DOING EXPERIMENTS FOR LINGUISTICS: Internet-based experimentation using the Ibex Farm

Brian Dillon
UMass Amherst

Rodica Ivan
UMass Amherst

LSA Institute Minicourse, Jan 4th 2018

TODAY:

PART 1: The acceptability judgment experiment / comprehension experiment

- **1:** Scientific + experimental methods
- **2:** Our hypothesis of the day: Interference and NPIs
- **3:** Experimental design: Basic principles

PART 2: The Ibex Farm

- **1:** The basic judgment experiment in the Ibex Farm
- **2:** Going beyond the basics in Ibex
- **3:** Collecting and analyzing data

PART 3: Collecting participants

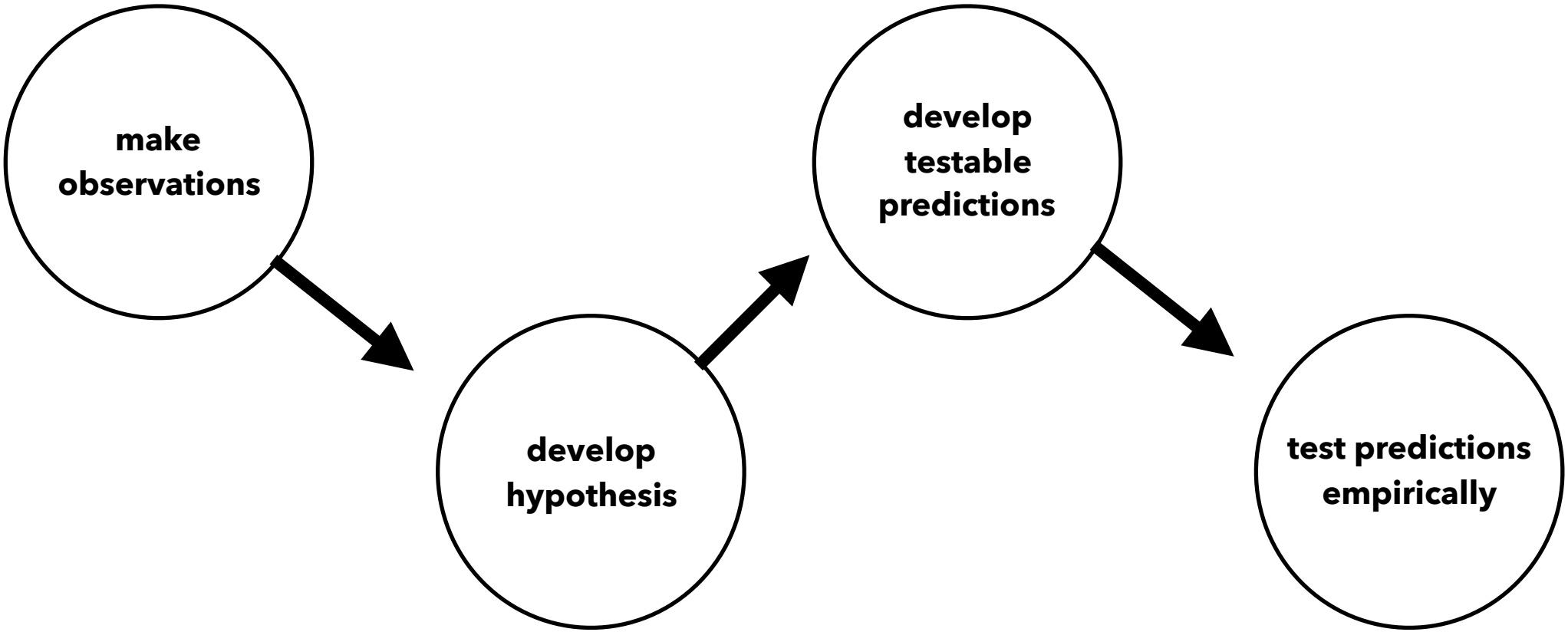
- **1:** Mechanical Turk and other crowdsourcing sites
- **2:** Other means of recruitment

PART 4: Quality control

- **1:** Strategies for quality control
- **2:** Detecting participant 'non-compliance'
- **3:** Participant non-naïveté

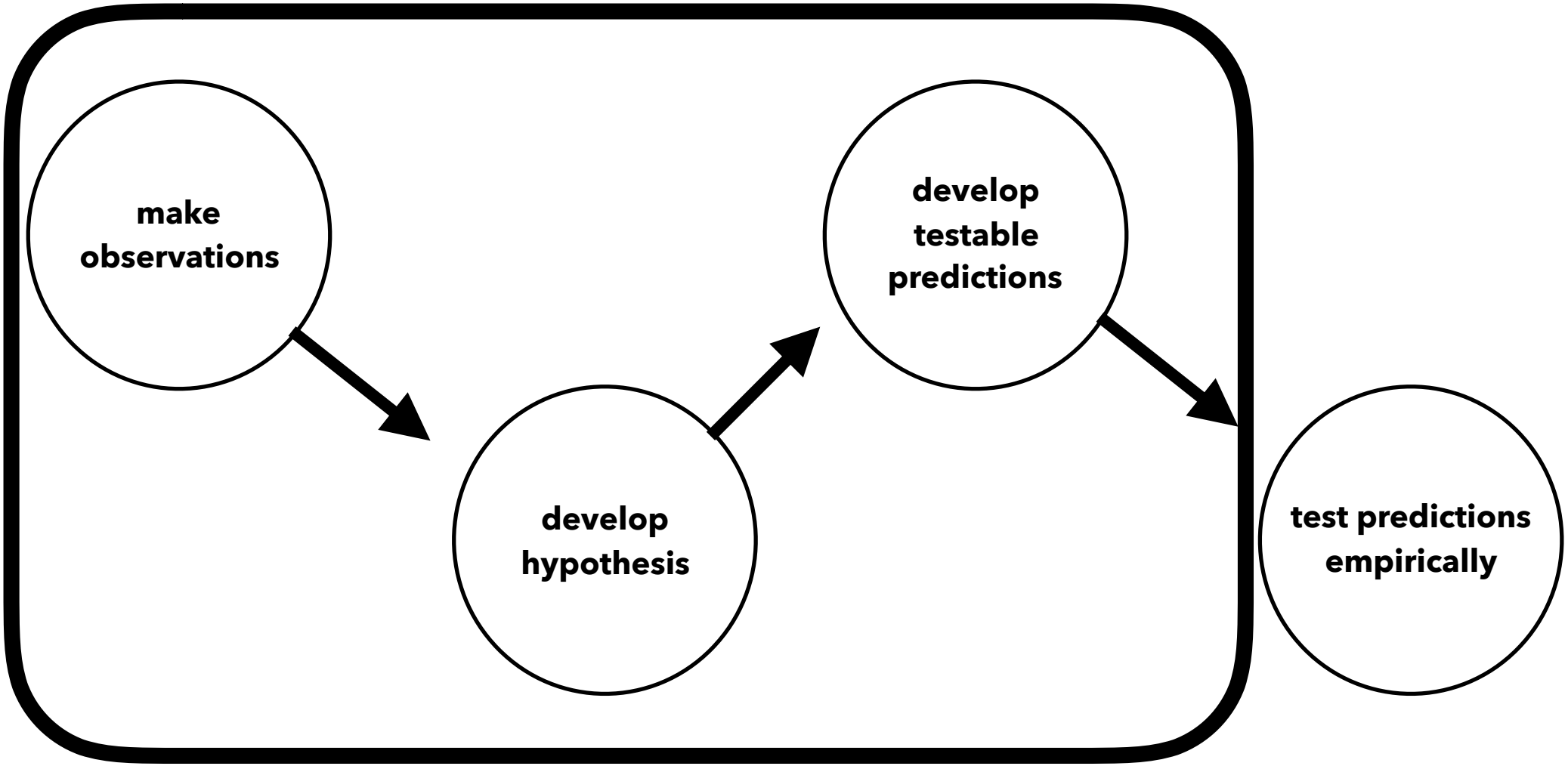
PART 1: The Experiment

The scientific method:



PART 1: The Experiment

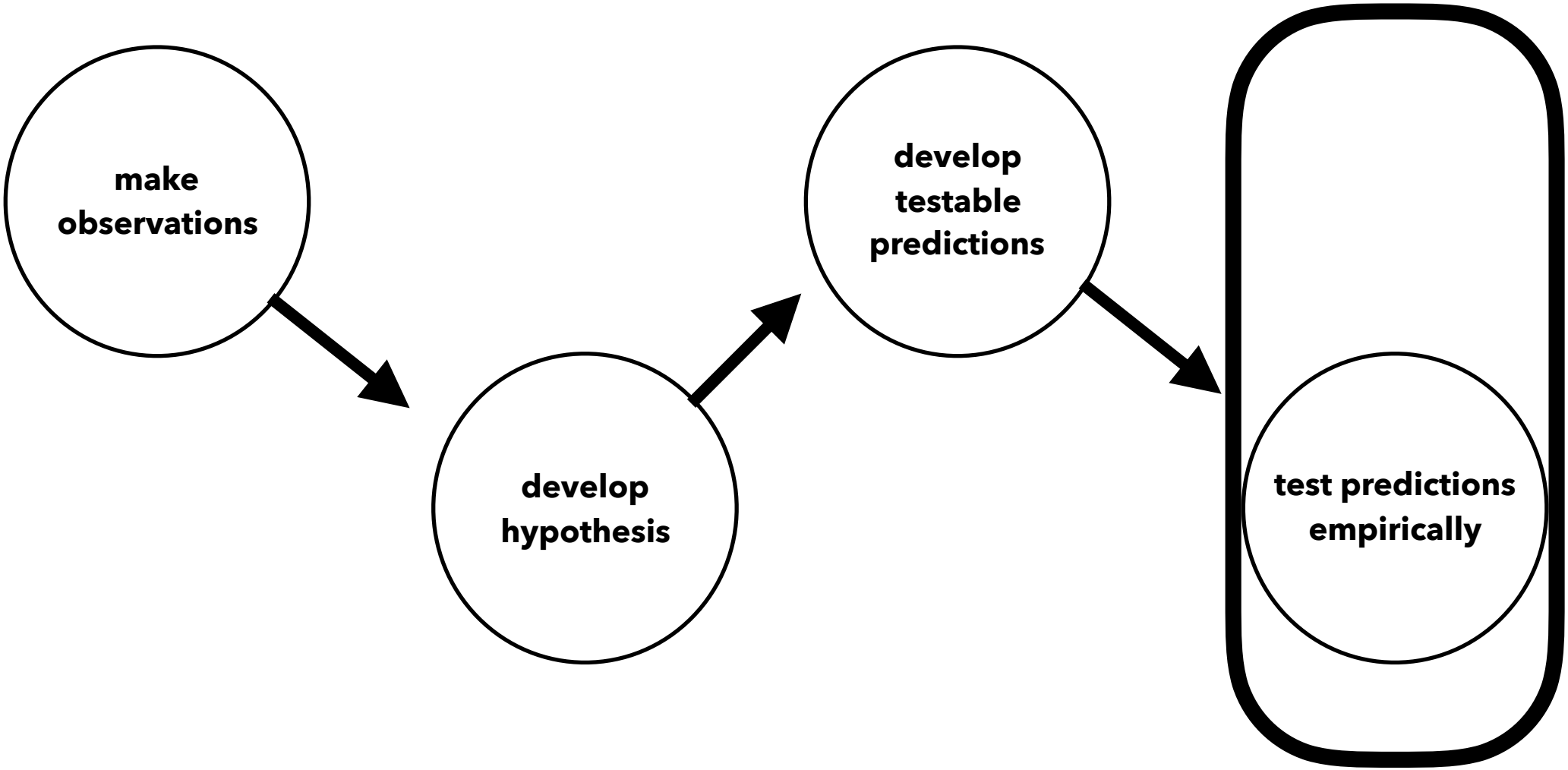
The scientific method:



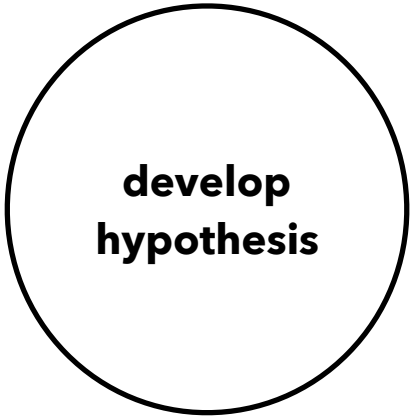
your day-to-day as working linguist

PART 1: The Experiment

The scientific method:



PART 1: The Experiment



Hypothesis: Licensing an NPI, like other syntactic dependents, involves memory retrieval to 'check' whether there is any syntactically available negation.

Search for +NEG

No duck that the goose chased has ever returned to our pond

PART 1: The Experiment

develop testable predictions

Predictions:

Acceptability should be decreased when memory retrieval fails.

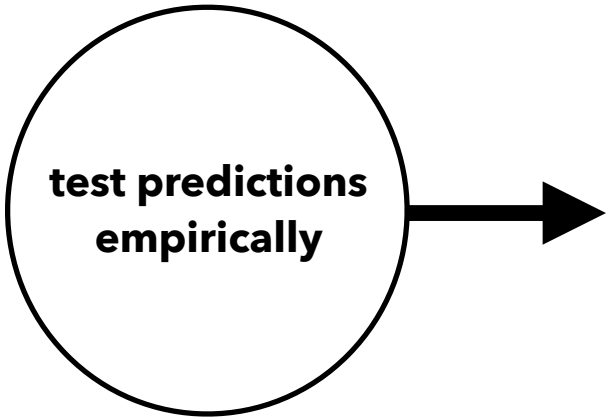
However, there should be 'NPI illusions' when there is a linearly preceding negation that can interfere with retrieval:

No duck that the goose chased has ever returned to our pond

The duck that no goose chased has ever returned to our pond

The duck that the goose chased has ever returned to our pond

PART 1: The Experiment



The experimental method:
Manipulating one variable (the *independent variable*), and determining whether this manipulation impacts another variable (the *dependent variable*)

Independent variable / Experimental factor:

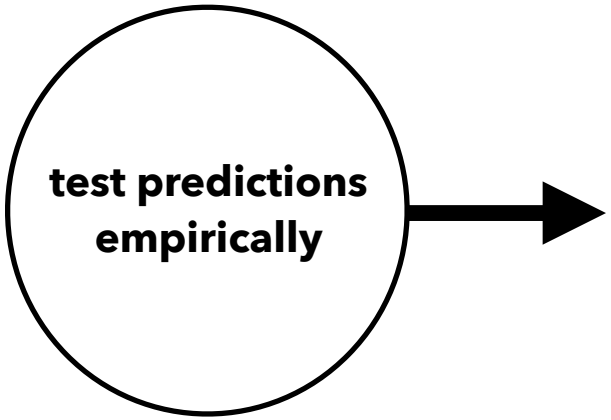
NPI licensing: {GRAM, ILLUSORY, UNGRAM}

No duck that the goose chased has ever returned to our pond

The duck that no goose chased has ever returned to our pond

The duck that the goose chased has ever returned to our pond

PART 1: The Experiment



The experimental method:
Manipulating one variable (the *independent variable*), and determining whether this manipulation impacts another variable (the *dependent variable*)

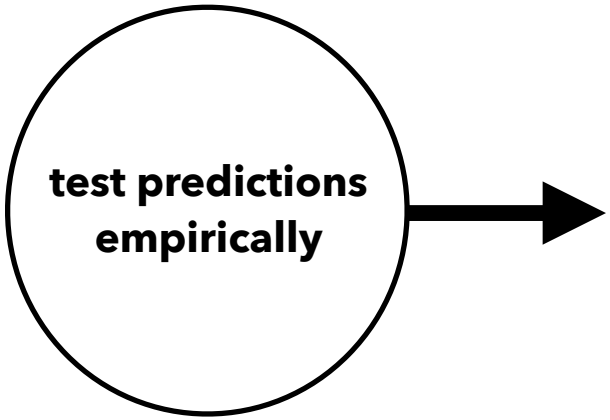
Dependent variable / Experimental measurement:

Rating of the sentence on a 1-7 scale OR

Binary classification of sentence acceptability (Y/N)

The duck that no goose chased has ever returned to our pond

PART 1: The Experiment



The experimental method:
Manipulating one variable (the *independent variable*), and determining whether this manipulation impacts another variable (the *dependent variable*)

Hypothesis predicts: Higher ratings for GRAM sentences than UNGRAM and ILLUSORY sentences, higher ratings for ILLUSORY sentences than UNGRAM sentences.

PART 1: The Experiment

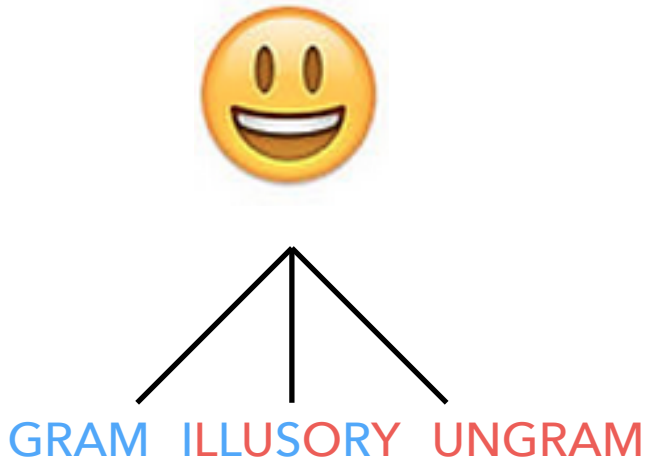
- **Condition:** Each distinct level of your independent variable
- **Participant:** A single participant in a questionnaire.
- **Item set:** A single set of minimal pair stimuli that differ only in the independent variable manipulated.
- **Experimental list:** The set of experimental stimuli that will be presented to a given participant.

