Experimental Linguistics with IBEX

A Walkthrough

2021-01-22

Presentation Outline



Introduction to **IBEX**

- Basic ideas
- Navigating the platform



Scripting an experiment

- Overview of critical components
- Code walkthrough



Data Analysis

- Understanding the output format
- Importing into R

Introduction to IBEX

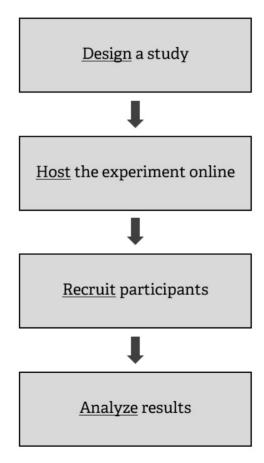
What is IBEX?

- Stands for (I)nternet-(B)ased (EX)periments
- DOES:
 - Host experiments with webpage links
 - Log user interactions
 - Store data on a secure server

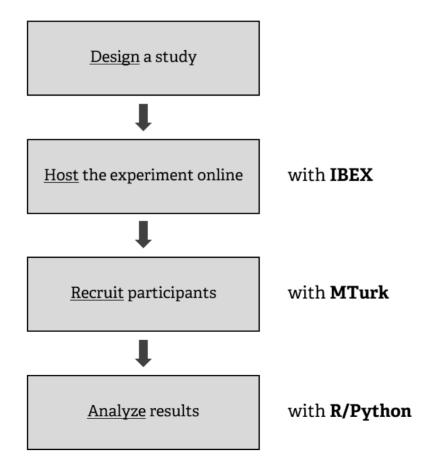
DOES NOT:

- Recruit participants (see Amazon Mechanical Turk)
- Provide an analysis of the results (see R, Python)

Where does IBEX fit?



Where does IBEX fit?



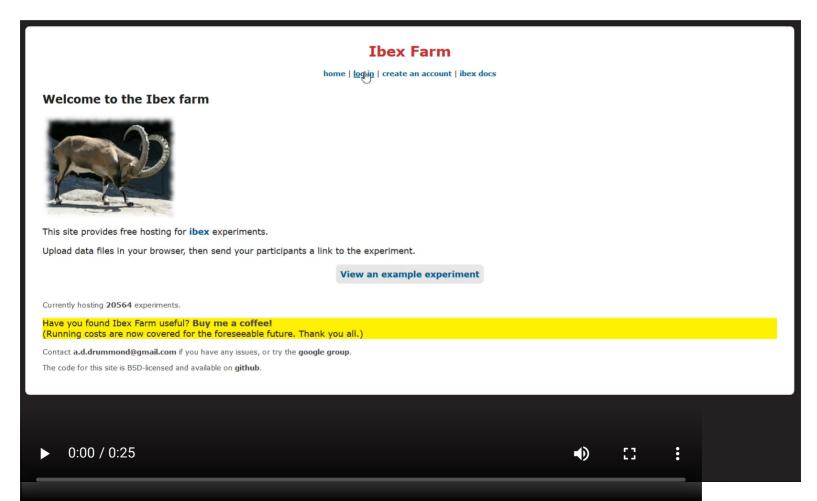
Navigating IBEX

- Go to https://spellout.net/ibexfarm
- Click create an account (or log in)
- Click manage my experiments
- Click Create a new experiment
- Give it a unique name like "workshop_example"
- Done!

Experiments

- workshop_example (ibex 0.3.9) (delete | rename)
- » Create a new experiment

Navigating IBEX



Experiment file structure

chunk_includes

Stand-alone files go here

css_includes

Style specifications go here

data_includes

• Experiment scripts go here

js_includes

Modules ("controllers") go here

results & server_state

Automatically generated/updated

Update from git repo» (help) chunk includes (upload a file to this directory | refresh) example intro.html (delete | rename | upload new version | edit) test1.mp3 (delete | rename | upload new version | edit) test2.mp3 (delete | rename | upload new version | edit) css includes (upload a file to this directory | refresh) DashedSentence.css FlashSentence.css Form.css global main.css Message.css Question.css Scale.css Separator.css data_includes (upload a file to this directory | refresh) example_data.js (delete | rename | upload new version | edit) js_includes (upload a file to this directory | refresh) AcceptabilityJudgment.js DashedAcceptabilityJudgment.js DashedSentence.js FlashSentence.is Form.js Message.js Ouestion.is Scale.js Separator.js VBox.js results (upload a file to this directory | refresh) This directory is not present server_state (upload a file to this directory | refresh) This directory is not present

Experiment 'workshop example' (ibex 0.3.9)

