jsPsych-tutorial

TOPICS

はじめに

今回のチュートリアルの目的

1. jsPsychを使った心理実験・調査を作成できるようになる（7/21）
2. jsPsych心理実験・調査で得られたデータをRで整頓できるようになる（7/28）

1コマで紹介できそうな課題としてフランカー課題を扱います。 質問紙尺度は何でも良かったのですが，ふと，日本語版BIS/BAS（高橋他, 2007[1]）を使うことにしました。

jsPsychの超基本

jsPsychは，心理学実験作成・実施のためのJavaScriptライブラリです。 刺激の提示・データ収集の機能がまとめられています。 例えば，以下のような機能が用意されています。

- 単語提示
- 画像提示
- 音声提示
- 選択ボタン提示
- 回答用テキストボックスの表示

JavaScript とは

jsPsychのベースとなっているJavaScriptは，Webページを動的に制御するためのプログラミング言語です。 映画のサイトで背景画像が切り替わったり，検索窓に文字を入力すると候補が表示されたりするのはJavaScriptによって実現されています。

jsPsychを用いた心理学実験とは

jsPsychを用いた心理学実験は，jsPsych（を含むJavaScript）が埋め込まれた**Webページ**です。 Google検索で入力された文字に応じて候補が表示されるように，jsPsychが埋め込まれたWebページでは時間経過やキー入力に応じて表示内容が更新されることで心理学実験が実現されます。

WebページなのでChromeやFirefox，Safariといったウェブブラウザで表示されます。 ネット上に公開してURLを共有すれば，世界中の誰でもその人のPCから参加可能になるため，jsPsychを用いてオンライン実験が実施できるわけです。 とはいえ，誰にもURLを共有せず実験室でだけ利用すれば，実験室実験も可能です。

Flanker task

フランカー課題については この記事 に簡単な説明があります。

刺激として今回は `<<<<<`, `>>>>>`, `<<><<`, `>><>>` の４つを使用します。参加者は真ん中の記号が`<`なら`f`キー，`>`なら`j`キーを押すこととします。

刺激を一つ提示する

`tutorial/start/` に保存されている `flanker.html` というファイルをエディタで開いてください。 右クリックでエディタアプリを指定して開くか，予め開いておいたエディタからファイルを選択することで開くことができます。 VScodeの場合はファイルをドラッグ＆ドロップすることで開くことも可能です。

さて，`flanker.html` は初期状態で以下のようになっています。説明のために資料にはコメントを付け足しています。`<!--ほげほげ-->` や `// ほげほげ` はそれぞれ `.html`，`.js` ファイルで実行時に処理されない部分になり，コードに関するコメントとして機能します。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
      <!-- jsPsych関連ツールの呼び出し -->
    <script src="../jspsych-6.3.1/jspsych.js"></script>
    <script src="../jspsych-6.3.1/plugins/jspsych-html-keyboard-
response.js"></script>
    <link rel="stylesheet" href="../jspsych-6.3.1/css/jspsych.css" />
  </head>
  <body></body>
  <script>
      // ここにjavascriptコードを書いて実験を作成する
      // 基本的にはここを編集すれば良い
  </script>
</html>
```

上で省略した`<script></script>` の間には以下のようなコードが書かれています。

```
// 実験の設定変数の作成
var trial = {
  type: 'html-keyboard-response',
  stimulus: '<<<<<',
  choices: ['f', 'j'],
    post_trial_gap: 500,
};

// 設定をもとに実験を実行する関数
jsPsych.init({
  timeline: [trial],
});
```

javascriptでは，`var 変数名 =` で変数を宣言します。`{}` はオブジェクト型（連想配列）になります。データ型の種類や詳細については [この記事](#) を参照してください。javascriptを書いているといろんなカッコ`{}`, `[]`, `()` を別々の用途で使うので，間違えないように注意してください。

この例のように，jsPsychでは，実験（ここであれば1試行）の設定が指定された変数を作成（`var trial =`）し，それを `jsPsych.init()` に入れることで実験として実行されます。ここでは次の3つの点を設定しています。

- type：試行のタイプ。`html-keyboard-response` は，html形式で刺激を指定し，キー入力を収集する試行
- stimulus：刺激。単に文字を提示する場合は文字列を指定すれば良い。
- choices：収集するキー入力。配列`[]` にキーの文字を指定する。
- post_trial_gap：試行終了後の空白時間。単位はミリ秒。

ブラウザで開く

`flanker.html` をフォルダ上でダブルクリックすると，デフォルトのブラウザで開いて，`<<<<<` がブラウザの中央に表示されるはずです。

演習

- 同じ試行を複数回実施してみよう
- 提示する刺激を変えてみよう

サンプルコードは `tutorial/exercise/FL01_single_trial.html`

異なる刺激で複数試行実施する

それぞれの試行で提示される刺激が変わるようにします。 例えば，最初のコードを以下のように変えると，異なる刺激が提示される試行が一つずつ実施されます。

