

生成AIを用いたjsPsych プログラミング

専修大学 国里愛彦

「生成AIを用いたjsPsychプログラミング」

目次

1. 生成AI時代に必要なプログラミング知識
2. jsPsychの基礎知識
3. NotebookLMを用いたjsPsychプログラミング
4. データ収集方法

「生成AIを用いたjsPsychプログラミング」

目次

1. 生成AI時代に必要なプログラミング知識
2. jsPsychの基礎知識
3. NotebookLMを用いたjsPsychプログラミング
4. データ収集方法

生成AI時代にプログラミングなんて必要ですか？

プログラムは生成AIが書いてくれるが、以下が明らかでなければ意味のあるものは作り出せない。

- 「どんなものを作るのか？」
- 「生成AIが書いたプログラムは何をしているか？正しいのか？」

→作りたいもの・課題を**整理するスキル**，プログラムコードを**読むスキル**が重要になる。

プログラムコードを読むために

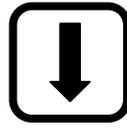
- **どの言語も上から順番に実行，変数(配列)，条件分け，繰り返し，関数(メソッド)を押さえれば良い。**
- 言語によって書き方は違うけど，その辺は生成AIに頼ることができる。
- 人間がするのは，処理の流れを適切に追うという抽象化された部分であり，どう書くかは生成AI時代にそれほど重要ではない。

コードを読んでみよう！

右のようなコード
(javascript)があった
とする。どう読もう
か？

```
function modifyVector(vector) {  
  var modifiedVector = [];  
  for (let i = 0; i < vector.length; i++) {  
    if (vector[i] % 2 === 0) {  
      modifiedVector.push(vector[i] * 10);  
    } else {  
      modifiedVector.push(vector[i]);  
    }  
  }  
  return modifiedVector;  
}
```

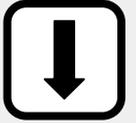
上から順番に実行



コードは上から順番に実行されるので、コードは上から読んでいく。

コードはある程度まとまったブロックになっていることも多い。

```
function modifyVector(vector) {  
  var modifiedVector = [];  
  for (let i = 0; i < vector.length; i++) {  
    if (vector[i] % 2 === 0) {  
      modifiedVector.push(vector[i] * 10);  
    } else {  
      modifiedVector.push(vector[i]);  
    }  
  }  
  return modifiedVector;  
}
```



変数(配列)



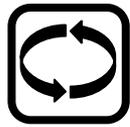
何か値などをいれる場所を変数と呼ぶ(複数の場合は、配列)。

変数には型があり、var(再宣言・再代入可), let(再宣言不可・再代入可), const(再宣言・再代入不可)がある。

pushメソッドで配列に要素を追加

```
function modifyVector(vector) {  
  var modifiedVector = [];  
  for (let i = 0; i < vector.length; i++) {  
    if (vector[i] % 2 === 0) {  
      modifiedVector.push(vector[i] * 10);  
    } else {  
      modifiedVector.push(vector[i]);  
    }  
  }  
  return modifiedVector;  
}
```

繰り返し



プログラミングで同じような処理を繰り返す場合はfor文を使う。

for文をみたら、どこからどこの範囲を繰り返すのかを確認する(今回はオレンジ色の{}の範囲内を繰り返す)

```
function modifyVector(vector) {  
  var modifiedVector = [];  
  for (let i = 0; i < vector.length; i++) {  
    if (vector[i] % 2 === 0) {  
      modifiedVector.push(vector[i] * 10);  
    } else {  
      modifiedVector.push(vector[i]);  
    }  
  }  
  return modifiedVector;  
}
```

条件分け

条件分けして処理する場合は, if文を使う。

if文の範囲とどのような場合分けかを確認する。

if(条件)

else if(条件)

else(それ以外)

```
function modifyVector(vector) {  
  var modifiedVector = [];  
  for (let i = 0; i < vector.length; i++) {  
     if (vector[i] % 2 === 0) {  
      modifiedVector.push(vector[i] * 10);  
    } else {  
      modifiedVector.push(vector[i]);  
    }  
  }  
  return modifiedVector;  
}
```

関数(メソッド)



ある程度まとまった処理をする場合は、その処理をする関数(メソッド)を作る。

functionで宣言することが多い。

関数の範囲, 引数(入力), 出力を確認する。

```
function modifyVector(vector) {  
  var modifiedVector = [];  
  for (let i = 0; i < vector.length; i++) {  
    if (vector[i] % 2 === 0) {  
      modifiedVector.push(vector[i] * 10);  
    } else {  
      modifiedVector.push(vector[i]);  
    }  
  }  
  return modifiedVector;  
}
```



コードを試す

javascriptを試す
ことができるサイト
で動かしてみま
しょう。

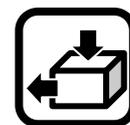
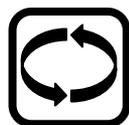
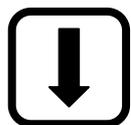
[こちらでコードをコ
ピペして,](#)

以下のサイトで動
かしてみましよう！

<https://js.do>

```
function modifyVector(vector) {  
  var modifiedVector = [];  
  for (let i = 0; i < vector.length; i++) {  
    if (vector[i] % 2 === 0) {  
      modifiedVector.push(vector[i] * 10);  
    } else {  
      modifiedVector.push(vector[i]);  
    }  
  }  
  return modifiedVector;  
}  
  
// 確認用のコードを追加  
const originalVector = [1, 2, 3, 4, 5];  
const modifiedVector=modifyVector(originalVector);  
document.write(originalVector + "をmodifyVectorで変換  
すると"+modifiedVector);
```

プログラムコードを読むために



- どの言語も上から順番に実行，変数(配列)，条件分け，繰り返し，関数(メソッド)を押さえれば基本的にOK!
- 各言語の違いは方言みたいなものなので，生成AIの支援があればなんとかなる(「Pythonでのif文の書き方は？」とかAIに質問すればいい)
- 基本的な構造を理解して設計する(読む)ことができれば，あとは生成AIを活用すればいい時代です。

生成AIとは？

- 生成AIは、通常のAIと同じ機械学習を用いたものだが、識別・分類だけでなく、新規に画像・文章・動画などを生成できるAIになる。
- GPT(Generative Pretrained Transformer)は、トランスフォーマーアーキテクチャーに基づくニューラルネットワーク(NN)モデルを用いて事前学習したモデル。
- 膨大なパラメータ数のNNで、大規模な言語データを用いて学習したモデルは、大規模言語モデル(LLM)とも呼ばれる。

LLM

<長所>

- ・柔軟に文章の要約・生成ができる
- ・文章を理解しているように処理できるので、自然言語でも指示すれば動く(プログラミングもしてくれる)。人が言葉を使ってできるようなことは大体できる(完璧ではないが)。

<短所>

- ・間違っていることを言う(ハルシネーション)ことがある(学習する情報を新しくしたり, ネット検索も併用するなどの工夫もあるが, ゼロにはならない)。

どれを使えば良い？

- 競争激化につき, GPT-4o(OpenAI), Claude 3 Opus(Anthropic), Gemini Pro 1.5(Google)については, そこまで大きな差はない(差は使い方, 言語に依存する印象)。
- どれでも良いが, 画像などの生成や便利な機能を選ぶならOpenAIの有料版?, Googleのサービスをよく使う方は今後の連携からGemini?
- ハルシネーション対策のために, 今回は Googleの実験的なサービスのNotebookLMを使います(今は無料!)。

「生成AIを用いたjsPsychプログラミング」

目次

1. 生成AI時代に必要なプログラミング知識
- 2. jsPsychの基礎知識**
3. NotebookLMを用いたjsPsychプログラミング
4. データ収集方法

jsPsychの準備をする

- jsPsychの準備には以下の3つがある

①CDN(Contents Delivery Network)上のjsPsychを利用

②jsPsychのソースをDLして利用

③npmを用いてimportして利用

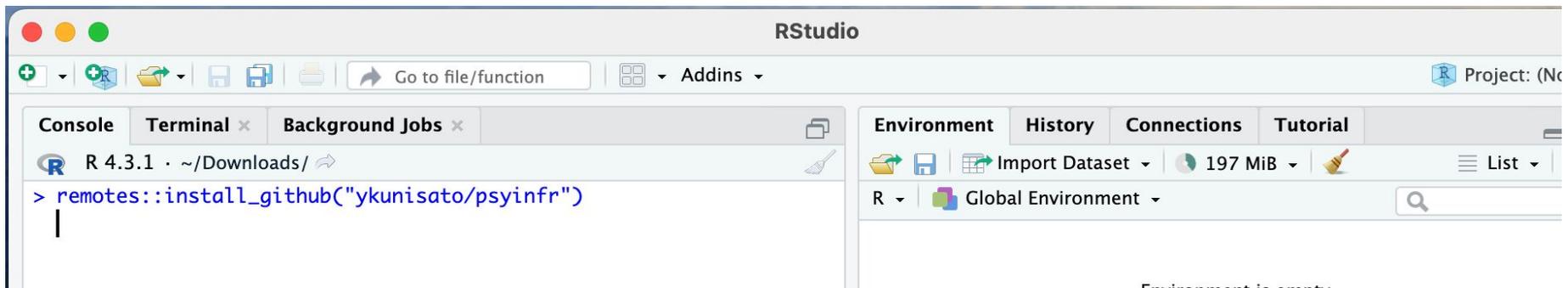
- ①は手軽だがオフラインだと使えない, ③は少し玄人向け
- ②はDLが面倒だが分かりやすいので今回採用する
(DLは自動化する)

psyinfrによるjsPsychの準備

- 国里の自作Rパッケージの[psyinfr](#)を使う。
- remotesパッケージのinstall_github()を使って、RStudioのConsoleからインストール(remotesがなければインストール)

[コードのリンク](#)

```
# install.packages("remotes")  
remotes::install_github("ykunisato/psyinfr")
```



カレントワーキングディレクトリについて

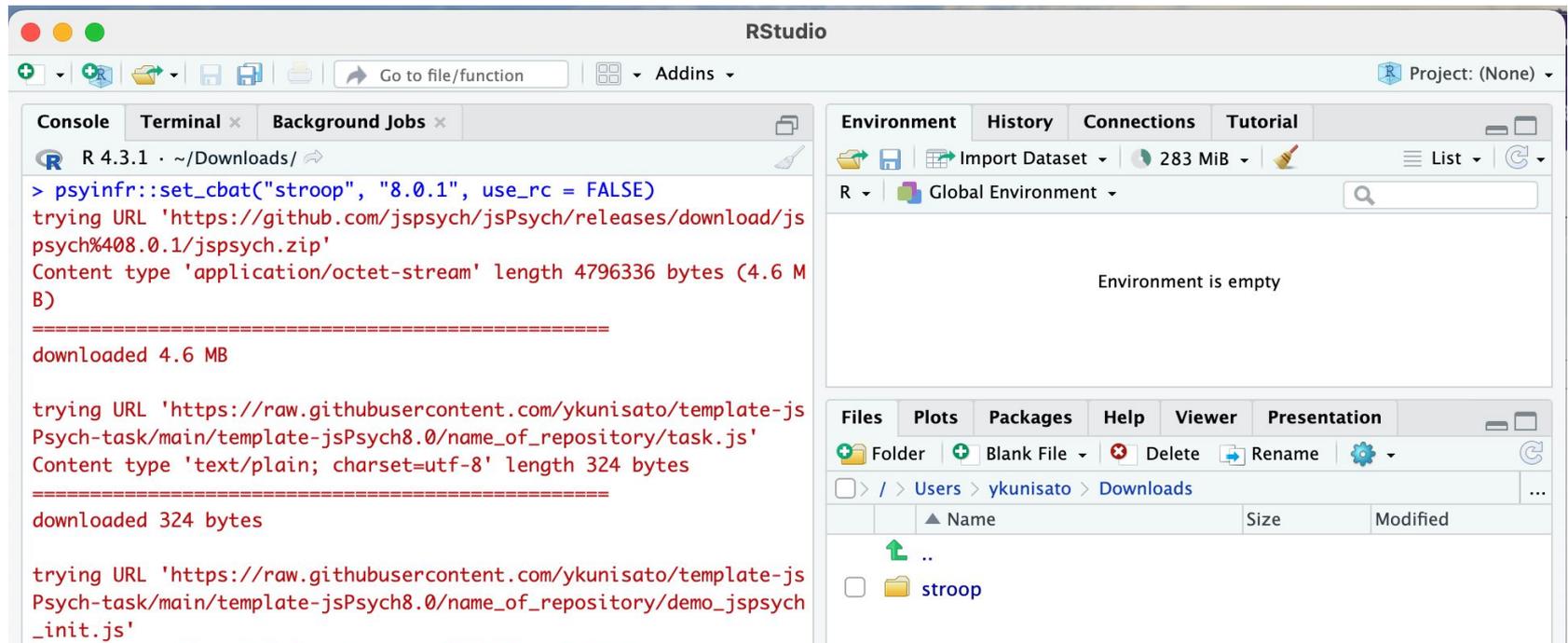
- プログラミングをする場合は、今どこのディレクトリで作業しているかが重要になります（カレントワーキングディレクトリ）。
- なにかプログラムを実行するにしても、場所がずれていると動かないので、カレントワーキングディレクトリを確認したり、設定する。
- RStudioの場合、Session > Set Working Directory > Choose Directory... で選択する

set_cbat()関数によるjsPsychの準備

- psyinfrのset_cbat()関数を使ってRStudio上で作業できるようにします。以下のコードを打ち込んでください(練習でstroop課題を作ります)。

[コードのリンク](#)

```
psyinfr::set_cbat("stroop", "8.0.1", use_rc = FALSE)
```



The screenshot shows the RStudio interface with the following content:

Console:

```
R 4.3.1 ~./Downloads/
> psyinfr::set_cbat("stroop", "8.0.1", use_rc = FALSE)
trying URL 'https://github.com/jspsych/jsPsych/releases/download/jspsych%408.0.1/jspsych.zip'
Content type 'application/octet-stream' length 4796336 bytes (4.6 MB)
=====
downloaded 4.6 MB

trying URL 'https://raw.githubusercontent.com/ykunisato/template-jsPsych-task/main/template-jsPsych8.0/name_of_repository/task.js'
Content type 'text/plain; charset=utf-8' length 324 bytes
=====
downloaded 324 bytes

trying URL 'https://raw.githubusercontent.com/ykunisato/template-jsPsych-task/main/template-jsPsych8.0/name_of_repository/demo_jspsych_init.js'
```

Environment: Global Environment (Empty)

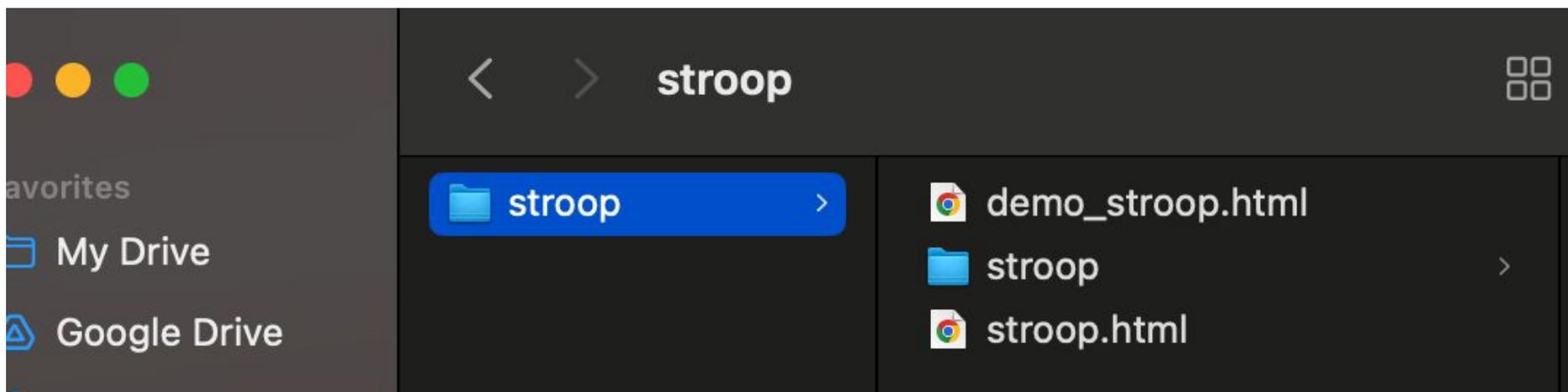
Files: Users > ykunisato > Downloads

	Name	Size	Modified
	..		
	stroop		

set_cbat()関数で設定したフォルダ構造

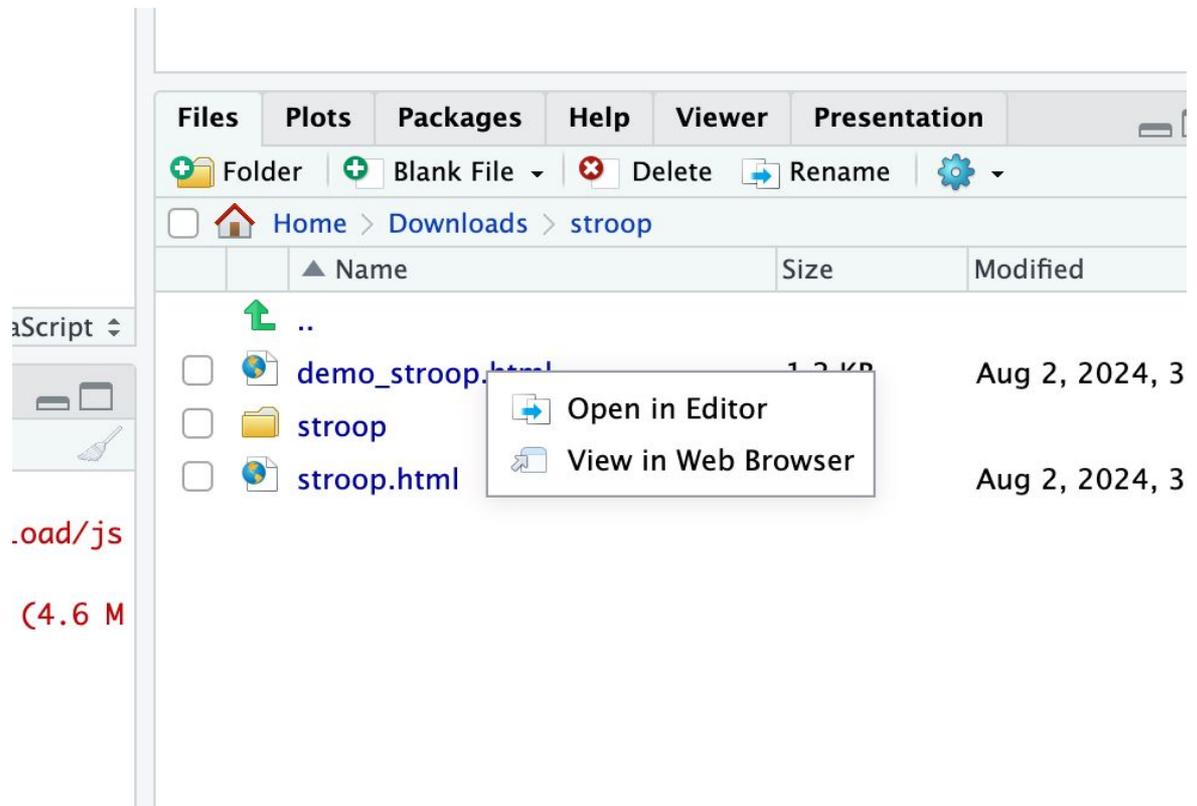
stroopフォルダ内に、(1)jsPsychのスク립トが入ったstroopフォルダ、(2)demo_stroop.html(動作確認用)、(3)stroop.html(JATOS用)があります。