Two types of experimental designs are :

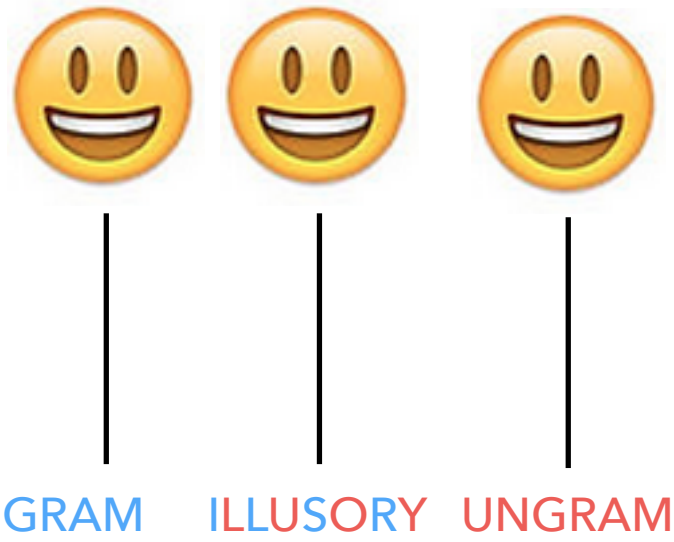
- ***within-subject designs***, meaning that a single **participant** will see all **conditions.** **(typical in linguistics)**
- ***between-subject designs***, a single **participant** will not see all **conditions.**

PART 1: The Experiment

Repeated Measures



Independent Measures



Within-subjects designs typically require fewer participants because each participant serves as their own control. They therefore give greater statistical power (i.e. chance of finding an effect if there is one to be found). However, your experimental conditions may interact.

PART 1: The Experiment

- **Condition:** Each distinct level of your independent variable
- **Participant:** A single participant in a questionnaire.
- **Item set:** A single set of minimal pair stimuli that differ only in the independent variable manipulated.
- **Experimental list:** The set of experimental stimuli that will be presented to a given participant.

Experimental designs can be:

- ***within-item designs***, meaning that a single **item set** contains all **conditions.**
- ***between-item designs***, a single **item** exists only in one **condition.**

PART 1: The Experiment

Item 1:

No duck that the goose chased has ever returned to our pond	**GRAM**
The duck that no goose chased has ever returned to our pond	**ILLUSORY**
The duck that the goose chased has ever returned to our pond	**UNGRAM**

Item 2:

No bills that the Democratic senators supported have ever become law.	**GRAM**
The bills that no Democratic senators supported have ever become law.	**ILLUSORY**
The bills that the Democratic senators supported have ever become law.	**UNGRAM**

within-item designs, meaning that a single **item set** will contain all **conditions**

PART 1: The Experiment

Item 1:

No duck that the goose chased has ever returned to our pond **GRAM**

Item 2:

The bills that no Democratic senators supported have ever become law. **UNGRAM**

between-items design, a single **item set** does not contain all **conditions.**

PART 1: The Experiment

Repeated Measures

No duck that the goose chased has ever returned to our pond

The duck that no goose chased has ever returned to our pond

The duck that the goose chased has ever returned to our pond

Within-items designs typically require less control over potentially confounding factors (plausibility of scenario, frequency of words), because each item set as its own control. Like within-subjects designs, they therefore give greater statistical power (i.e. chance of finding an effect if there is one to be found).

PART 1: The Experiment

- **Condition:** Each distinct level of your independent variable
- **Participant:** A single participant in a questionnaire.
- **Item set:** A single set of minimal pair stimuli that differ only in the independent variable manipulated.
- **Experimental list:** The set of experimental stimuli that will be presented to a given participant.
- **Fillers:** Experimental trials distinct from your critical stimuli, typically intended to mask the experimental manipulation.
- **Practice trials:** Experimental trials presented to familiarize a participant with the experimental design.
- **Latin Square distribution:** A technique for distributing a within-items design into distinct lists for experimental presentation.

PART 1: The Experiment

Item 1:

No duck that the goose chased has ever returned to our pond

The duck that no goose chased has ever returned to our pond

The duck that the goose chased has ever returned to our pond

Item 2:

No bills that the Democratic senators supported have ever become law.

The bills that no Democratic senators supported have ever become law.

The bills that the Democratic senators supported have ever become law.

Item 3:

No bears that the trainers worked with have ever attacked a handler.

The bears that no trainers worked with have ever attacked a handler.

The bears that the trainers worked with have ever attacked a handler.

- **Latin Square distribution:** Create *n* distinct experimental lists for an *n*-condition experiment such that each list has *exactly one condition from each item set*.

PART 1: The Experiment

Item 1:

No duck that the goose chased has ever returned to our pond

The duck that no goose chased has ever returned to our pond

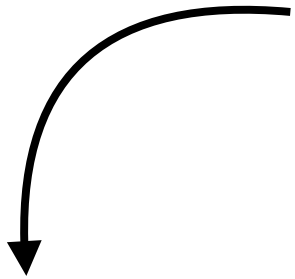
The duck that the goose chased has ever returned to our pond

Item 2:

No bills that the Democratic senators supported have ever become law.

The bills that no Democratic senators supported have ever become law.

The bills that the Democratic senators supported have ever become law.



Item 3:

No bears that the trainers worked with have ever attacked a handler.

The bears that no trainers worked with have ever attacked a handler.

The bears that the trainers worked with have ever attacked a handler.

LIST 1:

No duck that the goose chased has ever returned to our pond

The bills that no Democratic senators supported have ever become law.

The bears that the trainers worked with have ever attacked a handler.

LIST 2:

The duck that no goose chased has ever returned to our pond

The bills that the Democratic senators supported have ever become law.

No bears that the trainers worked with have ever attacked a handler.

PART 1: The Experiment

Summing up: key pieces to a judgment experiment are the following:

- **Experimental items.**
- **Filler items**.
- **Practice items**.
- **Instructions.**

Next up: Experimental software to facilitate administration of surveys and other experiments.

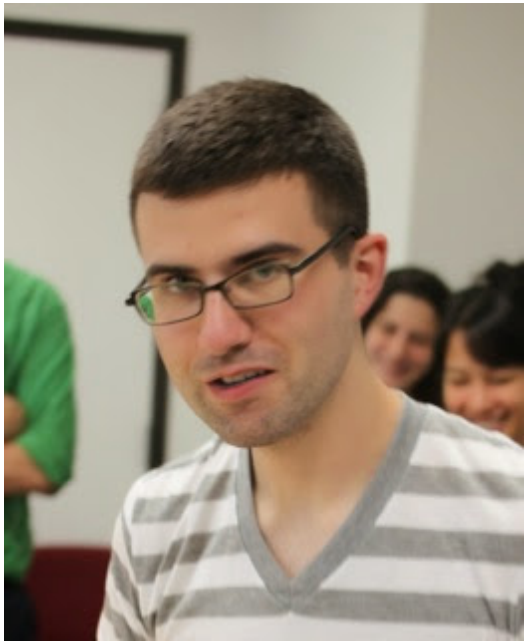
PART 2: The Ibex Farm

When deploying an experiment online, there are many pieces of experimental software that are designed to streamline or facilitate this process that you may find useful. We are going to focus on one:

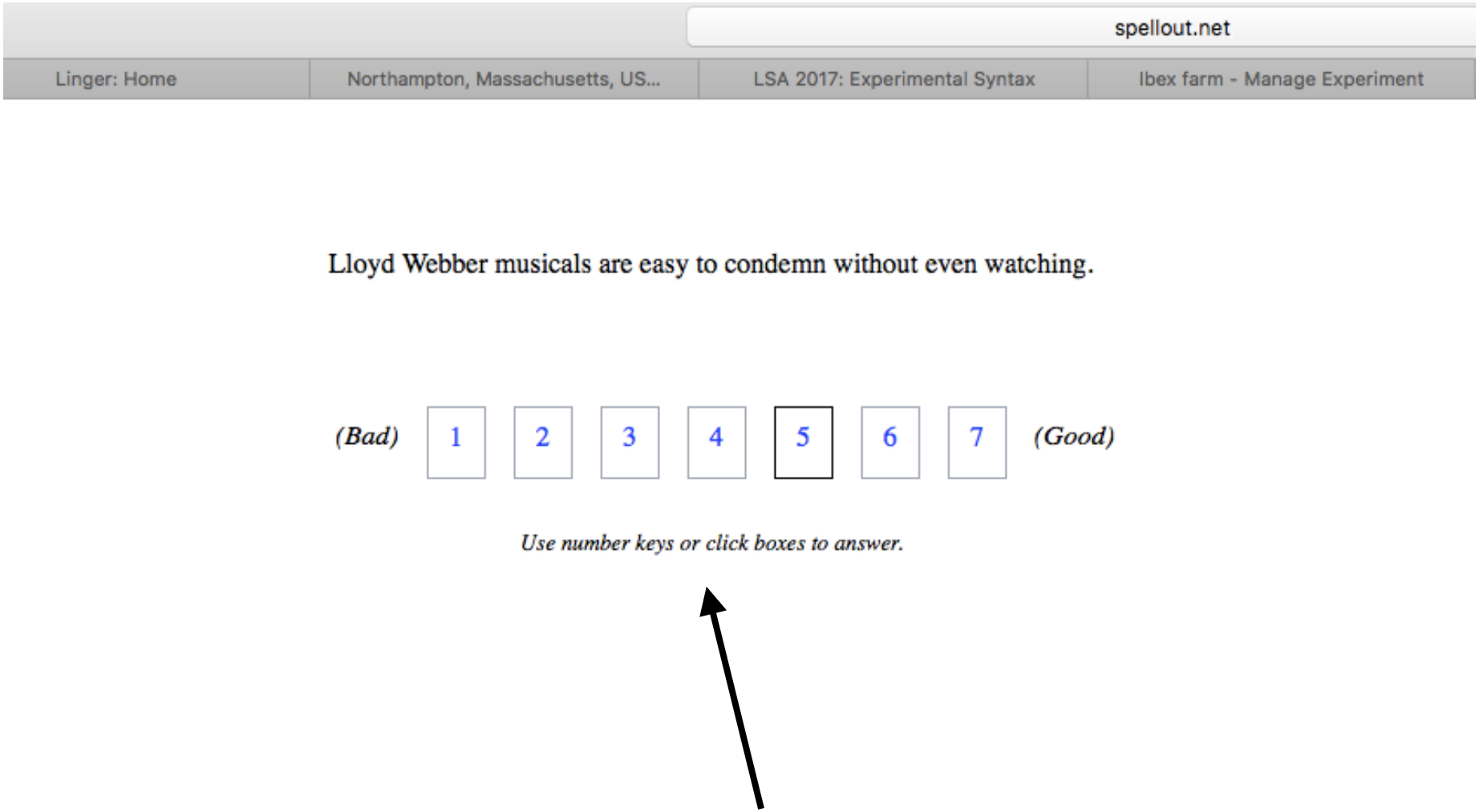
The Ibex Farm: **http://spellout.net/ibexfarm**.Javascript-based software for deploying web-based experiments. Can create a variety of judgment-based methodologies, in addition to other psycholinguistic paradigms (e.g. self-paced reading).

Pros: highly flexible platform, can be customized, can deploy experiments over the web;
Cons: requires familiarity with Javascript; can be difficult to debug.

Written by Alex Drummond, UCL:

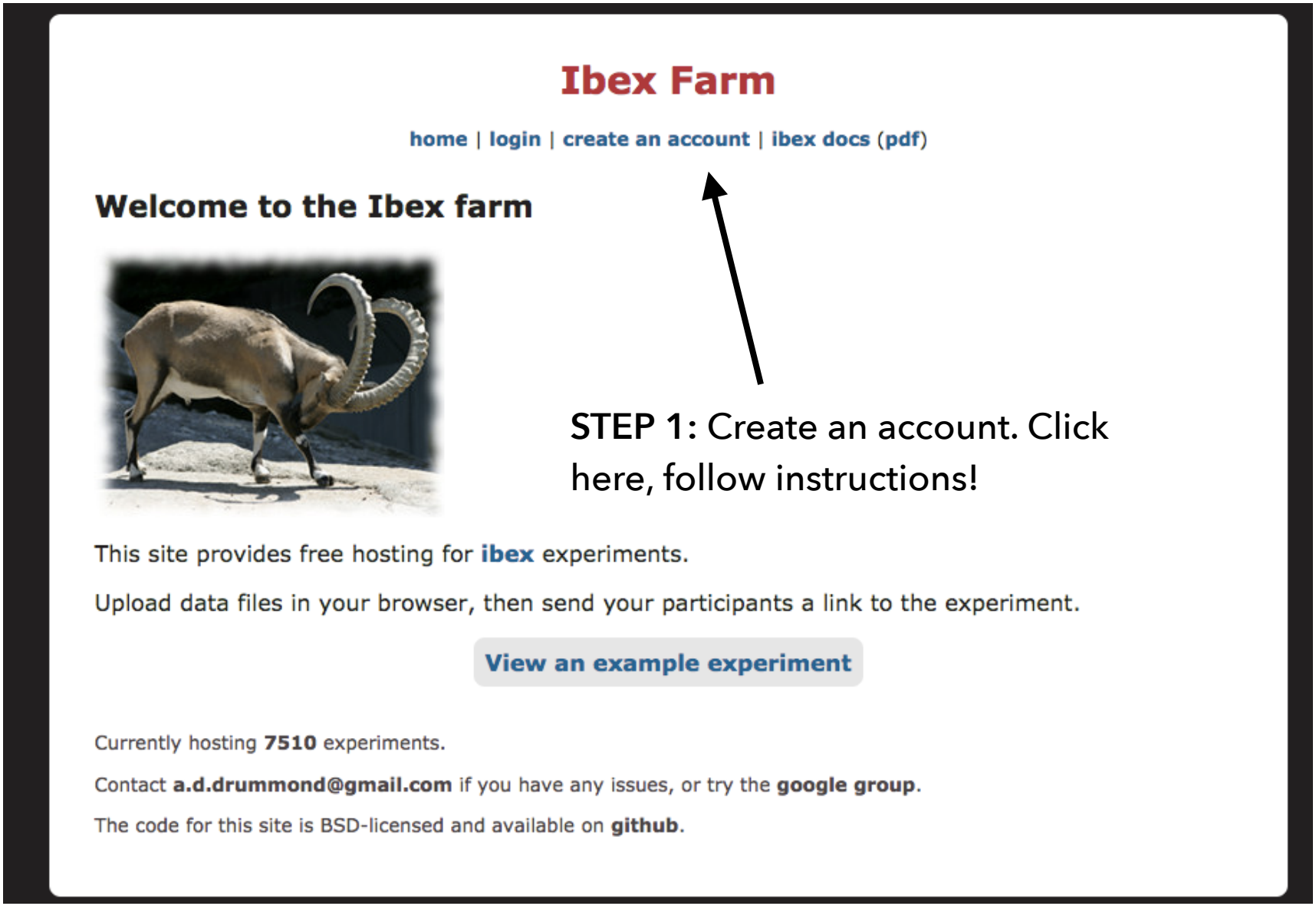


PART 2: The Ibex Farm



Easy to deploy judgment experiments; runs in a web-browser!

PART 2: The Ibex Farm



http://spellout.net/ibexfarm

PART 2: The Ibex Farm

Ibex Farm

home | ibex docs (pdf)

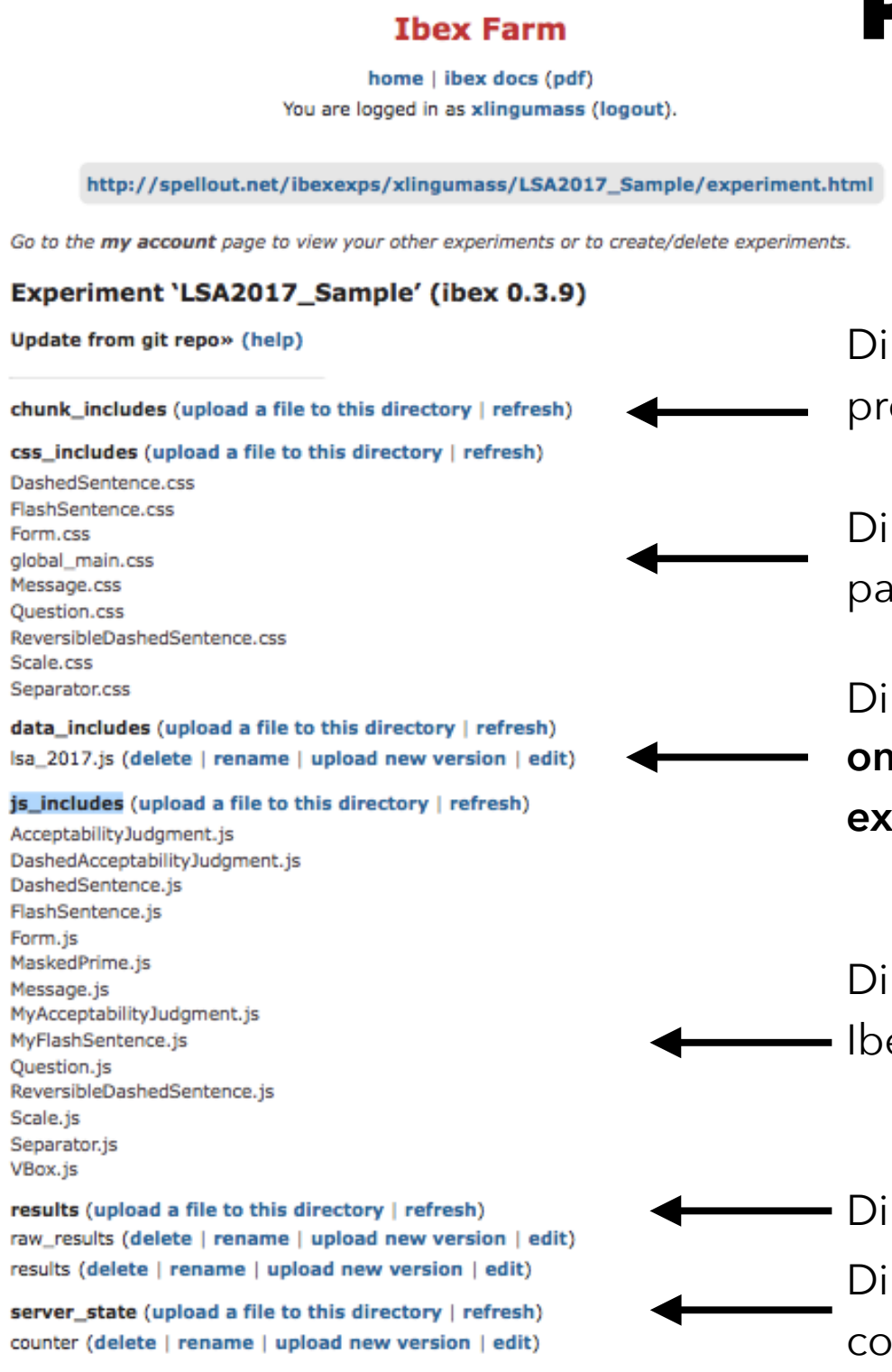
You are logged in as xlingumass (logout).

