

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



Analysis of the results

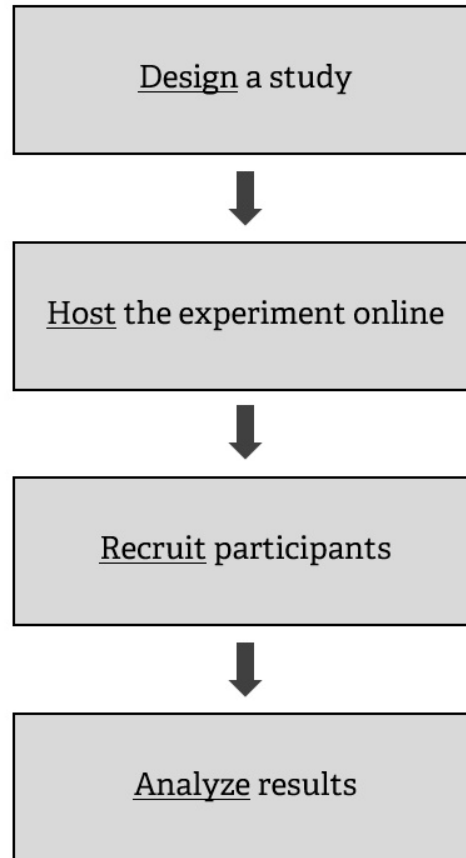
- Understanding the output format
- Importing into Excel and R

1. Introduction to IBEX

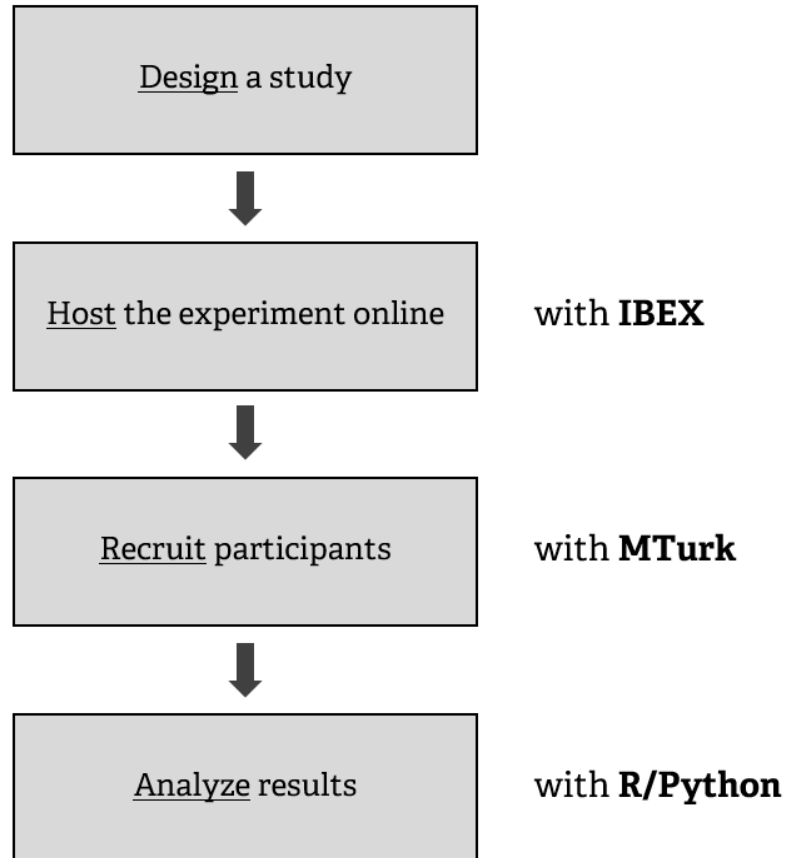
What is IBEX?

- Stands for (**I**)nternet-(**B**)ased (**EX**)periments
- **DOES:**
 - Host experiments on a webpage
 - Log user interactions with the experiment
 - 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

Ibex Farm

[home](#) | [login](#) | [create an account](#) | [ibex docs](#)

Welcome to the Ibex farm



This site provides free hosting for **ibex** experiments.

Upload data files in your browser, then send your participants a link to the experiment.

[View an example experiment](#)

Currently hosting **20564** experiments.

Have you found Ibex Farm useful? **Buy me a coffee!**
(Running costs are now covered for the foreseeable future. Thank you all.)

Contact a.d.drummond@gmail.com if you have any issues, or try the [google group](#).

The code for this site is BSD-licensed and available on [github](#).