※set_cbat()で”stroop”としたのでファイル名やフォルダ名がstroopになっています。set_cbat()で別の名前を指定すると変わります。

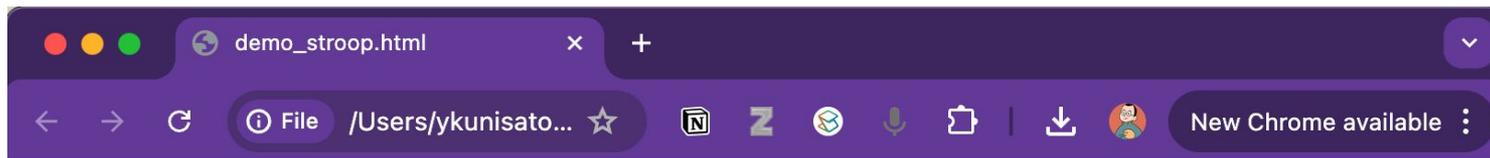


demo_XXXX.htmlで動作確認

- demo_XXXX.htmlをRstudio上でクリックすると「View in Web Browser」がでてくるので、クリックします。



ウェブブラウザ上で確認できます



それでは課題をはじめます。

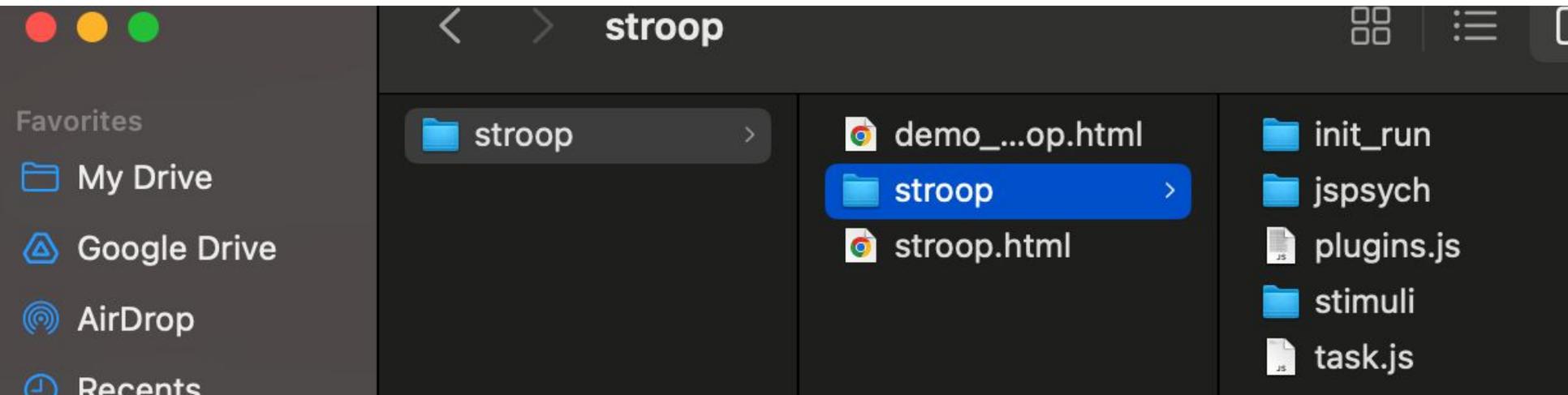
以下の「開始」を押すと、全画面になって課題が始まります。

開始

set_cbat()関数で設定したフォルダ構造

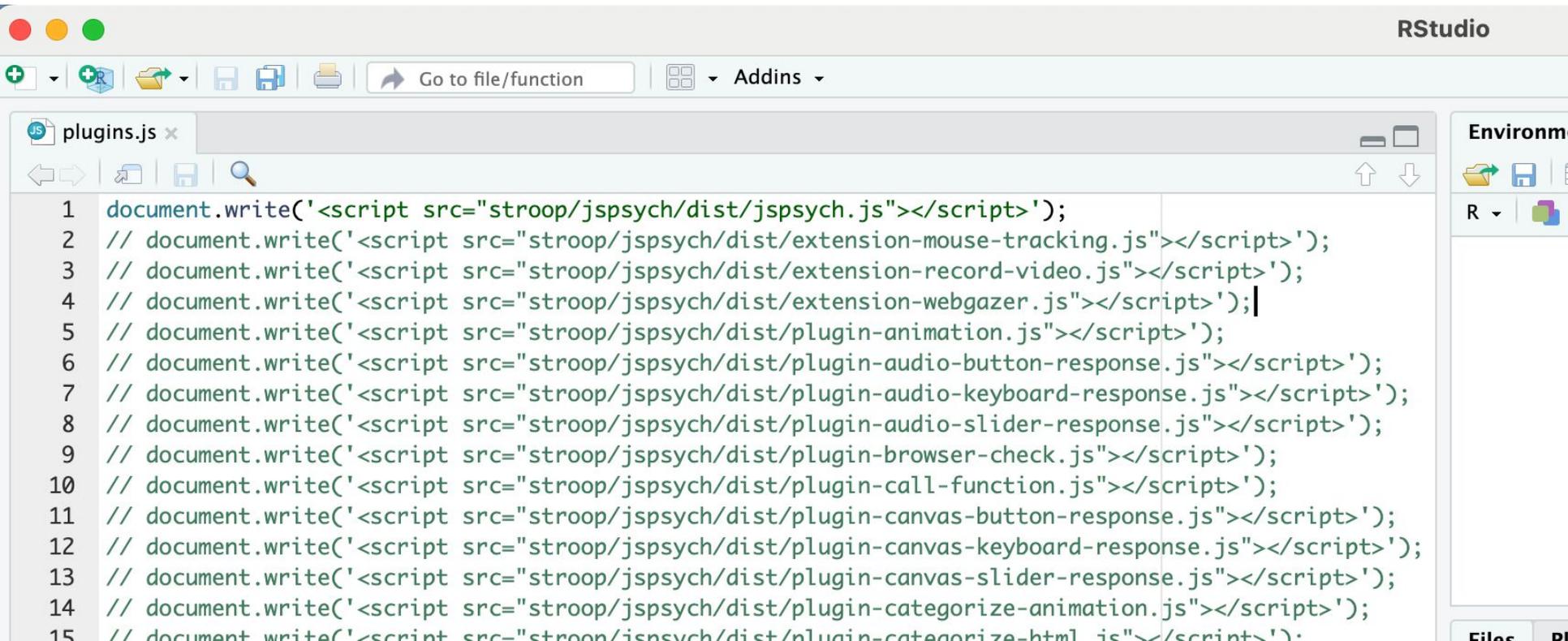
stroopフォルダ内のstroopフォルダ内に以下が含まれる。

- (1) init_run (jsPsychの初期化と実行についてのコードが入っている, 基本触らない)
- (2) jspsych (jsPsychのコードやプラグインが入っている, 基本さらわない)
- (3) plugins.js (後で説明)
- (4) stimuli (課題で使う刺激をいれる)
- (5) task.js (後で説明)



plugins.js

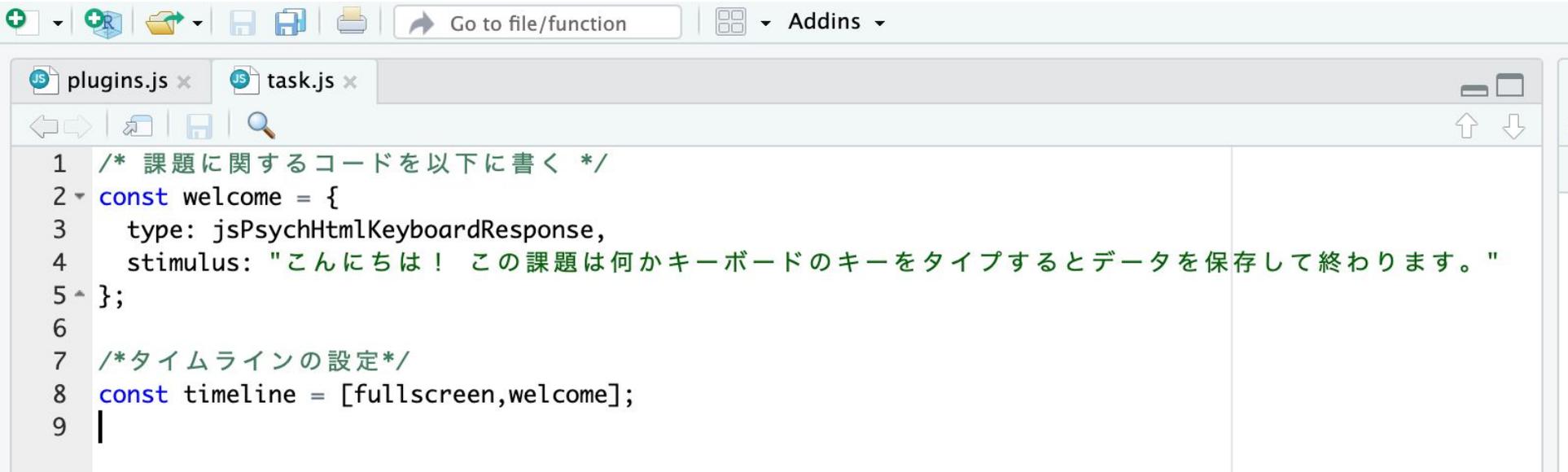
- plugins.jsでは、課題の作成に使うjsPsychプラグインを指定します。基本のjspsych.js, plugin-fullscreen.js, plugin-html-keyboard-response.js以外は //をいれており使われてないです。使うプラグインが出てきたら//をとってください。



```
1 document.write('<script src="stroop/jspsych/dist/jspsych.js"></script>');
2 // document.write('<script src="stroop/jspsych/dist/extension-mouse-tracking.js"></script>');
3 // document.write('<script src="stroop/jspsych/dist/extension-record-video.js"></script>');
4 // document.write('<script src="stroop/jspsych/dist/extension-webgazer.js"></script>');
5 // document.write('<script src="stroop/jspsych/dist/plugin-animation.js"></script>');
6 // document.write('<script src="stroop/jspsych/dist/plugin-audio-button-response.js"></script>');
7 // document.write('<script src="stroop/jspsych/dist/plugin-audio-keyboard-response.js"></script>');
8 // document.write('<script src="stroop/jspsych/dist/plugin-audio-slider-response.js"></script>');
9 // document.write('<script src="stroop/jspsych/dist/plugin-browser-check.js"></script>');
10 // document.write('<script src="stroop/jspsych/dist/plugin-call-function.js"></script>');
11 // document.write('<script src="stroop/jspsych/dist/plugin-canvas-button-response.js"></script>');
12 // document.write('<script src="stroop/jspsych/dist/plugin-canvas-keyboard-response.js"></script>');
13 // document.write('<script src="stroop/jspsych/dist/plugin-canvas-slider-response.js"></script>');
14 // document.write('<script src="stroop/jspsych/dist/plugin-categorize-animation.js"></script>');
15 // document.write('<script src="stroop/jspsych/dist/plugin-categorize-html.js"></script>');
```

task.js

- task.jsにjsPsychで行うことのメインの作業を書きます。
- この中身について、この後説明します。



```
1 /* 課題に関するコードを以下に書く */
2 const welcome = {
3   type: jsPsychHtmlKeyboardResponse,
4   stimulus: "こんにちは！ この課題は何かキーボードのキーをタイプするとデータを保存して終わります。"
5 };
6
7 /*タイムラインの設定*/
8 const timeline = [fullscreen, welcome];
9 |
```

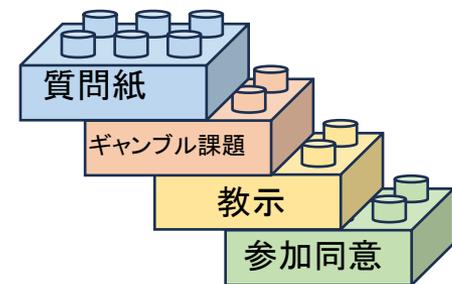
psyinfrによるjsPsychの準備のまとめ

- `set_cbat()`によって必要なファイルやフォルダ構造を用意する
- `demo_XXXX.html`で動作確認する(JATOS用の`XXX.html`については後で説明)
- フォルダ内の`plugins.js`(プラグインの設定), `stimuli`(課題で使う刺激の準備), `task.js`(課題についてのコード)を修正することで課題を作っていく。

なぜRやRstudioを使うのか？

- 心理系の学生が最も慣れている言語だからです。PythonやVSCodeでもいいのですが、もっとも導入しやすい言語から導入するのが良いかと思います（得意な人はpsyinfrで設定だけしてしまっって、VScodeで便利な補完機能とかを使っても良いと思います）
- いろいろなソフトを使わずに、RとRStudioでデータ収集(jsPsych)から、データ解析、論文執筆(Quarto)まで完結できるシステムを作るのが大切かな。

jsPsychとは？



- Josh de Leeuw によって開発されたWebブラウザをで行動実験を作るためのJavaScriptライブラリ。現在は, Josh de Leeuw とBecky GilbertとBjörn Luchterhandtが開発・管理をしている。
- 行動・認知実験に必要な要素をプラグインとしてまとめており, その組み合わせで課題が作れる。
- MITライセンス下で配布されているので, jsPsychを使って作った課題を公開することもでき, 簡単に試してもらうこともできる(オープンサイエンスに最適)
- 使ったらしっかり引用しよう！

de Leeuw, J.R., Gilbert, R.A., & Luchterhandt, B. (2023). jsPsych: Enabling an open-source collaborative ecosystem of behavioral experiments. *Journal of Open Source Software*, 8(85), 5351, <https://joss.theoj.org/papers/10.21105/joss.05351>.

jsPsychの基本構造

- 基本的には、以下の4つを書くことで認知・行動課題ができる。

①jsPsychの初期化

②タスクの定義(プラグイン)

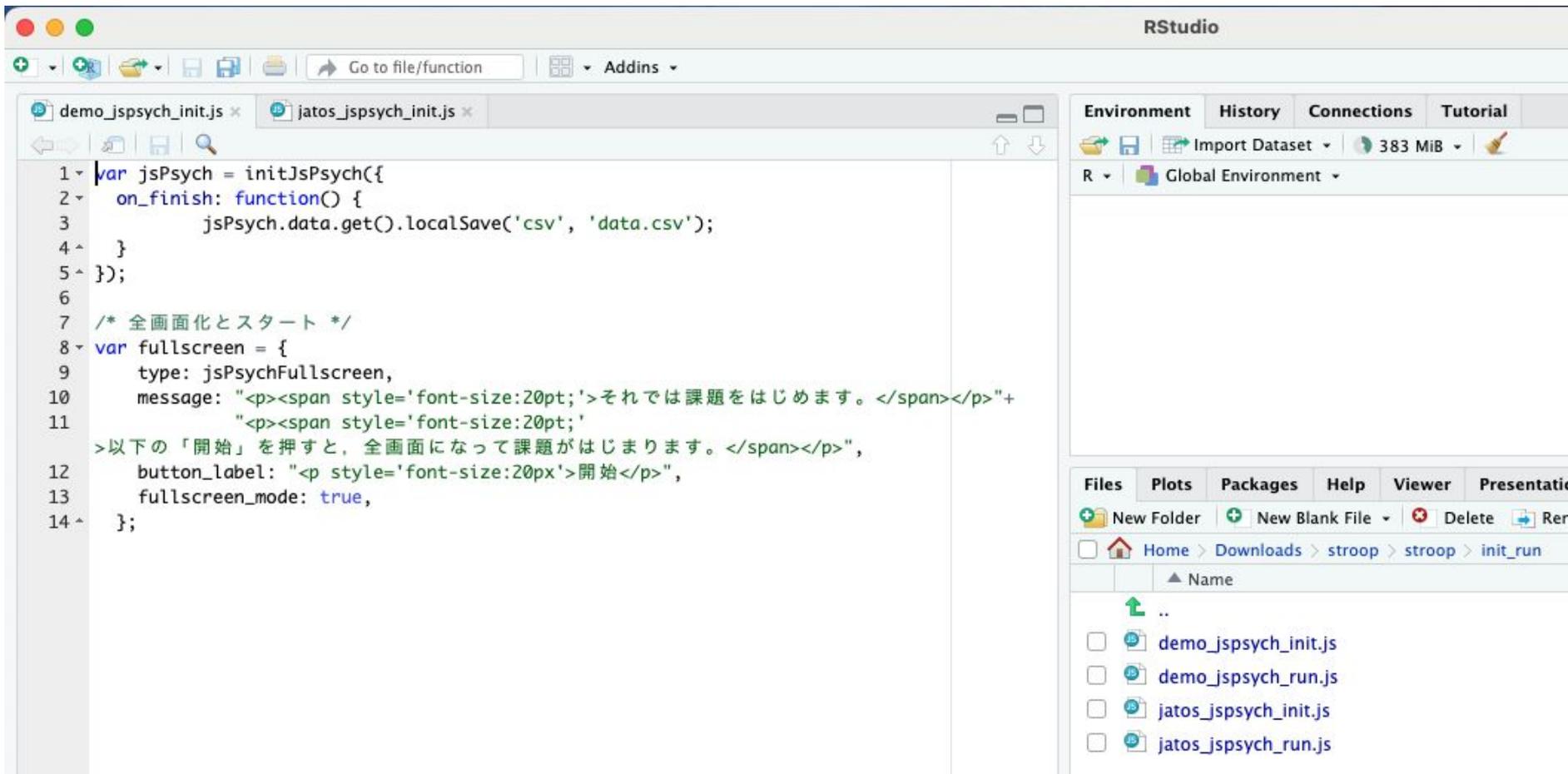
③タイムラインの定義

④タイムラインを走らせる

- `set_cbat()`を使えば、①と④は既に設定されているので、②と③を書けばよいだけです。

①jsPsychの初期化 (set_cbat())を使った場合)

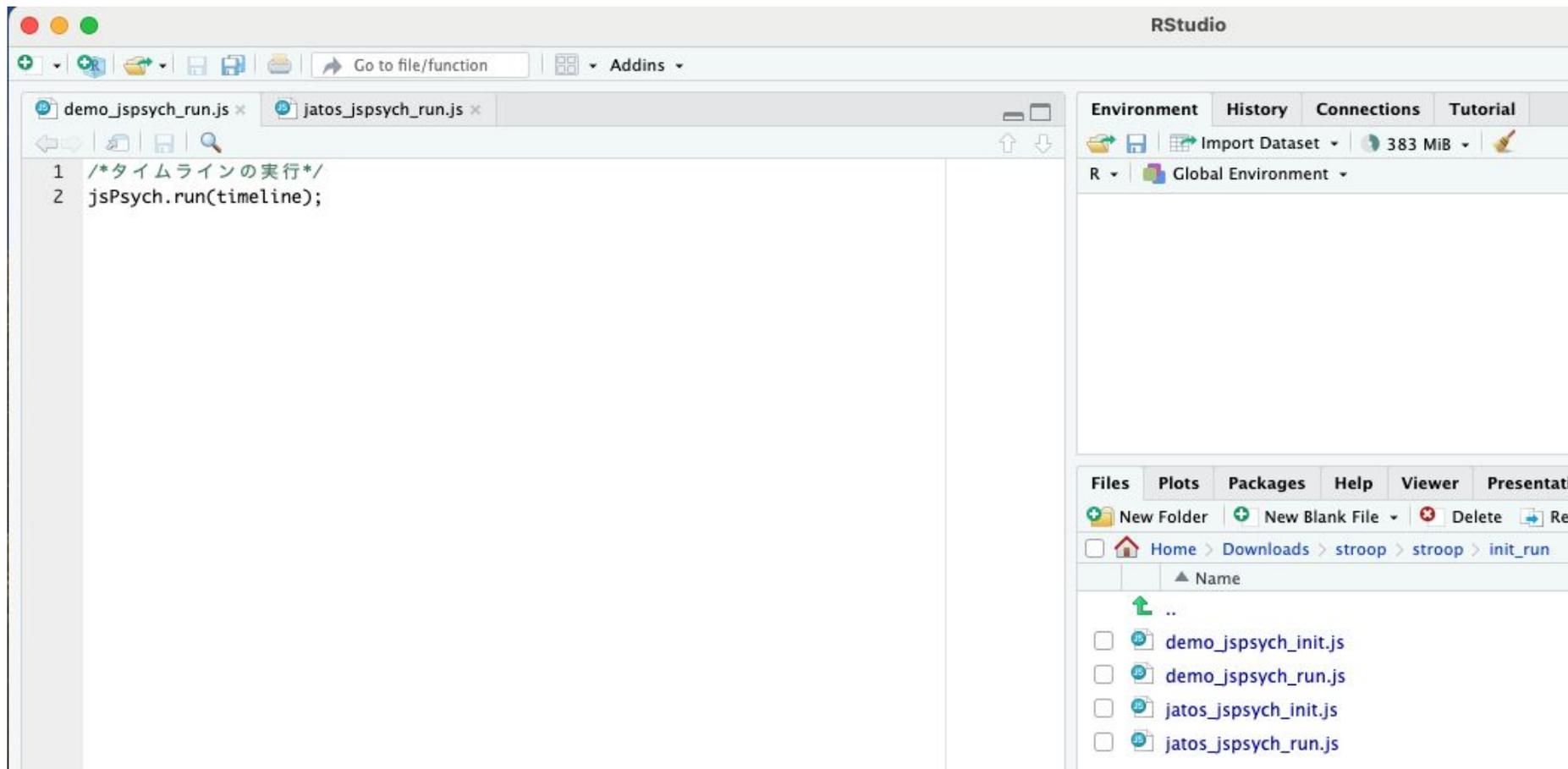
- set_cbat()では, 以下のようにjsPsychの初期化をしています(JATOS用とファイル分けています)。



```
1 var jsPsych = initJsPsych({
2   on_finish: function() {
3     jsPsych.data.get().localSave('csv', 'data.csv');
4   }
5 });
6
7 /* 全画面化とスタート */
8 var fullscreen = {
9   type: jsPsychFullscreen,
10  message: "<p><span style='font-size:20pt;'>それでは課題をはじめます。</span></p>"+
11           "<p><span style='font-size:20pt;'>
12 >以下の「開始」を押すと, 全画面になって課題が始まります。</span></p>",
13   button_label: "<p style='font-size:20px;'>開始</p>",
14   fullscreen_mode: true,
15 };
```