Experiments

- Agr_Pilot (ibex 0.3-beta16) (delete | rename)
- Anne-Test (ibex 0.3-beta19) (delete | rename)
- AppositiveNorm (ibex 0.3.7) (delete | rename)
- CAM-BRO (ibex 0.3.9) (delete | rename)
- CYCReboot (ibex 0.3.8) (delete | rename)
- Cheesecake (ibex 0.3.0) (delete | rename)
- Cheesecake-Redux (ibex 0.3.2) (delete | rename)
- Cheesecake-Redux-2 (ibex 0.3.2) (delete | rename)
- Cheesecake2 (ibex 0.3.0) (delete | rename)
- Cheesecake3 (ibex 0.3.0) (delete | rename)
- Cheesecake4 (ibex 0.3.0) (delete | rename)
- Cloze_Thesis (ibex 0.3.8) (delete | rename)
- CompA_Pilot (ibex 0.3-beta17) (delete | rename)
- Eaugis (ibex 0.3.7) (delete | rename)
- F1301 (ibex 0.3.2) (delete | rename)

List of experiments created under your account. Unlimited storage (or at least I haven't found the upper limit), and experiments stay indefinitely (but you should still back-up!)

PART 2: The Ibex Farm



← Link to your experiment

← Directory for *HTML* files you would like to present (not used in class example)

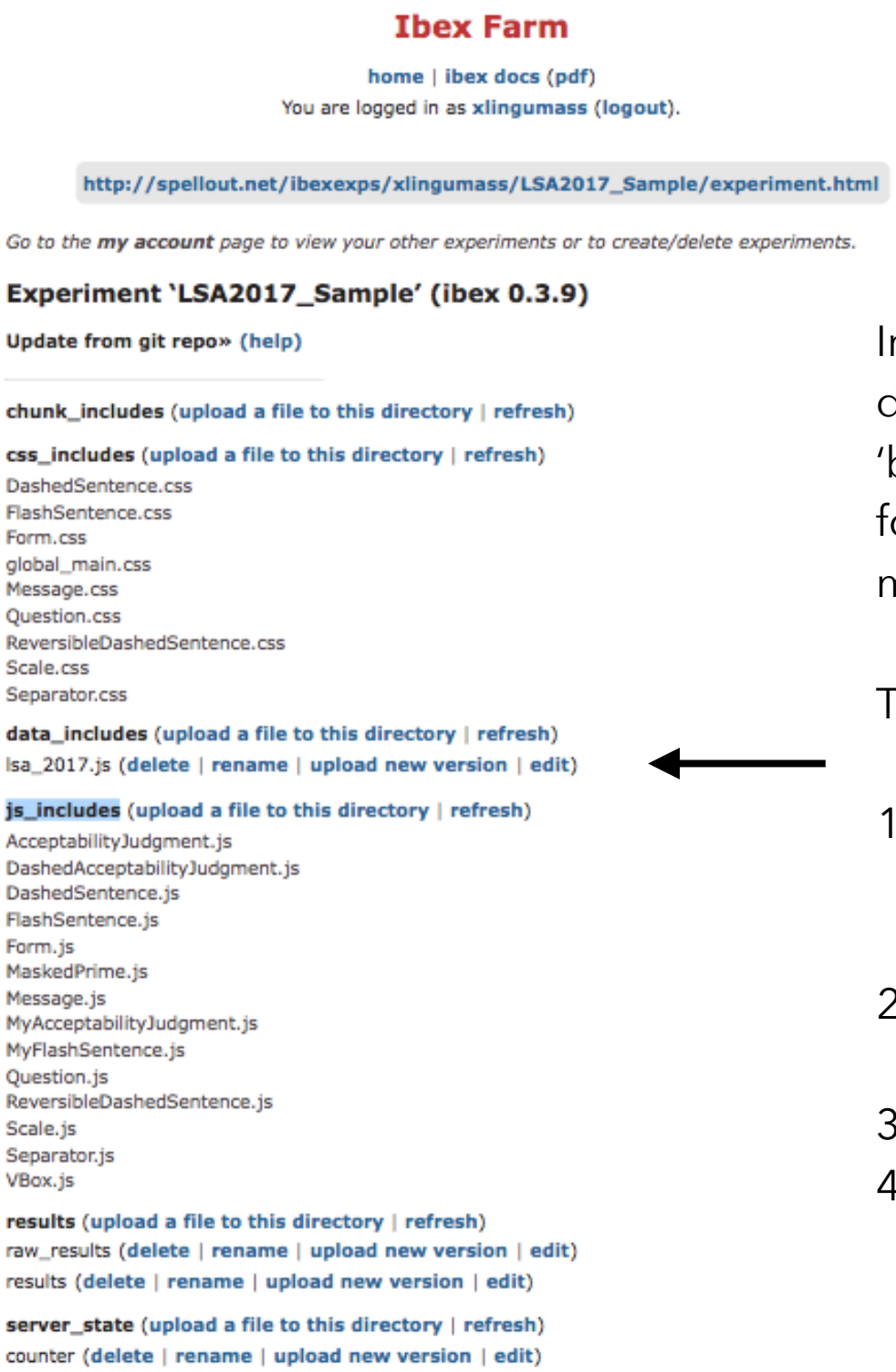
← Directory for *CSS* files that control presentation parameters (font, etc). Modify at your peril!

← Directory for main experiment file. **This is the only thing you need to edit for a basic experiment!**

← Directory with Javascript files that implement Ibex functions. Modify at your peril!

← Directory that stores results.

← Directory that stores current Latin Square counter.



In this tutorial, we will ***only*** set up the data_includes file; this implements the main 'body' of the experiment, and is all you need for a basic Ibex set-up (though you can do much more!).

The logic of this is the following:

1) Set up "trial" definitions: descriptions of each possible "trial" in the experiment. Do this for all sentences we will present.
2) Tell Ibex what you want a default "trial" to look like (how many response options, etc)
3) Write instructions/practice.
4) Tell Ibex presentation order.

PART 2: The Ibex Farm: Trial definition

We have coded up our experimental items in a text file or Excel spreadsheet and given them appropriate codes, e.g.:

```
npi.gram.01    No duck that the goose chased has ever returned to our pond.
```

To create an experimental trial from this sentence in Ibex, we have to embed it in the appropriate syntax:

```
[["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose
chased has ever returned to our pond."}]
```

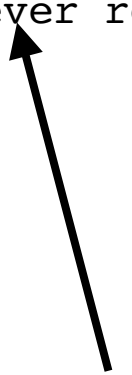
PART 2: The Ibex Farm: Trial definition

We have coded up our experimental items in a text file or Excel spreadsheet and given them appropriate codes, e.g.:

```
npi.gram.01    No duck that the goose chased has ever returned to our pond.
```

To create an experimental trial from this sentence in Ibex, we have to embed it in the appropriate syntax:

```
[["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose
chased has ever returned to our pond."}]
```



```
["CONDITION LABEL",ITEM SET #]
```

#1: Item label and item set #. For experimental items, we put the item label (condition label) **in quotes**, and in the first position in a mini-list here. After this, put the item set # (called group # in Ibex); **no quotes here.** This information allows Ibex to do automatic Latin Squaring.

PART 2: The Ibex Farm: Trial definition

We have coded up our experimental items in a text file or Excel spreadsheet and given them appropriate codes, e.g.:

```
npi.gram.01    No duck that the goose chased has ever returned to our pond.
```

To create an experimental trial from this sentence in Ibex, we have to embed it in the appropriate syntax:

```
[["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose
chased has ever returned to our pond."}]
```

"Controller name"

#2: Type of trial (a 'controller' in Ibex Lingo); **quotes needed.** Many options possible; we will only use 'AcceptabilityJudgment' here to do acceptability judgment trials.

PART 2: The Ibex Farm: Trial definition

We have coded up our experimental items in a text file or Excel spreadsheet and given them appropriate codes, e.g.:

`npi.gram.01  No duck that the goose chased has ever returned to our pond.`

To create an experimental trial from this sentence in Ibex, we have to embed it in the appropriate syntax:

```
[["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose
chased has ever returned to our pond."}]
```

#3: Arguments that define critical features of this acceptability judgment trial. **s** = 'sentence' argument. After **s:**, put the sentence for this trial **in quotes**.

```
{ … arguments … }

{ s: "SENTENCE IN QUOTES"}
```

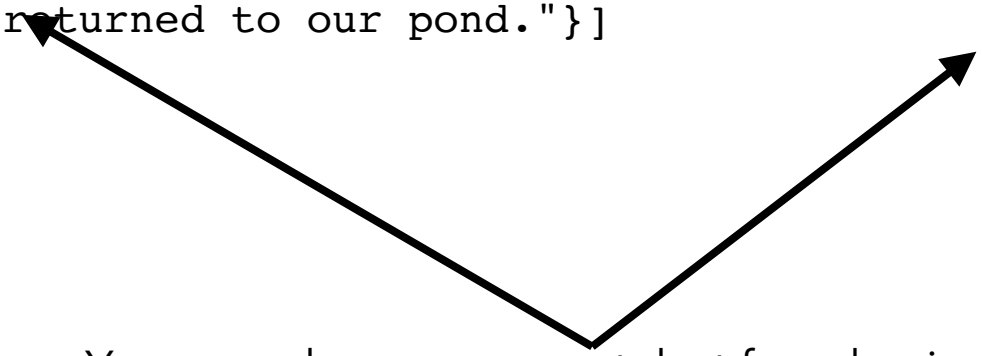
PART 2: The Ibex Farm: Trial definition

We have coded up our experimental items in a text file or Excel spreadsheet and given them appropriate codes, e.g.:

```
npi.gram.01    No duck that the goose chased has ever returned to our pond.
```

To create an experimental trial from this sentence in Ibex, we have to embed it in the appropriate syntax:

```
[["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose
chased has ever returned to our pond."}]
```



You may change any part; but for a basic acceptability judgment experiment we change *only* 's', and condition labels!

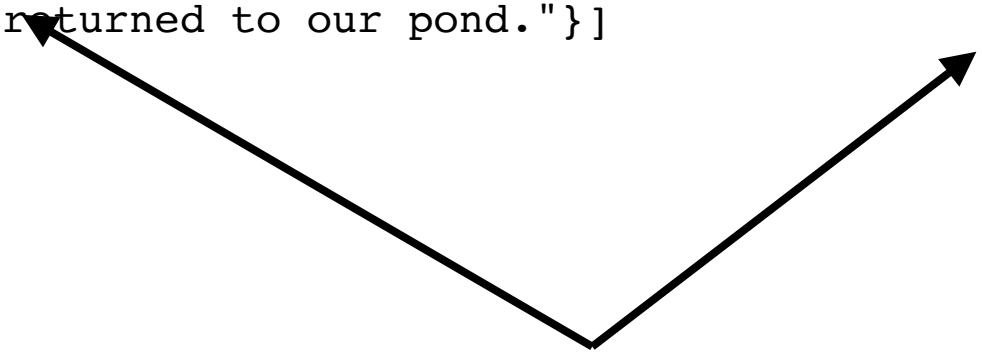
PART 2: The Ibex Farm: Trial definition

We have coded up our experimental items in a text file or Excel spreadsheet and given them appropriate codes, e.g.:

```
npi.gram.01    No duck that the goose chased has ever returned to our pond.
```

To create an experimental trial from this sentence in Ibex, we have to embed it in the appropriate syntax:

```
[["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose
chased has ever returned to our pond."}]
```

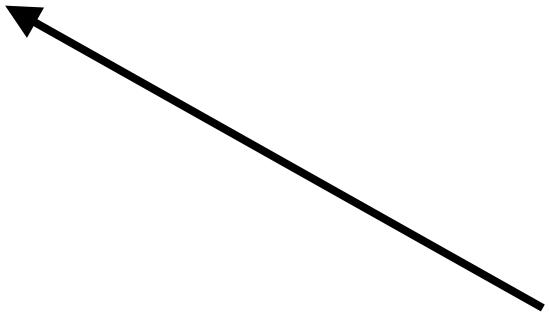


Other controllers include: “Question” (for posing a question and listing answers to choose from), “DashedAcceptabilityJudgment” (for speeded acceptability judgments), “DashedSentence” (for self-paced reading experiments) … see documentation for others!

PART 2: The Ibex Farm: Trial definition

If you have fillers that will not be Latin Squared, then you do not need to define an item set. Trial definitions are therefore simpler:

```
    [“Filler.Good.01”, "AcceptabilityJudgment", {s: "Mike prefers
tennis because it’s more exciting than squash."}]
```



Just a simple label in quotes, now!

PART 2: The Ibex Farm: The ‘items’ variable

Once you’ve defined all your trial variables, you have to collect them up into a single master list called the ‘items’ variable in Ibex Lingo. To define an items variable, start by typing…

```
var items = [
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all your trial variables, you have to collect them up into a single master list called the 'items' variable in Ibex Lingo. To define an items variable, start by typing… and adding in each trial variable, **followed by a comma** …

```
var items = [

    [["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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."}],
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all your trial variables, you have to collect them up into a single master list called the 'items' variable in Ibex Lingo. To define an items variable, start by typing… and adding in each trial variable, **followed by a comma** … until you get to the last trial in the list, which you add **but do not follow with a comma…**

```
var items = [

    [[“npi.gram”,1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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-GOOD-01”, "AcceptabilityJudgment", {s: "Mike prefers tennis because it's more exciting than squash."}]
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all your trial variables, you have to collect them up into a single master list called the 'items' variable in Ibex Lingo. To define an items variable, start by typing… and adding in each trial variable, **followed by a comma** … until you get to the last trial in the list, which you add **but do not follow with a comma…** and close it out with an ending bracket and semi-colon.

```
var items = [

    [[“npi.gram”,1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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-GOOD-01”, "AcceptabilityJudgment", {s: "Mike prefers tennis because it’s more exciting than squash."}]

];
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all your trial variables, you have to collect them up into a single master list called the 'items' variable in Ibex Lingo. To define an items variable, start by typing… and adding in each trial variable, **followed by a comma** … until you get to the last trial in the list, which you add **but do not follow with a comma…** and close it out with an ending bracket and semi-colon.

```
var items = [

    [[“npi.gram”,1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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-GOOD-01”, "AcceptabilityJudgment", {s: "Mike prefers tennis because it’s more exciting than squash."}]

];
```

NOTE: If you are familiar with Javascript, you will recognize that the items variable is just a particularly complicated instance of a Javascript **array.** You will see this structure throughout the Ibex file. Arrays are sequential collections of items, and have the general form:

```
[item1, item2, item3, …, item-n]

var array = [item1, item2, item3, …, item-n];
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all your trial variables, you have to collect them up into a single master list called the 'items' variable in Ibex Lingo. To define an items variable, start by typing… and adding in each trial variable, **followed by a comma** … until you get to the last trial in the list, which you add **but do not follow with a comma…** and close it out with an ending bracket and semi-colon.

```
var items = [

    [[“npi.gram”,1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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-GOOD-01”, "AcceptabilityJudgment", {s: "Mike prefers tennis because it’s more exciting than squash."}]

];
```

TROUBLESHOOTING TIP: By far the most common error you will see when trying to set up a new Ibex experiment is 'items variable not defined'. This means there is a syntax error **somewhere** in your file. First step in troubleshooting is to very carefully check that you have stuck to all syntax conventions above! If you are still having a hard time, options include:

- Open 'Javascript Console' in Chrome (View > Developer > Javascript Console); this can help you identify which line in your file throws the error.
- Use a text editor with text highlighting; this makes it easier to spot unbalanced quotes / brackets.

PART 2: The Ibex Farm: The 'items' variable

IC_Roads_code1_1_fullscreen.js | Sample_Items.txt | sample.js

```
var items = [

    [["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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.Good.01", "AcceptabilityJudgment", {s: "Mike prefers tennis because it's more exciting than squash."}]

];
```

- I like the **SUBLIME** text editor. When a file has a .js extension, it will color code the syntax for you. This makes it easy to see common syntax errors (omitting quotes, failing to put in close brackets, etc.).

IC_Roads_code1_1_fullscreen.js | Sample_Items.txt | sample.js

```
var items = [

    [["npi.gram",1], "AcceptabilityJudgment", {s: No duck that the goose chased has ever returned to our pond.}],
    [["npi.illus",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.Good.01", "AcceptabilityJudgment", {s: "Mike prefers tennis because it's more exciting than squash."}]

];
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all experimental trials and fillers, we put anything else we want to present inside the items variable.

A 'separator' controller… just a screen people see between trials.

```
var items = [

    ["sep", "Separator", { }],
    ["setcounter", "__SetCounter__", { }],

    ["introduction", Message, {consentRequired: false,
                    html: ["div",
                          ["p", "Welcome! Here are some instructions to the experiment."],
                          ["p", "Hope you have a great time! It'll be a blast."]
                             ]}],


    [["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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.Good.01", "AcceptabilityJudgment", {s: "Mike prefers tennis because it's more exciting than squash."}]
    ];
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all experimental trials and fillers, we put anything else we want to present inside the items variable.

A 'setcounter' controller… this is used to change the counter on the server, which determines which Latin Square list participants see.

```
var items = [

    ["sep", "Separator", { }],
    ["setcounter", "__SetCounter__", { }],

    ["introduction", Message, {consentRequired: false,
                    html: ["div",
                          ["p", "Welcome! Here are some instructions to the experiment."],
                          ["p", "Hope you have a great time! It'll be a blast."]
                             ]}],


    [["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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.Good.01", "AcceptabilityJudgment", {s: "Mike prefers tennis because it's more exciting than squash."}]
    ];
```

PART 2: The Ibex Farm: The 'items' variable

Once you've defined all experimental trials and fillers, we put anything else we want to present inside the items variable.

A 'Message' controller… this is just an untimed, unstructured message we can give participants. Here, I use it to present some instructions.