```
var trial1 = {
  type: 'html-keyboard-response',
  stimulus: '<<<<<',
  choices: ['f', 'j'],
    post_trial_gap: 500,
};

var trial2 = {
  type: 'html-keyboard-response',
  stimulus: '>>>>>',
  choices: ['f', 'j'],
    post_trial_gap: 500,
};

jsPsych.init({
  timeline: [trial1, trial2],
});
```

それぞれの試行について別々の変数名がついていることに注意にしてください。

演習

- 4 試行にしてみよう
- いろんな提示順序を試してみよう

サンプルコードは `tutorial/exercise/FL02_four_trials.html`

timeline_variablesの導入

一つ前のセクションで複数試行の実施を試してもらいましたが，コピペの繰り返しであまり効率的な作業とは言えません。 さらに，コピペを繰り返していると，刺激以外の設定はすべて同じなので，コードが冗長になり全体像が見にくくなります。 実際の実験ではさらに試行数が増えるため，効率さやコードの見た目がかなりひどいことになることは想像に易いです。

これを回避するための方法として，`timeline_variables` と `jsPsych.timelineVariable()` を使うことが挙げられます。 任意のキーと値で構成されたオブジェクト `{キー：値}` の配列 `[]` を作成し，それを`timeline_variables` に指定します。 そして，試行変数の該当の設定を，`jsPsych.timelineVariable('キー')` としておきます。 こうしておくと，実験の実行時にキーに対応する値が試行ごとに順番に適用され，異なった刺激が提示されるようになります。

```
var stims_flanker = [
  { stim: '<<<<<' }, //  「stim」は呼び出すためのキー
  { stim: '>>>>>' },
  { stim: '>><>>' },
  { stim: '<<><<' },
];

var trial = {
  type: 'html-keyboard-response',
  stimulus: jsPsych.timelineVariable('stim'),
  choices: ['f', 'j'],
    post_trial_gap: 500,
};

var block = {
  timeline: [trial],
  timeline_variables: stims_flanker,
};

jsPsych.init({
  timeline: [block],
});
```

なお，`timeline_variables` 等を使って変更できる設定は一つだけではありません。 例えば，以下のようにすれば，試行ごとに`post_trial_gap`を変更することも可能です。

```
var stims_flanker = [
  { stim: '<<<<<', gap: 500 },
  { stim: '>>>>>', gap: 1000 },
```

```
    // 省略
];

var trial = {
  type: 'html-keyboard-response',
  stimulus: jsPsych.timelineVariable('stim'),
  choices: ['f', 'j'],
    post_trial_gap: jsPsych.timelineVariable('gap'),
};
```

演習

- 2つの刺激 `--<--`, `-->--` を新たに提示できるようにしよう

サンプルコードは `tutorial/exercise/FL03_timeline_variables.html`

刺激のランダマイズ

ちなみに，`randomize_order: true` を足すと，適用するオブジェクトの順番をランダムにすることができます。

```
var block = {
  timeline: [trial],
  timeline_variables: stims_flanker,
  randomize_order: true
};
```

注視点

実験ではしばしば注視点が刺激（系列）の前に提示されます。 注視点を導入するためには，それ用の変数を作成し，`timeline` 配列に入れればいいです。