④タイムラインを走らせる (set_cbat())を使った場合)

- set_cbat()では、以下のようにjsPsychのタイムラインを走らせています(JATOS用とファイル分けています)。

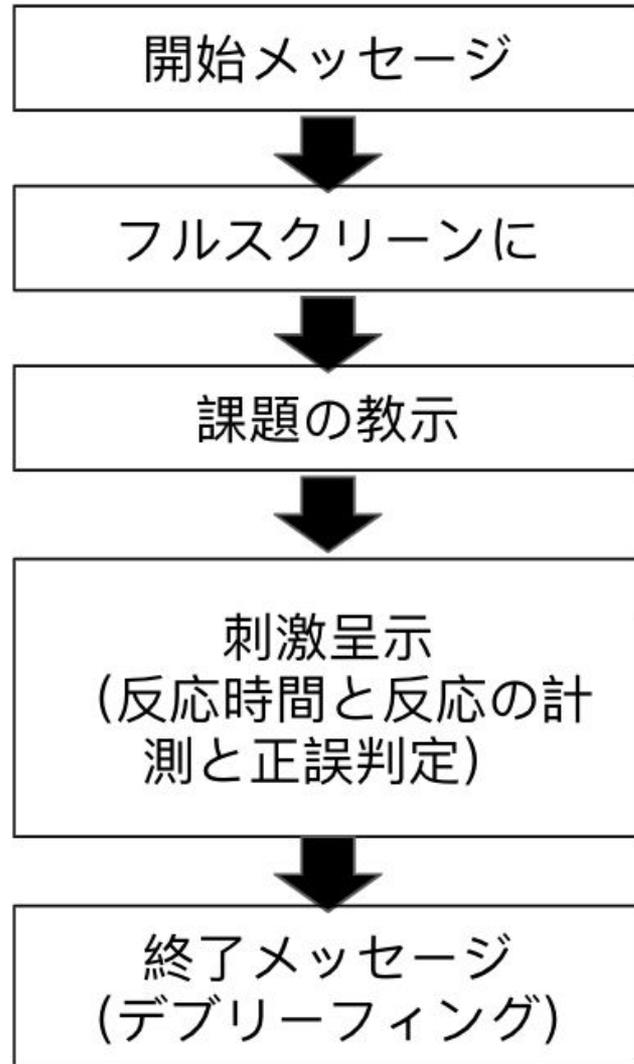


ストループ課題を作ろう！

- ストループ課題は、文字の意味と文字の色が同じだったり違ったりすることで、意味と色の情報が干渉しあうストループ効果を調べる課題です。
- 左のように意味と色が一致な刺激や右のように不一致な刺激を呈示し、意味ではなく色に関して反応するように求めます。その時の反応時間を計測して、一致条件よりも不一致条件の方が反応時間が長い場合にストループ効果が示されたとします。

あか あか

今回つくるストループ課題の構造



提示する刺激

- 1 赤色の「赤」
- 2 緑色の「赤」
- 3 黄色の「赤」
- 4 青色の「赤」
- 5 赤色の「緑」
- 6 緑色の「緑」
- 7 黄色の「緑」
- 8 青色の「緑」
- 9 赤色の「黄」
- 10 緑色の「黄」
- 11 黄色の「黄」
- 12 青色の「黄」
- 13 赤色の「青」
- 14 緑色の「青」
- 15 黄色の「青」
- 16 青色の「青」

刺激の数だけ繰り返すのを2回

プラグインとタイムライン

基本的に画面に何か文字を出してキーボードを反応をとるので、jsPsychHtmlKeyboardResponseプラグインを使う。

typeにプラグイン名、stimulusに刺激(なおHTMLの記法で装飾できる)をいれて、名前をつけて定義する(welcome)

定義したプラグインをtimelineとして定義する。

task.jsにコピーしてdemo_stroop.htmlで確認しよう！

```
/* 課題に関するコードを以下に書く */
```

```
/* 最初のメッセージ*/
```

```
const welcome = {  
  type: jsPsychHtmlKeyboardResponse,  
  stimulus: "実験にお越しいただき、ありがとうございます!キーボードのキーをどれか押して、開始してください。"  
};
```

```
/*タイムラインの設定*/
```

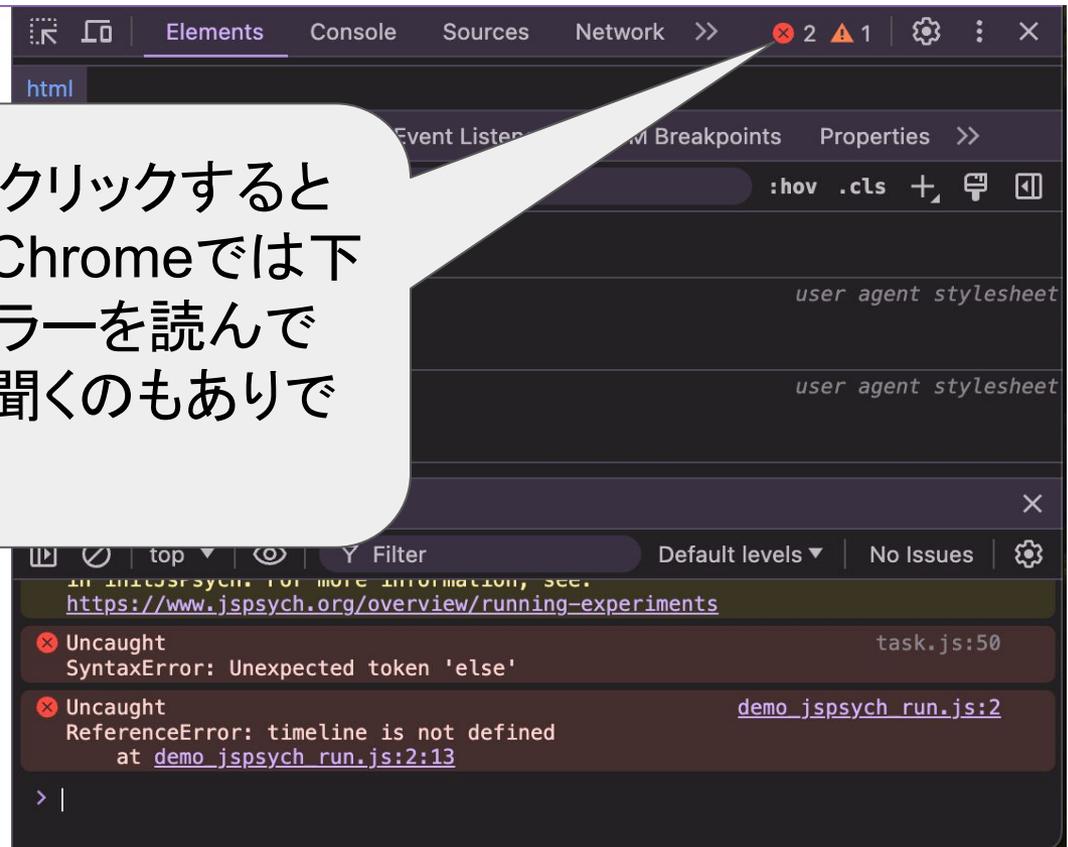
```
const timeline = [welcome, fullscreen];
```

[コードのリンク](#)

もし動かない場合

- demo_XXX.htmlなどをブラウザで確認していて、動かないor挙動がおかしい場合は、chromeなら画面を右クリックして「検証 (inspect)」をクリック、Microsoft EdgeならF12キーを押す。

いかにもエラーっぽい場所をクリックすると、エラーの内容が見えます。Chromeでは下の方にでていきますね。このエラーを読んで修正しても良いし、生成AIに聞くのもあります。



教示とストロープ課題の追加

- task.jsにコピーしてdemoで確認ください。
- 以下のように、教示を追加しています。welcomeと同じやり方です。紙面の制約で省略しています。

```
/* 教示*/  
  
const instruction = {  
  
  type: jsPsychHtmlKeyboardResponse,  
  
  stimulus: "ストロープの教示<省略>キーボードのキーをどれか押して、課題を開始してください。"  
  
};
```

[コードのリンク](#)

```
/* 課題に関するコードを以下に書く */  
  
/* 最初のメッセージ*/  
  
const welcome = {  
  
  type: jsPsychHtmlKeyboardResponse,  
  
  stimulus: "実験にお越しいただき、ありがとうございます。キーボードのキーをどれか押して、開始してください。"  
  
};  
  
/* 教示*/  
  
const instruction = {  
  
  type: jsPsychHtmlKeyboardResponse,  
  
  stimulus: "ストロープの教示<省略>キーボードのキーをどれか押して、課題を開始してください。"  
  
};  
  
/*ストロープ*/  
  
const stroop = {  
  
  type: jsPsychHtmlKeyboardResponse,  
  
  choices: ["d","f","j","k"],  
  
  stimulus: "<p style='color:green; font-size:60pt;'>赤</p>",  
  
  trial_duration: 2000,  
  
  prompt: '赤色ならd, 青色ならf, 緑色ならj, 黄色ならk',  
  
};  
  
/*タイムラインの設定*/  
  
const timeline = [welcome, fullscreen, instruction, stroop];
```

- ストロープを追加しています。choicesでキーを指定して, stimulusには色を使うためHTML記法を使っています。trial_durationを決めるとその時間内に反応がなければ次に進みます。promptで簡単な指示を書けます。最後にタイムラインの追加をしています。

```
/*ストロープ*/
```

```
const stroop = {  
  type: jsPsychHtmlKeyboardResponse,  
  choices: ["d", "f", "j", "k"],  
  stimulus: "<p style='color:green; font-size:60pt;'>赤</p>",  
  trial_duration: 2000,  
  prompt: '赤色ならd, 青色ならf, 緑色ならj, 黄色ならk',  
};
```

```
/* 課題に関するコードを以下に書く */
```

```
/* 最初のメッセージ*/
```

```
const welcome = {  
  type: jsPsychHtmlKeyboardResponse,  
  stimulus: "実験にお越しいただき, ありがとうございます。キーボードのキーをどれか押して, 開始してください。"  
};
```

```
/* 教示*/
```

```
const instruction = {  
  type: jsPsychHtmlKeyboardResponse,  
  stimulus: "ストロープの教示<省略>キーボードのキーをどれか押して, 課題を開始してください。"  
};
```

```
/*ストロープ*/
```

```
const stroop = {  
  type: jsPsychHtmlKeyboardResponse,  
  choices: ["d", "f", "j", "k"],  
  stimulus: "<p style='color:green; font-size:60pt;'>赤</p>",  
  trial_duration: 2000,  
  prompt: '赤色ならd, 青色ならf, 緑色ならj, 黄色ならk',  
};
```

```
/*タイムラインの設定*/
```

```
const timeline = [welcome, fullscreen, instruction, stroop];
```

複数のプラグインをタイムラインで定義

「固視点→ストロープ刺激」としたい場合に、右のようにtimeline:[]内にプラグインの定義をいれることで刺激呈示や反応取得の系列を作れる(choicesで"NO_KEYS"を指定すると反応をとりません)

```
/*ストロープ*/
const stroop = {
  timeline:[{
    type: jsPsychHtmlKeyboardResponse,
    choices: "NO_KEYS",
    stimulus: "<p style='font-size:60pt;'>+</p>",
    trial_duration: 1000,
  },{
    type: jsPsychHtmlKeyboardResponse,
    choices: ["d","f","j","k"],
    stimulus: "<p style='color:green; font-size:60pt;'>赤</p>",
    trial_duration: 2000,
    prompt: '赤色ならd, 青色ならf, 緑色ならj, 黄色ならk',
  }
];
```

[コードのリンク](#)

タイムライン変数 (timeline variables)

- 基本的にプラグインを用いた課題の一部の定義とタイムラインへの追加をすればいい。
- ただ、刺激呈示を繰り返したり、刺激を変えて呈示する場合は右のように複数定義をしていく形だとコードが長くなり不便すぎる。

→タイムライン変数を使う。

```
const stroop1 = {  
  <緑色の赤を呈示 コードは略>  
};  
  
const stroop2 = {  
  <緑色の緑を呈示 コードは略>  
};  
  
const stroop3 = {  
  type: jsPsychHtmlKeyboardResponse,  
  <赤色の緑を呈示 コードは略>  
};  
...  
  
/*タイムラインの設定*/  
  
const timeline = [welcome, fullscreen,  
instruction, stroop1, stroop2, stroop3...];
```

タイムライン変数 (timeline variables)

- 刺激を変えながら繰り返し呈示する刺激系列の情報をJSON(JavaScript Object Notation)形式で準備します。以下を追加ください。

*/*タイムライン変数の設定*/*

[コードのリンク](#)

```
const stimuli = [  
  {stimulus: "<p style='color:red; font-size:60pt;'>赤</p>"},  
  {stimulus: "<p style='color:green; font-size:60pt;'>赤</p>"},  
  {stimulus: "<p style='color:yellow;font-size:60pt;'>赤</p>"},  
  {stimulus: "<p style='color:blue; font-size:60pt;'>赤</p>"}  
];
```

CSV(エクセルなどの表形式)とJSON

- 心理学では表形式のCSVデータを使うことが多いが, JavaScriptを使うWeb系ではJSONが主流
- CSVにおける変数名とその内容をJSONでは, {変数名:内容}という形で記述する。毎回変数名があるので冗長であり見にくいだが, 複雑な構造も扱えるので便利になる。

stimulus
<p style='color:red;font-size:60pt;'>赤</p>
<p style='color:green;font-size:60pt;'>赤</p>
<p style='color:yellow;font-size:60pt;'>赤</p>
<p style='color:blue;font-size:60pt;'>赤</p>

```
[
```

```
{stimulus: "<p style='color:red; font-size:60pt;'>赤</p>"},
```

```
{stimulus: "<p style='color:green; font-size:60pt;'>赤</p>"},
```

```
{stimulus: "<p style='color:yellow;font-size:60pt;'>赤</p>"},
```

```
{stimulus: "<p style='color:blue; font-size:60pt;'>赤</p>"}
```

```
];
```

CSVからJSONへの変換

- エクセルで表形式で刺激系列などの準備をして、CSV出力します。それをRで読み込んで、jsonliteパッケージのtoJSON()でJSON形式に変換して、write()で書き出せばJSON形式になります。
- [data1.csvはこちらからDLください。](#)

```
library(jsonlite)
library(tidyverse)
csv_data <- read_csv("data1.csv")
json_data <- toJSON(csv_data)
write(json_data, "data.json")
```

[コードのリンク](#)

複数のプラグインをタイムラインで定義

- JSON形式で指定した stimuli とタイムライン変数に指定します(下のオレンジ色のコード)。
- 刺激呈示の設定の stimulus において `jsPsych.timelineVariable('stimulus')` のように、タイムライン変数の stimuli 内の stimulus を使うと宣言します。
- demo で4回刺激でるか確認しよう！

```
/*ストロープ*/
```

```
const stroop = {  
  timeline:[{  
    type: jsPsychHtmlKeyboardResponse,  
    choices: "NO_KEYS",  
    stimulus: "<p style='font-size:60pt;'>+</p>",  
    trial_duration: 1000,  
  },{  
    type: jsPsychHtmlKeyboardResponse,  
    choices: ["d","f","j","k"],  
    stimulus: jsPsych.timelineVariable('stimulus'),  
    trial_duration: 2000,  
    prompt: '赤色ならd, 青色ならf, 緑色ならj, 黄色  
    ならk',  
  }],  
  timeline_variables: stimuli  
};
```

[コードのリンク](#)

ランダムマイゼーション

- 用意した刺激をランダムする場合は
、 `randomize_order: true` でリストをランダムな順番にする。
- ランダムな順番で2回繰り返したい場合などは
、 `sample: {type: 'fixed-repetitions', size: 2}` を使う。

```
/*ストロープ*/
```

```
const stroop = {  
  timeline: [{  
    type: jsPsychHtmlKeyboardResponse,  
    choices: "NO_KEYS",  
    stimulus: "<p style='font-size:60pt;'>+</p>",  
    trial_duration: 1000,  
  }],  
  {  
    type: jsPsychHtmlKeyboardResponse,  
    choices: ["d", "f", "j", "k"],  
    stimulus: jsPsych.timelineVariable('stimulus'),  
    trial_duration: 2000,  
    prompt: '赤色ならd, 青色ならf, 緑色ならj, 黄色ならk',  
  }],  
  timeline_variables: stimuli,  
  //randomize_order: true  
  sample: {type: 'fixed-repetitions', size: 2}  
};
```

[コードのリンク](#)

動的パラメータ

- 参加者の反応に合わせてフィードバックを変えるようにする際には、パラメータに関数を設定する動的パラメータ (Dynamic parameters)を使う。
- 今回のストループ課題で参加者に正誤をFBとする (練習試行や動作確認で便利)。それにあたって、以下のようにタイムライン変数に使う変数を増やす (刺激の番号 (ID), 刺激のタイプ (一致, 不一致), 正反応)

	A	B	C	D
1	stimulus	no	stim_type	correct_key
2	<p style='color:red;font-size:60pt;'>赤</p>	1	congruent	d
3	<p style='color:green;font-size:60pt;'>赤</p>	2	incongruent	j
4	<p style='color:yellow;font-size:60pt;'>赤</p>	3	incongruent	k
5	<p style='color:blue;font-size:60pt;'>赤</p>	4	incongruent	f

タイムライン変数の変更

- [data2.csv](#)は[こちらからDL](#)して、右のRコードでJSONに変換ください。
- task.jsに↓タイムライン変数を設定します。

[コードのリンク](#)

```
library(jsonlite)
library(tidyverse)
csv_data <- read_csv("data2.csv")
json_data <- toJSON(csv_data)
write(json_data, "data.json")
```

*/*タイムライン変数の設定*/*

```
const stimuli = [
  {"stimulus": "<p style='color:red;font-size:60pt;'>赤</p>", "no": 1, "stim_type": "congruent", "correct_key": "d"},
  {"stimulus": "<p style='color:green;font-size:60pt;'>赤</p>", "no": 2, "stim_type": "incongruent", "correct_key": "j"},
  {"stimulus": "<p style='color:yellow;font-size:60pt;'>赤</p>", "no": 3, "stim_type": "incongruent", "correct_key": "k"},
  {"stimulus": "<p style='color:blue;font-size:60pt;'>赤</p>", "no": 4, "stim_type": "incongruent", "correct_key": "f"}];
```

[コードのリンク](#)

dataにタイムライン変数の値を保存する

data:{}を使って、保存されるデータにタイムライン変数の値を保存するように設定する
(オレンジ部分を追加ください)

```
/*ストロープ*/
```

```
const stroop = {
```

```
  timeline:{{
```

<固視点の内容を省略>

```
  }{{
```

```
    type: jsPsychHtmlKeyboardResponse,
```

```
    choices: ["d","f","j","k"],
```

```
    stimulus: jsPsych.timelineVariable('stimulus'),
```

```
    trial_duration: 2000,
```

```
    prompt: '赤色ならd, 青色ならf, 緑色ならj, 黄色ならk',
```

```
    data: {
```

```
      trial_id: jsPsych.timelineVariable('no'),
```

```
      stimulus_type: jsPsych.timelineVariable('stim_type'),
```

```
      correct_key: jsPsych.timelineVariable('correct_key')
```

```
    }
```

```
  }},
```

