

Repeated structure and trustworthy data

Meeting 2 | Timeline variables, randomization, scoring, and summaries

Bartosz Maćkiewicz

Programming Behavioral Experiments in jsPsych

Build one reusable trial, run it over many items, and show that the resulting rows match the design.

Artifact: a small randomized Stroop pilot with accurate summaries.

Retrieval: three quick checks

- Does a fixation produce a row?
- What does a keyboard timeout store?
- Does refreshing a local page save results?

Open last meeting's export and verify your answers.

Yes, a fixation produces a row. A timeout stores null response and null RT. Refreshing loses unsaved in-memory results.

Separate what varies from what stays fixed

Describe the design in an item table; define the procedure once.

A new item should not require copying the whole experiment.

```
const items = [  
  { text: "2 + 2", key: "f" },  
  { text: "3 + 4", key: "j" }  
];
```

This describes the design before any participant responds. Participant responses belong in the result data.

Two v8 tools with different jobs

// Direct parameter: a placeholder.

```
stimulus: jsPsych.timelineVariable("text")
```

// Inside a function: get the current value.

```
stimulus: () =>
```

```
  `

${jsPsych.evaluateTimelineVariable("text")}</p>`


```

Do not insert a placeholder object into an HTML string.

A repeated procedure

```
const procedure = {  
  timeline: [fixation, trial],  
  timeline_variables: items,  
  randomize_order: true  
};
```

One item supplies the variables for one pass through the procedure.

Live checkpoint L1: parity

Use all four arithmetic items in `materials/live-starter/`. Randomize their order.

F = even result; J = odd result.

Verify four distinct item IDs and correct visible arithmetic.

Randomization does not design the study

- Random order keeps the provided item counts.
- It does not balance an unbalanced item list.
- It does not guarantee alternation or remove confounds.
- Two random runs may have the same order.

Audit counts before inspecting response times.

A list with three congruent items and one incongruent item remains 3:1 after shuffling.

Sampling versus repetition

```
sample: { type: "without-replacement", size: 4 }
```

Use a sample only when the design requires a subset. **repetitions** repeats a procedure's item set.

Today's core task uses every supplied item once.

Data fields are a design decision

For every task row, store:

- task and block;
- item ID and condition;
- stimulus attributes and expected response;
- observed response, RT, correctness, and omission.

Keep instructions and fixation easy to filter out.

Score only after the response exists

```
on_finish: (data) => {  
  data.omission = data.response === null;  
  data.correct = !data.omission &&  
    jsPsych.pluginAPI.compareKeys(  
      data.response, data.correct_response);  
}
```

Keyboard scoring and button-index scoring are different.

Answer one parity item correctly, one incorrectly, and let one time out.

Reconstruct the expected key from the arithmetic itself. Compare it with the saved fields.

A saved answer key cannot validate itself.

Define the denominator

Accuracy = correct decisions / **all task decisions**.

Mean RT = mean of **correct responses with finite RT**.

For the eight-row teaching dataset: 50% accuracy, 800 ms mean correct RT.

Query a relevant subset

```
const rows = jsPsych.data.get()
  .filter({ task: "parity", block: "test" })
  .values();
const correct = rows.filter((r) =>
  r.correct && Number.isFinite(r.rt));
```

What happens if `correct.length` is zero?

An empty subset has no mean

```
const meanRT = correct.length  
  ? correct.reduce((sum, r) => sum + r.rt, 0)  
    / correct.length  
  : null;
```

Display N/A, not 0 ms or NaN. The supplied `Course.summarize` implements this convention.

Add the supplied summary to the timeline. Its stimulus is a function so it reads the completed data.

Run once normally and once with no correct answers. Explain the displayed accuracy, omissions, and RT.

Use `98_Resources/data/synthetic-pilot.csv`.

Eight task trials, four correct, two omissions. Correct RTs: 500, 700, 900, 1100 ms.

Calculate accuracy and mean correct RT before checking the key.

Respond to the **ink**, ignoring the word.

F = red; G = blue; H = green; J = black.

Congruent: word and ink match. Incongruent: word and ink differ.

Use the arrow alternative if color discrimination is unsuitable.

1. Display the correct word and ink for each item.
2. Randomize all eight items; score and label each row.
3. Add overall and condition summaries.
4. Audit counts and run the all-timeout case.

Extend to 16 balanced items in Assignment 2.

A difference between two tiny classroom averages is not sufficient evidence for a psychological effect.

Item confounds, errors, practice, devices, and sampling matter. Do not choose exclusions to make the expected pattern appear.

Explain when to use `timelineVariable` versus `evaluateTimelineVariable`.

Which rows enter accuracy? Which enter RT? What is displayed when no correct response exists?

Before Meeting 3: Assignment 2, a practice/test flow sketch, and the short local JATOS setup guide. Bring a working task.