```
var fixation = {
  type: 'html-keyboard-response',
  stimulus: '+',
  choices: jsPsych.NO_KEYS, // どのキー入力も受け付けない
  trial_duration: 500 // 提示時間のための設定項目（単位はミリ秒）
};
```

なお，`fixation` 変数では，これまでの`trial` 変数とは設定している項目が少し変わっています。

- `post_trial_gap` が指定されていません。
    - この場合，既定値の0が適用されます。
- `trial_duration` が設定されています。
    - これまでの例では指定されていませんでした。指定しない場合，既定値の0が適用されます。

このように，実は試行の変数に設定できる項目は色々あります。 設定しなかった場合はjsPsych側で用意されている既定値が適用されます。 また，設定項目は`type`によっても変わります（質問紙用の`survey-lickert`を見るとよくわかる）。 詳しくは，jsPsychの公式サイトを参照してください。

演習

- 注視点を毎試行提示してみよう
- 注視点を最初だけ提示してみよう

Hint! これまで作ってきたコードには，`timeline`の設定箇所が２つあります。

サンプルコードは`tutorial/exercise/FL04_flanker.html`

質問紙尺度の提示

質問紙尺度として今回は日本語版BIS/BAS（高橋他, 2007）を使用します。 項目は本資料の一番最後にリストアップされています。

質問を一つ提示する

フランカー課題のときと同じ要領で，`tutorial/start/`に保存されている`bis-bas.html`というファイルをエディタで開いてください。 ファイルのhtml部分は以下のようになっています。 フランカーのときと，読み込んでいるプラグインが変わっていることに注目してください。 なぜこの変更があるのかについては，javascript部分を見ると理解できると思います。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <script src="../jspsych-6.3.1/jspsych.js"></script>
        <!--    html-keyboard-response ではない    -->
    <script src="../jspsych-6.3.1/plugins/jspsych-survey-likert.js">
</script>
    <link rel="stylesheet" href="../jspsych-6.3.1/css/jspsych.css" />
  </head>
  <body></body>
  <script>
      // ここにjavascriptコードを書いて実験を作成する
      // 基本的にはここを編集すれば良い
  </script>
</html>
```

javascript部分（= `<script></script>`の間）は以下のようになっています。

```
var bis_bas = {
  type: 'survey-likert', // タイプが変わっている。
```

```
  questions: [], // 一旦省略
  scale_width: 500,
};

jsPsych.init({
  timeline: [bis_bas],
});
```

まず，フランカー課題の場合と異なり，`type` に `survey-likert` が指定されています。 html部分で読み込むプラグインが変わっていたのは，これが理由です。 jsPsychを使った実験を書く際には，コード内で登場した`type`に対応するプラグインをhtmlをすべて読み込んでおく必要があります。

また，`type` の変更に伴ってその他の設定項目も変わっていて，`questions`, `scale_width` になっています。 `questions` に，質問項目オブジェクト`{}`が格納された`配列`を指定することで，質問項目が表示されるようになります。

質問項目オブジェクトは以下のようにします。

```
{
  prompt: '競争に勝ったら，私は興奮するだろう', // 質問文
    // リッカートのラベル
  labels: ['あてはまらない', 'どちらかと言えばあてはまらない', 'どちらかと言えばあてはまる', 'あてはまる'],
  required: true, // 回答が必須かどうか
},
```

演習

- 質問文を変えてみよう
- 選択肢の数を4つ以外にしてみよう

質問文は，本資料の付録を見てください。

サンプルコードは `tutorial/exercise/BB01_change_item_scale.html`

質問を複数提示する

`questions`の配列に質問項目オブジェクトを複数入れることで，１ページに複数の質問を表示できるようになります。

```
var bis_bas = {
  type: 'survey-likert',
  questions: [{
          prompt: '競争に勝ったら，私は興奮するだろう',
          labels: ['あてはまらない', 'どちらかと言えばあてはまらない', 'どちらかと言えばあてはまる', 'あてはまる'],
          required: true
```

```
      },
        {
          prompt: '私は，興奮や新しい刺激を切望している',
          labels: ['あてはまらない', 'どちらかと言えばあてはまらない', 'どちらかと言えばあてはまる', 'あてはまる'],
          required: true
  }],
  scale_width: 500,
};
```

演習

- 表示する質問をあと２つ追加する

サンプルコードは `tutorial/exercise/BB02_four_items.html`

質問項目の配列をまとめて作る

さて，BIS/BASの質問項目は全部で20項目あります。 上の方法で表示される質問を増やしていくと，フランカー課題の説明でも述べたように，コード作成の効率性やコードの可読性が下がっていきます。 個人的には，すでに４項目でうんざりです。 もし，リッカートのラベルを変更することになったら，その編集作業は発狂ものです。