```
var items = [

    ["sep", "Separator", { }],
    ["setcounter", "__SetCounter__", { }],

    ["introduction", Message, {consentRequired: false,
                    html: ["div",
                          ["p", "Welcome! Here are some instructions to the experiment."],
                          ["p", "Hope you have a great time! It'll be a blast."]
                             ]}],


    [["npi.gram",1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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-GOOD-01", "AcceptabilityJudgment", {s: "Mike prefers tennis because it's more exciting than squash."}]
    ];
```

PART 2: The Ibex Farm: Controllers

Controllers are pre-defined, common types of experimental trial formats used in Ibex Farm. Common ones and their behavior include:

- **`AcceptabilityJudgment`** : Presents a sentence along with a user-defined rating scale. A single trial records i) response and ii) reaction time (ms) in results file.
- **`DashedAcceptabilityJudgment`** : Presents a sentence one 'chunk' at a time, speeded or user-paced, and presents a a user-defined rating scale after a trial. A single trial records i) response and ii) reaction time (ms) in results file. If it is user paced, a single trial records reaction time (ms) for each word/chunk in results file.
- **`Question`** : Presents a question along with a user-defined set of response options. A single trial records i) response and ii) reaction time (ms) in results file.
- **`DashedSentence`** : Presents a sentence one 'chunk' at a time, speeded or user-paced. If it is user paced, a single trial records reaction time (ms) for each word/chunk in results file.
- **`Message`** : Presents an HTML file. Does not record anything in results file.
- **`Form`** : Presents an HTML file. Records anything collected in HTML file in results file.
- **`Separator`** : Presents a screen between experimental trials

PART 2: The Ibex Farm: Controllers

Controllers can also be custom created if you know how to code in Javascript. For example, **Carolyn Anderson (UMass)** created this useful controller:

- **`ComicCaption`** : Presents a picture and sentence along with a user-defined rating scale. A single trial records i) response and ii) reaction time (ms) in results file. **https://github.com/canders1/ibex_pictures**

Carolyn Anderson (UMass)

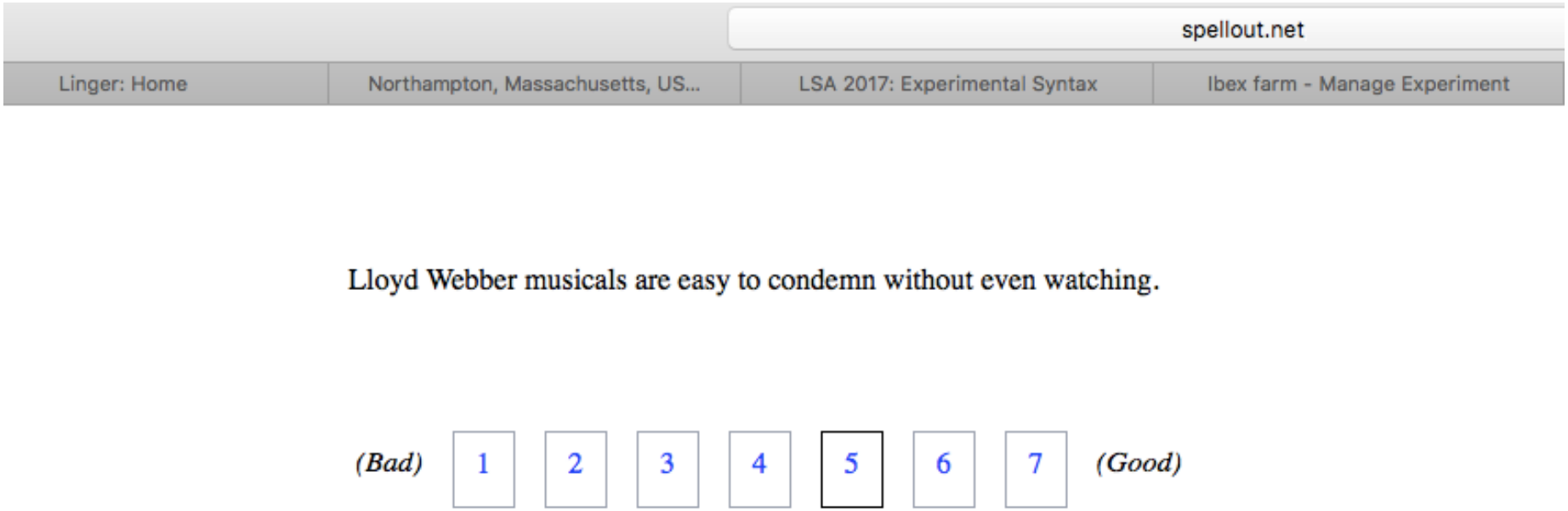
PART 2: The Ibex Farm: Controllers

Each controller has different properties or **arguments**; you can either set a given trial's properties item by item (like the 's' argument), or you can set defaults at the top of the file. These defaults will be shared by all instances of a given controller in your experiment. Like so:

```
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)"
];
```

You can edit any of these options; check documentation for further options you can specify!

The Ibex Farm




```
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)"
];
```

PART 2: The Ibex Farm: Controllers

Each controller has different properties or **arguments**; you can either set a given trial's properties item by item (like the 's' argument), or you can set defaults at the top of the file. These defaults will be shared by all instances of a given controller in your experiment. Like so:

```
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)"
];

var items = [
```

This item will be (exceptionally) presented with a 5 point scale, because the default argument has been overridden.

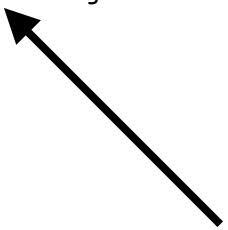
```
…

    [["npi.illus",1], "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our
pond.", as: ["1", "2", "3", "4", "5"]}],
```

PART 2: The Ibex Farm: Controllers

One important type of controller is the **Message** or **Form** controllers, used to display HTML messages. This is useful for instructions, practice.

```
["introduction", Message, {consentRequired: false,
                    html: ["div",
                          ["p", “Welcome! Here are some instructions to the experiment.”],
                          ["p", “Hope you have a great time! It’ll be a blast.”]
                             ]}],
```



In this Message trial, the HTML is defined in situ.

PART 2: The Ibex Farm: Controllers

One important type of controller is the **Message** or **Form** controllers, used to display HTML messages. This is useful for instructions, practice.

```
["introduction", Message, {consentRequired: false,
                    html: ["div",
                          ["p", “Welcome! Here are some instructions to the experiment.”],
                          ["p", “Hope you have a great time! It’ll be a blast.”]
                             ]}],
```

In this Message trial, we simply point to another HTML file. Ibex will search the current experiment’s ‘chunk includes’ folder and display the associated HTML.

```
["introduction", Message, {consentRequired: false,
                    html: {include: ‘intro.html’}}],
```

PART 2: The Ibex Farm: Controllers

A single experimental trial may consist of multiple, sequentially presented controllers. For example, we may want to ask a comprehension question after an acceptability judgment. Here's a basic trial definition:

```
    [["npi.illus",1], "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}],
```

Here's the same trial, but now with a following question:

```
    [["npi.illus",1], "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}, "Question", {q: "What did that sentence mean?", as: ["No idea.","No goose returned.", "The duck returned."]}],
```

PART 2: The Ibex Farm: ShuffleSequence

The last thing you have to set is the shuffleSequence. Think of this as an instruction to IbexFarm to tell it how to present the trials you defined in the 'items' variable:

First, we create a sequence using 'seq' …

```
var shuffleSequence = seq("setcounter",
  "introduction",
   sepWith("sep", rshuffle(startsWith("npi"), startsWith("F"))))
```

PART 2: The Ibex Farm: ShuffleSequence

The last thing you have to set is the shuffleSequence. Think of this as an instruction to IbexFarm to tell it how to present the trials you defined in the 'items' variable:

```
var shuffleSequence = seq("setcounter",
  "introduction",
   sepWith("sep", rshuffle(startsWith("npi"), startsWith("F"))))
```

… then we present the main body of the experiment.

sepWith(X,Y): "Separate all items in 'Y' with 'X'"; here, it puts a separator between all trials.
rshuffle(X): "Pseudorandomly shuffle X"
startsWith(X): "All trial definitions that begin with 'X'.

rshuffle's default behavior is to pseudorandomize, putting as much space as possible between classes of items defined by 'startsWith'. In this case, it will alternate fillers and NPI trials.

PART 2: The Ibex Farm: ShuffleSequence

An aside on (pseudo-)randomization:

Randomization is very important for all aspects of the experimental process.

Participants are randomly assigned to experimental Latin Square lists; this ensures that there are no systematic confounds between list identity and potentially important 'third factor' variables (such as time of day, gender, etc.).

Experimental trials are typically presented in a pseudo-random fashion, ideally randomized anew for each participant. Across the experiment, this guarantees no systematic confounds from order of presentation (e.g. priming effects that might occur from adjacent items).

One real strength of Ibex—and similar experimental software—is that this pseudo randomization is the default. It is the default for good reason: beware if you choose to turn off these functionalities!

PART 2: The Ibex Farm: Latin Squares

If you code your experiment in IbexFarm, you do not need to divide your experimental items into Latin Squared lists by hand. Instead, IbexFarm will automatically parse item sets into *n* lists, where *n* is the number of conditions you have.

It does this by storing a 'counter' on the server that tracks what list to present. When the counter is **0** , the first list will be selected. Following our Latin Square regime, the first list consists of:

```
    [[“npi.gram”,1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",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.”}],

    [[“npi.gram”,2], "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}],
    [["npi.illus",2], "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}],
    [["npi.ungram",2], "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}],
```

When the counter is 0, Ibex will *only* present the bolded items in the experiment.

PART 2: The Ibex Farm: Latin Squares

If you code your experiment in IbexFarm, you do not need to divide your experimental items into Latin Squared lists by hand. Instead, IbexFarm will automatically parse item sets into *n* lists, where *n* is the number of conditions you have.

It does this by storing a 'counter' on the server that tracks what list to present. When the counter is **1** , the first list will be selected. Following our Latin Square regime, the first list consists of:

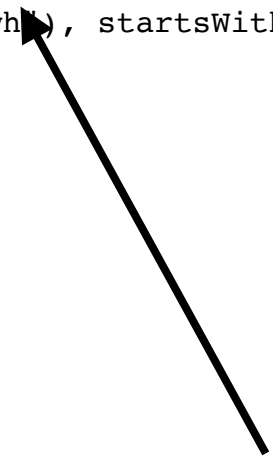
```
    [[“npi.gram”,1], "AcceptabilityJudgment", {s: "No duck that the goose chased has ever returned to our pond."}],
    [["npi.illus",1], "AcceptabilityJudgment", {s: "The duck that no goose chased has ever returned to our pond."}],
    [["npi.ungram",1], "AcceptabilityJudgment", {s: "The duck that the goose chased has ever returned to our pond.”}],

    [[“npi.gram”,2], "AcceptabilityJudgment", {s: "No bill that the Democratic senators supported has ever become law."}],
    [["npi.illus",2], "AcceptabilityJudgment", {s: "The bill that no Democratic senators supported has ever become law."}],
    [["npi.ungram",2], "AcceptabilityJudgment", {s: "The bill that the Democratic senators supported has ever become law."}],
```

When the counter is 1, Ibex will *only* present the bolded items in the experiment.

PART 2: The Ibex Farm: Latin Squares

```
var shuffleSequence = seq("setcounter",
  "introduction",
 sepWith("sep", rshuffle(startsWith("wh"), startsWith("F"))))
```

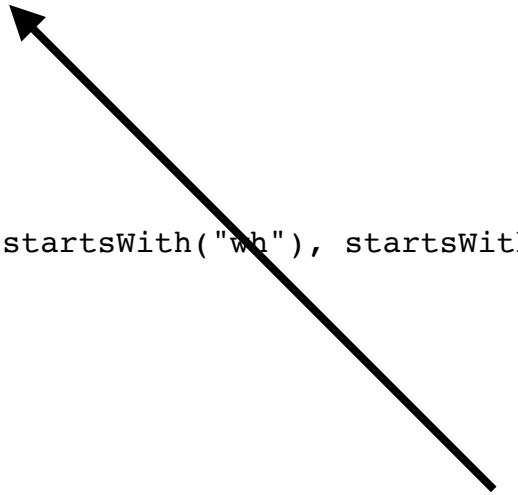


‘Setcounter’ will increment counter by 1 every time it is encountered (i.e. it will move to next list). So every person who starts the experiment will see a new list!

PART 2: The Ibex Farm: Latin Squares

```
var counterOverride = 2

var shuffleSequence = seq(
  "introduction",
  "prepractice",
  "practice",
  "getready",
  sepWith("sep", rshuffle(startsWith("wh"), startsWith("F"))))
```



An alternative to using set counter is to set the counterOverride variable manually. When this is set, all participants will see the same list (**remember, 2 means list 3!**)

PART 2: The Ibex Farm: The experiment

To recap, making an experiment in Ibex involves the following three pieces:

1) Define experimental trials, combine with fillers, practice into **items variable**.
2) Define **controller defaults**.
3) Define **shuffleSequence**

That should give Ibex all the information necessary to run your experiment! Save this all in a **single file with .js extension**, and upload it to the **data_includes** folder in your Ibex console.

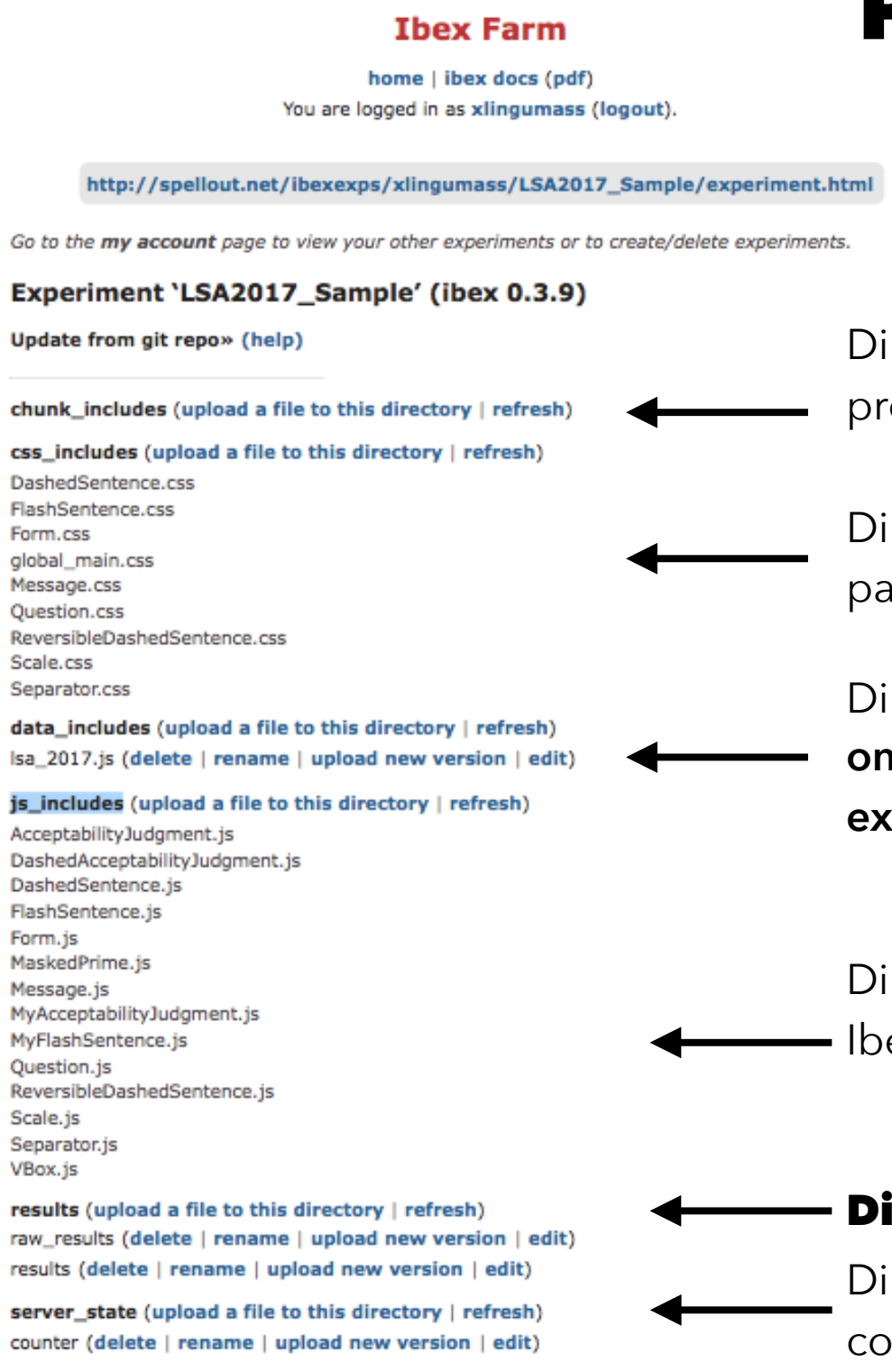
Then click the experiment… and you should be good to go.

PART 2: The Ibex Farm: The experiment

Now it's your turn.

- **Access LSA_Minicourse_NPIs.js from course dropbox.**
- **Open it in your text editor.**
- **Add a Form controller to present 'intro.html' to collect demographic data.**
- **Add two new experimental NPI items that you create.**
- **Add a comprehension question at the end of one filler item. It can be anything you want.**
- **Find the bug in the code. There is at least one bug that will prevent the experiment from running. Check your file carefully!**
- **Upload it, and run it.**

PART 2: The Ibex Farm



Link to your experiment

Directory for *HTML* files you would like to present (not used in class example)

Directory for *CSS* files that control presentation parameters (font, etc). Modify at your peril!

Directory for main experiment file. **This is the only thing you need to edit for a basic experiment!**

Directory with Javascript files that implement Ibex functions. Modify at your peril!

Directory that stores results.

Directory that stores current Latin Square counter.

PART 2: The Ibex Farm: Results

results.csv