▶ 0:00 / 0:25

🔊 🗨️ ⋮

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.js

Form.js

Message.js

Question.js

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 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.js

Form.js

Message.js

Question.js

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

2. 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("q1", "q2")));
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.

```
// Defaults and other settings //
```

```
//// 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.

A simple layout

```
template
1 var randomCode = Math.random().toString(36).substr(2,9);
2
3 var completionCode = String("MPI-" + randomCode);
4
5 var completionMessage = "Thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
6
7 var showProgressbar = false;
8
9 var defaults = {"acceptabilityJudgment": {
10   as: ["1", "2", "3", "4", "5", "6", "7"],
11   presentIsScale: true,
12   instructions: "Use number keys or click boxes to answer.",
13   leftComment: "(bad)",
14   rightComment: "(good)"
15 }};
16
17 var shuffleSequence = seq(
18   "set_counter",
19   "demographics",
20   "intro",
21   "practice",
22   seqWith("sep", rshuffle(startWith("opt"), startWith("filler"))))
23 );
24
25 var items = [
26
27   ["setCounter", "_setCounter_", {}],
28
29   ["intro", "Message", {
30     consentRequired: false,
31     html: ["div",
32       ["p", "Welcome! Here are some instructions to the experiment."],
33       ["p", "Hope you have a great time! It'll be a blast."]
34     ]
35   }],
36
37   ["practice", "Message", {
38     transfer: "keypress",
39     html: ["div",
40       ["p", "Before starting the questionnaire, let's do a couple of examples to get a feel for the task."]
41     ]
42   }],
43
44   ["practice", "acceptabilityJudgment", {s: "The car drove like a dream."}],
45   ["practice", "Message", {
46     transfer: "keypress",
47     html: ["div",
48       ["p", "How was that? Many people rate that sentence pretty good."]
49     ]
50   }],
51
52   ["practice", "Message", {
53     transfer: "keypress",
54     html: ["div",
55       ["p", "Let's try another one."]
56     ]
57   }],
58   ["practice", "acceptabilityJudgment", {s: "The cat ever has eaten cheese the."}],
59   ["practice", "Message", {
60     transfer: "keypress",
61     html: ["div",
62       ["p", "That sentence usually receives pretty poor ratings."],
63       ["p", "Ok! That's it. It's time to begin"]
64     ]
65   }],
66
67   ["sep", "separator", {}],
68
69   [{"mpi_graw", 1}], "acceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
70   [{"mpi_illus", 1}], "acceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}],
71   [{"mpi_ungraw", 1}], "acceptabilityJudgment", {s: "The duck that the goose chased has ever returned to our pond."}],
72   [{"mpi_graw", 1}], "acceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}],
73   [{"mpi_illus", 1}], "acceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}],
74   [{"mpi_ungraw", 1}], "acceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}],
75
76   [{"filler_4ad-81", "acceptabilityJudgment", {s: "The bored schoolchild kicked dog the without any apparent remorse."}],
77   [{"filler_810-82", "acceptabilityJudgment", {s: "Pauline left her sweater the train on the way home."}],
78   [{"filler_0000-83", "acceptabilityJudgment", {s: "Margaret spent all night working at the canery and was tired."}],
79   [{"filler_0000-84", "acceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}],
80
81   [{"filler-ATTENTIONCHECK-81", "acceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."}]
82 ];
83
84
85
86
87
88
89
```

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.

A simple layout

