "target_word": "apple",
    "type": "novel",
    "relatedness": "novel",
    "correct_response": 1,
    "image_path": "applehorse.png",
    "correct_key": "A"
  },

```

```
var target = {
  type: jsPsychImageKeyboardResponse,
  stimulus: jsPsych.timelineVariable('image_path'),
  choices: ['A', 'L'],
  stimulus_width: 500,
  maintain_aspect_ratio: true,
  prompt: function(){
    return "<span style= 'font-size:200%><br>" + String(jsPsych.tim
  },
  data: {
    typeoftrial: 'target',
    target: jsPsych.timelineVariable('target_word'),
    prime: jsPsych.timelineVariable('prime_word'),
    type: jsPsych.timelineVariable('type'),
    relatedness: jsPsych.timelineVariable('relatedness'),
    correct_key: jsPsych.timelineVariable('correct_key')
  },
}
```

tagging target trials

- we add more data to the critical target trials using the key-value pairs from priming.js
- save, reload, and view the data

```
var practice_stimuli = [
  {
    "block_number": "practice",
    "triad": 1,
    "target_image_pair": "apple-horse",
    "part": "priming",
    "prime_word": "boff",
    "target_word": "apple",
    "type": "novel",
    "relatedness": "novel",
    "correct_response": 1,
    "image_path": "applehorse.png",
    "correct_key": "A"
  },

```

```
{
  "rt": 683,
  "stimulus": "applehorse.png",
  "response": "a",
  "typeoftrial": "target",
  "target": "horse",
  "prime": "nuppical",
  "type": "novel",
  "relatedness": "novel",
  "correct_key": "L",
  "trial_type": "image-keyboard-response",
  "trial_index": 3,
  "time_elapsed": 2002,
  "internal_node_id": "0.0-0.0-3.0"
},

```

recording accuracy

- we can also automatically record whether the correct key was pressed by comparing the response from each trial (`data.response`) to the `correct_key` using the `on_finish` parameter and `jsPsych.pluginAPI.compareKeys()`

```
var target = {
  type: jsPsychImageKeyboardResponse,
  stimulus: jsPsych.timelineVariable('image_path'),
  choices: ['A', 'L'],
  stimulus_width: 500,
  maintain_aspect_ratio: true,
  prompt: function(){
    return "<span style= 'font-size:200%'><br>" + String(jsPsych.timelineVariable('t
  },
  data: {
    typeoftrial: 'target',
    target: jsPsych.timelineVariable('target_word'),
    prime: jsPsych.timelineVariable('prime_word'),
    type: jsPsych.timelineVariable('type'),
    relatedness: jsPsych.timelineVariable('relatedness'),
    correct_key: jsPsych.timelineVariable('correct_key')
  },
  on_finish: function(data){
    data.correct = jsPsych.pluginAPI.compareKeys(data.response, data.correct_key);
  }
}
```

```
{
  "rt": 1323,
  "stimulus": "applehorse.png",
  "response": "a",
  "typeoftrial": "target",
  "target": "apple",
  "prime": "boff",
  "type": "novel",
  "relatedness": "novel",
  "correct_key": "A",
  "trial_type": "image-keyboard-response",
  "trial_index": 3,
  "time_elapsed": 2646,
  "internal_node_id": "0.0-0.0-3.0",
  "correct": true
},
```

recording attention check accuracy

- to record whether the participants are typing the novel words, we first need to **record the novel words** from each sentence
- can we use the information from sentences.js to help us?

```
var sentences = [  
  {  
    "Triad": 1,  
    "Pair": "foobly apple",  
    "Part": "training",  
    "sentence": "I went to Zimziland because I heard you can get a foobly apple.",  
    "novel1": "foobly",  
    "novel2": "Zimziland",  
    "novel3": ""  
  },  
  {  
    "Triad": 1,  
    "Pair": "foobly apple",  
    "Part": "training",  
    "sentence": "My sister doesn't like to have a foobly apple.",  
    "novel1": "foobly",  
    "novel2": "",  
    "novel3": ""  
  }  
]
```


tagging sentence trials

- we first add the sentence itself and the novel words to the data derived from the sentence trials
- run the training_procedure to verify this is working
- what about attention trials?