```
#
# Results on Tuesday July 18 2017 19:38:31 UTC.
# USER AGENT: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/601.4.4 (KHTML, like Gecko) Version/9.0.3 Safari/601.4.4
# Design number was non-random = 20
#
# The lines below this comment are in groups of 2.
# The formats of the lines in each of these groups are as follows:
#
# Line 1:
#     Col. 1: Time results were received.
#     Col. 2: MD5 hash of participant's IP address.
#     Col. 3: Controller name.
#     Col. 4: Item number.
#     Col. 5: Element number.
#     Col. 6: Type.
#     Col. 7: Group.
#     Col. 8: Sentence (or sentence MD5).
# Line 2:
#     Col. 1: Time results were received.
#     Col. 2: MD5 hash of participant's IP address.
#     Col. 3: Controller name.
#     Col. 4: Item number.
#     Col. 5: Element number.
#     Col. 6: Type.
#     Col. 7: Group.
#     Col. 8: Question (NULL if none).
#     Col. 9: Answer.
#     Col. 10: Whether or not answer was correct (NULL if N/A).
#     Col. 11: Time taken to answer.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,3,0,unannounced.practice,NULL,She was the winner.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,3,0,unannounced.practice,NULL,NULL,7,NULL,1012
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,4,0,unannounced.practice,NULL,Promise to wash%2C Neal did the car.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,4,0,unannounced.practice,NULL,NULL,2,NULL,1112
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,5,0,unannounced.practice,NULL,The brother and sister that were playing all the time had to be sent to bed.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,5,0,unannounced.practice,NULL,NULL,5,NULL,2619
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,6,0,unannounced.practice,NULL,The children were cared for by the adults and the teenagers.
```

‘Item’ is position in ‘items’ list; ‘Element’ is position in Controllers for that trial; ‘Type’ is the label associated with that trial’; ‘Group’ is what we’ve been calling ‘item set’

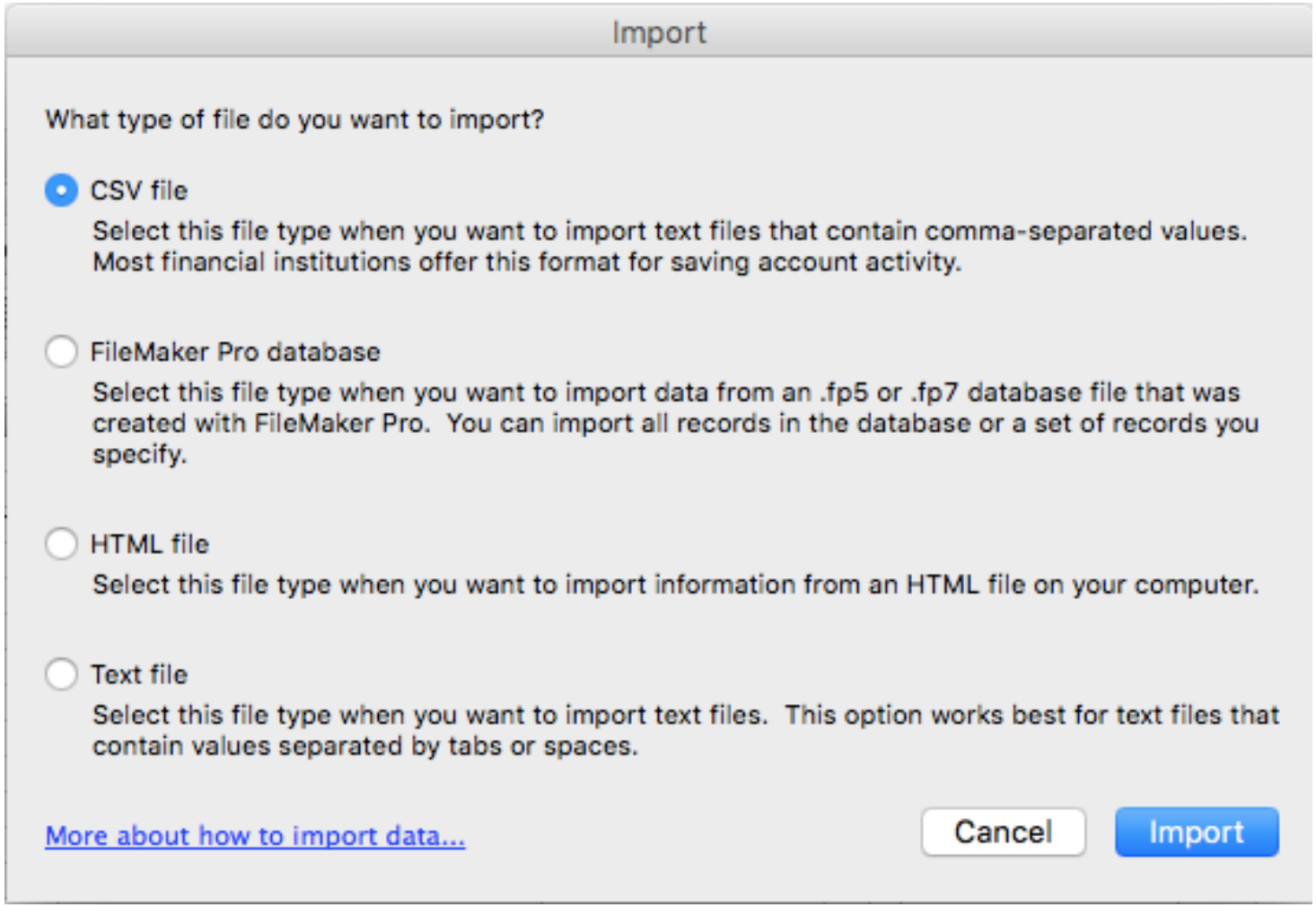
PART 2: The Ibex Farm: Results

results.csv