```
template
1 var randomCode = Math.random().toString(36).substr(2,9);
2
3 var completionCode = String("MPI-" + randomCode);
4
5 var completionMessage = "Thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
6
7 var showProgressbar = false;
8
9 var defaults = {"acceptabilityJudgment": {
10   ss: ["1", "2", "3", "4", "5", "6", "7"],
11   presentIsScale: true,
12   instructions: "Use number keys or click boxes to answer.",
13   leftComment: "(bad)",
14   rightComment: "(good)"
15 }};
16
17 var shuffleSequence = seq(
18   "set_counter",
19   "demographics",
20   "intro",
21   "practice",
22   seqWith("sep", rshuffle(startWith("opt"), startWith("filler"))))
23 );
24
25 var items = [
26
27   ["setCounter", "_setCounter_", { }],
28
29   ["intro", "Message", {
30     consentRequired: false,
31     html: ["div",
32       ["p", "Welcome! Here are some instructions to the experiment."],
33       ["p", "Hope you have a great time! It'll be a blast."]
34     ]
35   }],
36
37   ["practice", "Message", {
38     transfer: "keypress",
39     html: ["div",
40       ["p", "Before starting the questionnaire, let's do a couple of examples to get a feel for the task."]
41     ]
42   }],
43
44   ["practice", "AcceptabilityJudgment", {s: "The car drove like a dream."}],
45   ["practice", "Message", {
46     transfer: "keypress",
47     html: ["div",
48       ["p", "How was that? Many people rate that sentence pretty good."]
49     ]
50   }],
51
52   ["practice", "Message", {
53     transfer: "keypress",
54     html: ["div",
55       ["p", "Let's try another one."]
56     ]
57   }],
58   ["practice", "AcceptabilityJudgment", {s: "The cat ever has eaten cheese the."}],
59   ["practice", "Message", {
60     transfer: "keypress",
61     html: ["div",
62       ["p", "That sentence usually receives pretty poor ratings."],
63       ["p", "Ok! That's it. It's time to begin!"]
64     ]
65   }],
66
67   ["sep", "separator", {}],
68
69   [{"mpi_graw", 1}], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
70   [{"mpi_illus", 1}], "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}],
71   [{"mpi_ungraw", 1}], "AcceptabilityJudgment", {s: "The duck that the goose chased has ever returned to our pond."}],
72   [{"mpi_graw", 1}], "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}],
73   [{"mpi_illus", 1}], "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}],
74   [{"mpi_ungraw", 1}], "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}],
75
76   [{"filler_4ad-81", "AcceptabilityJudgment", {s: "The bored schoolchild kicked dog the without any apparent remorse."}],
77   [{"filler_810-82", "AcceptabilityJudgment", {s: "Pauline left her sweater the train on the way home."}],
78   [{"filler_0000-83", "AcceptabilityJudgment", {s: "Margaret spent all night working at the canery and was tired."}],
79   [{"filler_0000-82", "AcceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}],
80
81   [{"filler-ATTENTIONCHECK-81", "AcceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."}]
82
83 ];
84
85
86
87
88
89
90
```

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)

A simple layout

```
1 var randomCode = Math.random().toString(16).substr(2,9);
2
3 var completionCode = String("MPI-" + randomCode);
4
5 var completionMessage = "Thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
6
7 var showProgressBar = false;
8
9 var defaults = { "acceptabilityJudgment": {
10   seq: ["1", "2", "3", "4", "5", "6", "7"],
11   presentIsScale: true,
12   instructions: "Use number keys or click boxes to answer.",
13   leftComment: "(bad)",
14   rightComment: "(good)"
15 }};
16
17 var shuffleSequence = seq({
18   "setCounter",
19   "demographic",
20   "intro",
21   "practice",
22   seqWith("sep", rshuffle(startWith("opt"), startWith("filler")))});
23
24
25 var items = [
26
27   ["setCounter", "_setCounter_", {}],
28
29   ["intro", "Message", {
30     consentRequired: false,
31     html: ["div",
32       ["p", "Welcome! Here are some instructions to the experiment."],
33       ["p", "Hope you have a great time! It'll be a blast."]
34     ]
35   }],
36
37   ["practice", "Message", {
38     transfer: "keypress",
39     html: ["div",
40       ["p", "Before starting the questionnaire, let's do a couple of examples to get a feel for the task."]
41     ]
42   }],
43
44   ["practice", "AcceptabilityJudgment", {s: "The car drove like a dream."}],
45   ["practice", "Message", {
46     transfer: "keypress",
47     html: ["div",
48       ["p", "How was that? Many people rate that sentence pretty good."]
49     ]
50   }],
51
52   ["practice", "Message", {
53     transfer: "keypress",
54     html: ["div",
55       ["p", "Let's try another one."]
56     ]
57   }],
58
59   ["practice", "AcceptabilityJudgment", {s: "The cat ever has eaten cheese the."}],
60   ["practice", "Message", {
61     transfer: "keypress",
62     html: ["div",
63       ["p", "That sentence usually receives pretty poor ratings."],
64       ["p", "Ok! That's it. It's time to begin"]
65     ]
66   }],
67
68   ["sep", "separator", {}],
69
70   ["mpi грам", "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
71   ["mpi иллю", "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}],
72   ["mpi уграм", "AcceptabilityJudgment", {s: "The duck that the goose chased has ever returned to our pond."}],
73
74   ["mpi грам", "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}],
75   ["mpi иллю", "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}],
76   ["mpi уграм", "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}],
77
78   ["filler-сад-81", "AcceptabilityJudgment", {s: "The bored schoolchild kicked dog the without any apparent remorse."}],
79   ["filler-био-82", "AcceptabilityJudgment", {s: "Pauline left her sweater the train on the way home."}],
80   ["filler-0000-83", "AcceptabilityJudgment", {s: "Margaret spent all night working at the camera and has tired."}],
81   ["filler-0000-84", "AcceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}],
82
83   ["filler-ATTENTIONCHECK-84", "AcceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."}]
84
85 ];
```

Parts of the Script

Settings:

- Sets various options for the experiment

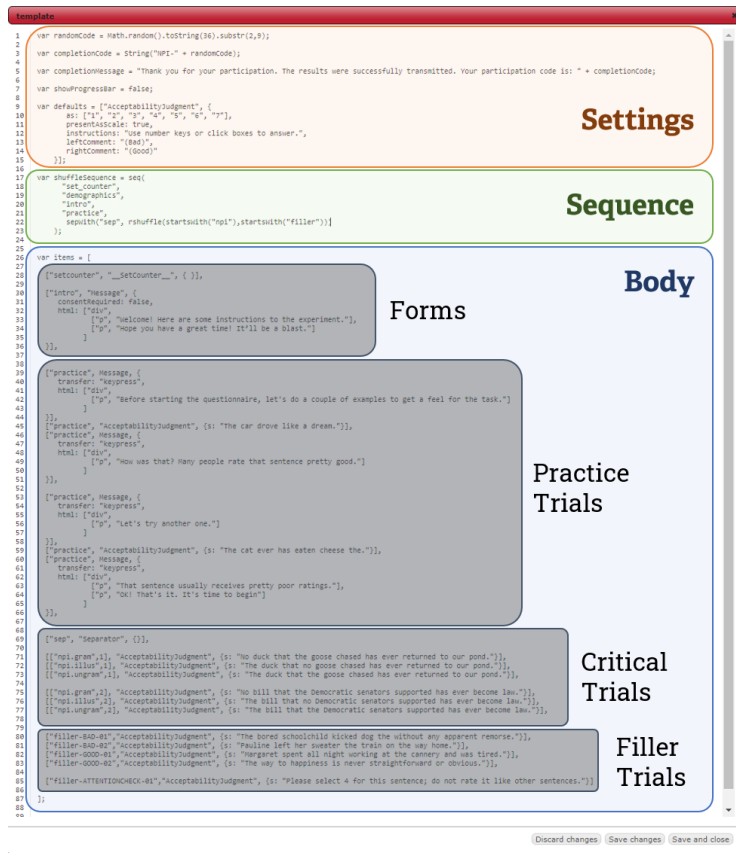
Sequence (**shuffleSequence**):

- Specifies the ordering of the different parts of the experiment

Body (**items**):

- Includes the actual material that will be shown to the participants

A simple layout



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

template