今回は質問項目オブジェクトの**配列**を効率よく作るというのが目標になります。 フランカー課題で使った`timeline_variables` ではなく，`for` 文を使った方法が適しているでしょう。

for 文

`for` 文は同じ処理を繰り返すときに使う構文です。 `for` 文の使い方を知るために，以下の例を見てみましょう。

```
var num = 1
for (var i = 0; i < 10; i++) {
    num = num + 1
}
```

`for ([初期化式]; [条件式]; [加算式])` で何回繰り返すかを設定します。 上の例では，それぞれ，

- 初期化式：繰り返しを制御するためのカウンター用変数 `i` を `0` からスタートさせる
- 条件式： `i` が10未満の間だけ `{}` 内の処理を繰り返す（`i < 10` ）
- 加算式： `{}` 内の処理が終わったら `i` に1を足す（`i++` ）

となっています。 これによって，ここでは `num = num + 1` という処理を10回繰り返しています。

本題

それでは，`for` 文を使って質問項目の配列を作っていきます。 `for` 文で扱いやすくするように，事前に 質問文が列挙された配列 `items` を作成しています。 ここでは省略していますが，20個の質問文を

全部入れます。 また，`for` 文内の処理を見やすくするために，リッカートのラベルのための配列を `scale` という変数として事前に宣言しています。

```
var scale = ['あてはまらない', 'どちらかと言えばあてはまらない', 'どちらかと言えばあてはまる', 'あてはまる'];
var items = [
  'たとえ何かよくないことが私の身に起ころうとしていても，怖くなったり神経質になったりすることはほとんどない',
  '私は，欲しいものを手に入れるためには格別に努力する',
  // 中略
  '競争に勝ったら，私は興奮するだろう',
  '私は，間違いを犯すことを心配している',
];

var questions = []; // 空の配列
for (var i = 0; i < items.length; i++) {
  questions.push({ prompt: items[i], labels: scale, required: true });
} // 質問オブジェクトを一つずつ追加

var bis_bas = {
  type: 'survey-likert',
  questions: questions, // 完成したquestions配列をここで指定
  randomize_question_order: true, // 質問文の順番をランダム化する
  scale_width: 500,
};
```

`for` 文を詳しく見ていきましょう。 この例では，一つ前の例から条件式の部分が変更されています。

- 条件式： `i` が質問項目の配列（`items`）の要素数より小さい間だけ `{}` 内の処理を繰り返す（`i < items.length`）

つまり， 今回の例では，`i = 19` のときまで処理を繰り返すことになります。 `i` の初期値が 0 なので，0 ~ 19 の計20回処理が繰り返されます。

`{}` 内で行っている処理では，質問項目オブジェクトを `.push()` で 事前に作っておいた空の `questions` 配列に追加しています。

追加する質問項目オブジェクトですが，`prompt:` の部分に `items[i]` とあります。 これは，質問文配列`items` の `i` 番目の質問文をとるということをしています。 `i` は`for` の繰り返しごとに1ずつ足されていくので，初期値の `i = 0` 番目から19番目まで質問文が順番に取り出されることになります。0番目は奇妙に見えますが，javascirptでは，日常的な数え方とは異なり，配列の最初の要素は `0` 番目となっています [2]。 したがって，0番目から19番目まで順番に取り出していくことで，20個の質問文が取り出せるわけです。

これで，最初は空だった `questions` 配列に 20個の質問項目オブジェクトが格納されます。

演習

- 他の質問紙尺度でも練習してみよう（授業内では扱いません）

今回紹介しなかった大事なこと

- 保存するデータの追加・変更
- 参加者情報の取得
- 作成したコードをサーバーにアップロードしオンラインに公開する方法
- サーバーにデータを保存する方法

実用的なオンライン実験を作り，実施するためには，この辺の話をしないといけないのですが，もう1コマあればな...という感じです。 また，サーバー関連の話については，手軽に利用できるサーバーは基本的に有料なので，デモ用途で手軽に使えるものを用意できなかったというのも理由の一つです。

上の2点については，`tutorial/goal` に保存されている `bis-bas.html` や `flanker.html`，`exp/exp1_casual.html` を見ると，一応確認できるようにはしています。