<タイムライン変数の内容を省略>

```
};
```

保存されたデータの確認

- demoで確認をすると、以下のようなデータが保存されます。responseとcorrect_keyが一致していたら正答、一致しなかったら誤答ですね。

rt	stimulus	response	trial_type	trial_index	plugin_version	time_elapsed	success	trial_id	stimulus_type	correct_key
905	実験にお越しいただき、ありがとうございます。キ		html-keyboard-re	0	2.0.0	906				
962			fullscreen	1	2.0.0	2871	TRUE			
3208	ストループの教示<省略>キーボードのキーをどれ		html-keyboard-re	2	2.0.0	6079				
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	3	2.0.0	7082				
1035	<p style='color:red;font-size:60pt;'>赤</p>	d	html-keyboard-re	4	2.0.0	8118		1	congruent	d
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	5	2.0.0	9120				
1377	<p style='color:green;font-size:60pt;'>赤</p>	j	html-keyboard-re	6	2.0.0	10498		2	incongruent	j
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	7	2.0.0	11500				
909	<p style='color:blue;font-size:60pt;'>赤</p>	k	html-keyboard-re	8	2.0.0	12410		4	incongruent	f
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	9	2.0.0	13412				
468	<p style='color:yellow;font-size:60pt;'>赤</p>	d	html-keyboard-re	10	2.0.0	13880		3	incongruent	k
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	11	2.0.0	14883				
997	<p style='color:yellow;font-size:60pt;'>赤</p>	f	html-keyboard-re	12	2.0.0	15881		3	incongruent	k
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	13	2.0.0	16883				
406	<p style='color:blue;font-size:60pt;'>赤</p>	d	html-keyboard-re	14	2.0.0	17290		4	incongruent	f
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	15	2.0.0	18291				
281	<p style='color:red;font-size:60pt;'>赤</p>	d	html-keyboard-re	16	2.0.0	18573		1	congruent	d
null	<p style='font-size:60pt;'>+</p>	null	html-keyboard-re	17	2.0.0	19574				
323	<p style='color:green;font-size:60pt;'>赤</p>	d	html-keyboard-re	18	2.0.0	19898		2	incongruent	j

```
/*ストロープ*/
```

```
const stroop = {
```

```
  timeline:[{
```

<固視点の内容を省略>

```
  },{
```

<ストロープの前半の内容を省略>

```
    data: {
```

```
      trial_id: jsPsych.timelineVariable('no'),
```

```
      stimulus_type: jsPsych.timelineVariable('stim_type'),
```

```
      correct_key: jsPsych.timelineVariable('correct_key')
```

```
    }
```

```
    on_finish: function(data){
```

```
      if(jsPsych.pluginAPI.compareKeys(data.response,data.correct_key)){
```

```
        data.correct = 1;
```

```
      }else{
```

```
        data.correct = 0;
```

```
      }
```

```
    }
```

```
  }
```

```
],
```

コードのリンク

on_finishで試
行が終わった
ら動作する関
数を指定する。
dataを引数とし
てresponseと
correct_keyを
比較して、一致
してたら
、correctに
1, 違うと0を入
れる

保存されたデータの確認

- demoで確認をすると、以下のようなデータが保存されます。correctの列が追加されました。

rt	stimulus	response	trial_type	trial_index	plugin_version	time_elapsed	success	trial_id	stimulus_type	correct_key	correct
5009	実験にお起		html-keybc	0	2.0.0	5009					
1363			fullscreen	1	2.0.0	7377	TRUE				
1648	ストループ		html-keybc	2	2.0.0	9027					
null	<p style='f	null	html-keybc	3	2.0.0	10031					
null	<p style='c	null	html-keybc	4	2.0.0	12035		2	incongruent	j	0
null	<p style='f	null	html-keybc	5	2.0.0	13038					
null	<p style='c	null	html-keybc	6	2.0.0	15042		4	incongruent	f	0
null	<p style='f	null	html-keybc	7	2.0.0	16045					
null	<p style='c	null	html-keybc	8	2.0.0	18049		3	incongruent	k	0
null	<p style='f	null	html-keybc	9	2.0.0	19051					
1142	<p style='c	d	html-keybc	10	2.0.0	20195		1	congruent	d	1
null	<p style='f	null	html-keybc	11	2.0.0	21197					
1081	<p style='c	f	html-keybc	12	2.0.0	22279		3	incongruent	k	0
null	<p style='f	null	html-keybc	13	2.0.0	23282					
914	<p style='c	j	html-keybc	14	2.0.0	24197		4	incongruent	f	0
null	<p style='f	null	html-keybc	15	2.0.0	25200					
1247	<p style='c	k	html-keybc	16	2.0.0	26448		2	incongruent	j	0
null	<p style='f	null	html-keybc	17	2.0.0	27451					
719	<p style='c	d	html-keybc	18	2.0.0	28172		1	congruent	d	1

```
/*ストロープ*/
```

```
const stroop = {
```

```
  timeline:{{
```

<固視点の内容を省略>

```
  },{
```

<ストロープの内容を省略>

```
  },{
```

```
    type: jsPsychHtmlKeyboardResponse,
```

```
    choices: "NO_KEYS",
```

```
    stimulus: function(){
```

```
      if(jsPsych.data.get().last(1).values()[0].correct == 1){
```

```
        return "<p style='font-size:60pt;'>正解</p>";
```

```
      }else{
```

```
        return "<p style='font-size:60pt;'>不正解</p>";
```

```
      }
```

```
    },
```

```
    trial_duration: 1000,
```

```
  }],
```

<タイムライン変数の内容を省略>

```
};
```

コードのリンク

フィードバックする

jsPsych.data.get().last(1).values()[0]を使うと、1つ前の内容のデータにアクセスできる。

correctが1なら正解、それ以外なら不正解を返す。

demoを動かすと正誤がフィードバックされるようになっていると思います。

おまけ: 固視点の呈示間隔を変動させる

固視点のtrial_durationに以下の用に関数をしこむと、固視点の呈示間隔を変動させることができる。

動的パラメータは使いこなすと色々できる。

```
const stroop = {  
  timeline: {  
    type: jsPsychHtmlKeyboardResponse,  
    choices: "NO_KEYS",  
    stimulus: "<p style='font-size:60pt;'>+</p>",  
    trial_duration: function(){  
      var random_duration = jsPsych.randomization.sampleWithoutReplacement([800, 1000, 1200, 1400])[0];  
      return random_duration;  
    },  
  },  
};
```

<以下省略>

[コードのリンク](#)

jsPsychのミニマムエッセンス

- 基本的には以下の4つを設定(set_cbat()なら②と③)

①jsPsychの初期化

②タスクの定義(プラグイン)

③タイムラインの定義

④タイムラインを走らせる

- jsPsychのプラグインでタスクを定義して、それをタイムラインで流す。繰り返しや指定した刺激を流す場合は、タイムライン変数を活用する。参加者の反応に合わせて呈示を変える場合は、動的パラメータを使う。

「生成AIを用いたjsPsychプログラミング」

目次

1. 生成AI時代に必要なプログラミング知識
2. jsPsychの基礎知識
- 3. NotebookLMを用いたjsPsychプログラミング**
4. データ収集方法

生成AIの活用

- 基本的には、課題の構造を決めたら、プラグインとタイムライン変数を駆使して課題を作って、タイムラインに入れる。
- インターネット上に情報も多くあるので、検索しながら進めることもできるが、生成AIと対話をしつつ作成するのが効率的である。
- 生成AIはハルシネーションが怖いので(jsPsychはバージョンの問題もある)、ソースがはっきりした生成AIを使う。

NotebookLMを使おう！

- NotebookLMは GoogleのGeminiベースのパーソナライズAI(実験的なサービスで現状無料)
- ソース(Google Doc, PDFなど)を読み込んでそれに基づいて回答してくれるので、ハルシネーションが抑えられる。

<https://notebooklm.google/>

- jsPsychだけでなく、気になるトピックの記事やPDFをどんどんつつこめば、要約したり、質問に回答してくれるので、研究にフル活用できる(かなり便利な機能です)。

NotebookLMに入れるソースを準備しよう！

(1) jsPsych本家の情報

- jsPsychのウェブサイトソースにできるといいが、全てのページを指定する必要がある(50を超えるとNotebookLMは読み込めない。今後変わる可能性はある)
- jsPsychのウェブサイトにある内容をGoogle Docに貼り付けて使う(↓)

[プラグイン jsPsych v8用notebokLMソース](#)

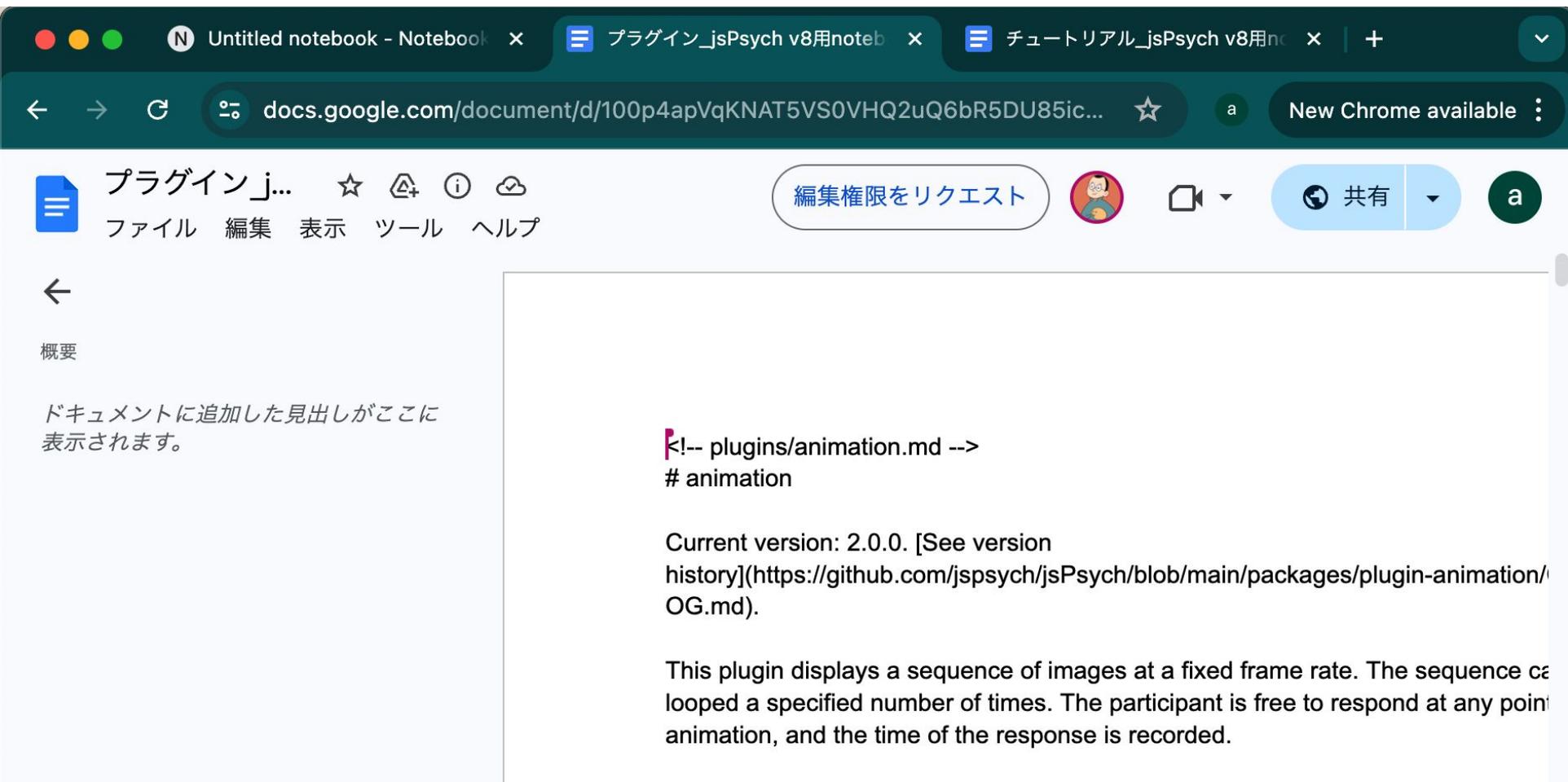
[チュートリアル jsPsych v8用notebokLMソース](#)

参考: jsPsychのサイトの情報を Google Docに保存する方法

- [jsPsychのGithubリポジトリ](#)の中身をダウンロードして、チュートリアルやプラグインの説明をしたMarkdownファイル(.md)が入っているdocsフォルダを開く
- vsCodeのMarkdownCatプラグインをつかって、関連するMarkdownファイルを選択して統合する(プラグインとその他で分けました)。統合したMarkdownファイルの中身をグーグルドキュメントに貼り付ける(2つのファイルで10万語ある)

Google Docをご自身のGoogle Driveに保存する

リンクをクリックして開く



Untitled notebook - Notebook x プラグイン_jsPsych v8用noteb x チュートリアル_jsPsych v8用nc x +

docs.google.com/document/d/100p4apVqKNAT5VS0VHQ2uQ6bR5DU85ic... ☆ a New Chrome available

プラグイン_j... ☆ 編集 ツール ヘルプ

編集権限をリクエスト

共有

概要

ドキュメントに追加した見出しがここに
表示されます。

```
k!-- plugins/animation.md -->
# animation

Current version: 2.0.0. [See version
history](https://github.com/jspsych/jsPsych/blob/main/packages/plugin-animation/
OG.md).

This plugin displays a sequence of images at a fixed frame rate. The sequence ca
looped a specified number of times. The participant is free to respond at any point
animation, and the time of the response is recorded.
```

Google Docをご自身のGoogle Driveに保存する

ファイル > コピーを作成 をクリックします。



編集権限をリクエスト



概要

ドキュメント

表示

新規作成

開く

コピーを作成

共有

メール

ダウンロード

名前を変更

移動

ドライブにショートカットを追加

ゴミ箱に移動

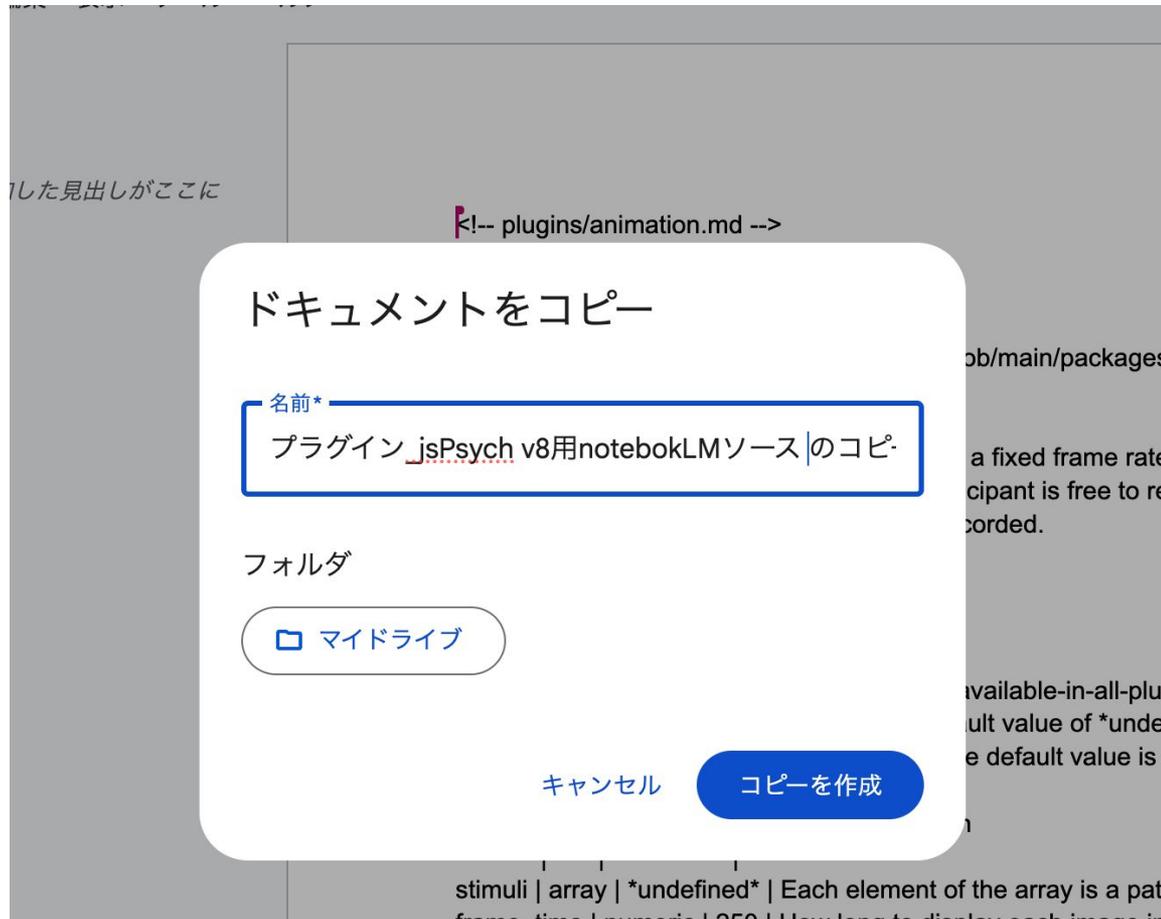
```
<!-- plugins/animation.md -->  
# animation
```

Current version: 2.0.0. [See version history](<https://github.com/jspsych/jsPsych/blob/main/OG.md>).

This plugin displays a sequence of images at a fixed looped a specified number of times. The participant i animation, and the time of the response is recorded.