```
1 var randomCode = Math.random().toString(36).substr(2,9);
2
3 var completionCode = String("MPI-" + randomCode);
4
5 var completionMessage = "Thank you for your participation. The results were successfully transmitted. Your participation code is: " + completionCode;
6
7 var showProgressBar = false;
8
9 var defaults = {"AcceptabilityJudgment", {
10   as: ["1", "2", "3", "4", "5", "6", "7"],
11   presentAsScale: true,
12   instructions: "use number keys or click boxes to answer.",
13   leftComment: "(bad)",
14   rightComment: "(good)"
15 }};
16
17 var shuffleSequence = seq(
18   "set_counter",
19   "demographics",
20   "intro",
21   "practice",
22   sepwith("sep", rshuffle(startswith("mpi"), startswith("filler")))
23 );
24
25 var items = [
26
27   [{"setCounter", "setCounter", { }},
28
29   [{"intro", "Message", {
30     consentRequired: false,
31     html: ["div",
32       ["p", "Welcome! Here are some instructions to the experiment."],
33       ["p", "Hope you have a great time! It'll be a blast."]
34     ]
35   }},
36
37   [{"practice", "Message", {
38     transfer: "keypress",
39     html: ["div",
40       ["p", "Before starting the questionnaire, let's do a couple of examples to get a feel for the task."]
41     ]
42   }},
43   [{"practice", "AcceptabilityJudgment", {s: "The car drove like a dream."}},
44   [{"practice", "Message", {
45     transfer: "keypress",
46     html: ["div",
47       ["p", "How was that? Many people rate that sentence pretty good."]
48     ]
49   }},
50   [{"practice", "Message", {
51     transfer: "keypress",
52     html: ["div",
53       ["p", "Let's try another one."]
54     ]
55   }},
56   [{"practice", "AcceptabilityJudgment", {s: "The cat ever has eaten cheese the."}},
57   [{"practice", "Message", {
58     transfer: "keypress",
59     html: ["div",
60       ["p", "That sentence usually receives pretty poor ratings."],
61       ["p", "OK! That's it. It's time to begin"]
62     ]
63   }},
64   [{"sep", "Separator", { }},
65
66   [{"mpi.gram", 1}, {"AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}},
67   [{"mpi.illus", 1}, {"AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}},
68   [{"mpi.ungram", 1}, {"AcceptabilityJudgment", {s: "The duck that the goose chased has ever returned to our pond."}},
69
70   [{"mpi.gram", 2}, {"AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}},
71   [{"mpi.illus", 2}, {"AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}},
72   [{"mpi.ungram", 2}, {"AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}},
73
74   [{"filler-BAD-01", "AcceptabilityJudgment", {s: "The bored schoolchild kicked dog the without any apparent remorse."}},
75   [{"filler-BAD-02", "AcceptabilityJudgment", {s: "Pauline left her sweater the train on the way home."}},
76   [{"filler-GOOD-01", "AcceptabilityJudgment", {s: "Margaret spent all night working at the cannery and was tired."}},
77   [{"filler-GOOD-02", "AcceptabilityJudgment", {s: "The way to happiness is never straightforward or obvious."}},
78   [{"filler-ATTENTIONCHECK-01", "AcceptabilityJudgment", {s: "Please select 4 for this sentence; do not rate it like other sentences."}}]
79 ];
80
81
82
83
84
85
86
87
88
89
90
```

Settings

Sequence

Body

Forms

Practice Trials

Critical Trials

Filler Trials

Our first experiment

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

"While Anna dressed the kitten paid attention."

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

... ✓[_{VP} dressed], the kitten ...

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

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

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

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

Translating the design to code

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".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

Translating the design to code

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".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

Translating the design to code

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".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

Translating the design to code

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".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```


Translating the design to code

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".

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "While Anna trained the kitten paid attention."}  
]
```