Experiment file structure

chunk_includes

Stand-alone files go here

css_includes

Style specifications go here

data_includes

• Experiment scripts go here

js_includes

Modules ("controllers") go here

results & server_state

Automatically generated/updated

Experiment 'workshop example' (ibex 0.3.9) Update from git repo» (help) chunk includes (upload a file to this directory | refresh) example intro.html (delete | rename | upload new version | edit) test1.mp3 (delete | rename | upload new version | edit) test2.mp3 (delete | rename | upload new version | edit) css includes (upload a file to this directory | refresh) DashedSentence.css FlashSentence.css Form.css global main.css Message.css Question.css Scale.css Separator.css data_includes (upload a file to this directory | refresh) example_data.js (delete | rename | upload new version | edit) js_includes (upload a file to this directory | refresh) AcceptabilityJudgment.js DashedAcceptabilityJudgment.js DashedSentence.js FlashSentence.is Form.js Message.js Ouestion.is Scale.js Separator. is VBox.js results (upload a file to this directory | refresh) This directory is not present server_state (upload a file to this directory | refresh)

This directory is not present

Scripting an experiment

The script

Only need to modify **one file**: *example_data.js* (can also be renamed later)

At creation, the default file looks like this:

```
var shuffleSequence = seq("intro", sepWith("sep", seq("practice", rshuffle("s1", "s2"))),
sepWith("sep", rshuffle("g1", "g2")));
var practiceItemTypes = ["practice"];
var defaults = [
    "Separator", {
        transfer: 1000,
       normalMessage: "Please wait for the next sentence.",
       errorMessage: "Wrong. Please wait for the next sentence."
    "DashedSentence", {
       mode: "self-paced reading"
    },
    "AcceptabilityJudgment", {
        as: ["1", "2", "3", "4", "5", "6", "7"],
       presentAsScale: true,
       instructions: "Use number keys or click boxes to answer.",
       leftComment: "(Bad)", rightComment: "(Good)"
```

The above script creates this self-paced reading experiment.

Writing your own script

The *example_data.js* script works, but is not very friendly.

We'll use **our own** - **template.js** - to demonstrate how the script works.

```
// Options and Other Variables //
//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode
/// Show a progress bar at the top? (true/false)
var showProgressBar = false
/// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
    "AcceptabilityJudgment", {
      as: ["1", "2", "3", "4", "5", "6", "7"],
      presentAsScale: true,
      instructions: "Use number keys or click boxes to answer.",
      leftComment: "(Bad)",
      rightComment: "(Good)"
```

The above script creates this acceptability rating experiment.

```
var randomCode = Math.random().toString(36).substr(2.9)
var completionCode = String("NPT." + randomCode):
 var completionNessage = "Thank you for your participation. The results were successfully transmitted, your participation code is: " + completionCode
var defaults = {"acceptability]udgment", {
    ss: ["1", "2", "3", "4", "5", "6", "7"],
    precentscale: true,
    instructions: "use number keys or click boxes to answer.",
    letComment: "(800",
    rightComment: "(6005)"
]];
            sepwith("sep", rshuffle(startswith("npi"),startswith("filler"))1
  ["setrounter", " setrounter ", { }].
                    oiv",
["p", "Before starting the questionnaire, let's do a couple of examples to get a feel for the task."]
  ]], ["Gractice", "AcceptabilityJudgment", (s: "The car drove like a drean.")], ["Gractice", Message, ( transfer: "keypress", html: ["div", "hum wax hhat) Many decole rate that sentence pretty good
                    "GIV", "How was that? Many people rate that sentence pretty good."]
  [['npi.gram',1], "AcceptabilityJudgment", (s: "No duck that the goose chased has ever returned to our pond.")], [['npi.llus',1], "AcceptabilityJudgment", (s: "The duck that no goose chased has ever returned to our pond.")], ['['npi.ungram',1], "AcceptabilityJudgment", (s: "The duck that the goose chased has ever returned to our pond.")],
    ["filler-ATTENTIONCHECK-01", "AcceptabilityJudgment", (s: "Please select 4 for this sentence; do not rate it like other sentences.")]
                                                                                                                                                                       Discard changes Save changes Save and close
```