Parameters

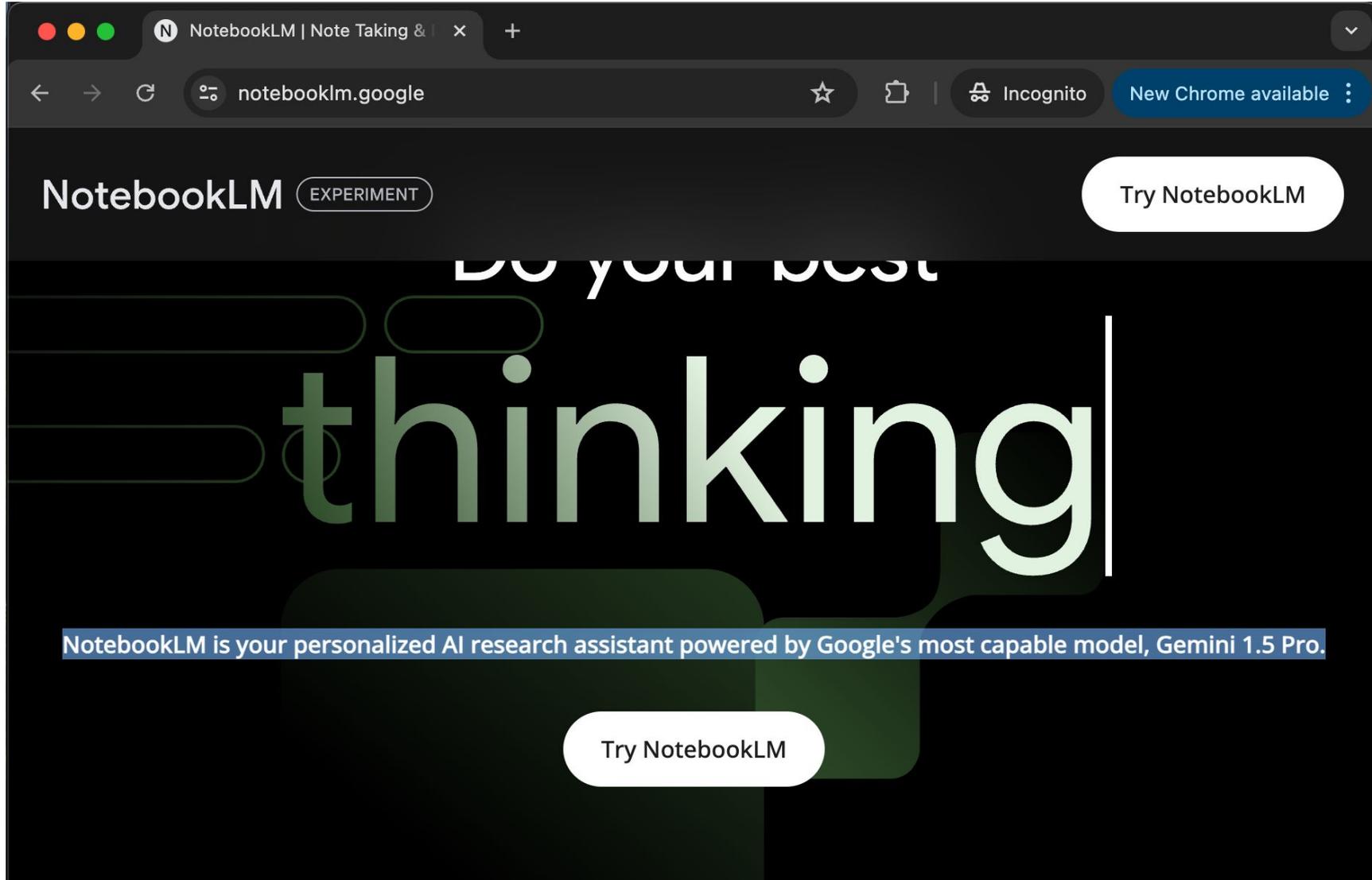
名前をつけてDriveにコピーする。



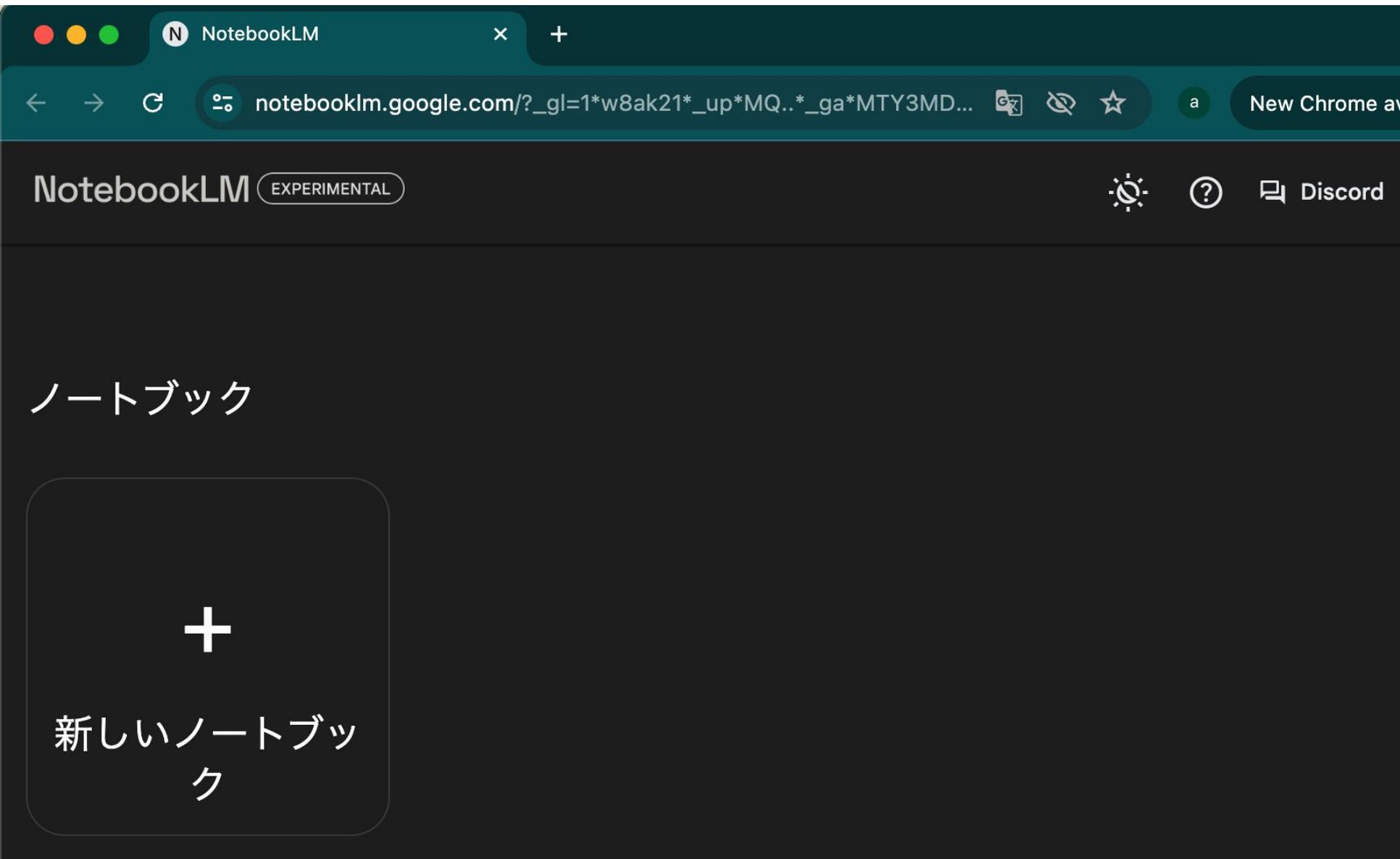
チュートリアルの方のファイルも同様にコピーする。

生成AI (NotebookLM) を使おう！

- Try NotebookLMをクリックする

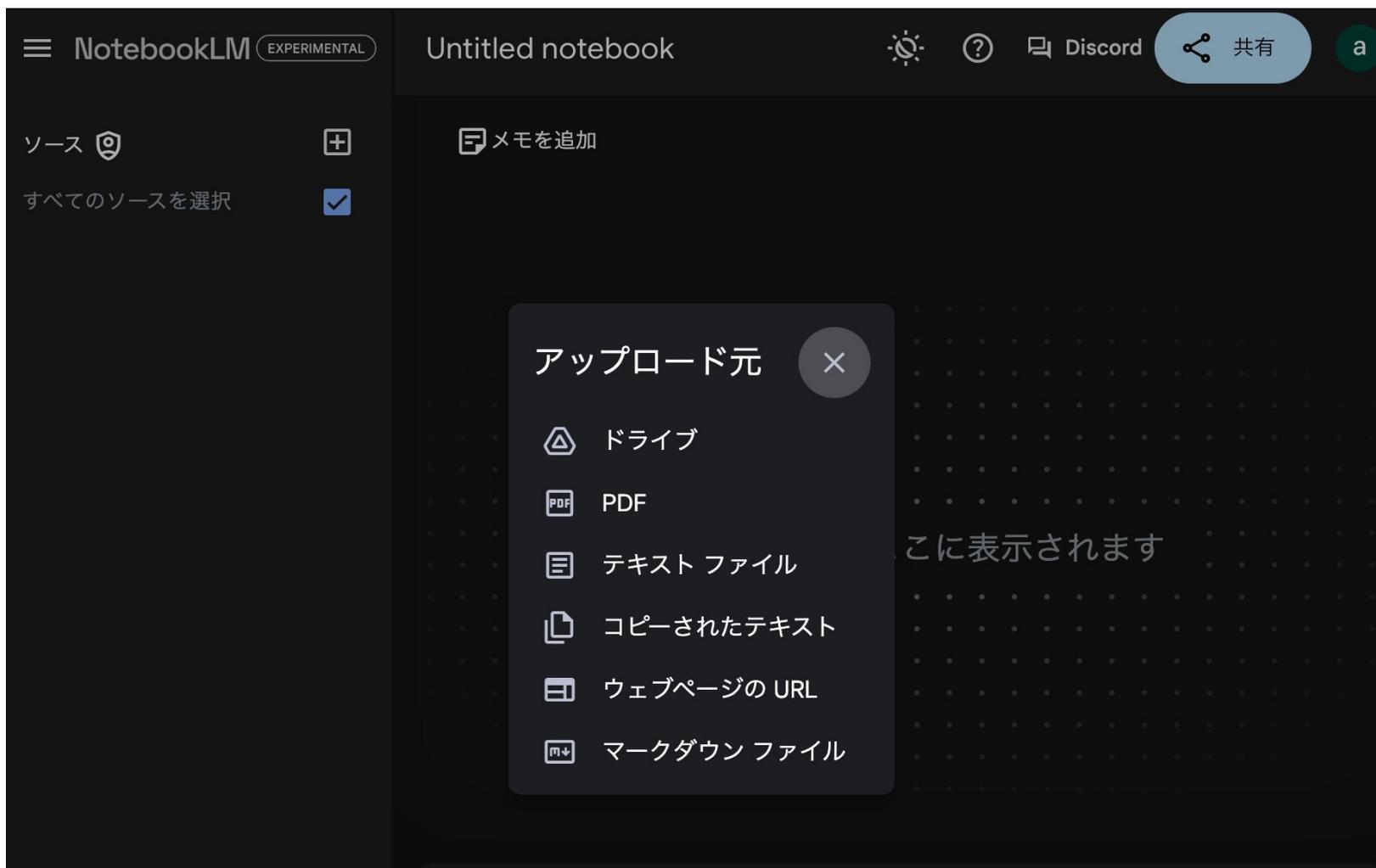


ホーム画面で、「+新しいノートブック」をクリック



ソースを指定

ドライブをクリックする。



ソースを指定

さきほどコピーしたGoogle Docを選択する。



チュートリアルもプラグインも両方追加する。

ソースをもとに賢くなっている！

☰ NotebookLM EXPERIMENTAL

Untitled notebook

☰ ソース ⓘ +

すべてのソースを選択 ✓

☰ チュートリアル_jsPsy... ✓

☰ プラグイン_jsPsych v8... ✓

* ノートブック ガイド

生成

- よくある質問
- 学習ガイド
- 目次
- タイムライン
- ブリーフィング・ドキュメント

概要

このテキストは、行動科学実験用のブラウザベースのソフトウェアであるjsPsychの開発者向けドキュメントからの抜粋です。**プラグインと拡張機能の作成方法、実験の実行とデータの保存に関する考慮事項、テストと貢献のためのガイドラインについて説明します。** また、マウス追跡、ビデオ録画、視線追跡などの機能を追加する拡張機能と、WebGazerライブラリとの統合についても詳しく説明しています。最後に、動的なパラメータの使用、スタイル設定のカスタマイズ、タイムラインの作成など、jsPsychの主要な機能と構成について詳しく説明します。

☰ チャットを表示 2 個のソース 入力を開始します... → * ノートブック ガイド

指導教員よりも優しく的確に教えてくれる

jsPsychについて教えて

jsPsych は、Web ブラウザで動作する行動実験を作成するための JavaScript ライブラリです。 ① jsPsych は、行動科学者が実験をオンラインで簡単に実施できるように設計されています。 ② jsPsych は、実験中に発生するイベントの種類を定義するプラグインで構成されています。 ③ jsPsych は、最も一般的に使用されている 4 つの Web ブラウザの最新バージョン (Chrome、Firefox、Safari、Edge) をサポートしています。 ④ jsPsych はオープンソースプロジェクトであり、多くの貢献者がいます。 ① jsPsych は現在、Josh de Leeuw、Becky Gilbert、Björn Luchterhandt のコアチームによって管理されています。 ① jsPsych を学術研究で使用する場合の推奨される引用は次のとおりです。

de Leeuw, J.R., Gilbert, R.A., & Luchterhandt, B. (2023). jsPsych: Enabling an open-source collaborative ecosystem of behavioral experiments. *Journal of Open Source Software*, 8(85), 5351, <https://joss.theoj.org/papers/10.21105/joss.05351>. ⑤

この論文は jsPsych の最新の記述であり、現在のコアチームメンバー全員が含まれています。 ⑤



jsPsychの主な貢献者は？

jsPsych実験のデータはどこに保存される？

jsPsych実験で使用するプラグイン

指導教員が聞かれても答えられないことも答えてくれる

jsPsychのversion 8.0.0の主な変更点は？

jsPsych バージョン 8.0.0 の主な変更点は以下のとおりです。

- **jsPsych.timelineVariable() の変更点:** `jsPsych.timelineVariable()` は、試行が実行されるときに jsPsych がタイムライン変数の値を評価するために使用するプレースホルダーオブジェクトを返すようになりました。 1 以前は、`jsPsych.timelineVariable()` は現在のタイムライン変数の値をすぐに返していました。この変更により、`jsPsych.timelineVariable()` の動作と `jsPsych.evaluateTimelineVariable()` の動作がより明確に区別されるようになりました。 2
- **Timeline変数の保存:** `jsPsych.getAllTimelineVariables()` のサポートが削除され、試行レベルのプロパティ `save_timeline_variables` に置き換えられました。試行のすべてのタイムライン変数を jsPsych のデータに保存する必要がある場合は、試行で `save_timeline_variables: true` を設定できます。 3
- **試行パラメータ:** プラグインの `info` オブジェクトで `array: true` とマークされているパラメータがある場合、jsPsych はパラメータが配列でない場合はエラーをスローするようになりました。 3 以前は、一部のプラグインでは、一部のパラメータを配列または非配列のいずれかにすることが許可されていました。jsPsych のメインリポジトリに含まれているプラグインの場合、影響を受け

とても指導教員には言えない口調でも、色々と教えてもらえる。

質問紙でデータを取りたいのですが、どのプラグインを使えばいい？使い方は？

jsPsychを使ってアンケート調査を行う場合、**jsPsych Surveyプラグイン**を使うと便利です。ソースに掲載されているSurveyプラグインは以下のとおりです。



- **jsPsychSurveyText**: 自由記述式の質問を作成できます。 1 2 3
- **jsPsychSurveyHtmlForm**: HTMLを用いてより自由度の高いフォームを作成できます。 4 5
- **jsPsychSurveyLikert**: リッカート尺度を使った質問を作成できます。 6 7 8 9
- **jsPsychSurveyMultiChoice**: 単一選択式の質問を作成できます。 10 11 12
- **jsPsychSurveyMultiSelect**: 複数選択式の質問を作成できます。 13 14 15

これらのプラグインを使用すると、質問項目、回答形式、表示オプションなどを柔軟に設定できます。

例：jsPsychSurveyTextを使った簡単な質問

```
var trial = {  
  type: jsPsychSurveyText,  
  questions: [  

```

jsPsychには詳しいのですが...

- NotebookLMはソースに基づいて回答するので、ソースにないことは回答しない。ストループ課題は知らない。

ストループ課題について教えてください。

提供されたソースは、jsPsychの技術的なドキュメントに焦点を当てており、ストループ課題自体の説明は含まれていません。ストループ課題についてより詳しく知りたい場合は、心理学の教科書や、ストループ課題に関する学術論文を調べてみることをお勧めします。これはソースに含まれていない情報なので、ご自身で確認する必要があるかもしれません。



NotebookLMに入れるソースを準備しよう！

(2)関連したサイトの情報

国内には色々と有用なサイトがあるので、それをソースにいれることもできます。

- [国里のウェブサイト](#)
- [西山さんのZennの記事](#)
- [黒木先生のQiitaの記事](#)

今回は、作成する課題についてGoogle Docにまとめたので、こちらを先ほどと同じくドライブにコピーしてNotebookLMに読ませます。

- [ストループ課題について](#)

ストループ課題に詳しくなった

ストループ課題について教えてください

ストループ課題は、文字の意味と文字の色が一致する場合と不一致する場合で、意味と色の情報が干渉し合うストループ効果を調べる課題です。① 典型的なストループ課題では、意味と色が一致する刺激と不一致する刺激を呈示し、被験者には色の判断を求めます。① 色の判断にかかる反応時間を測定し、不一致条件の方が一致条件よりも反応時間が長い場合にストループ効果が示されたとみなされます。これは、不一致条件では色の判断に意味の情報が干渉するため、反応が遅くなるためです。①

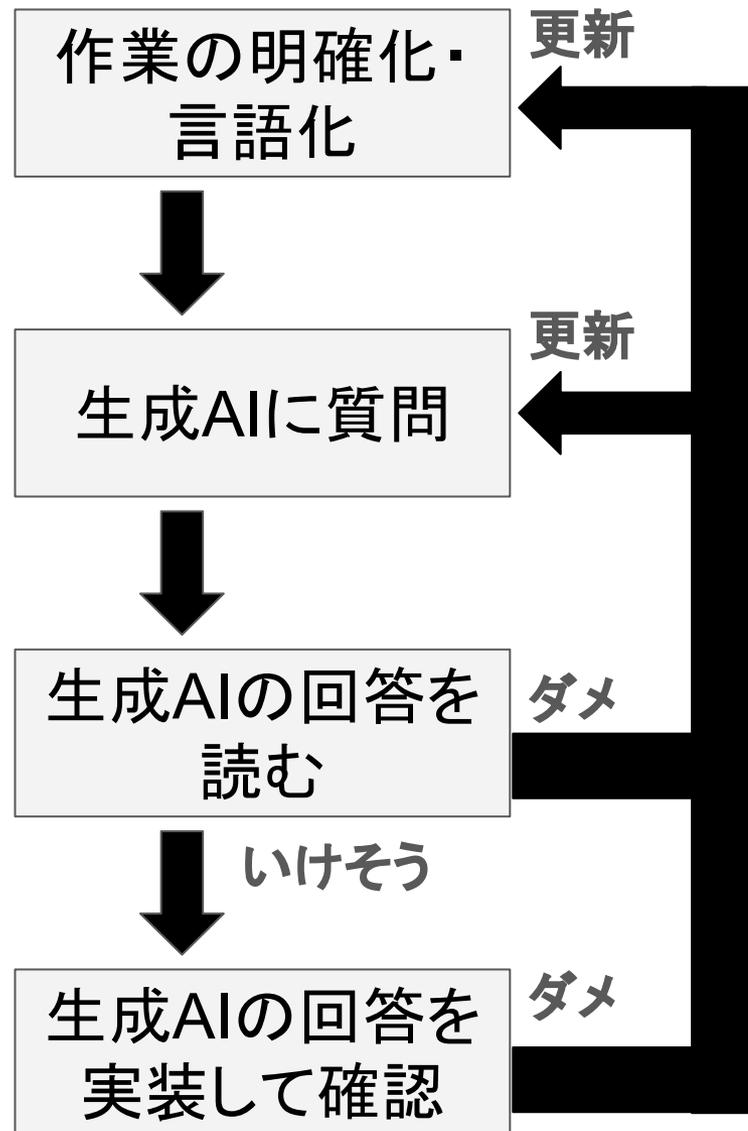
ストループ課題には様々な種類がありますが、赤色、青色、緑色、黄色の4色の刺激を用いて、一致条件と不一致条件を作成し、キーボードのボタン押しで反応時間を測定するタイプがあります。①

ソースは、ストループ課題のプログラミング方法に関する情報を提供しており、ストループ課題の設計図の作成方法などについて説明しています。① しかし、ソースはストループ課題の心理学的な側面や、ストループ効果に関する詳細な理論については説明していません。これらの情報については、心理学の教科書やストループ課題に関する学術論文を参照する必要があります。この情報はソースには含まれていないため、ご自身で確認する必要があるかもしれません。



生成AIとストループ作ってみましょう！

- 最新のjsPsych v8(2024/7/17に出た)に対応し, ストループ課題についてもある程度情報をもっている生成AIが準備できました。
- 右のように生成AIと対話しつつ作成してみましょう！



ちゃんと意図を読んで回答してくれてる。

「この課題では、以下のような色のついた単語を見ていただきます。青<赤色> 単語の意味は無視して、それぞれの単語の「色」を以下のキーボードのキーを押して回答してください。左手の人差し指と中指でdとf、右手の人差し指と中指でjとkを押してください。・赤色<赤色>の単語ならdを押す・青色<青色>の単語ならfを押す・緑色<緑色>の単語ならjを押す・黄色<黄色>の単語ならkを押す上の例の場合だと、赤色で「青」と書いてありますので、dを押します キーボードのキーをどれか押すと課題が始まります」という教示をjsPsychで作りたいのですが、改行もちゃんとしたいです。

提供された情報に基づいて、jsPsych で改行を含めた教示を作成する方法について説明します。

ソース **1** **2** **3** は、jsPsych で実験を作成する際のコード例を示しています。これらの例では、jsPsychHtmlKeyboardResponse プラグインを使用して刺激を提示し、参加者からのキーボード入力を取得する方法が示されています。

jsPsychHtmlKeyboardResponse プラグインを使用して、指定された教示を作成するコード例を以下に示します。

```
var instructions = {
  type: jsPsychHtmlKeyboardResponse,
  stimulus: `
    <p>この課題では、以下のような色のついた単語を見ていただきます。</p>
    <p style="color: red;">青</p>
    <p>単語の意味は無視して、それぞれの単語の「色」を以下のキーボードのキーを押して回答してください。</p>
    <p>左手の人差し指と中指で「d」と「f」、右手の人差し指と中指で「j」と「k」を押してください。</p>
    <ul>
      <li><span style="color: red;">赤色</span>の単語ならdを押す</li>
      <li><span style="color: blue;">青色</span>の単語ならfを押す</li>
      <li><span style="color: green;">緑色</span>の単語ならjを押す</li>
      <li><span style="color: yellow;">黄色</span>の単語ならkを押す</li>
    </ul>
  `
}
```