```
var sentence = {
  type: jsPsychHtmlKeyboardResponse,
  stimulus: jsPsych.timelineVariable('sentence'),
  choices: [''],
  trial_duration: 100,
  on_finish: function(data) {
    last_trial_index = (data.trial_index + 1) % 40;
    console.log("last_trial_index= " + last_trial_index);
  },
  data: {
    typeoftrial: 'sentence',
    sentence: jsPsych.timelineVariable('sentence'),
    novel1: jsPsych.timelineVariable('novel1'),
    novel2: jsPsych.timelineVariable('novel2'),
    novel3: jsPsych.timelineVariable('novel3')
  }
}
```

```
{
  {
    "rt": null,
    "stimulus": "I'm a dodish geck fan!",
    "response": null,
    "typeoftrial": "sentence",
    "sentence": "I'm a dodish geck fan!",
    "novel1": "dodish",
    "novel2": "geck",
    "novel3": "",
    "trial_type": "html-keyboard-response",
    "trial_index": 0,
    "time_elapsed": 106,
    "internal_node_id": "0.0-0.0-0.0"
  },
  {
    "rt": 1488,
    "response": {
      "Q0": "dsa"
    },
    "typeoftrial": "attention",
    "trial_type": "survey-text",
    "trial_index": 7,
    "time_elapsed": 2230,
    "internal_node_id": "0.0-0.0-1.6-0.6"
  },
}
```

evaluating attention responses: 1

- we need to first retrieve the novel words from the **preceding sentence** into the attention trial and add that to its own data
- save and run **training_procedure** again to see if these novel words appearing on attention trials
- also check **inspector**

```
var attention = {
  type: jsPsychSurveyText,
  questions: [{prompt: "Type any ONE novel word from the previous sentence:"}],
  data: {
    typeoftrial: 'attention',
  },
  on_finish: function(data) {
    var last_trial_data = jsPsych.data.get().filter({typeoftrial: 'sentence'}).last(1).values()[0];
    console.log("last_trial_data= ", last_trial_data);

    data.novel1 = last_trial_data.novel1;
    data.novel2 = last_trial_data.novel2;
    data.novel3 = last_trial_data.novel3;
    data.response = data.response.Q0;
  }
};
```

```
{
  "rt": 644,
  "response": "dsa",
  "typeoftrial": "attention",
  "trial_type": "survey-text",
  "trial_index": 4,
  "time_elapsed": 2050,
  "internal_node_id": "0.0-2.0-0.0-1.1-0.1",
  "ID": 756809708,
  "novel1": "foobly",
  "novel2": "mipp",
  "novel3": ""
},
```

```
last_trial_data= Object
  internal_node_id: "0.0-0.0-0.21"
  novel1: "dodish"
  novel2: "geck"
  novel3: ""
  response: null
  rt: null
  sentence: "I'm a dodish geck fan!"
  stimulus: "I'm a dodish geck fan!"
  time_elapsed: 2334
  trial_index: 21
  trial_type: "html-keyboard-response"
  typeoftrial: "sentence"
  [[Prototype]]: Object
index.html:99
```

evaluating attention responses: 2

- what could we use now to compare the participant response to these novel words?
- we can use `jsPsych.pluginAPI.compareKeys()`!
- note the use of the **OR (||)** operator: if ANY of the novel words are mentioned, it is recorded as correct