Editing the script

When you open the .js file in the data_includes section of your experiment on Ibex, it will open up a text editor.

```
var randomCode = Math.random().toString(36).substr(2.9)
var completionCode = String("NPT." + randomCode):
   var completionNessage w "Thank you for your participation. The results were successfully transmitted, your participation code is: " + completionCode
var defaults = ["Acceptability)udgment", {
    ss: ["1", "2", "3", "4", "5", "7"],
    presentscade: row,
    presentscade: row,
    leftcoment: "[dad]",
    leftcoment: "[dad]",
    rightcoment: "[dad]",
                              sepwith("sep", rshuffle(startswith("npi"),startswith("filler"))1
 ["setcounter", " setcounter ", ( )].
        ["practice", "AcceptabilityJudgment", {s: "The car drove like a dream.")],
["practice", Message, {
    transfer: "keypress",
    html: ["div",
                                                  "GIV", "How was that? Many people rate that sentence pretty good."
   ]],
['prectice', "AcceptabilityJudgment", (s: "The cat ever has eaten cheese the.")],
['prectice', Message, ['
'prectice', Message, Message, ['
'prectice', Message, Message, Message, Message, Message, Message, Message, Mes
       [("npi.gram",1), "AcceptabilityJudgment", (s: "No duck that the goose chased has ever returned to our pond.")]
[("npi.lluo",1), "AcceptabilityJudgment", (s: 'The duck that no goose chased has ever returned to our pond.")
[("npi.ungma",1), "AcceptabilityJudgment", (s: 'The duck that the goose chased has ever returned to our pond.")
                                                                                                                                                                                                                                                                                                                                                                                                                                                           Discard changes Save changes Save and close
```

Editing the script

When you open the .js file in the data_includes section of your experiment on Ibex, it will open up a text editor.

Options for editing the file:

- Make changes directly on IBEX
- Download the file and open with an editor that supports JavaScript syntax checking (e.g., Atom)

```
Settings
                                                                                         Sequence
["setcounter", " SetCounter ", ( )]
                                                                                                   Body
        "Olv",
["o", "How was that? Many people rate that sentence pretty good."
```

Discard changes Save changes Save and close

Parts of the Script

Settings:

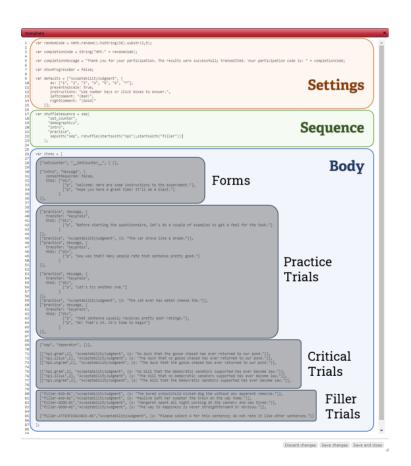
 Sets various options for the experiment

Sequence (shuffleSequence):

 Specifies the ordering of the different parts of the experiment

Body (items):

 Includes the actual material that will be shown to the participants



Body

Forms

 Introduction page, consent form, directions, language background, demographic information, etc.

Trials

- All stimuli for the experiment
- Practice, Critical, Filler
- Can be accompanied by messages, questions, etc.

Walkthrough of the components

```
var randomCode = Math.random().toString(36).substr(2,9);
 var completionCode = String("NPI-" + randomCode);
 var completionMessage = "Thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
 var showProgressRar = false:
Settings
 var shuffleSequence = seq(
          "set counter
         "demographics",
"intro",
                                                                                                                                                                                   Sequence
           sepwith("sep", rshuffle(startswith("npi"),startswith("filler"))]
                                                                                                                                                                                                      Body
   ["intro", "Message", {
       Forms
     ["practice", Message,
     "practice", "AcceptabilityJudgment", (s: "The car drove like a dream.")],
     ["practice", Message, {
transfer: "keypress",
html: ["div",
                   ["p", "How was that? Many people rate that sentence pretty good."]
                                                                                                                                                                       Practice
    ["practice", Message, {
                                                                                                                                                                       Trials
       transfer: "keypress",
html: ["div",
["p", "Let's try another one."]
    ["practice", "AcceptabilityJudgment", (s: "The cat ever has eaten cheese the.")],
["practice", Message, {
    transfer: "keypress",
       html: ["div", "That sentence usually receives pretty poor ratings."], ["p", "ok! That's it. It's time to begin"]
    ["sep", "Separator", {}],
    [["mpi.gram",1], "AcceptabilityJudgment", (s: "No duck that the goose chased has ever returned to our pond.")],
[["mpi.lllus",1], "AcceptabilityJudgment", (s: "The duck that no goose chased has ever returned to our pond.")],
[["mpi.ungram",1], "AcceptabilityJudgment", (s: "The duck that the goose chased has ever returned to our pond.")],
                                                                                                                                                                                       Critical
                                                                                                                                                                                        Trials
    [["mpi.gram",2], "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law.")],
[["mpi.lllus",2], "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law.")],
[["mpi.ungram",2], "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law.")],
    ['Siler-BLO.87','Acceptability).ognest', is: "The bores consciouls sized day the without any apparent remorps.

"Yeiler-BLO.87', "Acceptability).ognest', is: "Busine let the researce the train on the way home "],

"filer-BLO.87', "Acceptability.ougnest', is: "margaret sport all night working at the camery and was tired."]],

"filer-BLO.87', "Acceptability.ougnest', is: "her way to happoints is inever traightformed or actions."]],
                                                                                                                                                                                                   Filler
                                                                                                                                                                                                    Trials
    ["filler-ATTENTIONCHECK-01","AcceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."
```

Discard changes Save changes Save and close

But first, the experiment design

Study: We are interested in how people recover from **garden-path sentences**.