この課題では、以下のような色のついた単語を見ていただきます。

青

単語の意味は無視して、それぞれの単語の「色」を以下のキーボードのキーを押して回答してください。

左手の人差し指と中指で「d」と「f」、右手の人差し指と中指で「j」と「k」を押してください。

- 赤色の単語ならdを押す
- 青色の単語ならfを押す
- 緑色の単語ならjを押す
- 黄色の単語ならkを押す

上の例の場合だと、赤色で「青」と書いてありますので、dを押します

キーボードのキーをどれか押すと課題が始まります

AIを信じすぎずに確認する。いい感じ。

でも上の赤色の「青」は大きくしたいな。あとリスト形式微妙だな。→指示をより明確にしよう！

「この課題では、以下のような色のついた単語を見ていただきます。青<赤色> 単語の意味は無視して、それぞれの単語の「色」を以下のキーボードのキーを押して回答してください。左手の人差し指と中指でdとf, 右手の人差し指と中指でjとkを押してください。・赤色<赤色>の単語ならdを押す・青色<青色>の単語ならfを押す・緑色<緑色>の単語ならjを押す・黄色<黄色>の単語ならkを押す上の例の場合だと、赤色で「青」と書いてありますので、dを押します キーボードのキーをどれか押すと課題が始まります」という教示をjsPsychで作りたいのですが、改行もちゃんとしたいです。「青<赤色>」をもっと大きなサイズのフォントにして、「赤色<赤色>の単語ならdを押す」のリスト形式はやめてください。

ソースに基づいて、jsPsychでストループ課題の教示を作成する方法を説明します。

教示の例



```
<!DOCTYPE html>
<html>
  <head>
    <title>ストループ課題</title>
    <script src="https://unpkg.com/jspsych@8.0.1"></script>
    <script src="https://unpkg.com/@jspsych/plugin-html-keyboard-response@2.0.0"></script>
    <link href="https://unpkg.com/jspsych@8.0.1/css/jspsych.css" rel="stylesheet" type="text/css" />
  </head>
  <body></body>
```

してほしいことをちゃんと伝える。人間もAIも同じ。

この課題では、以下のような色のついた単語を見ていただきます。

青

単語の意味は無視して、それぞれの単語の「色」を以下のキーボードのキーを押して回答してください。

左手の人差し指と中指で「d」と「f」、右手の人差し指と中指で「j」と「k」を押してください。

- ・ 赤色の単語なら「d」を押す
- ・ 青色の単語なら「f」を押す
- ・ 緑色の単語なら「j」を押す
- ・ 黄色の単語なら「k」を押す

上の例の場合だと、赤色で「青」と書いてありますので、「d」を押します

キーボードのキーをどれか押すと課題が始まります

いい感じ！

ストループ課題をjsPsychで作ろうと思います。そのための刺激系列をJSON形式で用意したいのですが、できますか？

はい、jsPsychでストループ課題を作成するための刺激系列をJSON形式で用意することができます。

ソースでは、jsPsychでアンケート調査を行う際に使用できるプラグインとその基本的な使い方が紹介されています。しかし、ストループ課題の刺激系列を作成するための具体的な情報は提供されていません。

ストループ課題を作成するためには、**jsPsychHtmlKeyboardResponse** プラグインや **jsPsychImageKeyboardResponse** プラグインなどを利用し、色の名前と異なるインクの色で表示された単語に対する反応を測定する必要があります。

刺激系列のJSON形式の例

```
[
  {
    "word": "赤",
    "color": "blue",
    "correct_response": "left"
  },
  {
```



合っているけど意図とと違う。こっちの意図をちゃんと伝えるように指示を出すようにする。

これくらいはっきり言わないと・・・

ストループ課題をjsPsychで作ろうと思います。そのための刺激系列をJSON形式で用意してください。キーは, stimulus, no, stim_type, correct_keyの4つです。stimulusは, '

ストループ課題をjsPsychで作ろうと思います。そのための刺激系列をJSON形式で用意してください。キーは、stimulus, no, stim_type, correct_keyの4つです。stimulusは、'赤'のように、HTML記法で色とフォントサイズと文字を指定します。colorは red,green, yellow,blueの4種類、文字は赤、黄、緑、青の4種類あります。すべての組み合わせで16種類を用意してください。noはただの番号なので、1から16の数字を順番に割り振ってください。stim_typeは、stimulusで指定するcolorと文字が一致していたらcongruent、不一致だったらincongruentを指定してください。correct_keyは、stimulusで指定するcolorがredならd、blueならf、greenならj、yellowならkを指定してください。

ソースに基づいて、jsPsychのストループ課題で使用するための16種類の刺激系列をJSON形式で記述します。

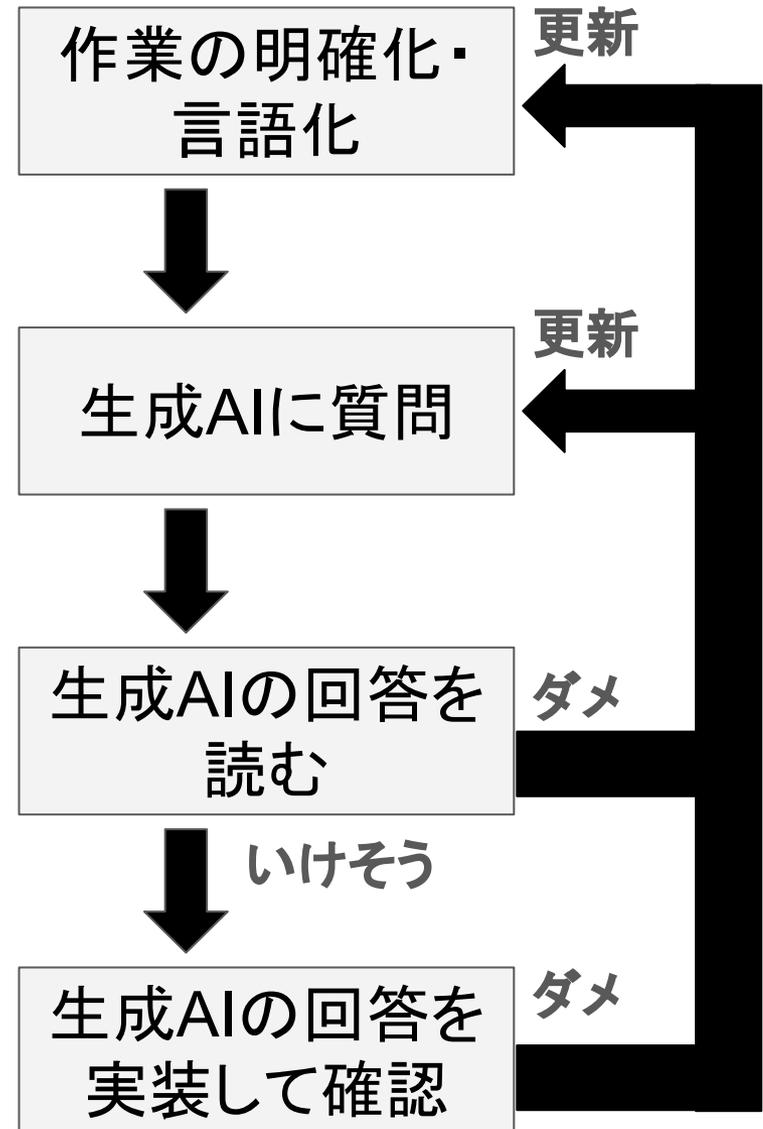


```
[
  {
    "stimulus": "<p style='color:red; font-size:60pt;'>赤</p>",
    "no": 1,
    "stim_type": "congruent",
    "correct_key": "d"
  },
  {
    "stimulus": "<p style='color:red; font-size:60pt;'>黄</p>",
    "no": 2,
    "stim_type": "incongruent",
    "correct_key": "d"
  },
]
```

- 望み通りのJSONが用意できた。さっきのcsvファイルからRでJSON変換が不要に！

生成AIを活用したjsPsychプログラミング

- ハルシネーションを考慮してできるだけソースを指定できる生成AIを使う (NotebookLM)
- 必要なソースを網羅的に集めて入れる(現状では、少し工夫が必要だが、今後はもっと楽になるだろう)
- ソースの工夫で自分の目的にフィットした生成AIと対話しつつ課題を作成できる。



オマケ：研究で使う刺激の生成

- 生成AIは、プログラミング以外にも活用できる。
- 画像刺激の生成
- 動画刺激の生成
- 音声刺激の生成
- 呈示する刺激となる文の生成

→生成AIによって生成した刺激についての論文もでてきているので、ご自身の分野で使えるものがないか確認するとよい。

「生成AIを用いたjsPsychプログラミング」

目次

1. 生成AI時代に必要なプログラミング知識
2. jsPsychの基礎知識
3. NotebookLMを用いたjsPsychプログラミング
- 4. データ収集方法**

データ収集(ローカル)

- jsPsychをつかってローカルPCでデータ収集ができる。
- HTMLを開いて実験を実施(psyinfrを使う場合は, demo_XXX.html)

→簡単だが, データの保存名などの工夫をする(都度名前をつけて保存or乱数でファイル名をつける)

- JATOSを使う

→PCにJATOSをいれて管理をする。ただしJATOSを自分で管理するのが難しいかもしれない。

データ収集(オンライン)

- Mindprobeを使う(ヨーロッパ認知心理学会が無償で提供しているサービス)

<https://mindprobe.eu/>

- レンタルサーバーを借りてJATOSをおいて使う

[国里愛彦\(2022\). チュートリアルワークショップ「心理学研究における研究室インフラの整備」日本心理学会第86回大会 東京\(日本大学\)](#)

→レンタルサーバーを自分で借りるのはちょっと大変かもしれないので, Mindprobeを使う方法を説明します。

psyinfr::set_cbat()でjsPsychファイルを準備して、課題を作成する。

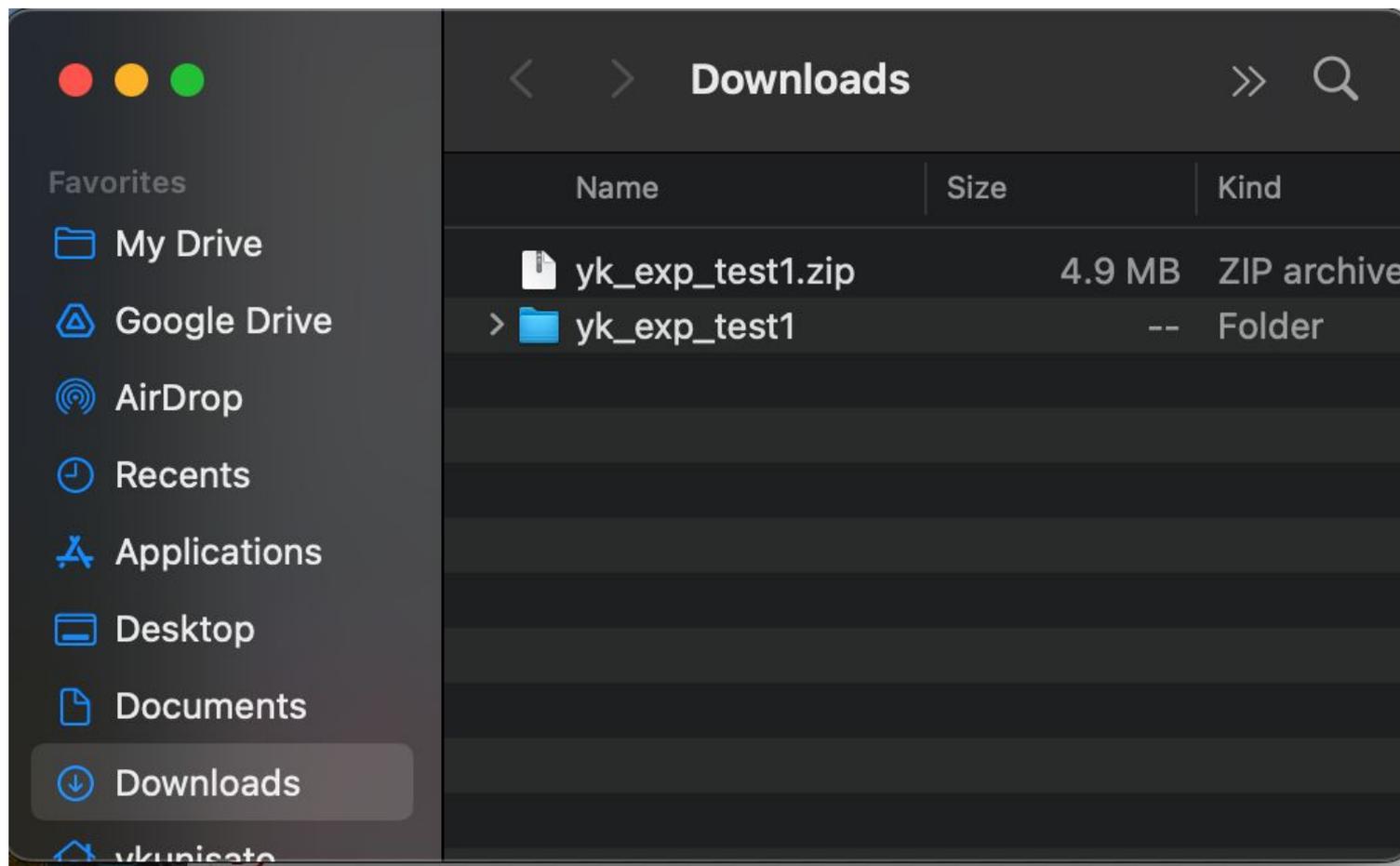
The screenshot shows the RStudio interface. The editor window displays a JavaScript file named `task.js` with the following code:

```
1 /* 課題に関するコードを以下に書く */
2 const welcome = {
3   type: jsPsychHtmlKeyboardResponse,
4   stimulus: "mindprobeのテスト!"
5 };
6
7 /*タイムラインの設定*/
8 const timeline = [fullscreen, welcome];
9
```

The console window shows the execution of the command `psyinfr::set_cbat("yk_exp_test1", "8.0.1", use_rc = FALSE)`. The output indicates that the URL `'https://github.com/jspsych/jsPsych/releases/download/jspsych%408.0.1/jspsych.zip'` was used, and the content type is `'application/octet-stream'` with a length of 4796336 bytes (4.6 MB). The download progress is shown as `downloaded 4.6 MB`.

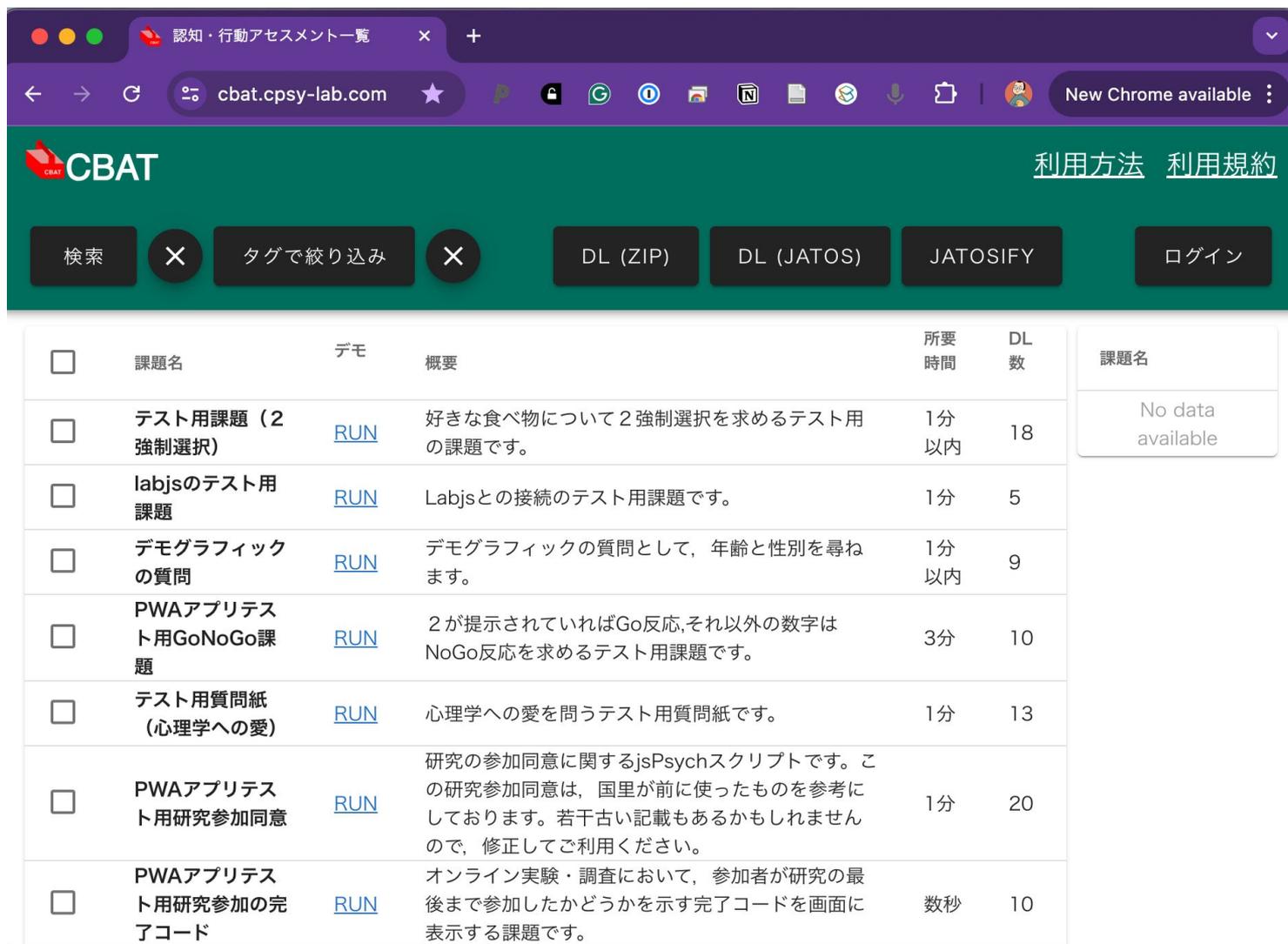
The Environment pane shows the Global Environment, which is currently empty. The Files pane shows the current directory is `~/Downloads`, and a folder named `yk_exp_test1` is visible.

作成したjsPsych課題をフォルダごとzip化する



Macの場合、標準の「圧縮」だとうまくいかないことがあるので、WinZipperを使う。

CBAT(<https://cbat.cpsy-lab.com/>)で「JATOSIFY」をクリックする。



The screenshot shows the CBAT website interface. At the top, there is a navigation bar with the CBAT logo on the left and links for "利用方法" (Usage) and "利用規約" (Terms of Service) on the right. Below the navigation bar is a search bar with the text "検索" (Search) and a search icon. To the right of the search bar are several buttons: "タグで絞り込み" (Filter by tag), "DL (ZIP)", "DL (JATOS)", "JATOSIFY", and "ログイン" (Login). The main content area displays a table of tasks with columns for checkboxes, task names, demo links, descriptions, required time, and download count. A search filter box on the right shows "No data available".