```
#
# Results on Tuesday July 18 2017 19:38:31 UTC.
# USER AGENT: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/601.4.4 (KHTML, like Gecko) Version/9.0.3 Safari/601.4.4
# Design number was non-random = 20
#
# The lines below this comment are in groups of 2.
# The formats of the lines in each of these groups are as follows:
#
# Line 1:
#     Col. 1: Time results were received.
#     Col. 2: MD5 hash of participant's IP address.
#     Col. 3: Controller name.
#     Col. 4: Item number.
#     Col. 5: Element number.
#     Col. 6: Type.
#     Col. 7: Group.
#     Col. 8: Sentence (or sentence MD5).
# Line 2:
#     Col. 1: Time results were received.
#     Col. 2: MD5 hash of participant's IP address.
#     Col. 3: Controller name.
#     Col. 4: Item number.
#     Col. 5: Element number.
#     Col. 6: Type.
#     Col. 7: Group.
#     Col. 8: Question (NULL if none).
#     Col. 9: Answer.
#     Col. 10: Whether or not answer was correct (NULL if N/A).
#     Col. 11: Time taken to answer.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,3,0,unannounced.practice,NULL,She was the winner.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,3,0,unannounced.practice,NULL,NULL,7,NULL,1012
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,4,0,unannounced.practice,NULL,Promise to wash%2C Neal did the car.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,4,0,unannounced.practice,NULL,NULL,2,NULL,1112
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,5,0,unannounced.practice,NULL,The brother and sister that were playing all the time had to be sent to bed.
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,5,0,unannounced.practice,NULL,NULL,5,NULL,2619
1500406711,8a5e67b77b5907b78a8d61ad8d616fe9,AcceptabilityJudgment,6,0,unannounced.practice,NULL,The children were cared for by the adults and the teenagers.
```

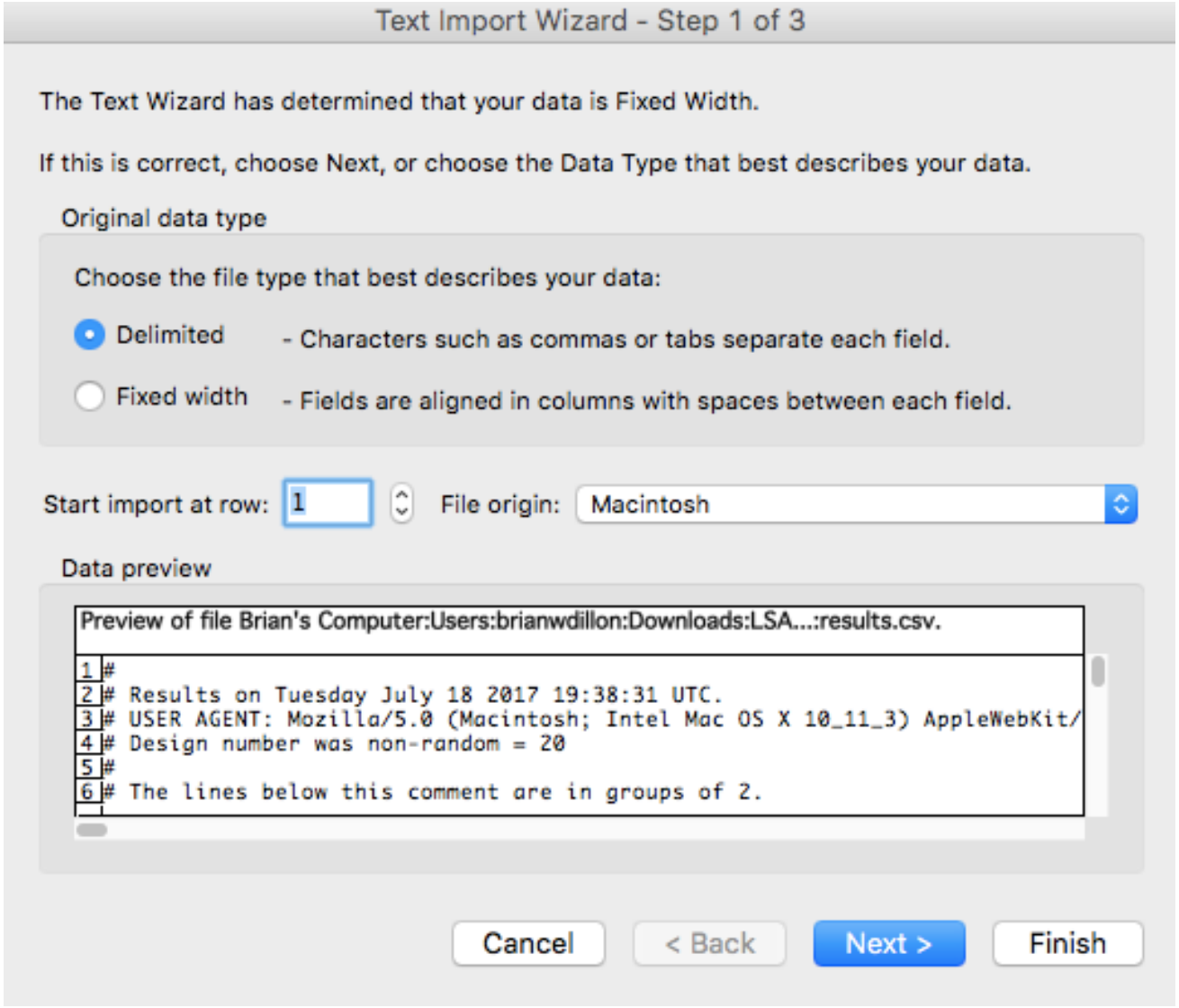
‘Answer’ is response on 7 point scale

PART 2: The Ibex Farm: Results

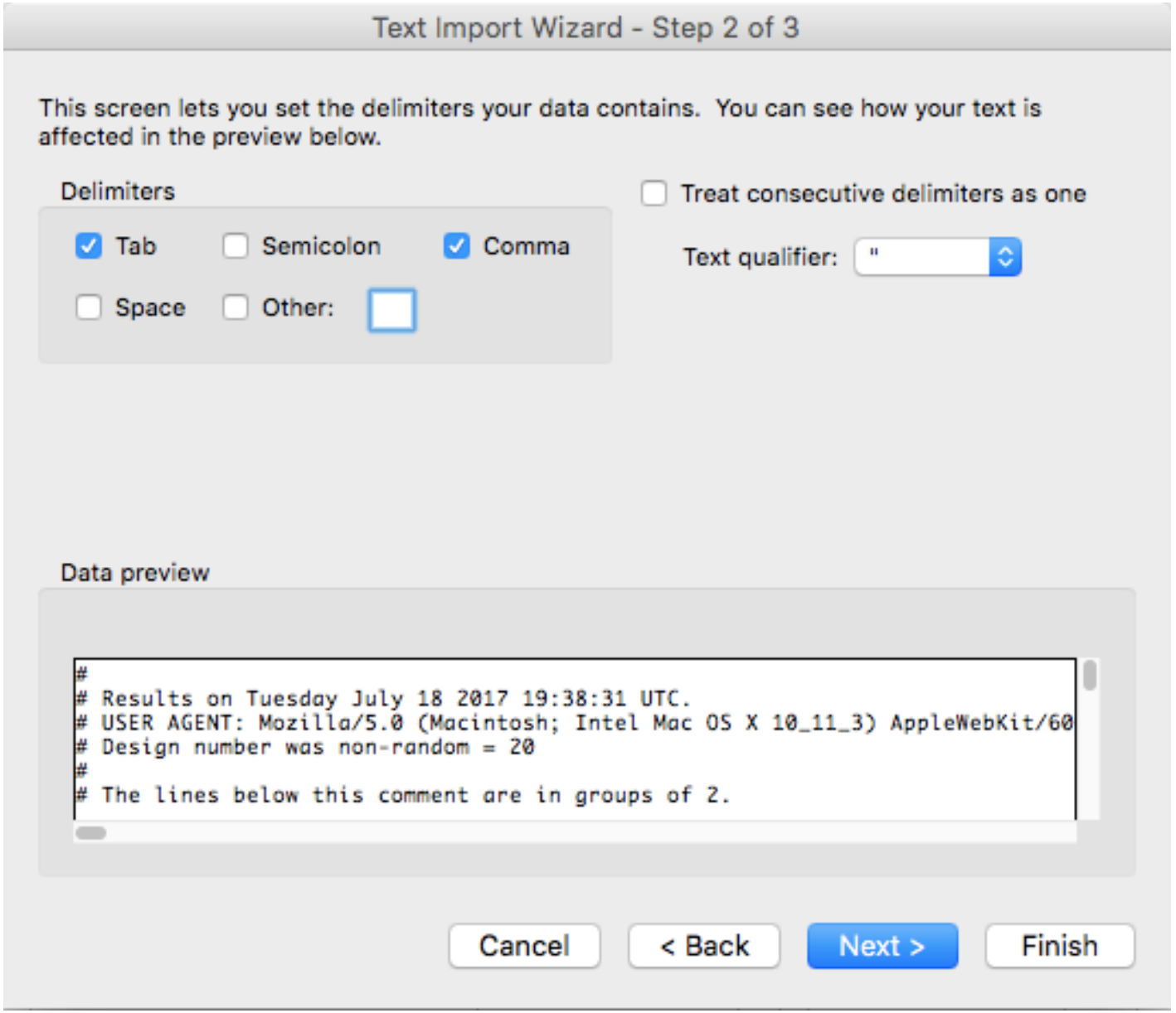


Ibex files are essentially CSV (comma-separated values) files with comments interspersed. They can be read directly into **R** or **Excel.** Here's how to read them into **Excel**.

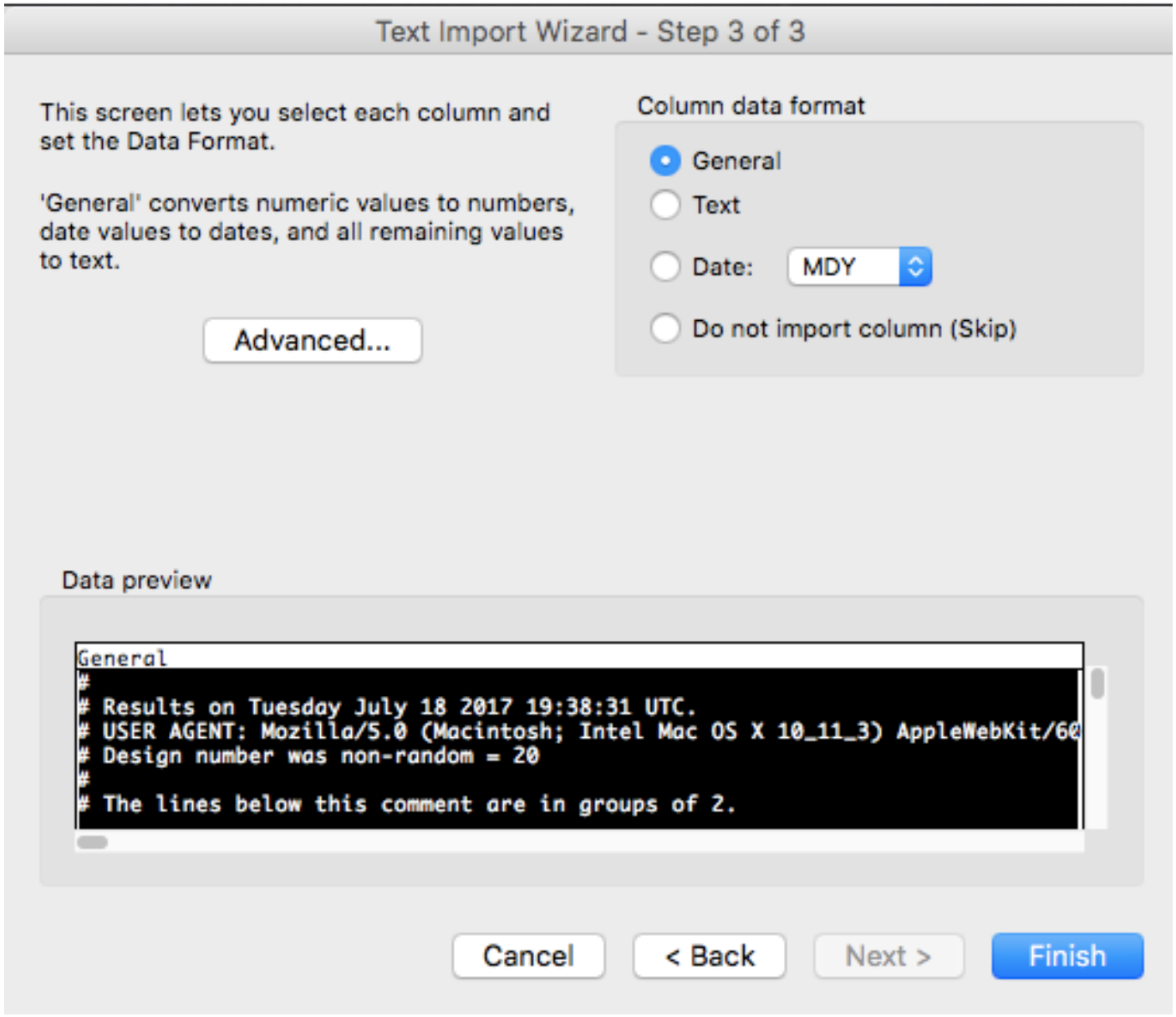
PART 2: The Ibex Farm: Results



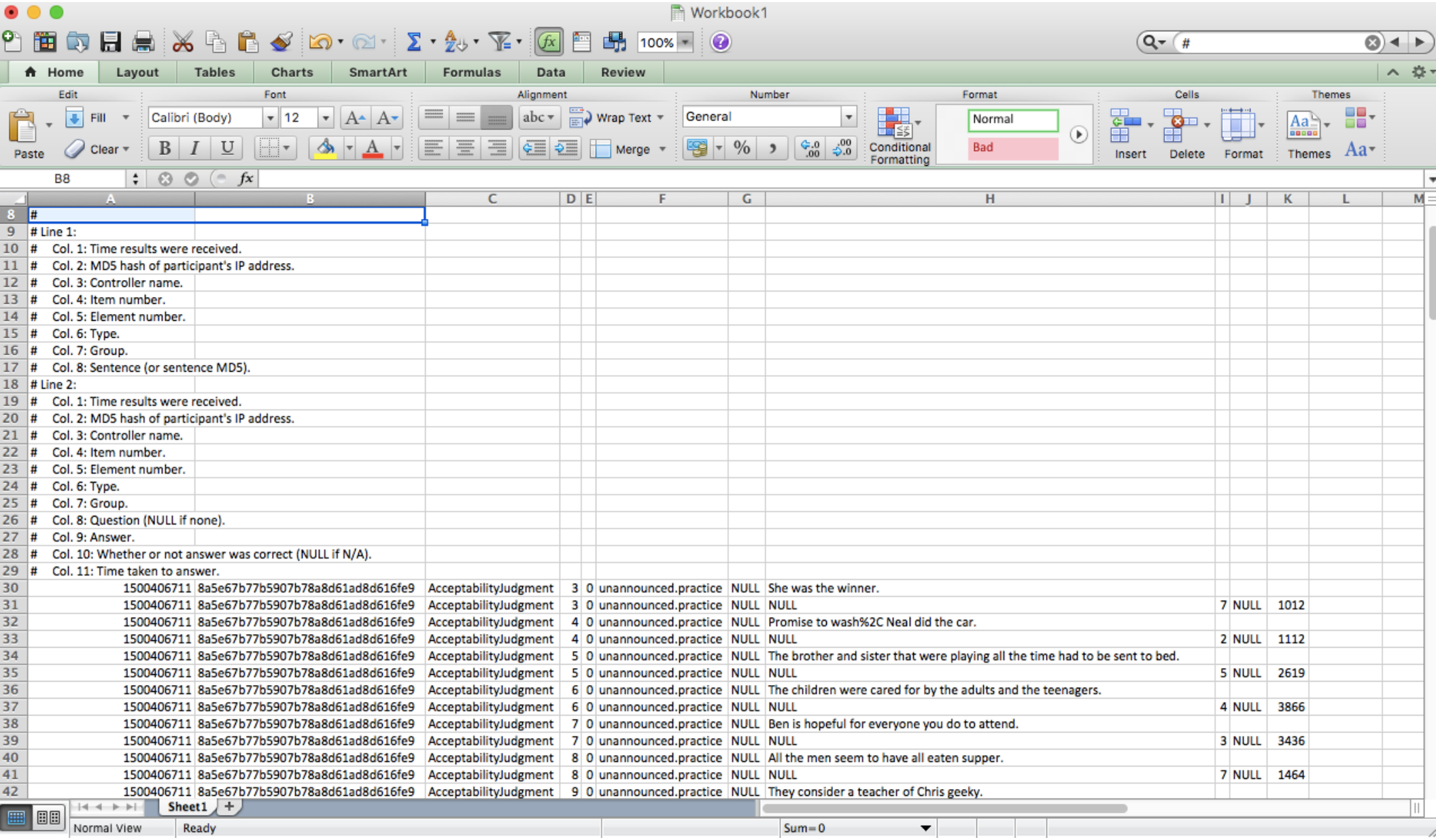
PART 2: The Ibex Farm: Results



PART 2: The Ibex Farm: Results



PART 2: The Ibex Farm: Results



	A	B	C	D	E	F	G	H	I	J	K	L	M
8	#												
9	# Line 1:												
10	# Col. 1: Time results were received.												
11	# Col. 2: MD5 hash of participant's IP address.												
12	# Col. 3: Controller name.												
13	# Col. 4: Item number.												
14	# Col. 5: Element number.												
15	# Col. 6: Type.												
16	# Col. 7: Group.												
17	# Col. 8: Sentence (or sentence MD5).												
18	# Line 2:												
19	# Col. 1: Time results were received.												
20	# Col. 2: MD5 hash of participant's IP address.												
21	# Col. 3: Controller name.												
22	# Col. 4: Item number.												
23	# Col. 5: Element number.												
24	# Col. 6: Type.												
25	# Col. 7: Group.												
26	# Col. 8: Question (NULL if none).												
27	# Col. 9: Answer.												
28	# Col. 10: Whether or not answer was correct (NULL if N/A).												
29	# Col. 11: Time taken to answer.												
30	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	3	0	unannounced.practice	NULL	She was the winner.					
31	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	3	0	unannounced.practice	NULL	NULL	7	NULL	1012		
32	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	4	0	unannounced.practice	NULL	Promise to wash%2C Neal did the car.					
33	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	4	0	unannounced.practice	NULL	NULL	2	NULL	1112		
34	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	5	0	unannounced.practice	NULL	The brother and sister that were playing all the time had to be sent to bed.					
35	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	5	0	unannounced.practice	NULL	NULL	5	NULL	2619		
36	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	6	0	unannounced.practice	NULL	The children were cared for by the adults and the teenagers.					