"While Anna dressed the kitten paid attention."

```
... *[_{VP} dressed the kitten], ...
```

... $\sqrt{[v_P]}$ dressed], the kitten ...

Hypothesis: Verbs that are frequently **transitive** make recovery more difficult, compared to verbs that are frequently **intransitive**.

(transitive) - "While Anna *trained* the kitten paid attention."

(intransitive) - "While Anna *dressed* the kitten paid attention."

Prediction: Lower **acceptability ratings** in the transitive condition (*gp.trans*) than in the intransitive condition (*gp.intrans*).

Note: this is a within-participant design!

Trial Syntax

For each trial in our acceptability judgment experiment, we write this code:

```
[["Trial name", Set #], "Trial Type", {s: "Sentence"}]
```

This is a list (array) of three elements (clearer with spacing):

```
[

["Trial name", Set # ],

"Trial Type",

{s: "Sentence"}
]
```

```
[ ["Trial name", Set # ],
"Trial Type",
{s: "Sentence"} ]
```

We have one big bracket which contains **three elements**:

- 1. Another list, consisting of the *name of the trial* and *set number*
- 2. A string specifying the *type of the trial* (called **Controllers**)
- 3. A curly bracket (braces), which has "s" and the sentence separated by a colon

Example 1:

Example 1:

Example 1:

Example 1:

Example 1:

Example 1:

For the **gp.trans condition**, the **first stimuli** in our **acceptability judgment experiment** is the **sentence** "While Anna trained the kitten paid attention".

This can be put into a single line:

```
[["gp.trans",1], "AcceptabilityJudgment", {s: "While Anna trained the k
```

Example 2:

For the **gp.intrans condition**, the **first stimuli** in our **acceptability judgment experiment** is the **sentence** "While Anna dressed the kitten paid attention".

This can be put into a single line:

```
[["gp.intrans",1], "AcceptabilityJudgment", {s: "While Anna dressed the
```

Practice

Item set #2:

```
(transitive) - "Since Dave improved the department was satisfied." (intransitive) - "Since Dave worried the counselor devised a plan."
```

<code here>

Hover for answer

Putting together the stimuli

Wrap in var items = [] and separate by a comma:

```
var item = [

//// Set #1
  [["gp.trans",1], "AcceptabilityJudgment", {s: "While Anna trained the kitten paid attention."}],
  [["gp.intrans",1], "AcceptabilityJudgment", {s: "While Anna dressed the kitten paid attention."}],

//// Set #2
  [["gp.trans",2], "AcceptabilityJudgment", {s: "Since Dave improved the department was satisfied."}],
  [["gp.intrans",2], "AcceptabilityJudgment", {s: "Since Dave worried the counselor devised a plan."}]
```

Good scripting habits:

- Grouping item sets together and separating sets with new line
- Adding comments (starts with two or more slashes //)
- Saving often! (editing in IBEX not recommended)

Practice and Fillers

Practice Trials:

• Presented at the beginning, accompanied by instructions and feedback

Filler Trials:

Mixed in with the critical trials

Both types of trials are invariant across conditions

This means that the *first element* of the big bracket can just be the *name of the stimuli*.

```
Practice Filler-good Filler-bad Filler-catch
```

Practice #1: "The car drove like a dream"

```
["practice-1",
  "AcceptabilityJudgment",
  {s: "The car drove like a dream."}]
```

Putting together the Body

```
var it.em = [
 //// Practice
 ["practice-1", "AcceptabilityJudgment", {s: "The car drove like a dream."}],
 // Critical Trials //
 //// Set. #1
 [["qp.trans",1], "AcceptabilityJudgment", {s: "While Anna trained the kitten paid attention."}],
 [["gp.intrans",1], "AcceptabilityJudgment", {s: "While Anna dressed the kitten paid attention."}],
 //// Set. #2
 [["qp.trans",2], "AcceptabilityJudgment", {s: "Since Dave improved the department was
satisfied." }],
  [["qp.intrans",2], "AcceptabilityJudgment", {s: "Since Dave worried the counselor devised a
plan."}],
 //// Fillers (Good)
 ["filler-good-01", "AcceptabilityJudgment", {s: "When Harry fell, the audience was shocked."}],
 //// Fillers (Bad)
 ["filler-bad-01", "AcceptabilityJudgment", {s: "When Tyler sneezed the driver, he passed a
tissue."}],
 //// Fillers (Catch)
 ["filler-catch-01", "AcceptabilityJudgment", {s: "Please select 4 for this sentence."}]
```