<input type="checkbox"/>	課題名	デモ	概要	所要時間	DL数	課題名
<input type="checkbox"/>	テスト用課題（2強制選択）	RUN	好きな食べ物について2強制選択を求めるテスト用の課題です。	1分以内	18	No data available
<input type="checkbox"/>	labjsのテスト用課題	RUN	Labjsとの接続のテスト用課題です。	1分	5	
<input type="checkbox"/>	デモグラフィックの質問	RUN	デモグラフィックの質問として、年齢と性別を尋ねます。	1分以内	9	
<input type="checkbox"/>	PWAアプリテスト用GoNoGo課題	RUN	2が提示されていればGo反応,それ以外の数字はNoGo反応を求めるテスト用課題です。	3分	10	
<input type="checkbox"/>	テスト用質問紙（心理学への愛）	RUN	心理学への愛を問うテスト用質問紙です。	1分	13	
<input type="checkbox"/>	PWAアプリテスト用研究参加同意	RUN	研究の参加同意に関するjsPsychスクリプトです。この研究参加同意は、国里が前に使ったものを参考にしております。若干古い記載もあるかもしれませんので、修正してご利用ください。	1分	20	
<input type="checkbox"/>	PWAアプリテスト用研究参加の完了コード	RUN	オンライン実験・調査において、参加者が研究の最後まで参加したかどうかを示す完了コードを画面に表示する課題です。	数秒	10	

タイトル(フォルダ名), JATOSのバージョン(3)を入力して, ソースコードで先程のzipファイルを選ぶ

JATOS 設定

タイトル
yk_exp_test1

JATOS のバージョン
3

説明

コメント

ソースコード
 yk_exp_test1.zip (4.9 MB)

1 files (4.9 MB in total)

OK キャンセル

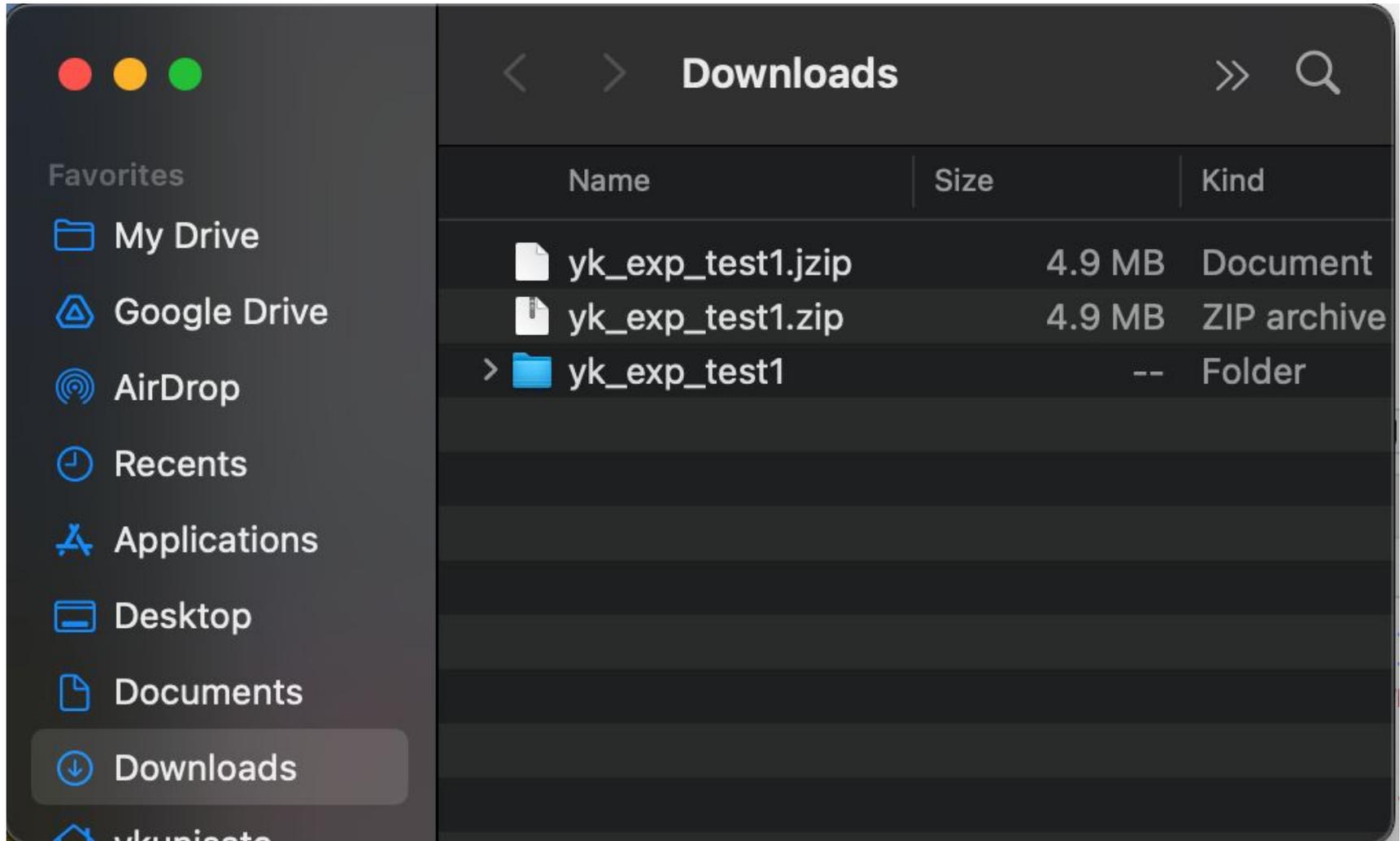
ダウンロードを待つ

The screenshot shows a web browser window with the URL `cbat.cpsy-lab.com`. The page header includes the CBAT logo and navigation links for "利用方法" and "利用規約". Below the header is a search bar with a "検索" button and a "タグで絞り込み" button. There are also buttons for "DL (ZIP)", "DL (JATOS)", "JATOSIFY", and "ログイン".

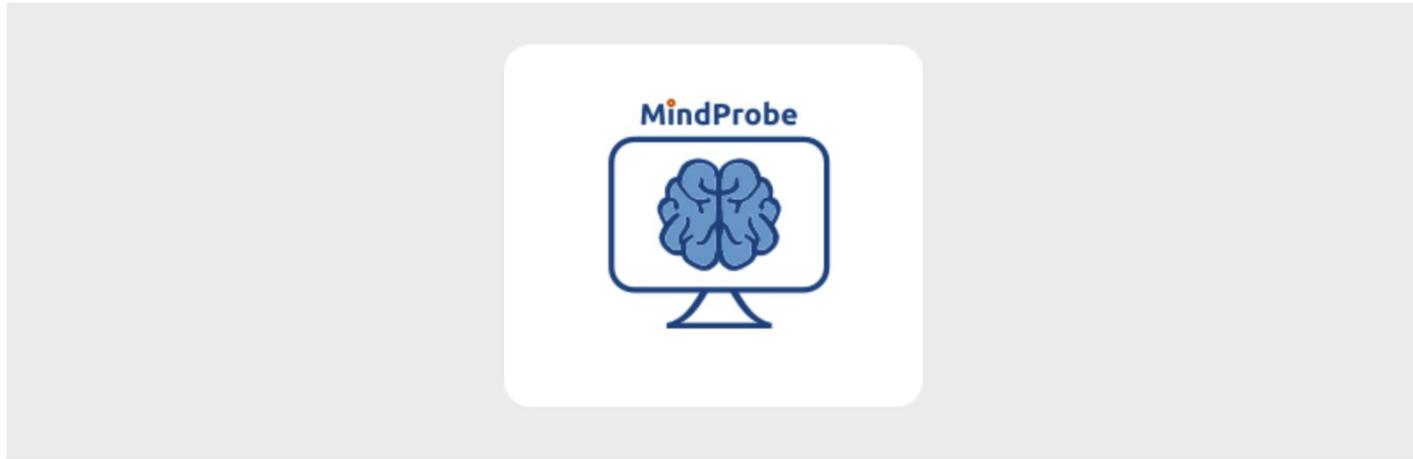
The main content is a table of test topics. The first row is selected, and a white dialog box with the text "ダウンロード準備中です。しばらくお待ちください。" is overlaid on the table.

<input type="checkbox"/>	課題名	デモ	概要	所要時間	DL数	課題名
<input checked="" type="checkbox"/>	テスト用課題（2強制選択）	RUN	好きな食べ物について2強制選択を求めるテスト用の課題です。	1分以内	17	テスト用課題（2強制選択）
<input type="checkbox"/>	labjsのテスト用課題	RUN	Labjsとの接続のテスト用課題です。	1分	5	
<input type="checkbox"/>	デモグラフィックの質問				9	
<input type="checkbox"/>	PWAアプリテスト用GoNoGo課題	RUN	2が提示されていればGo反応,それ以外の数字はNoGo反応を求めるテスト用課題です。	3分	10	
<input type="checkbox"/>	テスト用質問紙（心理学への愛）	RUN	心理学への愛を問うテスト用質問紙です。	1分	13	
<input type="checkbox"/>	PWAアプリテスト用研究参加同意	RUN	研究の参加同意に関するjsPsychスクリプトです。この研究参加同意は、国里が前に使ったものを参考にしております。若干古い記載もあるかもしれませんので、修正してご利用ください。	1分	20	
<input type="checkbox"/>	PWAアプリテスト用研究参加の完了コード	RUN	オンライン実験・調査において、参加者が研究の最後まで参加したかどうかを示す完了コードを画面に表示する課題です。	数秒	10	
<input type="checkbox"/>	PWAアプリテスト用気分の質問紙	RUN	テスト用の気分を測定する質問紙です。	1分	8	
<input type="checkbox"/>	PWAアプリテスト		スマートフォン用に調整したテスト用の2腕バンディット課題で			

JATOS用のjzipファイルができる



Mindprobe(<https://mindprobe.eu/>) へ移動する。ログインはORCID(オーキッド, 研究者識別子を提供するサービス)が便利だけど, メール認証も可能



MindProbe is a server to host online experiments. This server runs the open-source software [JATOS](#), which is compatible with several free software packages for developing online experiments, including [OpenSesame/OSWeb](#), [jsPsych](#), and [Lab.JS](#). MindProbe is sponsored by [ESCOP](#), with [Journal of Cognition](#) as their official journal. MindProbe is free.

[Getting an account](#)

[Log in](#)

[Support](#)

[Privacy and ethics](#)

[Terms of use](#)

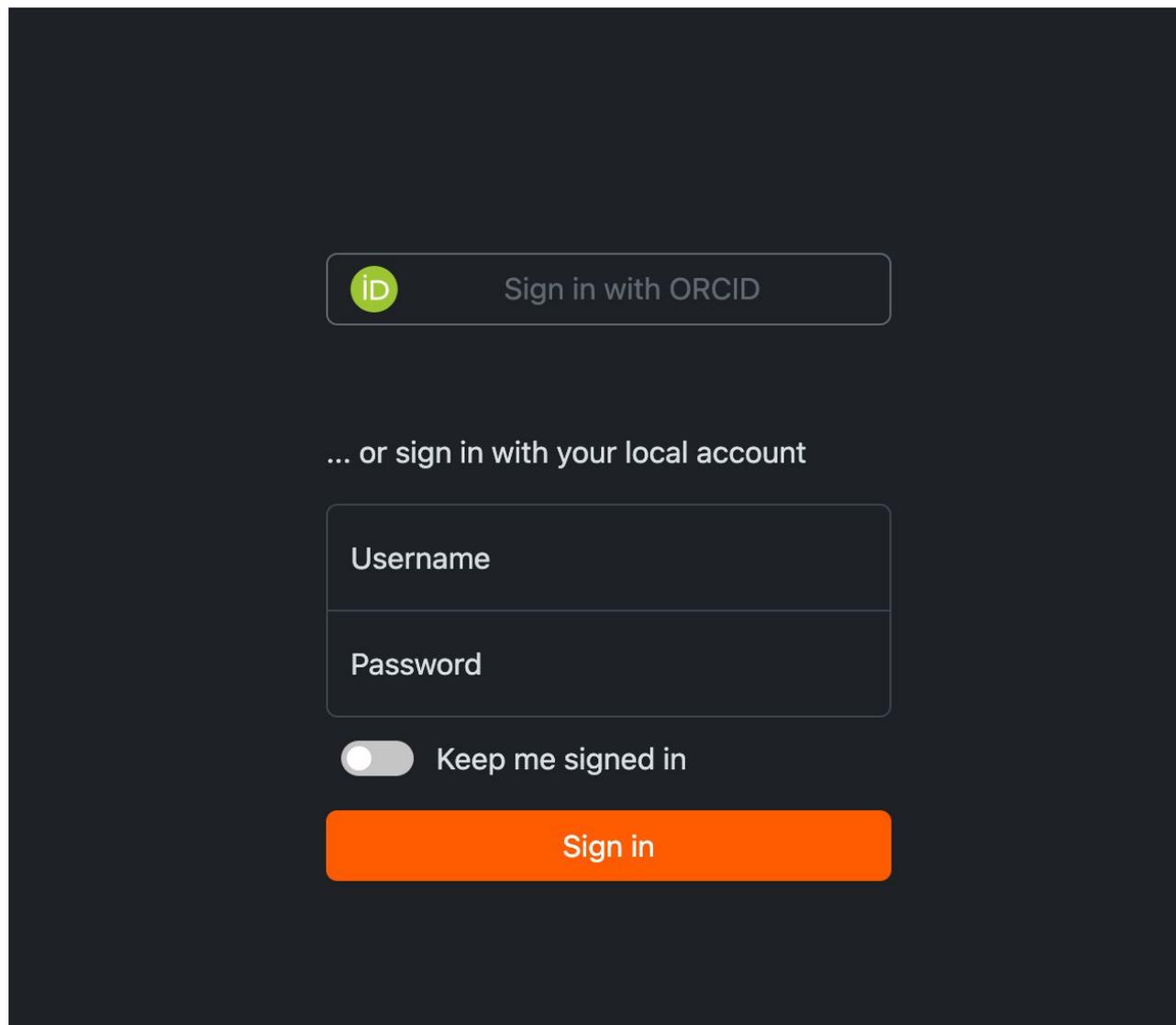


Getting an account

By logging into MindProbe, you agree with our [privacy and ethics policy](#) and [terms of use](#).

If you have an [ORCID](#) ID, you can use this to [log in directly](#).

ORCIDがあれば、それを使って簡単にログインできる。



The image shows a dark-themed login interface. At the top, there is a button with a green circular icon containing the letters 'id' and the text 'Sign in with ORCID'. Below this, the text '... or sign in with your local account' is displayed. Underneath, there are two stacked input fields: the top one is labeled 'Username' and the bottom one is labeled 'Password'. Below the input fields is a toggle switch that is currently turned off, followed by the text 'Keep me signed in'. At the bottom of the form is a large, bright orange button with the text 'Sign in' in white.

ORCIDアカウントで認 証してログインする

Authorize access

You are currently signed in as:



Yoshihiko Kunisato

<https://orcid.org/0000-0002-5830-7182>

[Sign out](#)

JATOS

[Show details](#)

This organization has asked for the following access to your ORCID record:

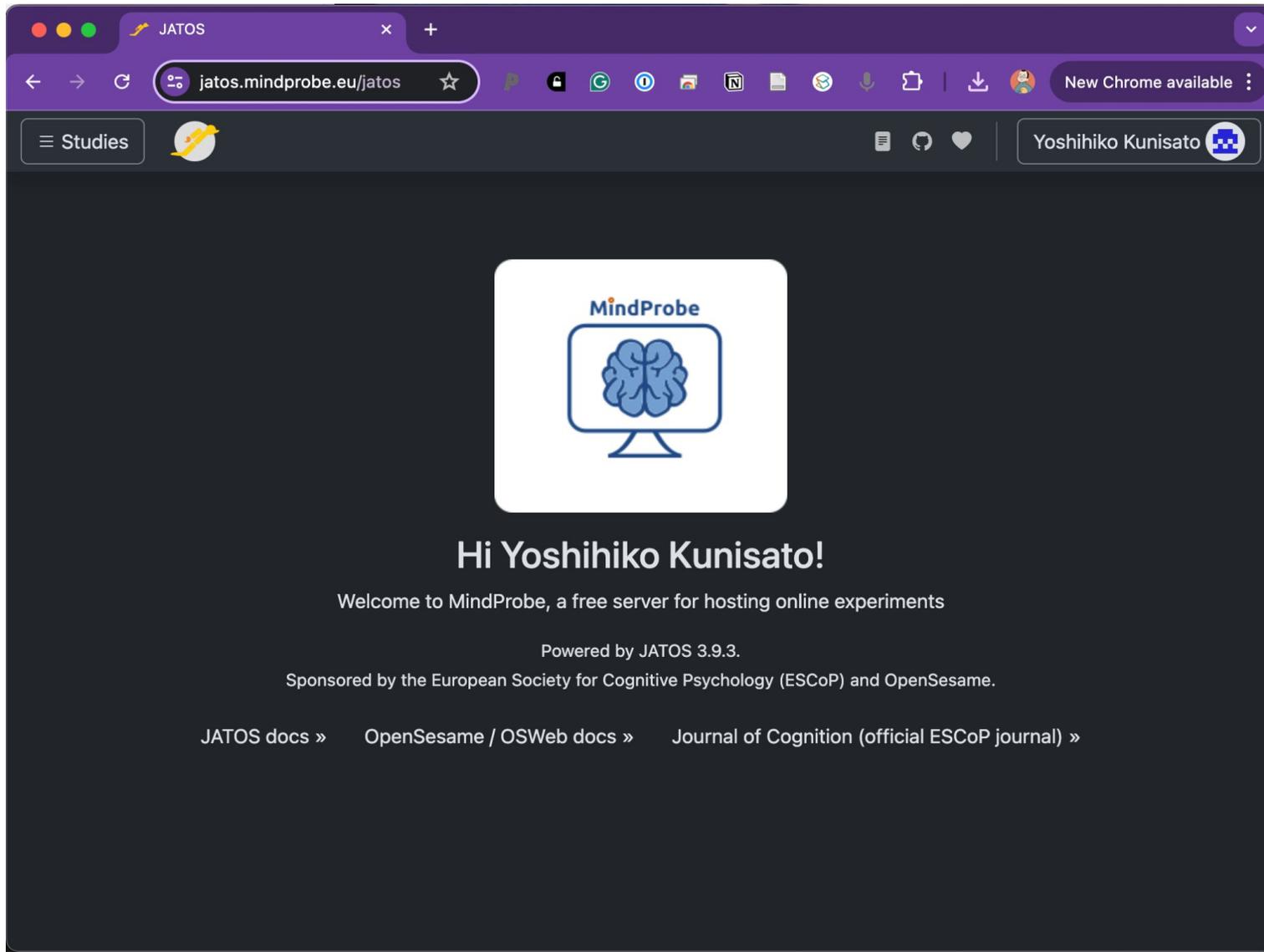
 Get your ORCID iD

If authorized, this organization will have access to your ORCID record, as outlined above and described in further detail in [ORCID's privacy policy](#).

You can manage access permissions for this and other Trusted Organizations from within your list of [trusted parties](#).

Authorize access

Mindprobeの自分のページ



The screenshot shows a web browser window with a purple theme. The address bar displays "jatos.mindprobe.eu/jatos". The page content includes a "MindProbe" logo featuring a brain on a monitor, a personalized greeting "Hi Yoshihiko Kunisato!", and a welcome message. It also mentions the software version "JATOS 3.9.3" and sponsors "ESCoP" and "OpenSesame". At the bottom, there are links to "JATOS docs", "OpenSesame / OSWeb docs", and "Journal of Cognition".

JATOS

jatos.mindprobe.eu/jatos

Studies

Yoshihiko Kunisato

MindProbe

Hi Yoshihiko Kunisato!

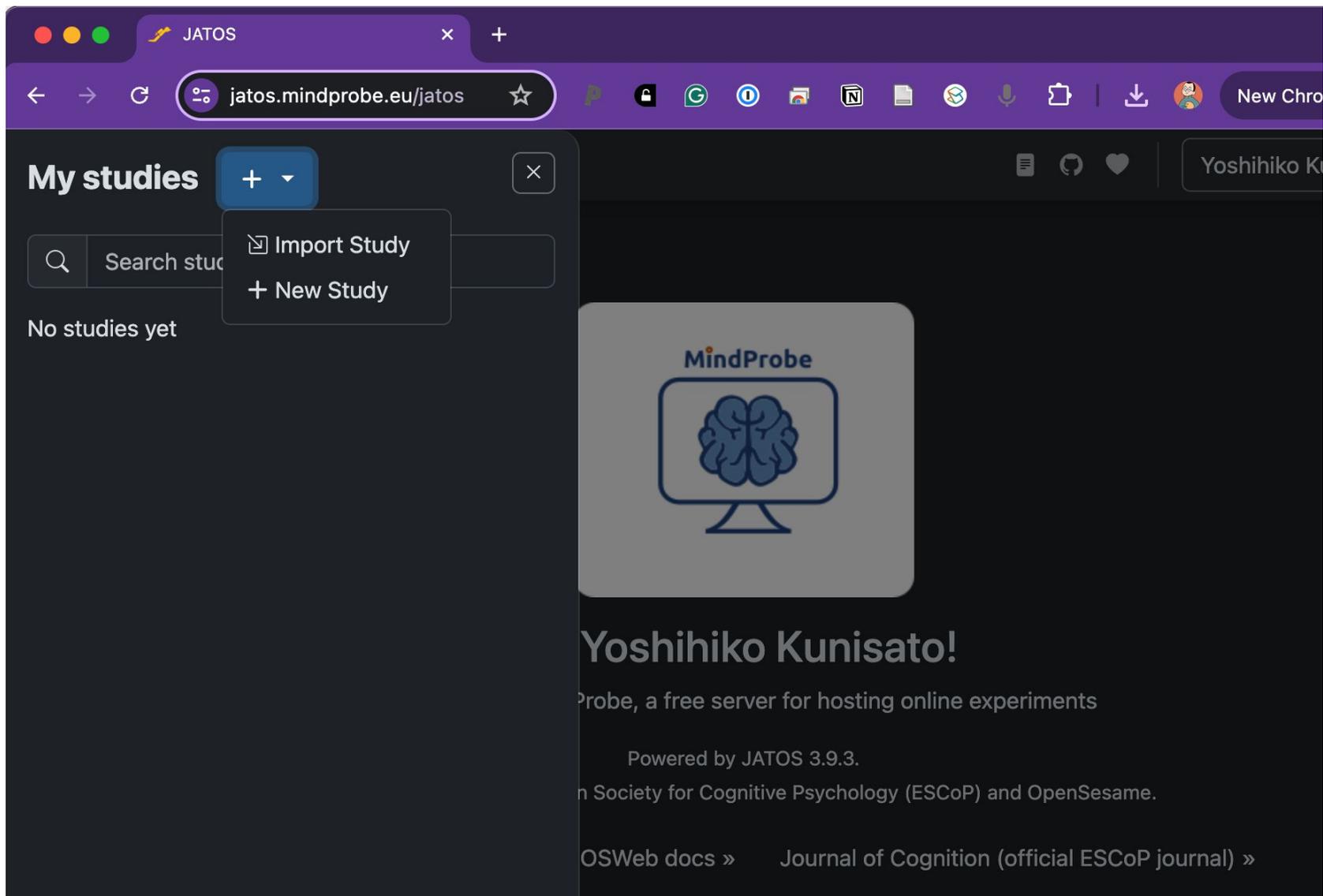
Welcome to MindProbe, a free server for hosting online experiments

Powered by JATOS 3.9.3.