Translating the design to code

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"**.

```
[  
  [ "gp.trans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "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-
```

Translating the design to code

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"**.

```
[  
  [ "gp.intrans", 1 ],  
  "AcceptabilityJudgment",  
  {s: "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 items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment", {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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!

Defining the Sequence

```
var items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment",  
   {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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"
```


Defining the Sequence

```
var items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment",  
   {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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"  
)
```

Defining the Sequence

```
var items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment",  
   {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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"  
)
```

Defining the Sequence

```
var items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment",  
   {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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.

Defining the Sequence

```
var items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment",  
   {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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.trans",  
  "gp.intrans",  
  "filler-good-01",  
  "filler-bad-01",  
  "filler-catch-01"  
)
```

1. A lot of typing ("-02", "-03", ...)

Defining the Sequence

```
var items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment",  
   {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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.trans",  
  "gp.intrans",  
  "filler-good-01",  
  "filler-bad-01",  
  "filler-catch-01"  
)
```

1. A lot of typing ("-02", "-03", ...)
2. Presentation order of some trials should be random

Defining the Sequence

```
var items = [  
  
  //// Practice  
  ["practice-1", "AcceptabilityJudgment",  
   {s: "The car drove like a dream."}],  
  
  // Critical Trials //  
  
  //// 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."}],  
  
  //// 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.trans",  
  "gp.intrans",  
  "filler-good-01",  
  "filler-bad-01",  
  "filler-catch-01"  
)
```

1. A lot of typing ("-02", "-03", ...)
2. Presentation order 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.trans",  
  "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], ...]`



`[“gp.trans”, 1]`

`[“gp.intrans”, 2]`



`[“gp.intrans”, 1]`

`[“gp.trans”, 2]`



`[“gp.trans”, 1]`

`[“gp.intrans”, 2]`



`[“gp.intrans”, 1]`

`[“gp.trans”, 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 items = [

  //// Counter
  ["setcounter", "__SetCounter__", { }],

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

  // Critical Trials //

  //// 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
```

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`)

```
// Defaults and other settings //
```

```
//// 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 [documentation](#).

defaults variable

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)"  
  }  
]
```

While Anna trained the kitten paid attention.

(Bad) (Good)

Use number keys or click boxes to answer.

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 confir  
var completionMessage = "Thank you for your participation. Your particip
```

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 [documentation](#).

A minimal experiment

We now have a minimally working experiment!

```
// Defaults and other settings //
```

```
//// 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 many others, 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!

→ [Click here to continue.](#)

Putting everything together

Here's a complete experiment with several **Message** controllers added.

```
// Defaults and other settings //
```

```
//// 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 IBEX website 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!

 spellout.net/ibexexps/skku_ibex/workshop_acceptabilityJudgment/experiment.html

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 documentation 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).

Let's go look at the documentation!