Important: The ordering of the trials here is just for human readability. We haven't yet told the program what order to present them in!

```
var item = [
  //// Practice
  ["practice-1", "AcceptabilityJudgment",
{s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1],
"AcceptabilityJudgment", {s: "While Anna
trained the kitten paid attention." }],
  [["gp.intrans",1],
"AcceptabilityJudgment", {s: "While Anna
dressed the kitten paid attention." }],
  //// Set. #2
  [["gp.trans",2],
"AcceptabilityJudgment", {s: "Since Dave
improved the department was satisfied."}],
  [["qp.intrans",2],
"AcceptabilityJudgment", {s: "Since Dave
worried the counselor devised a plan."}],
 //// Fillers (Good)
  ["filler-good-
01", "AcceptabilityJudgment", {s: "When
Harry fell, the audience was shocked." }],
```

List the *names* of each trial in order:

```
"practice-1",
"gp.trans",
"gp.intrans",
"gp.trans",
"gp.intrans",
"filler-good-01",
"filler-bad-01",
"filler-catch-01"
```

```
var item = [
  //// Practice
  ["practice-1", "AcceptabilityJudgment",
{s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1],
"AcceptabilityJudgment", {s: "While Anna
trained the kitten paid attention." }],
  [["gp.intrans",1],
"AcceptabilityJudgment", {s: "While Anna
dressed the kitten paid attention." }],
  //// Set. #2
  [["qp.trans",2],
"AcceptabilityJudgment", {s: "Since Dave
improved the department was satisfied."}],
  [["qp.intrans",2],
"AcceptabilityJudgment", {s: "Since Dave
worried the counselor devised a plan."}],
 //// Fillers (Good)
  ["filler-good-
01", "AcceptabilityJudgment", {s: "When
Harry fell, the audience was shocked."}],
```

Wrap them in seq():

```
seq(
  "practice-1",
  "gp.trans",
  "gp.intrans",
  "gp.trans",
  "gp.intrans",
  "filler-good-01",
  "filler-bad-01",
  "filler-catch-01"
)
```

```
var item = [
  //// Practice
  ["practice-1", "AcceptabilityJudgment",
{s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1],
"AcceptabilityJudgment", {s: "While Anna
trained the kitten paid attention." }],
  [["gp.intrans",1],
"AcceptabilityJudgment", {s: "While Anna
dressed the kitten paid attention." }],
  //// Set. #2
  [["qp.trans",2],
"AcceptabilityJudgment", {s: "Since Dave
improved the department was satisfied."}],
  [["qp.intrans",2],
"AcceptabilityJudgment", {s: "Since Dave
worried the counselor devised a plan."}],
 //// Fillers (Good)
  ["filler-good-
01", "AcceptabilityJudgment", {s: "When
Harry fell, the audience was shocked."}],
```

Assign to shuffleSequence:

```
var shuffleSequence = seq(
  "practice-1",
  "gp.trans",
  "gp.intrans",
  "gp.intrans",
  "gp.intrans",
  "filler-good-01",
  "filler-bad-01",
  "filler-catch-01"
)
```

```
var item = [
  //// Practice
  ["practice-1", "AcceptabilityJudgment",
{s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1],
"AcceptabilityJudgment", {s: "While Anna
trained the kitten paid attention." }],
  [["gp.intrans",1],
"AcceptabilityJudgment", {s: "While Anna
dressed the kitten paid attention." }],
  //// Set. #2
  [["qp.trans",2],
"AcceptabilityJudgment", {s: "Since Dave
improved the department was satisfied." }],
  [["gp.intrans",2],
"AcceptabilityJudgment", {s: "Since Dave
worried the counselor devised a plan."}],
  //// Fillers (Good)
  ["filler-good-
01", "AcceptabilityJudgment", {s: "When
Harry fell, the audience was shocked." }],
```

Assign to shuffleSequence:

```
var shuffleSequence = seq(
  "practice-1",
  "gp.trans",
  "gp.intrans",
  "gp.trans",
  "gp.intrans",
  "filler-good-01",
  "filler-bad-01",
  "filler-catch-01"
)
```