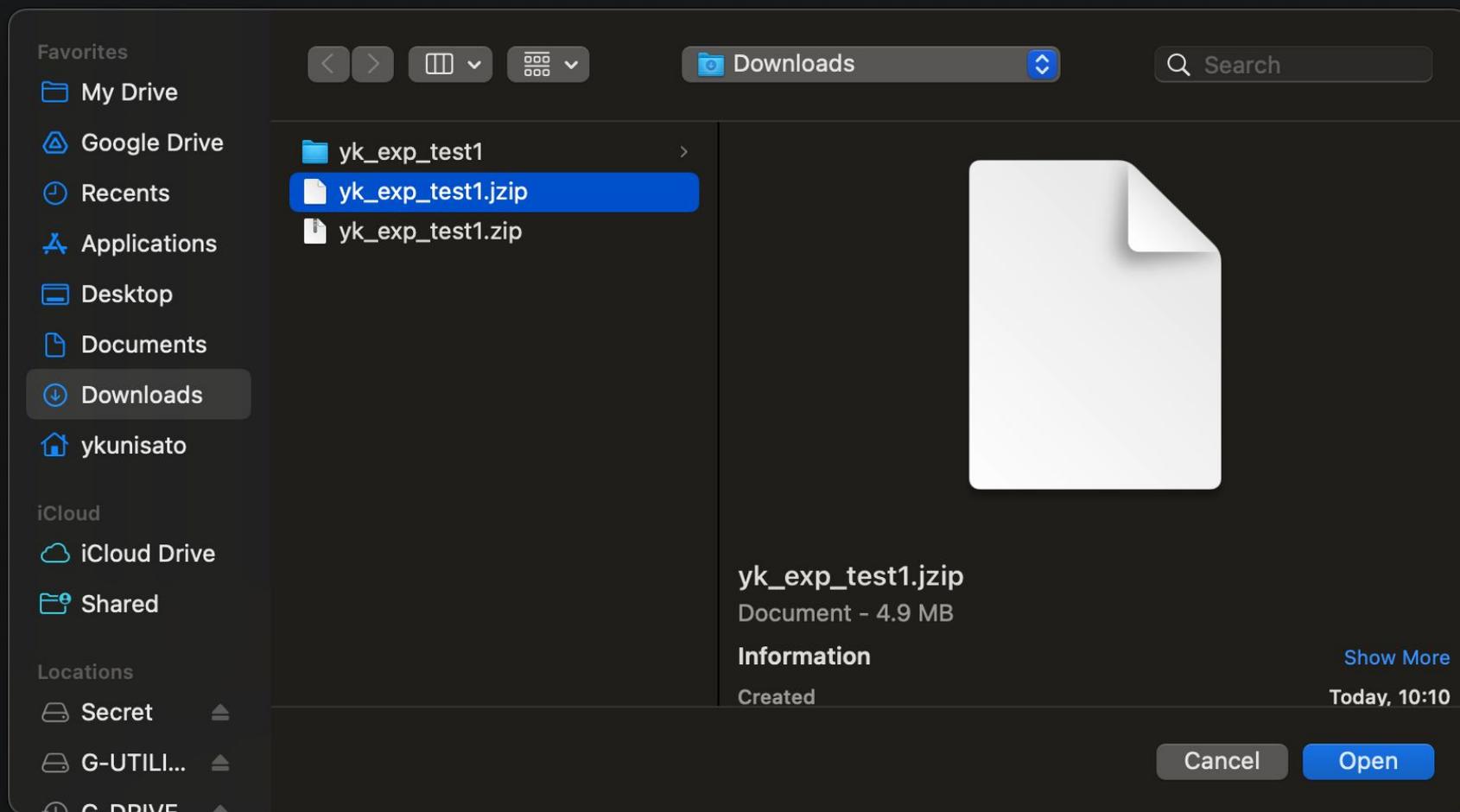
Sponsored by the European Society for Cognitive Psychology (ESCoP) and OpenSesame.

[JATOS docs »](#) [OpenSesame / OSWeb docs »](#) [Journal of Cognition \(official ESCoP journal\) »](#)

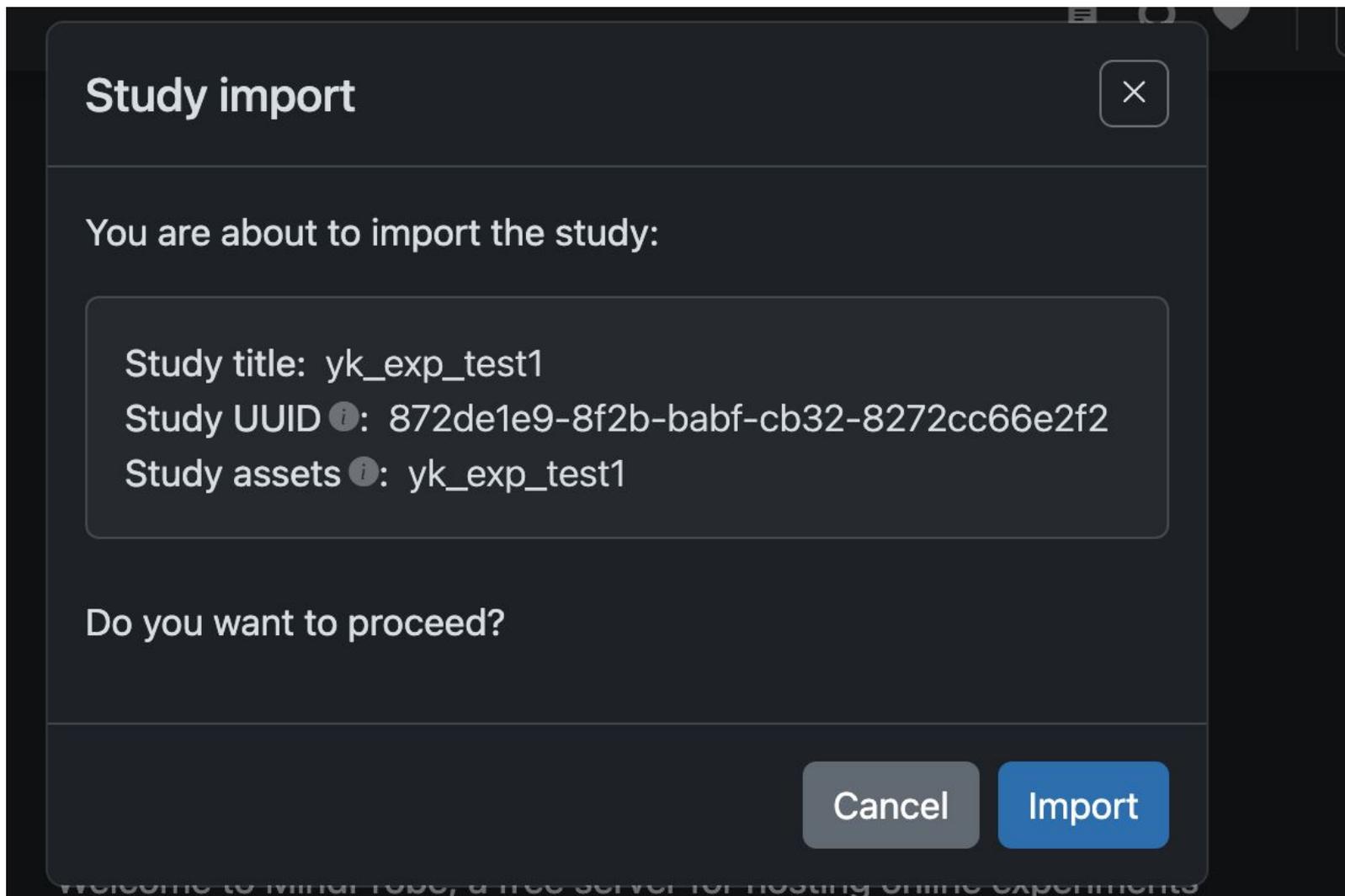
＋の▼をクリックして、import Studyをクリックする。



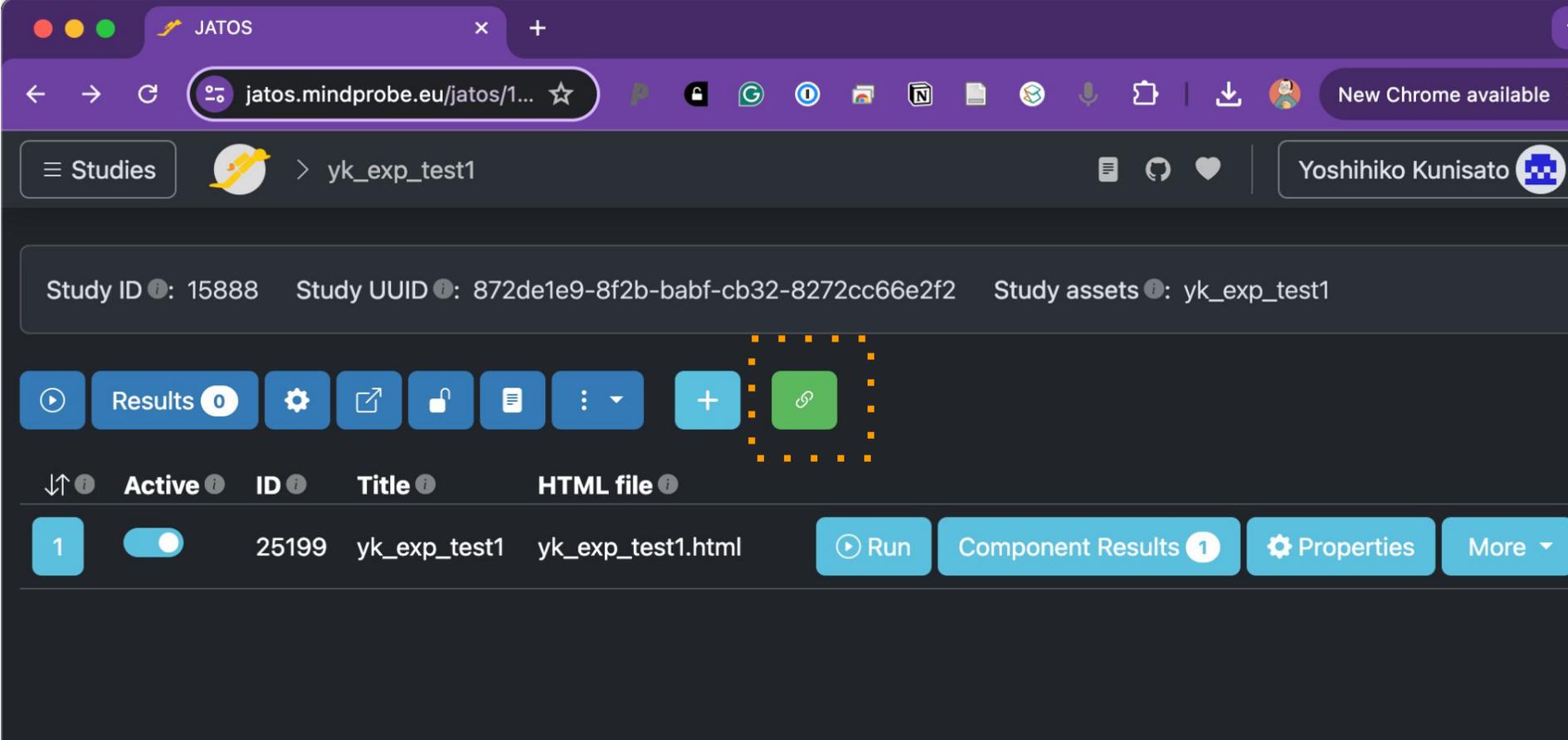
作成したjzipファイルを選択する



jzipファイルをインポートする



すぐに使えるようになっていています。RUNをクリックすればすぐに試すことができます。
オレンジ点線で囲ったアイコンをクリックします。



The screenshot shows the JATOS web interface in a browser. The browser tab is titled "JATOS" and the address bar shows "jatos.mindprobe.eu/jatos/1...". The page displays study information: "Study ID: 15888", "Study UUID: 872de1e9-8f2b-babf-cb32-8272cc66e2f2", and "Study assets: yk_exp_test1". Below this is a toolbar with several icons. A green icon with a link symbol is highlighted with an orange dashed box. Below the toolbar is a table with columns: "Active", "ID", "Title", and "HTML file". The table contains one row with the following data: "1", "25199", "yk_exp_test1", and "yk_exp_test1.html". To the right of the table are buttons for "Run", "Component Results (1)", "Properties", and "More".

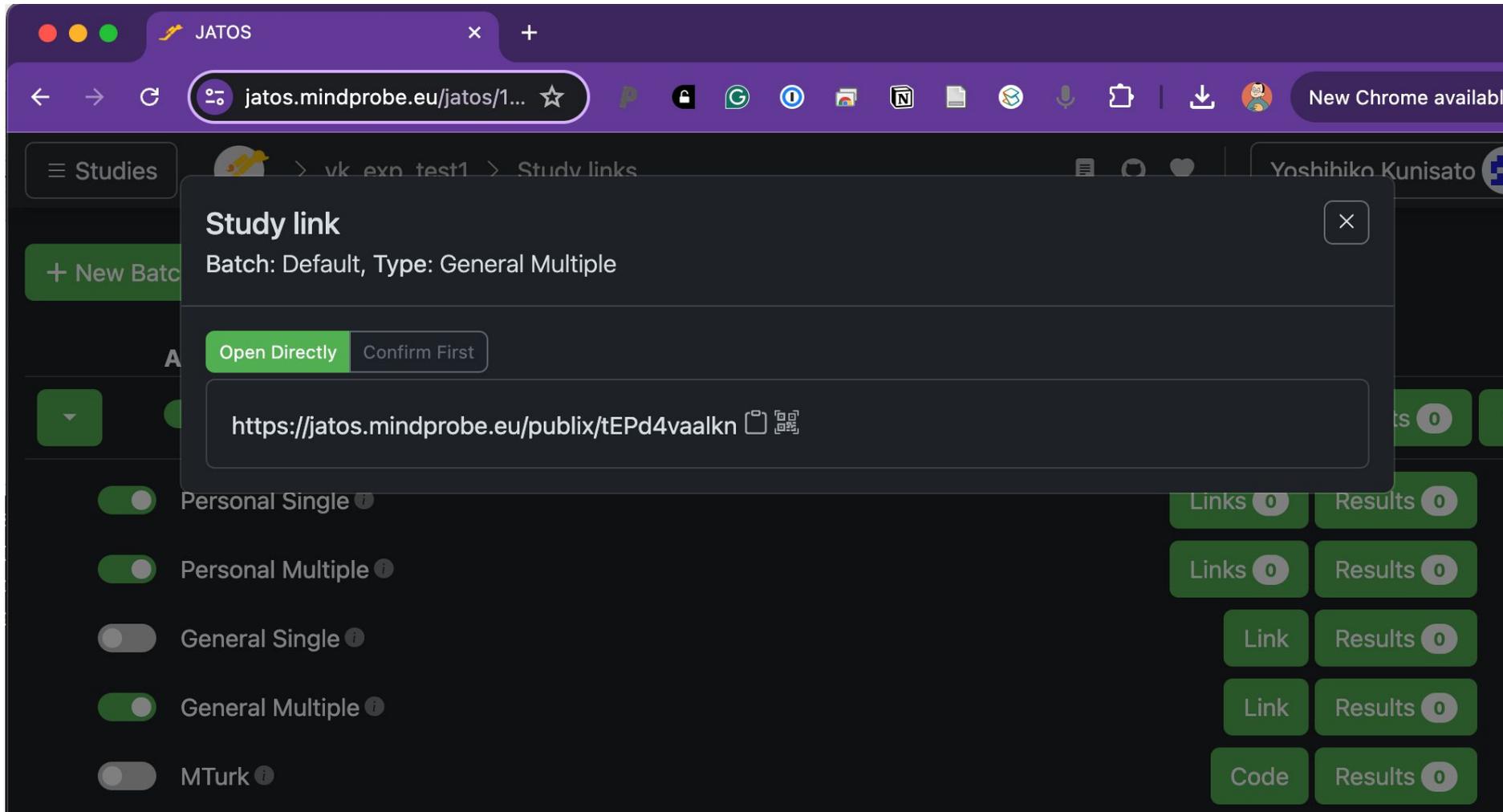
Active	ID	Title	HTML file
<input checked="" type="checkbox"/>	25199	yk_exp_test1	yk_exp_test1.html

オンライン実験で使えるリンクを発行します。1回だけアクセスを許可する場合はGeneral Single, 複数回を可能にする場合は General Multipleをチェックします。

The screenshot shows the JATOS web interface. The browser address bar displays `jatos.mindprobe.eu/jatos/1...`. The page title is "Study links" under the batch "yk_exp_test1". The user is identified as Yoshihiko Kunisato. The interface features two main buttons: "+ New Batch" and "Get Study Links".

Active	ID	Batch title	Settings	Links	Results
<input checked="" type="checkbox"/>	18271	Default	<input checked="" type="checkbox"/> Personal Single <input checked="" type="checkbox"/> Personal Multiple <input type="checkbox"/> General Single <input type="checkbox"/> General Multiple <input type="checkbox"/> MTurk	0	0

リンクを発行して、そのリンクを参加者に共有すれば、オンライン実験ができます。



The screenshot shows a web browser window with the URL `jatos.mindprobe.eu/jatos/1...`. The page displays a sidebar with a 'Studies' menu and a main area with a 'Study link' dialog box. The dialog box contains the following information:

- Study link** (with a close button)
- Batch: Default, Type: General Multiple
- Buttons: **Open Directly** (highlighted in green) and **Confirm First**
- URL: `https://jatos.mindprobe.eu/publix/tEPd4vaalkn` (with a QR code icon)

The background interface includes a list of study types with toggle switches and a table of results:

Study Type	Link	Results
Personal Single	Links 0	Results 0
Personal Multiple	Links 0	Results 0
General Single	Link	Results 0
General Multiple	Link	Results 0
MTurk	Code	Results 0

それでは課題をはじめます。

以下の「開始」を押すと、全画面になって課題が始まります。

開始

作業フローの整理

- (1) psyinfrでjsPsychの準備
- (2) 作成する課題・質問紙などの整理(NotebookLMへ)
- (3) jsPsychプログラミングwith NotebookLM
- (4) CBATでJATOSIFY (jzip化)
- (5) Mindprob or 自分で用意したJATOSでオンライン調査
・実験の実施

参考資料

- [jsPsychの本家のウェブサイト](#)
- [国里のウェブサイト](#)
- [西山さんのZennの記事](#)
- [黒木先生のQiitaの記事](#)
- 黄竹・岸山・野口(2023)『jsPsychによるオンライン音声実験レシピ』教養検定会議[書籍,1650円]