37	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	6	0	unannounced.practice	NULL	NULL	4	NULL	3866		
38	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	7	0	unannounced.practice	NULL	Ben is hopeful for everyone you do to attend.					
39	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	7	0	unannounced.practice	NULL	NULL	3	NULL	3436		
40	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	8	0	unannounced.practice	NULL	All the men seem to have all eaten supper.					
41	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	8	0	unannounced.practice	NULL	NULL	7	NULL	1464		
42	1500406711	8a5e67b77b5907b78a8d61ad8d616fe9	AcceptabilityJudgment	9	0	unannounced.practice	NULL	They consider a teacher of Chris geeky.					

PART 3: Recruiting participants

Now that you've got your experiment coded up in Ibex, time to collect some data. Today we will focus on…

CROWDSOURCING: getting a service (e.g. completing an experimental survey) by drawing on a relatively open, internet-based population.

PART 3: Recruiting participants

Now that you've got your experiment coded up in Ibex, time to collect some data. Today we will focus on…

CROWDSOURCING: getting a service (e.g. completing an experimental survey) by drawing on a relatively open, internet-based population.

WARNING! We will not discuss how to secure the appropriate ethics approval you need to work with human subjects collected in this way. You will need approval from your university's Institutional Review Board (the IRB) before running any studies using Mechanical Turk or other populations. This is generally a painless process, but since it varies from institution to institution, I can't give you detailed instructions. Your best resource here are local experimentalists at your home university who are versed with local ethics norms / requirements. Ask them, they'll be glad to help!

PART 3: Recruiting participants

There are many popular sites popping up that facilitate crowdsourcing. Some well-known ones:

MECHANICAL TURK: A 'marketplace for human intelligence tasks' (HITS). The largest and most well-known crowdsourcing site. Population is primarily users in the USA and in India. ***http://www.mturk.com***.

PROLIFIC ACADEMIC: Like MTurk, but specifically for academic studies. ***http://www.prolific.ac***

Le RISC (Relais d'Information sur les Sciences de la Cognition): Free crowdsourcing platform for finding participants in France. http://expesciences.risc.cnrs.fr

Zhubajie (猪八戒网): Crowdsourcing site active in China. **http://www.zbj.com**

PART 3: Recruiting participants

MECHANICAL TURK:

Pros:

- Easy and fast to collect participants
- More diverse subject pool than university subject pools

Cons:

- Usually costs more than in-lab experiments
- More quality control is needed to weed out 'non-compliant' participants
- Less control can be exercised over subject population

PART 3: Recruiting participants

MECHANICAL TURK:

Pros:

- Easy and fast to collect participants
- **More diverse subject pool than university subject pools**

Cons:

- Usually costs more than in-lab experiments
- More quality control is needed to weed out 'non-compliant' participants
- Less control can be exercised over subject population

PART 3: Recruiting participants

MECHANICAL TURK:

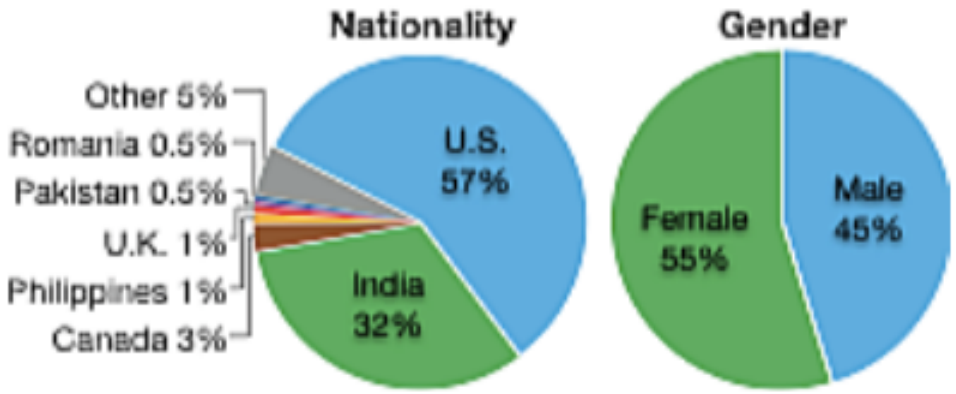


Figure 1. Nationality and gender of MTurk workers.

Figure 4. Reported reliance on money earned on MTurk.

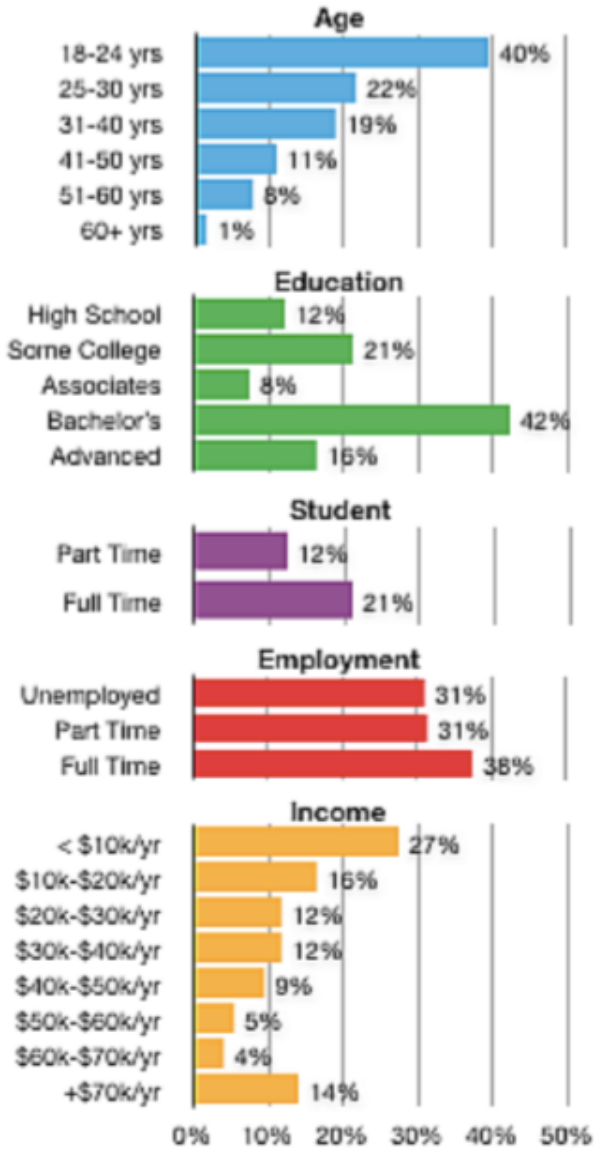
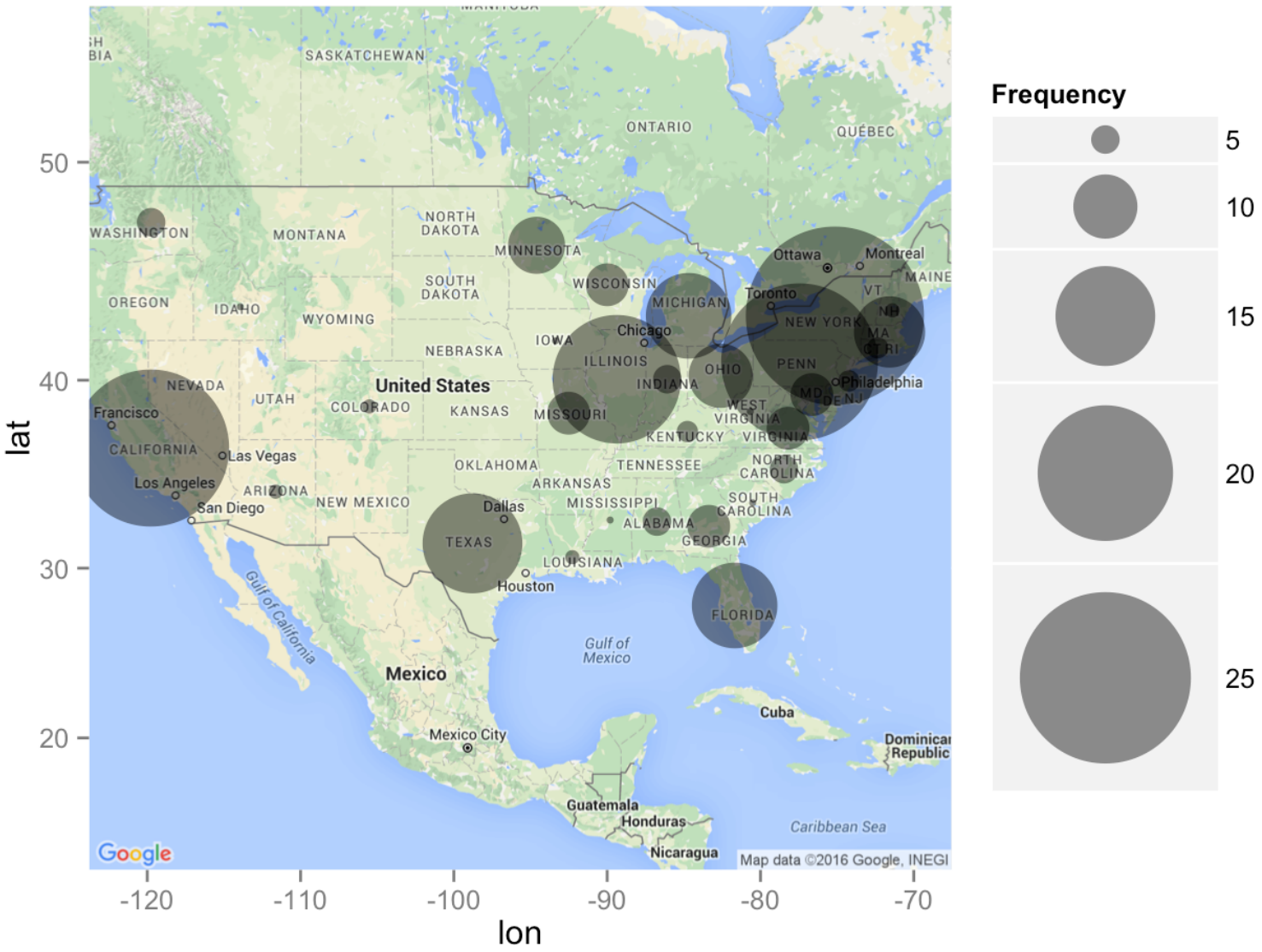


Figure 2. Demographics of MTurk workers.

Source: Ross, Zaldivar, Irani & Tomlinson, 2010

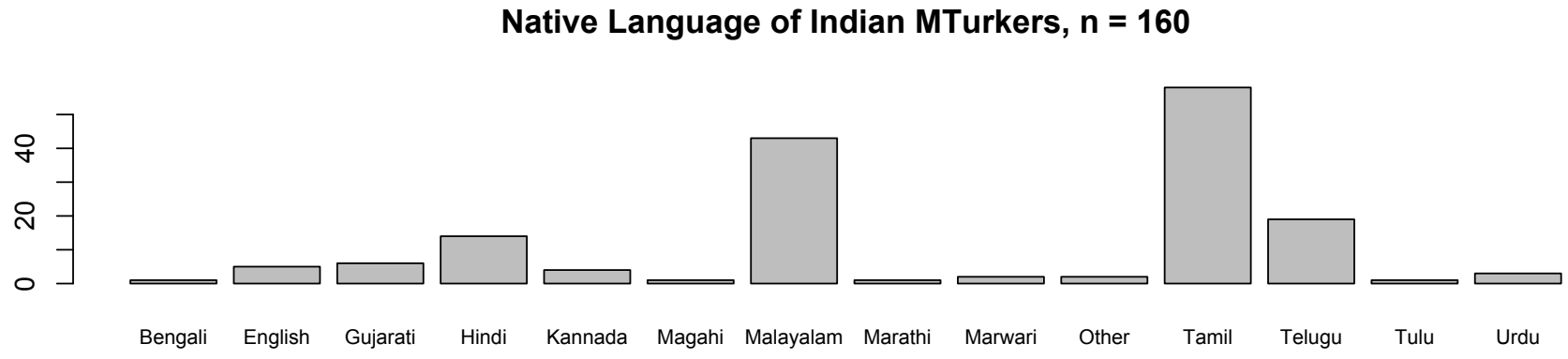
PART 3: Recruiting participants

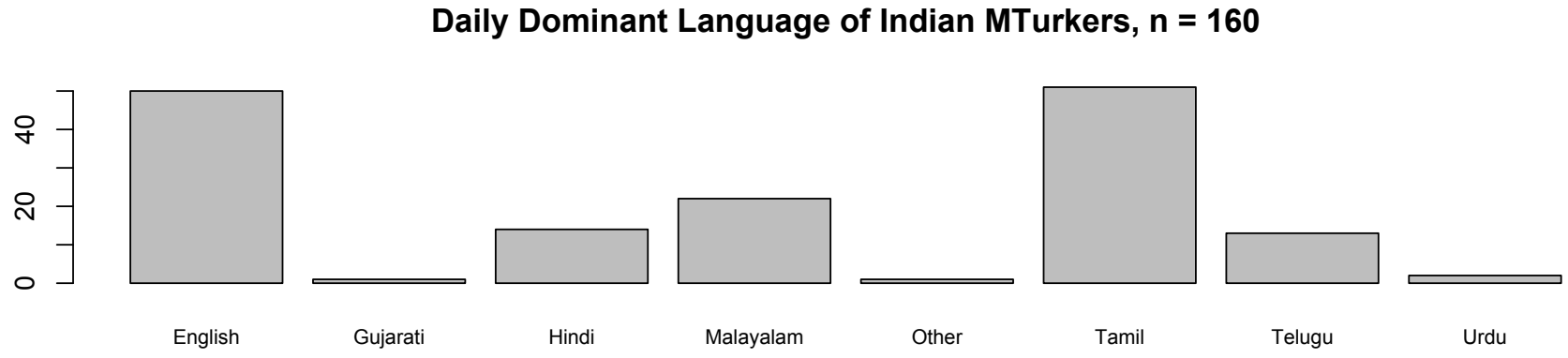
An analysis of 272 US-based participants collected in 7 experiments in our lab revealed most participants came from NY, PA, CA, IL, MI, TX and FL:



PART 3: Recruiting participants

Our lab's analysis of 160 participants based in India reveals most participants come from Southern India: Tamil, English and Malayalam speakers are well represented. Hindi speakers less so.





PART 3: Recruiting participants

MECHANICAL TURK:

- **HIT:** Human Intelligence Task. A task that will be completed by a single MTurk human.
- **Assignment:** A single instance of a HIT that is posted in a single **batch** of a **project.**