The shuffleSequence variable handles the *order of presentation* of the **experiment materials** that are stored inside the items variable.

```
var item = [
  //// Practice
  ["practice-1", "AcceptabilityJudgment",
{s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1],
"AcceptabilityJudgment", {s: "While Anna
trained the kitten paid attention." }],
  [["gp.intrans",1],
"AcceptabilityJudgment", {s: "While Anna
dressed the kitten paid attention." }],
  //// Set. #2
  [["qp.trans",2],
"AcceptabilityJudgment", {s: "Since Dave
improved the department was satisfied."}],
  [["gp.intrans",2],
"AcceptabilityJudgment", {s: "Since Dave
worried the counselor devised a plan."}],
 //// Fillers (Good)
 ["filler-good-
01", "AcceptabilityJudgment", {s: "When
Harry fell, the audience was shocked." }],
```

Problems

```
var shuffleSequence = seq(
   "practice-1",
   "gp.trans",
   "gp.intrans",
   "gp.intrans",
   "gp.intrans",
   "filler-good-01",
   "filler-bad-01",
   "filler-catch-01"
)
```

1. Lots of repetition ("-02", "-03", ...)

Defining the Sequence

```
var item = [
  //// Practice
  ["practice-1", "AcceptabilityJudgment",
{s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1],
"AcceptabilityJudgment", {s: "While Anna
trained the kitten paid attention."}],
  [["gp.intrans",1],
"AcceptabilityJudgment", {s: "While Anna
dressed the kitten paid attention." }],
  //// Set. #2
  [["qp.trans",2],
"AcceptabilityJudgment", {s: "Since Dave
improved the department was satisfied." }],
  [["gp.intrans",2],
"AcceptabilityJudgment", {s: "Since Dave
worried the counselor devised a plan."}],
 //// Fillers (Good)
 ["filler-good-
01", "AcceptabilityJudgment", {s: "When
Harry fell, the audience was shocked." }],
```

Problems

```
var shuffleSequence = seq(
  "practice-1",
  "gp.trans",
  "gp.intrans",
  "gp.intrans",
  "gp.intrans",
  "filler-good-01",
  "filler-bad-01",
  "filler-catch-01"
)
```

- 1. Lots of repetition ("-02", "-03", ...)
- 2. Presentation of some trials should be random

Defining the Sequence

```
var item = [
  //// Practice
  ["practice-1", "AcceptabilityJudgment",
{s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1],
"AcceptabilityJudgment", {s: "While Anna
trained the kitten paid attention."}],
  [["gp.intrans",1],
"AcceptabilityJudgment", {s: "While Anna
dressed the kitten paid attention." }],
  //// Set. #2
  [["qp.trans",2],
"AcceptabilityJudgment", {s: "Since Dave
improved the department was satisfied." }],
  [["gp.intrans",2],
"AcceptabilityJudgment", {s: "Since Dave
worried the counselor devised a plan."}],
 //// Fillers (Good)
 ["filler-good-
01", "AcceptabilityJudgment", {s: "When
Harry fell, the audience was shocked." }],
```

Problems

```
var shuffleSequence = seq(
  "practice-1",
  "gp.trans",
  "gp.intrans",
  "gp.intrans",
  "filler-good-01",
  "filler-bad-01",
  "filler-catch-01"
)
```

- 1. Lots of repetition ("-02", "-03", ...)
- 2. Presentation of some trials should be random
- 3. How do we counterbalance critical trials?

Sequence: multiple selection

To save us from writing repetitive code, we use startsWith()

```
var shuffleSequence = seq(
  "practice-1",
  "gp.trans",
  "gp.intrans",
  "gp.intrans",
  "filler-good-01",
  "filler-bad-01",
  "filler-catch-01"
)
```

```
var shuffleSequence = seq(
  startsWith("practice"),
  startsWith("gp")
  startsWith("filler")
)
```

The function startsWith() matches all names that starts with the given string.

Sequence: randomization

To mix critical and filler trials in random order, we use rshuffle()

```
var shuffleSequence = seq(
  startsWith("practice"),
  startsWith("gp")
  startsWith("filler")
)
```

```
var shuffleSequence = seq(
  startsWith("practice"),
  rshuffle(
      startsWith("gp"),
      startsWith("filler")
    )
)
```

By wrapping both the critical trials (*gp...*) and the filler trials (*filler...*) in rshuffle(), they are mixed together and presented in random order.