"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 kitten  
  ["gp.trans",2], "DashedSentence", {s: "Since Dave improved the depart  
  ["gp.intrans",2], "DashedSentence", {s: "Since Dave worried the couns  
  ...  
]
```

Our second experiment

Finally, after re-writing some of the "**Message**" items, we have a new experiment!

```
// Defaults and other settings //
```

```
//// 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",
```


3. Analysis of the results

Where to find the data

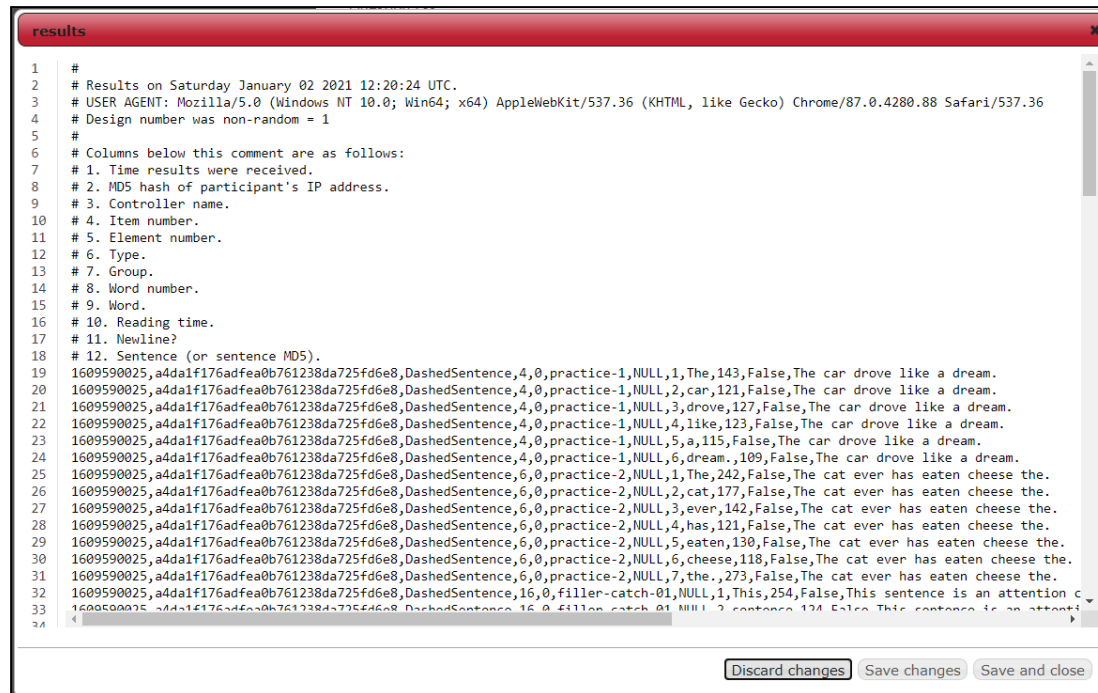
The data collected in the experiment can be found on the experiment page, under the **results** section.

```
results (upload a file to this directory | refresh)  
raw_results (delete | rename | upload new version | edit)  
results (delete | rename | upload new version | edit)
```

The section comprises of two files, of which *results* is the better formatted one.

The *results* file

results is a **csv** file (**c**omma **s**eparated **v**alues), meaning that each line contains a set of values that are separated by a comma.



```
1 #
2 # Results on Saturday January 02 2021 12:20:24 UTC.
3 # USER AGENT: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
4 # Design number was non-random = 1
5 #
6 # Columns below this comment are as follows:
7 # 1. Time results were received.
8 # 2. MD5 hash of participant's IP address.
9 # 3. Controller name.
10 # 4. Item number.
11 # 5. Element number.
12 # 6. Type.
13 # 7. Group.
14 # 8. Word number.
15 # 9. Word.
16 # 10. Reading time.
17 # 11. Newline?
18 # 12. Sentence (or sentence MD5).
19 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,4,0,practice-1,NULL,1,The,143,False,The car drove like a dream.
20 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,4,0,practice-1,NULL,2,car,121,False,The car drove like a dream.
21 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,4,0,practice-1,NULL,3,drove,127,False,The car drove like a dream.
22 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,4,0,practice-1,NULL,4,like,123,False,The car drove like a dream.
23 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,4,0,practice-1,NULL,5,a,115,False,The car drove like a dream.
24 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,4,0,practice-1,NULL,6,dream,109,False,The car drove like a dream.
25 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,6,0,practice-2,NULL,1,The,242,False,The cat ever has eaten cheese the.
26 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,6,0,practice-2,NULL,2,cat,177,False,The cat ever has eaten cheese the.
27 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,6,0,practice-2,NULL,3,ever,142,False,The cat ever has eaten cheese the.
28 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,6,0,practice-2,NULL,4,has,121,False,The cat ever has eaten cheese the.
29 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,6,0,practice-2,NULL,5,eaten,130,False,The cat ever has eaten cheese the.
30 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,6,0,practice-2,NULL,6,cheese,118,False,The cat ever has eaten cheese the.
31 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,6,0,practice-2,NULL,7,the,273,False,The cat ever has eaten cheese the.
32 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,16,0,filler-catch-01,NULL,1,This,254,False,This sentence is an attention c
33 1609590025,a4da1f176adfea0b761238da725fd6e8,DashedSentence,16,0,filler-catch-01,NULL,2,sentence,124,False,This sentence is an attenti
34
```

The lines that start with a pound symbol "#" are **comments** that include *metadata*.

They tell us what **variables** each comma-separated values correspond to.

The variables

All IBEX experiments return these **7** variables:

1. **Time**
2. **Participant ID**
3. **Controller Name**
4. **Item number**
5. **Element number**
6. **Type**
7. **Group**

The variables

Generally, we only care about **4** of these:

1. Time
2. **Participant ID**: A unique ID for each participant.
3. **Controller Name**: The controller that the values are from.
4. Item number
5. Element number
6. **Type**: The name of the trial.
7. **Group**: The item group number.

The last two variables uniquely identify each trial created in **items**:

```
[["Type", Group], "Trial Type", {s: "Sentence"}]
```

```
[["gp.trans", 1], "DashedSentence", {s: "..."}]
```

The variables

Depending on the experiment, IBEX returns other variables specific to the design.

For self-paced reading with "**DashedSentence**", we get **5** more variables:

1. **Word number** - The index of the word in the sentence.
2. **Word** - The text of that word.
3. **Reading time** - Reading time for that word.
4. **Newline?** - 0 or 1 indicating whether there was a line break.
5. **Sentence** - The text of the sentence.

These are also outlined in the [documentation](#) for "**DashedSentence**", so you know what variables to expect beforehand.

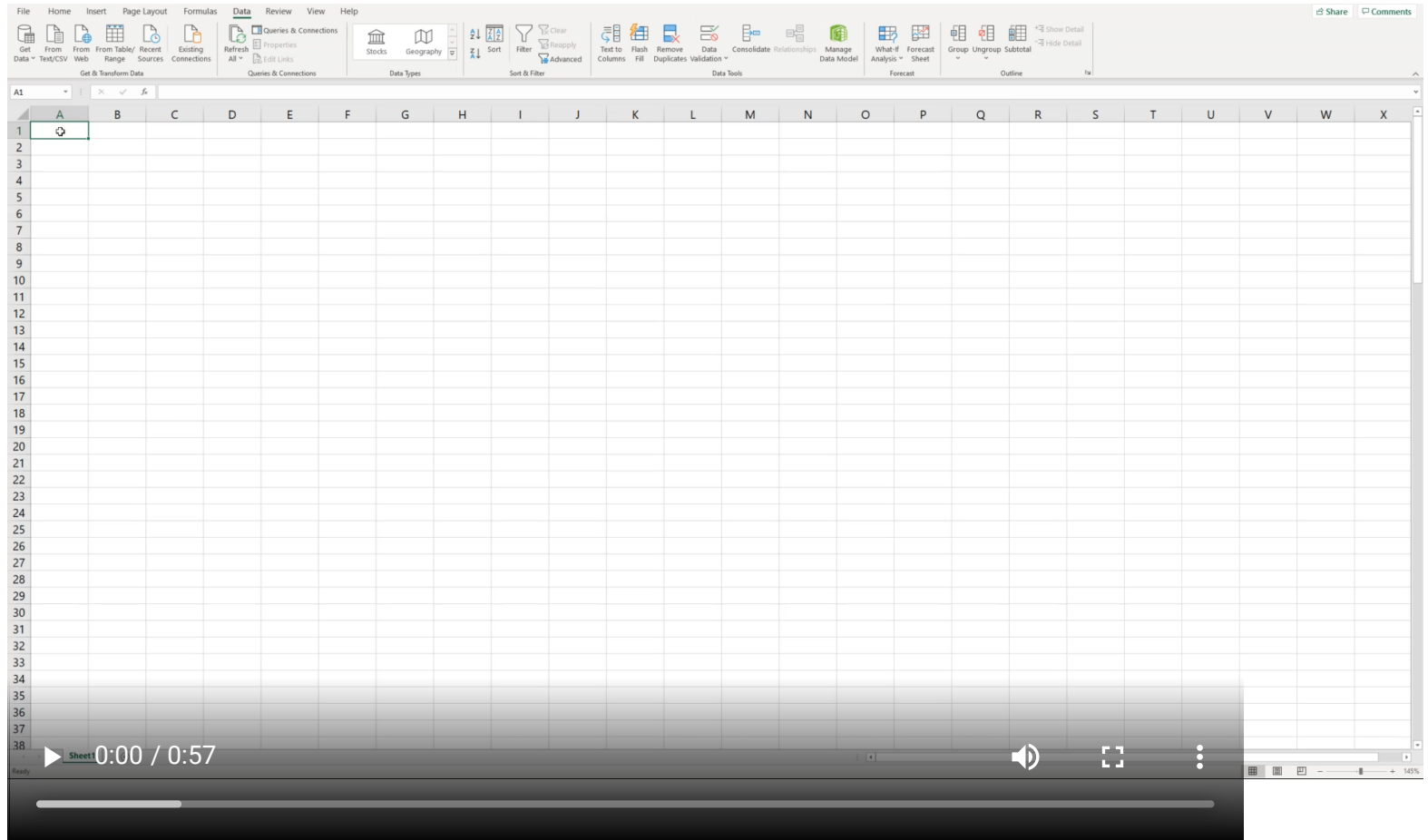
For an actual analysis of the results, we need the results to be imported somewhere as a **data frame** (where every value is a cell and each variable forms a column).

Importing - Excel

Steps for importing the data into Excel:

1. Copy the text of the *results* file from IBEX
2. Paste into the first column of an Excel spreadsheet
3. Highlight that column and click **Data** tab -> **Filter**
4. Click on the dropdown arrow that appears at the top cell
5. Click **Text Filters** -> **Begins With...** and type in "#"
6. Go to **Home** tab -> **Find & Select**, check "Visible cells only", and click OK
7. Right click on any part of the sheet and select **Delete Row**
8. Click the first column again and go to **Data** tab -> **Text to Columns**
9. Check "Delimited", "Comma", then "General" for each prompt
10. Add an empty row at the top and manually type in the column names from the comments of the original *results* file

Importing - Excel



Importing - R

Steps for importing into R:

1. Download the *results* file from IBEX
2. Open it with the `read_ibex()` function using a script