今回は紹介しませんが，試してみてもらうといいと思ったのは，cognition.run というサービスです。私が使ってみた感想は，超便利！です（しかも無料）。 特にびっくりしたのは，今回紹介を回避したサーバーにデータを保存するためのコードを書かなくても勝手に保存してくれるという点です。 とはいえ，本番の実験で使って大丈夫なのかはよくわかりません。 まだβ版（試作版）ということで，不測の事故もあるかもしれません。 データがサービス側にアクセスされることはないと Privacy and Data Policy に書いてあったので，そこは大丈夫なんだと思います。 少なくとも課題作成のちょっとした練習やオンライン実験の体験にはいいと思いました。

他にも，googleが提供する Firebase という基本無料のホスティングサービスがあります。 Firebaseを使う場合は 専修大学の国里先生が作成されたページ が参考になると思います。

付録

日本語版BIS/BASの項目

1. たとえ何かよくないことが私の身に起ころうとしていても，怖くなったり神経質になったりすることはほとんどない
2. 私は，欲しいものを手に入れるためには格別に努力する
3. 何かがうまくいっているときは，それを続けることがとても楽しいと思う
4. 面白そうだと思えば，私はいつも何か新しいものを試したいと考えている
5. 私は，欲しいものを手に入れたとき，興奮し，活気づけられる
6. 非難されたり怒られたりすると，私はかなり傷つく
7. 欲しいものがあると，私はたいていそれを手に入れるために全力を挙げる
8. 楽しいかもしれないから，というだけの理由で何かをすることがよくある
9. 欲しいものを手に入れるチャンスを見つけると，すぐに動き出す

10. 誰かが私のことを怒っていると考えたり，知ったりすると，私はかなり心配になったり動揺したりする
11. 何か好きなことをするチャンスをみつけると，私はすぐに興奮する
12. 私はしばしば時のはずみで行動する
13. 何かよくないことが起ころうとしていると考えると，私はたいていくよくよ悩む
14. よいことが私の身に起こると，そのことは，私に強い影響を与える
15. 何か重要なことをあまりうまくできなかったと考えると不安になる
16. 私は，興奮や新しい刺激を切望している
17. 私は，何かを追い求めているときには徹底的にやる
18. 私は，友達と比べると不安の種はとても少ない
19. 競争に勝ったら，私は興奮するだろう
20. 私は，間違いを犯すことを心配している

実験作成ソフトウェアの紹介

jsPsychの他にも，オンライン実験の作成ツールは色々あります。

- lab.js
- PsychoPy
- GORILLA

lab.jsとPsychoPyは無料で使えます。 ただし，オンラインで実施するためのサーバー代は必要です。 GORILLAは有料ソフトです。 有料な分，サポートは手厚いのかもしれません。

どのツールもGUIベースで，基本マウスでポチポチして実験を作ることができます。 プログラミングが苦手な人にとってはこっちのほうがとっつきやすいかもしれません。 ただ，実験の手続きにちょっとこだわりポイントがあったりすると結局コードが必要になったりします。

lab.jsについては 山形大学 小林先生と大杉先生のチュートリアル がとても参考になります。

今後の発展に参考になりそうなサイト

- 西山の作ったいくつかのオンライン記事
  - jsPsychによる心理学実験作成チュートリアルまとめ
    - 最新のjsPsychに対応していない部分があるので注意。
  - jsPsychのためのjavascriptのキほん

本番に近いコードを見るのが何より勉強になります。

- jsPsych 公式サイト
- 高橋先生の「キソジオンライン」
- 国里先生の「jsPsych を用いた認知課題の作成」
- 小林先生の「jsPsych チュートリアル」

公式からjsPsychをダウンロード（ページの右の方にあるreleasesから）すると，サンプルコードが同梱されています（examples フォルダ内）。 サンプルコードはそれぞれのプラグインの使い方を確認するのにとても便利です。

---

1. 高橋 雄介・山形 伸二・木島 伸彦・繁桝 算男・大野 裕・安藤 寿康 (2007). Grayの気質モデル. パーソナリティ研究, 15(3), 276–289. https://doi.org/10.2132/personality.15.276 ↩
2. なお， Rでは最初の要素は1番目です。Pythonは0番目です。 ↩