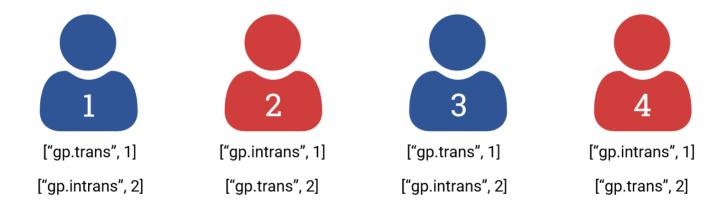
We can also write this out more compactly:

```
var shuffleSequence = seq(
  startsWith("practice"),
  rshuffle(startsWith("gp"), startsWith("filler"))
)
```

Sequence: counterbalancing

The **set number** in our critical trials automatically handles **counterbalancing**:

[["gp.trans", 1], ...], [["gp.intrans", 1], ...], [["gp.trans", 2], ...], [["gp.intrans", 2], ...]



We just need to add a **counter** inside items to track group assignment:

```
["setcounter", "__SetCounter__", { }]
```

Sequence: counterbalancing

Adding the counter to the experiment is simple:

```
var shuffleSequence = seq(
   "setcounter",
   ...
)

var items = [
   ["setcounter", "__SetCounter__", { }],
   ...
]
```

Note: Sometimes you want to place "setcounter" in the *middle* of the experiment

```
var shuffleSequence = seq(
   "intro"
   "consent",
   "setcounter",
   ...
)
```

Body and Sequence together

```
// Presentation Order //
var shuffleSequence = seq(
  "setcounter",
 startsWith("practice"),
 rshuffle(startsWith("gp"), startsWith("filler"))
// Experiment Materials //
var it.ems = [
 //// Counter
 ["setcounter", " SetCounter ", { }],
 //// Practice
 ["practice-1", "AcceptabilityJudgment", {s: "The car drove like a dream."}],
  // Critical Trials //
 //// Set. #1
 [["qp.trans",1], "AcceptabilityJudgment", {s: "While Anna trained the kitten paid attention."}],
 [["qp.intrans",1], "AcceptabilityJudgment", {s: "While Anna dressed the kitten paid attention."}],
 //// Set. #2
 [["qp.trans",2], "AcceptabilityJudgment", {s: "Since Dave improved the department was
satisfied." }],
  [["qp.intrans",2], "AcceptabilityJudgment", {s: "Since Dave worried the counselor devised a
```

Just need one more step: Settings

Settings (Basic)

Consist of miscellaneous options that we can put at the top of the script.

At the very least, we want to do two things:

- 1. Generate unique participant IDs
- 2. Specify the parameters for the **method design**

Settings (Basic)

Consist of miscellaneous options that we can put at the top of the script.

At the very least, we want to do two things:

- 1. Generate unique **participant IDs** (randomCode)
- 2. Specify the parameters for the **method design** (defaults)

```
// Options and Other Variables //

//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)

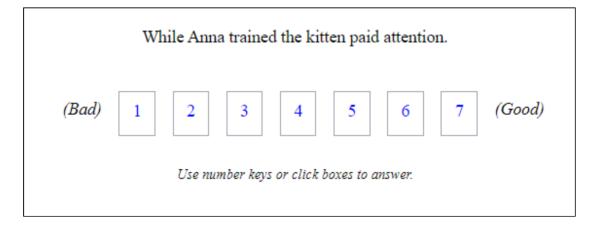
//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
   "AcceptabilityJudgment", {
     as: ["1", "2", "3", "4", "5", "6", "7"],
     presentAsScale: true,
     instructions: "Use number keys or click boxes to answer.",
     leftComment: "(Bad)",
     rightComment: "(Good)"
   }]
```

More details in the **AcceptabilityJudgment** section of the <u>documentation</u>.

A minimal experiment

The specifications in defaults set the design of the trials:

```
var defaults = [
   "AcceptabilityJudgment", {
      as: ["1", "2", "3", "4", "5", "6", "7"],
      presentAsScale: true,
      instructions: "Use number keys or click boxes to answer.",
      leftComment: "(Bad)",
      rightComment: "(Good)"
}]
```



Settings (Miscellaneous)

You can also change other options, such as showing a message at the end:

```
var completionMessage
```

```
//// A message to show to participants at completion (useful for confirm var completionMessage = "Thank you for your participation. Your participation."
```

And whether to show a progress bar:

var showProgressBar

```
//// Show a progress bar at the top? (true/false)
var showProgressBar = false
```

You can learn more about these various elements in the **Miscellaneous options** section of the <u>documentation</u>.

A minimal experiment

We now have a minimally working experiment!

```
// Options and Other Variables //
//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode
//// Show a progress bar at the top? (true/false)
var showProgressBar = false
//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
    "AcceptabilityJudgment", {
       as: ["1", "2", "3", "4", "5", "6", "7"],
       presentAsScale: true,
       instructions: "Use number keys or click boxes to answer.",
       leftComment: "(Bad)",
       rightComment: "(Good)"
   } ]
// Presentation Order //
var shuffleSequence = seq(
  "setcounter".
```

Interim Summary #1

What we've covered:

- We use a special syntax to create stimuli
- We store all the materials for our experiment inside items
- We specify the order of presentation inside shuffleSequence
- We set various options at the top of the script, such as defaults for the experiment *method* and the assignment of *participant ID*

A few more things to know:

- How can we add *plain text*?
 - Introduction page, consent form, directions, etc.
- How can we extend this workflow for *other experimental designs*?
 - Self-paced reading, comprehension tasks, etc.