```
source("read_ibex.R")
results <- read_ibex("SPR_results.txt")
```

```
##      Controller name      Type Group Word number      Word Reading time
## 92   DashedSentence      gp.intrans      2          8    plan.         261
## 75   DashedSentence      gp.trans       1          5   kitten         277
## 95   DashedSentence      filler-bad-01   NULL          3 sneezed         339
##                                     Sentence (or sentence MD5)
## 92      Since Dave worried the counselor devised a plan.
## 75      While Anna trained the kitten paid attention.
## 95 When Tyler sneezed the driver%2C he passed a tissue.
```

Preferred method because data can get more complicated.

Resources

All materials from this presentation can be found on [github](#).

More resources from the community:

- The official [IBEX documentation](#).
- The official [google group](#) to ask questions.

UPDATE (01/01/2021)

There is a **new version of IBEX** at a different link: <https://ibex.spellout.net>



Alex Drummond
to ibexperiments

Jan 1

Happy new year!

I'm announcing a complete rewrite of the Ibex Farm deployed at

<https://ibex.spellout.net>

The user-facing changes are conservative and incremental, but there are some significant improvements and new features. You can find an exhaustive list of these at

<https://gist.github.com/addrummond/b345d3d4f23436838e52afbe880ebbd9>. The first section should tell you

Resources


All materials from this presentation can be found on [github](#).

More resources from the community:

- The official [IBEX documentation](#).
- The official [google group](#) to ask questions.

UPDATE (01/01/2021)

New feature: results can be downloaded directly as a spreadsheet.

 **workshop_DashedSentence**

PUBLIC [make private](#)

Ibex version: 0.3.9 (imported from original Ibex Farm)
The counter is set to 2.

● 1 set of results.

Results for workshop_DashedSentence

This experiment has 1 set of results.

[Results as CSV spreadsheet](#)

[Results as Excel .xlsx spreadsheet](#)

[Results in original Ibex format \(with no column name comments\)](#)

[Results in original Ibex format \(with per-line column name comments\)](#)

[Raw JSON results](#)

Resources

All materials from this presentation can be found on [github](#).

More resources from the community:

- The official [IBEX documentation](#).
- The official [google group](#) to ask questions.

Other platforms

- PCIBex (Penn Controller for IBEX) [Link](#)
- PsychoPy [Link](#)
- _magpie (minimal architecture for the generation of portable interactive experiments) [Link](#)