```
var attention = {
  type: jsPsychSurveyText,
  questions: [{prompt: "Type any ONE novel word from the previous sentence:"}],
  data: {
    typeoftrial: 'attention',
  },

  on_finish: function(data){
    var last_trial_data = jsPsych.data.get().filter({typeoftrial: 'sentence'}).last(1).values()[0];
    console.log("last_trial_data= ", last_trial_data);

    data.novel1 = last_trial_data.novel1;
    data.novel2 = last_trial_data.novel2;
    data.novel3 = last_trial_data.novel3;
    data.response = data.response.Q0;

    if(
      jsPsych.pluginAPI.compareKeys(data.response, data.novel1) ||
      jsPsych.pluginAPI.compareKeys(data.response, data.novel2) ||
      jsPsych.pluginAPI.compareKeys(data.response, data.novel3)
    ){
      data.correct = 1;
      console.log("correct= " + data.correct);
    } else {
      data.correct = 0;
      console.log("correct= " + data.correct);
    }
  },
};
```

evaluating attention responses: 2

- save and re-run the `training_procedure`
- check that the attention trial now has a key storing whether the response typed in is correct or not

```
{  
    "rt": 775,  
    "response": "dsadas",  
    "typeoftrial": "attention",  
    "trial_type": "survey-text",  
    "trial_index": 15,  
    "time_elapsed": 4301,  
    "internal_node_id": "0.0-2.0-0.0-1.12-0.12",  
    "ID": 64255361,  
    "novel1": "dodish",  
    "novel2": "geck",  
    "novel3": "",  
    "correct": 0  
},
```

other data?

- adding a subject ID to the data
- each time the code is run, generate a random number and store it as the ID
- print this id using console.log
- save and reload, open your inspector

```
const jsPsych = initJsPsych(  
  {  
    on_finish: function(data) {  
      jsPsych.data.displayData();  
    }  
  }  
);  
  
// generate random ID  
  
var id = Math.floor(Math.random() * 1000000000);  
console.log("id =", id)
```

adding subject ID to trials

- we also want to attach this ID to all our trials
- two options:
 - manually by using the data parameter for all plugins
 - jsPsych also has a shortcut for this

```
// generate random ID

var id = Math.floor(Math.random() * 1000000000);
console.log("id =", id)

// tag all trials with ID

jsPsych.data.addProperties({
  ID: id
});
```

checking ID is being recorded

- look at the data being generated by the experiment
- ALL trials should have an ID associated with them

```
[
  {
    "rt": null,
    "stimulus": "Sometimes I wish it were easier to get a foobly mipp.",
    "response": null,
    "typeoftrial": "sentence",
    "sentence": "Sometimes I wish it were easier to get a foobly mipp.",
    "novel1": "foobly",
    "novel2": "mipp",
    "novel3": "",
    "trial_type": "html-keyboard-response",
    "trial_index": 0,
    "time_elapsed": 107,
    "internal_node_id": "0.0-0.0-0.0",
    "ID": 88255443
  },
  {
    "rt": null,
    "stimulus": "I would love to see a dodish horse.",
    "response": null,
    "typeoftrial": "sentence",
    "sentence": "I would love to see a dodish horse.",
    "novel1": "dodish",
    "novel2": "",
    "novel3": "",
    "trial_type": "html-keyboard-response",
    "trial_index": 1,
    "time_elapsed": 209,
    "internal_node_id": "0.0-0.0-0.1",
    "ID": 88255443
  }
]
```

other nuts and bolts?

- HW: adding a thank you screen
- still remaining:
 - adding a practice session
 - preloading images
 - going online!

next class

- **before** class
 - *prep*: running online experiments
 - *apply*: add a thank you screen to your experiment
 - *apply*: formative assignment #1 (solo, due Oct 13)
- **during** class
 - tying up loose ends
 - going online!!

logistics: project

- your next milestone is the full experiment code (Oct 22)
- feedback will be provided on design draft
- stop by office hours for more input / help
- class after fall break (Oct 12) is devoted to this

logistics: formative assignment #1

- coding a **new experiment** from start to finish
- due October 13, but start early
- open-resource, but **no collaboration**
- goal is to push you to **code independently**
 - full credit for a reasonable first attempt on all questions (3%)
 - second attempt (after feedback) will be worth 10%
- learning to debug: use the [jsPsych debugging checklist](#)
- practice jsPsych: [jsPsych practice questions](#)