The "Message" controller

The "Message" controller shows text on a new page.

It is a list of 3 elements, similar to the "AcceptabilityJudgement" items:

```
["Trial name", "Message", {html: text}]
```

- *Trial name* is used to reference the trial in sequencing, as seen earlier
- "Message" tells IBEX that this trial just shows text on a new screen
- Inside of the curly braces {} we can add text in the **html** parameter:

```
["intro", "Message", {html: ["p", "Welcome to the experiment!"]}]
```

Notes on **html**:

The code ["p", "<your text here>"] prints a single paragraph of text.

The "p" is called a **tags** and there are <u>many others</u>, but usually messages don't get more complicated than simple paragraphs.

Message examples

1-paragraph n-paragraphs consent keypress separator

A message composed of a single paragraph:

```
["intro", "Message", {html: ["p", "Welcome to the experiment!"]}]
```

Welcome to the experiment!

 \rightarrow Click here to continue.

Putting everything together

Here's a <u>complete experiment</u> with several **Message** controllers added.

```
// Options and Other Variables //
//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode
//// Show a progress bar at the top? (true/false)
var showProgressBar = false
//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
    "AcceptabilityJudgment", {
       as: ["1", "2", "3", "4", "5", "6", "7"],
      presentAsScale: true,
       instructions: "Use number keys or click boxes to answer.",
      leftComment: "(Bad)",
      rightComment: "(Good)"
   } ]
// Presentation Order //
var shuffleSequence = seq(
  "intro",
  "consent",
```

Getting the experiment up

So we have a new script, but how do we host the experiment?

- 1. Go to the <u>IBEX website</u> and log in.
- 2. Click on your experiment (or create one if you haven't already).
- 3. Click **edit** next to the *example_data.js* file in the **data_includes** section.
- 4. Delete its contents and copy paste your new code.
- 5. Click on the link at the top of the page.

/ibexexps/skku_ibex/workshop_acceptabilityJudgment/experiment.html

The URL shown in your browser is the link to your experiment!



https://spellout.net/ibexexps/skku ibex/workshop minimal/experiment.html

Making a different experiment

Suppose that after a *pilot experiment*, we find acceptability judgments to be inappropriate for answering our research question.

We want a *finer-grained measure* of recovery difficulty, so we'd like to change the experiment to **self-paced reading** and look at differences in *reading time*.

Given our existing template, we take the following steps:

- 1. Go to the <u>documentation</u> and find a Controller for self-paced reading.
- 2. Specify the design of that controller in the defaults variable.
- 3. Change our trials in items from "acceptabilityJudgment" to that Controller.
- 4. Make changes to the text of the "**Messages**" items (e.g., directions).

"DashedSentence" Controller

The "DashedSentence" Controller creates self-paced reading trials.

We first re-write defaults to specify appropriate settings for "DashedSentence".

```
var defaults = [
   "DashedSentence", {
      mode: "self-paced reading",
      display: "dashed"
}]
```

Then, we replace "acceptabilityJudgment" with "DashedSentence" in items.

```
var items = [
...
[["gp.trans",1], "DashedSentence", {s: "While Anna trained the kitten
[["gp.intrans",1], "DashedSentence", {s: "While Anna dressed the kitte
[["gp.trans",2], "DashedSentence", {s: "Since Dave improved the depart
[["gp.intrans",2], "DashedSentence", {s: "Since Dave worried the couns
...
]
```

Our second experiment

After re-writing some of the "Messages" items, we have our <u>new experiment!</u>

```
// Options and Other Variables //
//// Generates random number assigned to participants (Participant ID)
var randomCode = Math.random().toString(36).substr(2,9)
//// A message to show to participants at completion (useful for confirmation, raffle entry, etc.)
var completionMessage = "Thank you for your participation. Your participation code is: " +
randomCode
//// Show a progress bar at the top? (true/false)
var showProgressBar = false
//// Override default settings for controllers (parameters go inside the curly braces { })
var defaults = [
    "DashedSentence", {
      mode: "self-paced reading",
      display: "dashed"
   } ]
// Presentation Order //
var shuffleSequence = seq(
  "intro",
  "consent",
 "directions",
 startsWith("practice"),
  "setcounter",
```