- **Worker:** An MTurk user who completes hits.

PART 3: requester.mturk.com

← go to MTurk.com | Create an Account | Already have an account? Sign In: Requester

amazonmechanical turk beta | REQUESTER

Home | Create | Manage | Developer | Help

Overview | Tour | Case Studies | Pricing | Business Solutions | We're Hiring! Learn More

Work Distribution Made Easy

Mechanical Turk gives businesses and developers access to an on-demand, scalable workforce

- **Flexibility:** Scale your workforce up and down quickly
- **Accuracy:** Get high-quality, cost-effective results
- **Speed:** Start receiving results in minutes

Get Started

Work Distribution Made Easy

Mechanical Turk gives businesses and developers access to an on-demand, scalable workforce

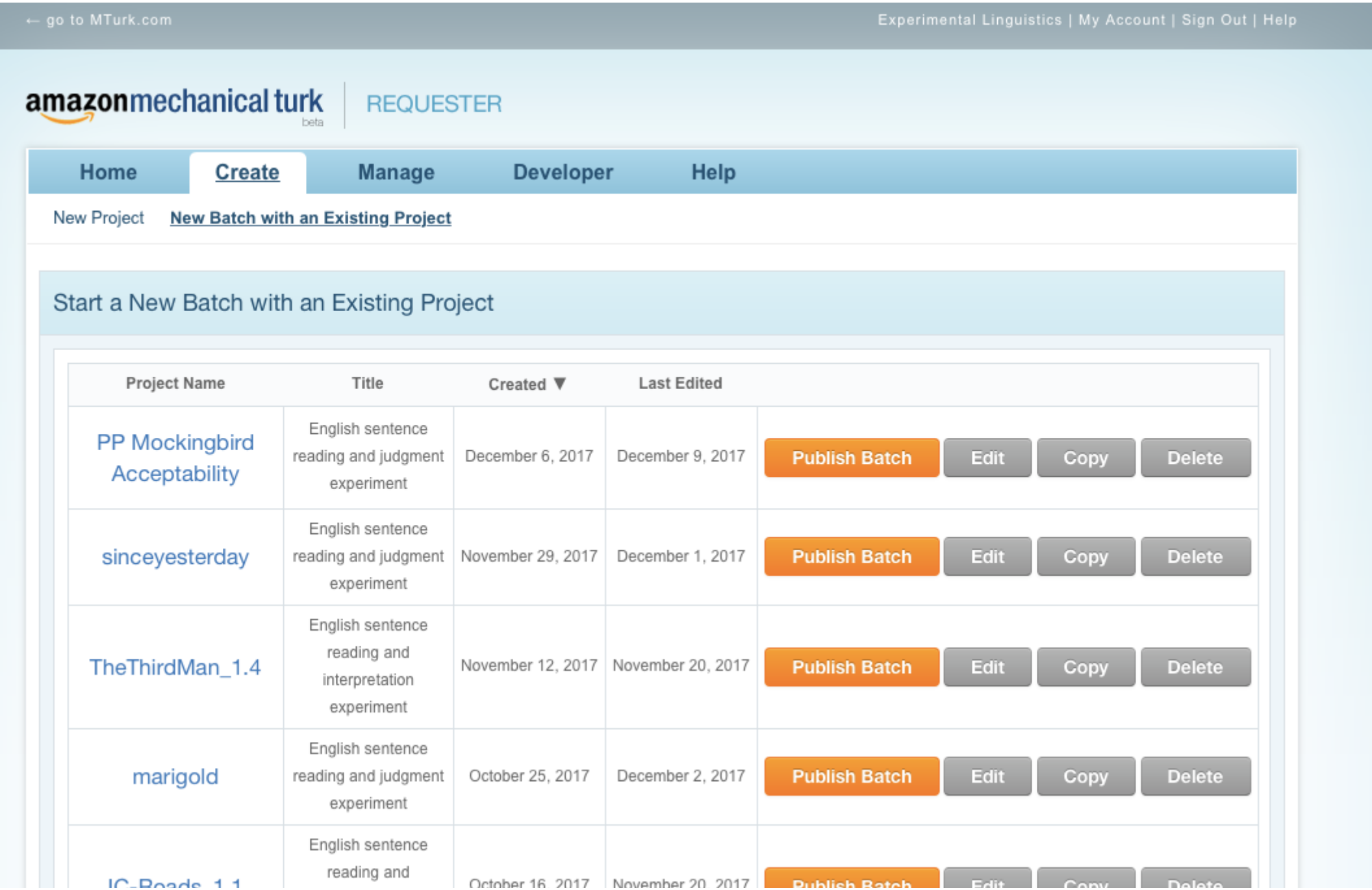
Categorization on Mechanical Turk

The Mechanical Turk **Categorization App** makes it simple to get fast, accurate results on your Categorization Project!

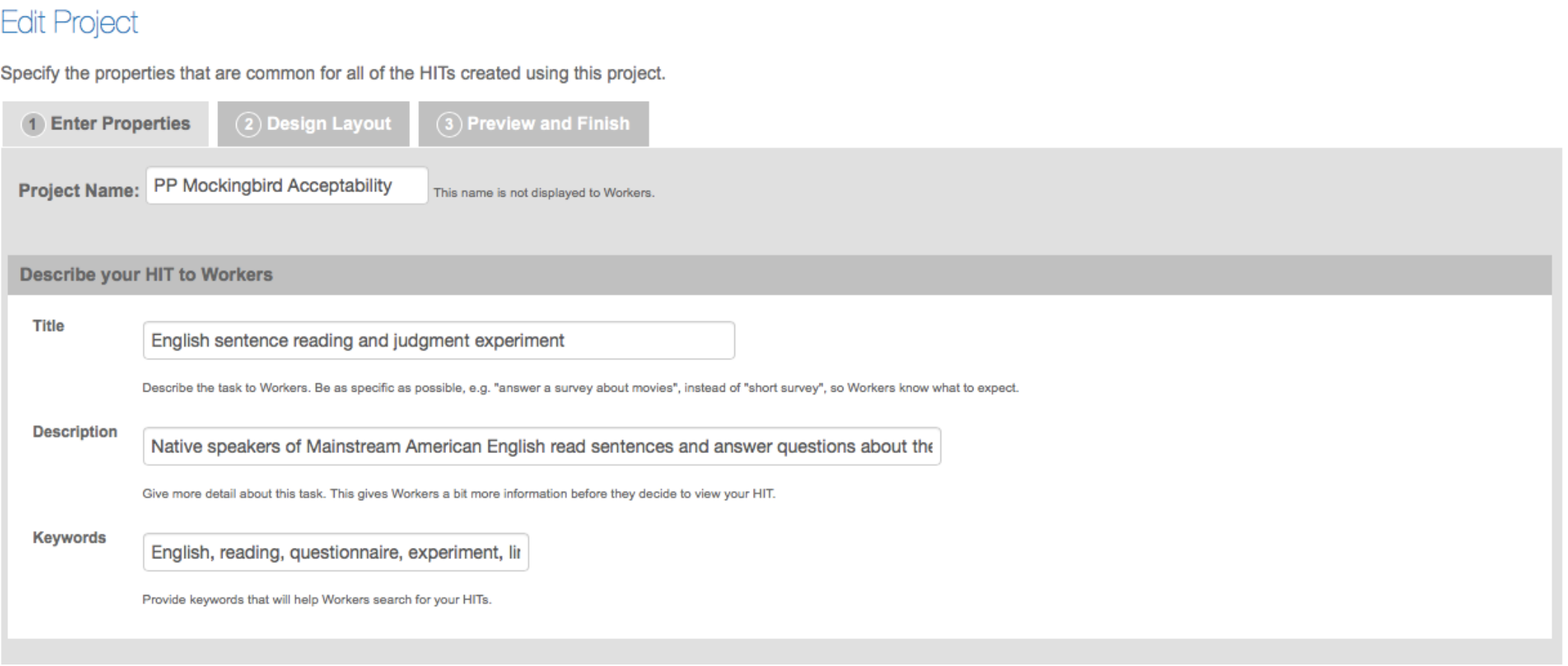
Sentiment Rating Simplified

The Mechanical Turk **Sentiment App** makes it simple to collect and understand sentiment on your data!

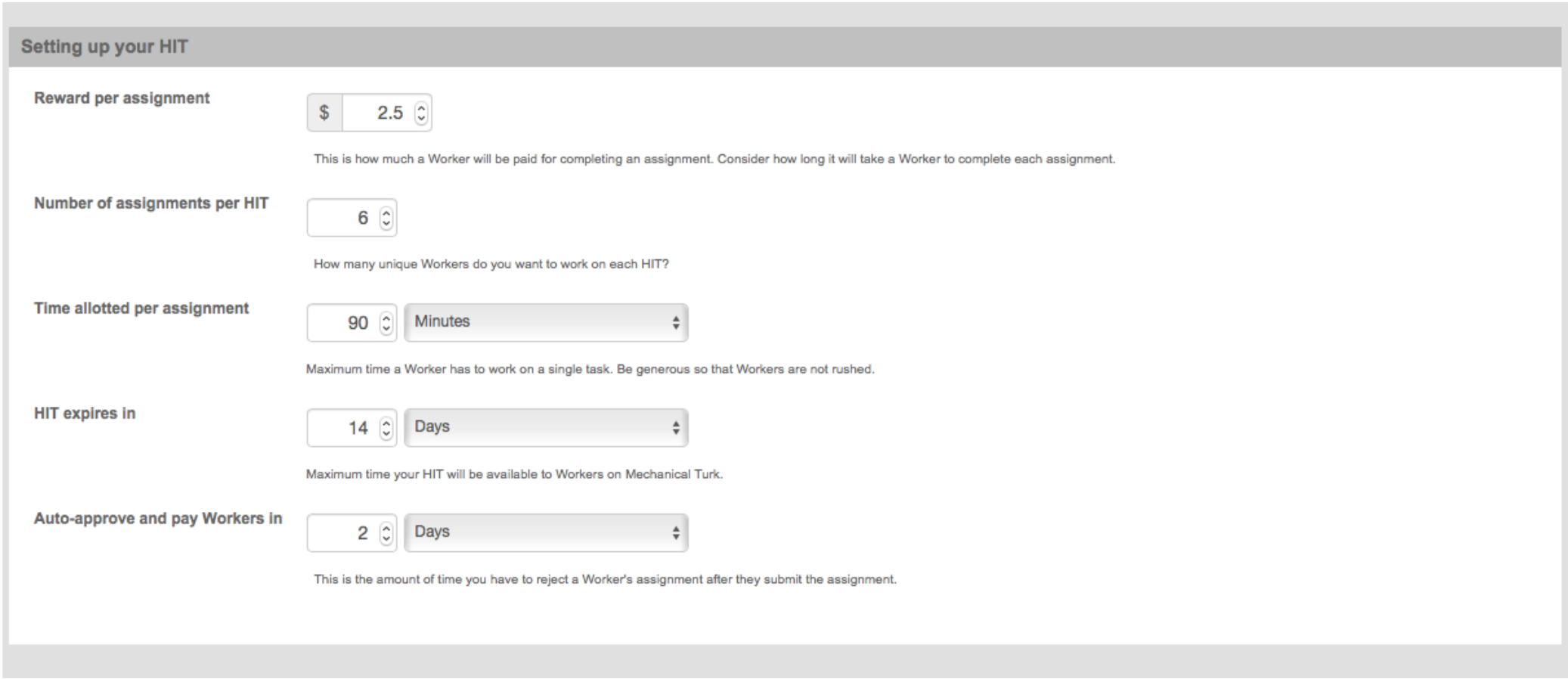
PART 3: requester.mturk.com



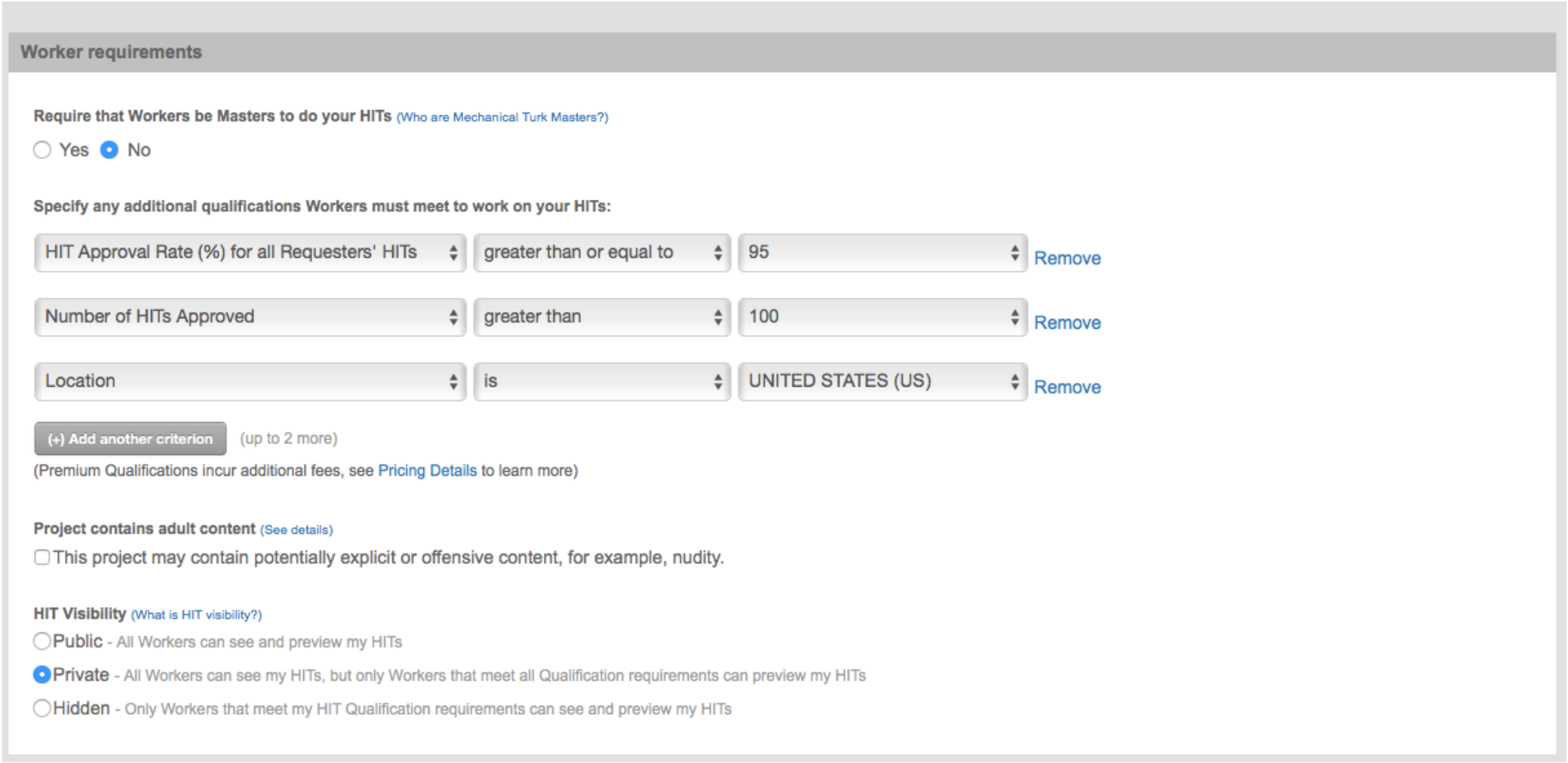
PART 3: requester.mturk.com



PART 3: requester.mturk.com



PART 3: requester.mturk.com



PART 3: requester.mturk.com

Use the HTML editor below to design the layout of your HIT. This layout is common for all of the HITs created with this project. You can define variables for data that will vary from HIT to HIT (Learn more).

1 Enter Properties | 2 Design Layout | 3 Preview and Finish

Project Name: PP Mockingbird Acceptability This name is not displayed to Workers.

Frame Height 400 Height in pixels of the frame your HIT will be displayed in to Workers. Adjust the height appropriately to minimize scrolling for Workers.

Normal | Font | Source

Overview:

In this experiment, you will read sentences of English and then judge how acceptable or unacceptable you find them using a seven-point rating scale. It should approximately 15-20 minutes if your attention is fully on this task. For example, you will read sentences like this:

1. The dogs that were playing in the lake lost the yellow frisbee.

Then you would answer the following question: "How acceptable did you find that sentence?" Your response can range from 1 (unacceptable) to 7 (acceptable).

Requirements:

Language: You MUST be a native speaker of American English to participate in this HIT. That is, English must be the first language you learned, and must have been your dominant langauge during your childhood. If American English is not your native language, you are not eligible for this HIT

Payment:

The compensation for this HIT is $2.50 USD.

Procedure:

If you would like to continue with the experiment, please accept the HIT, and then click the link below to go to our external testing website:

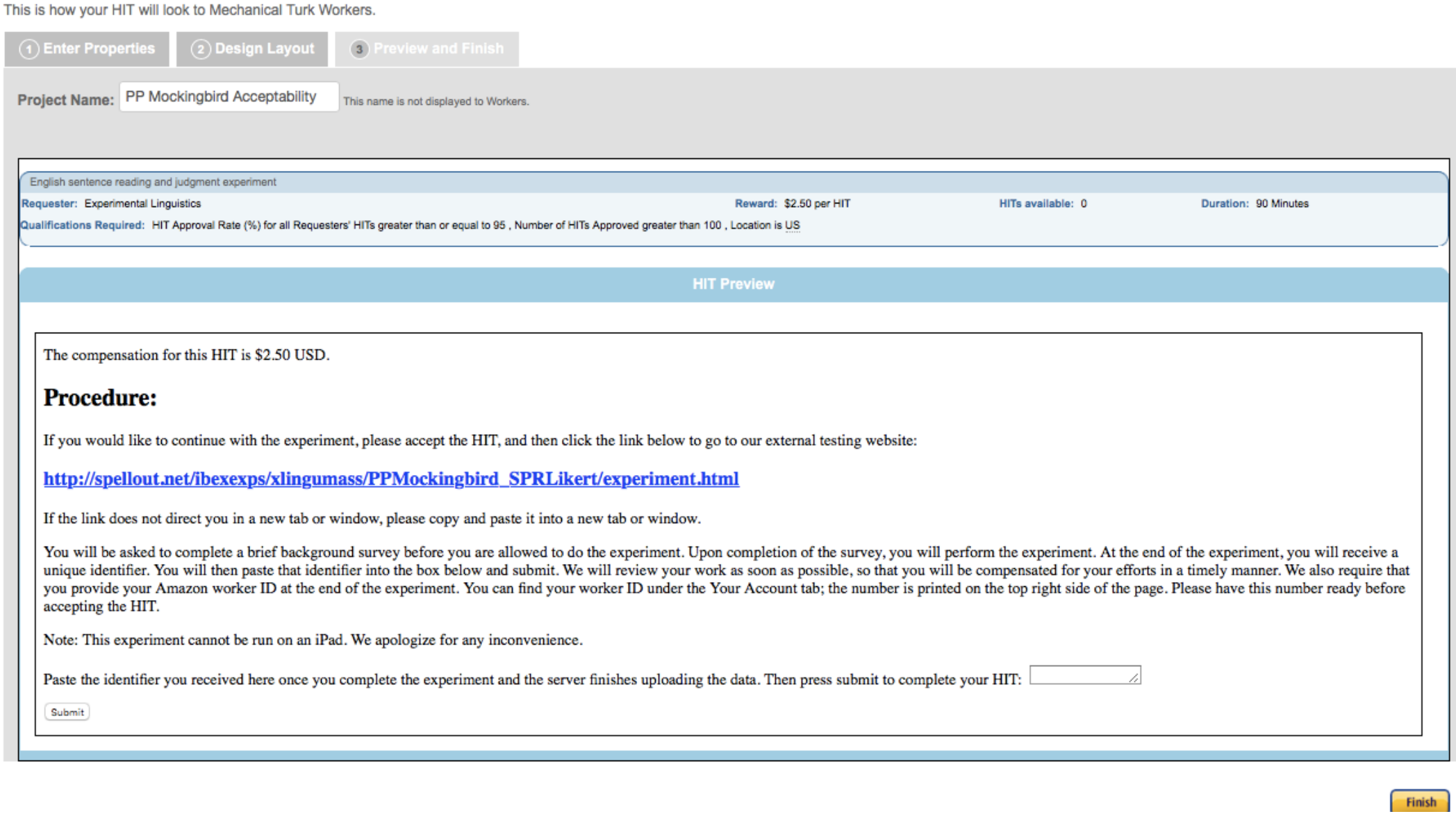
http://spellout.net/ibexexps/xlingumass/PPMockingbird_SPRLikert/experiment.html

If the link does not direct you in a new tab or window, please copy and paste it into a new tab or window.

You will be asked to complete a brief background survey before you are allowed to do the experiment. Upon completion of the survey, you will perform the experiment. At the end of the experiment, you will receive a unique identifier. You will

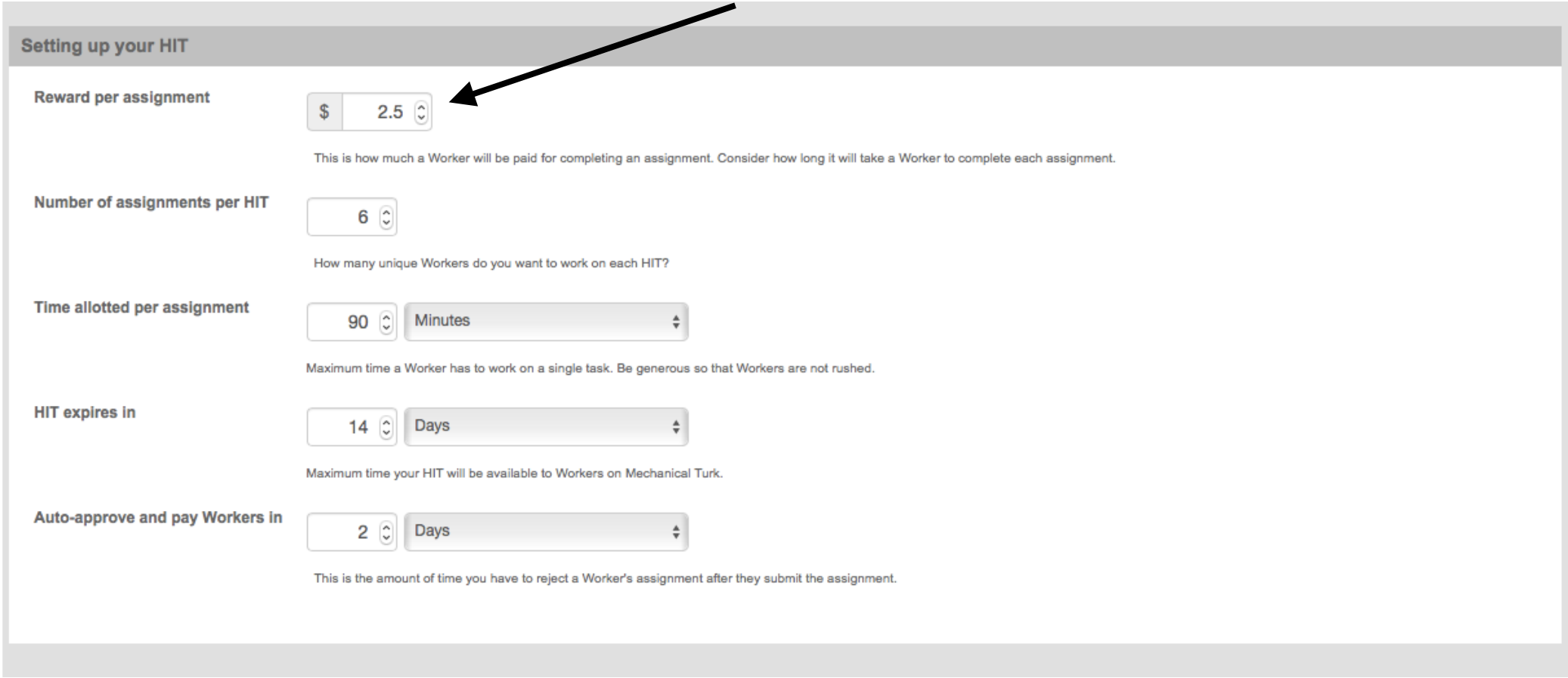
body p strong span

PART 3: requester.mturk.com



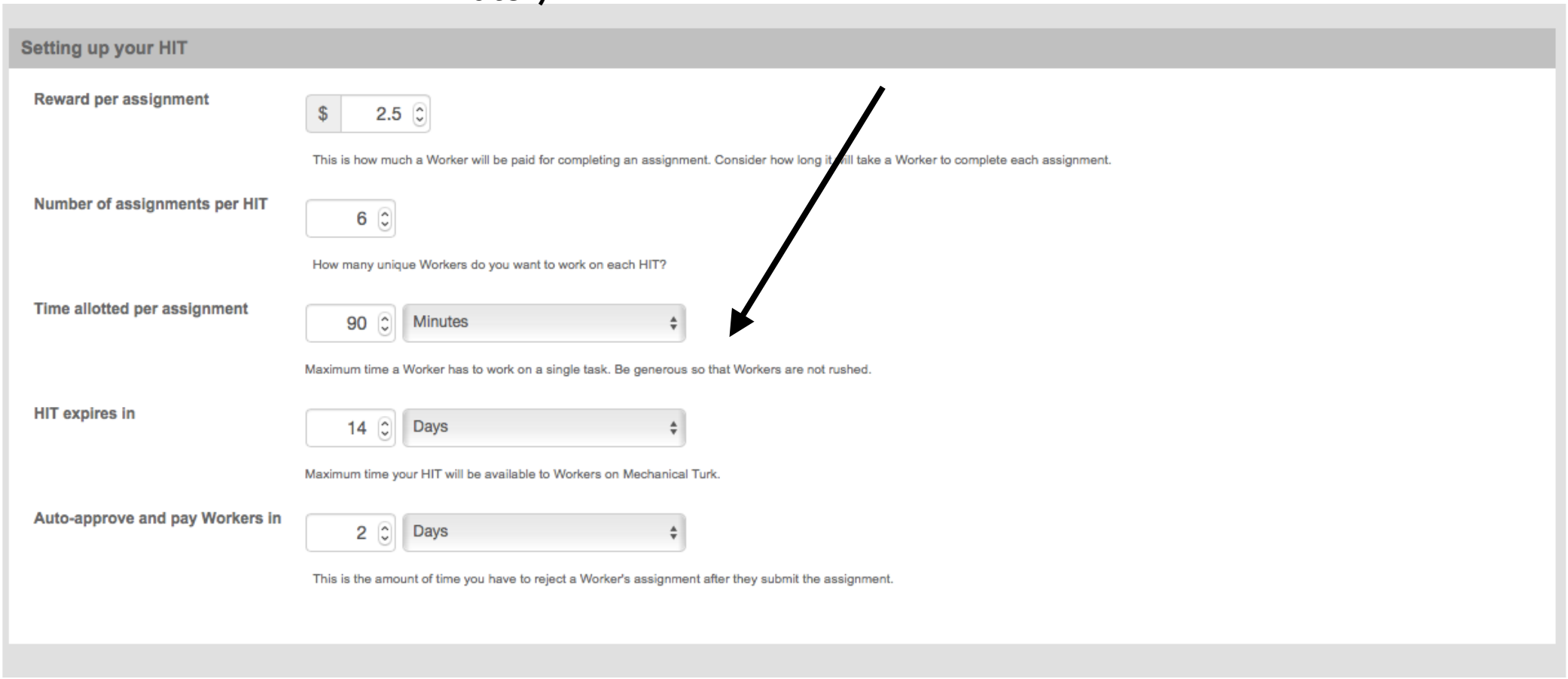
PART 3: requester.mturk.com

How much to pay ? Our lab pays equivalent to 10$ an hour, comparable to what we pay in lab. On top of the reward payment, you as requester will pay an additional 20% to MTurk.



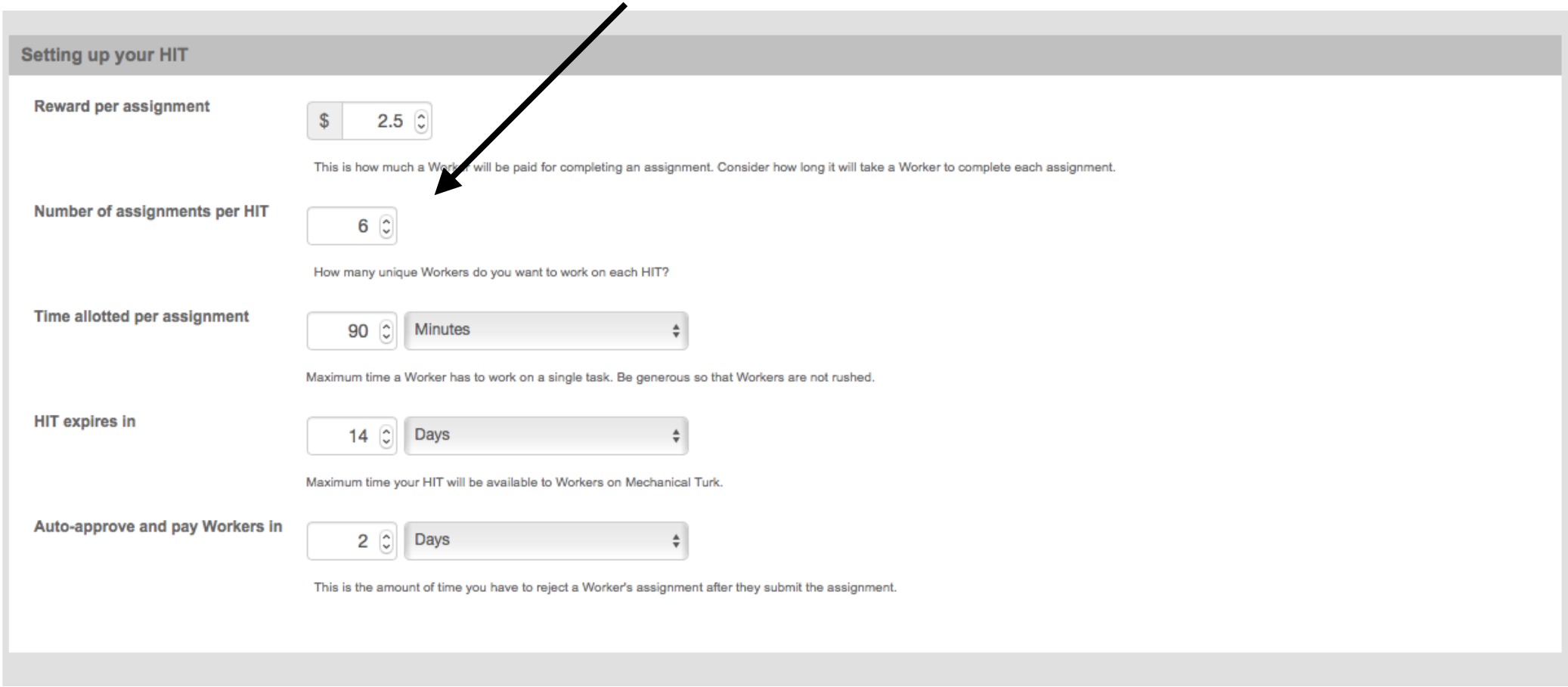
PART 3: requester.mturk.com

Time per assignment should be *significantly* longer than you think it will take. If a worker overruns this time, they will not be allowed to complete the HIT (which will make them angry… more later).



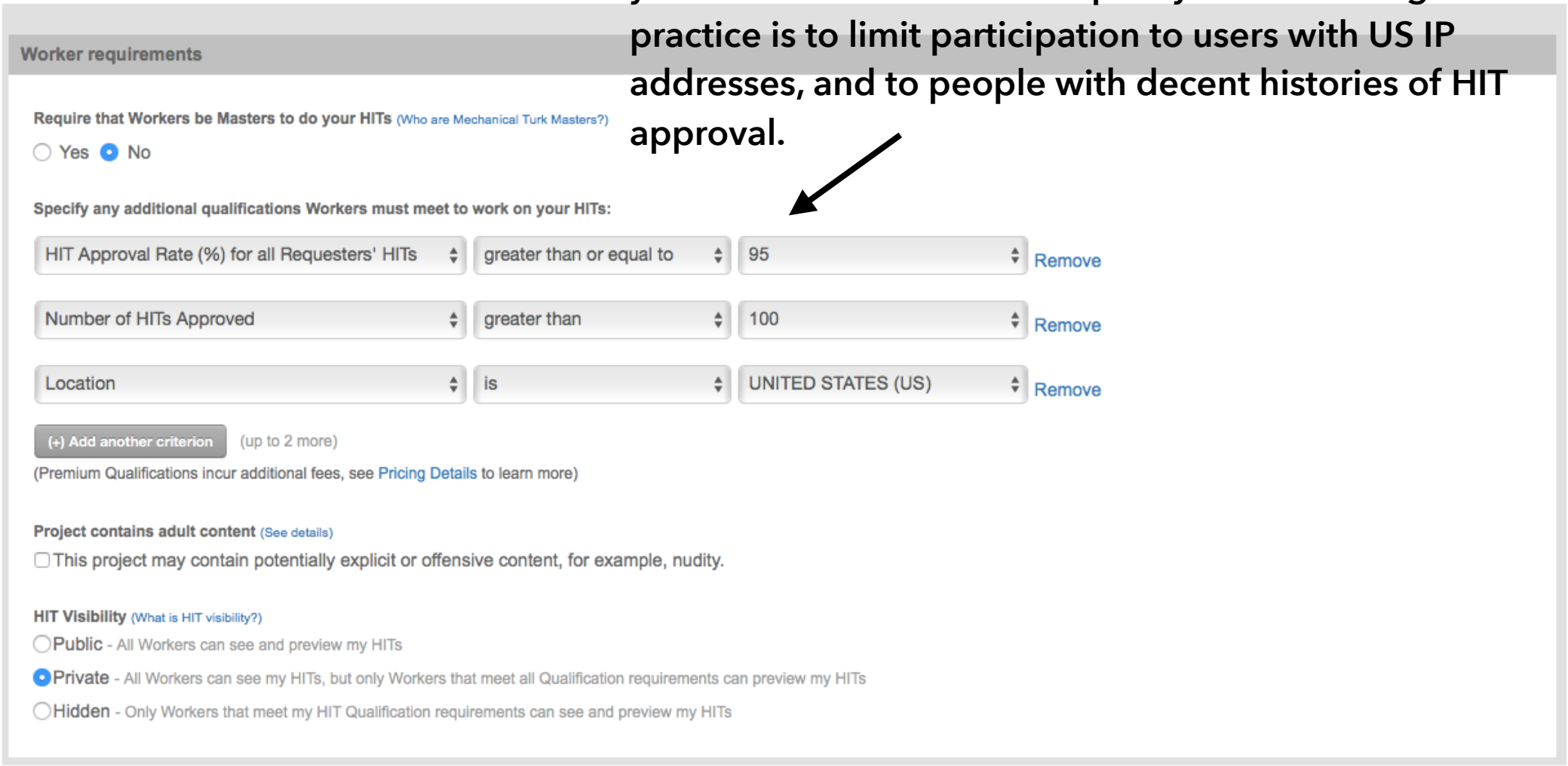
PART 3: requester.mturk.com

How many assignments per HIT? A single assignment = 1 'subject', assuming no repeats (more on that later). However, MTurk levies an additional 20% fee for batches of more than 10 assignments.

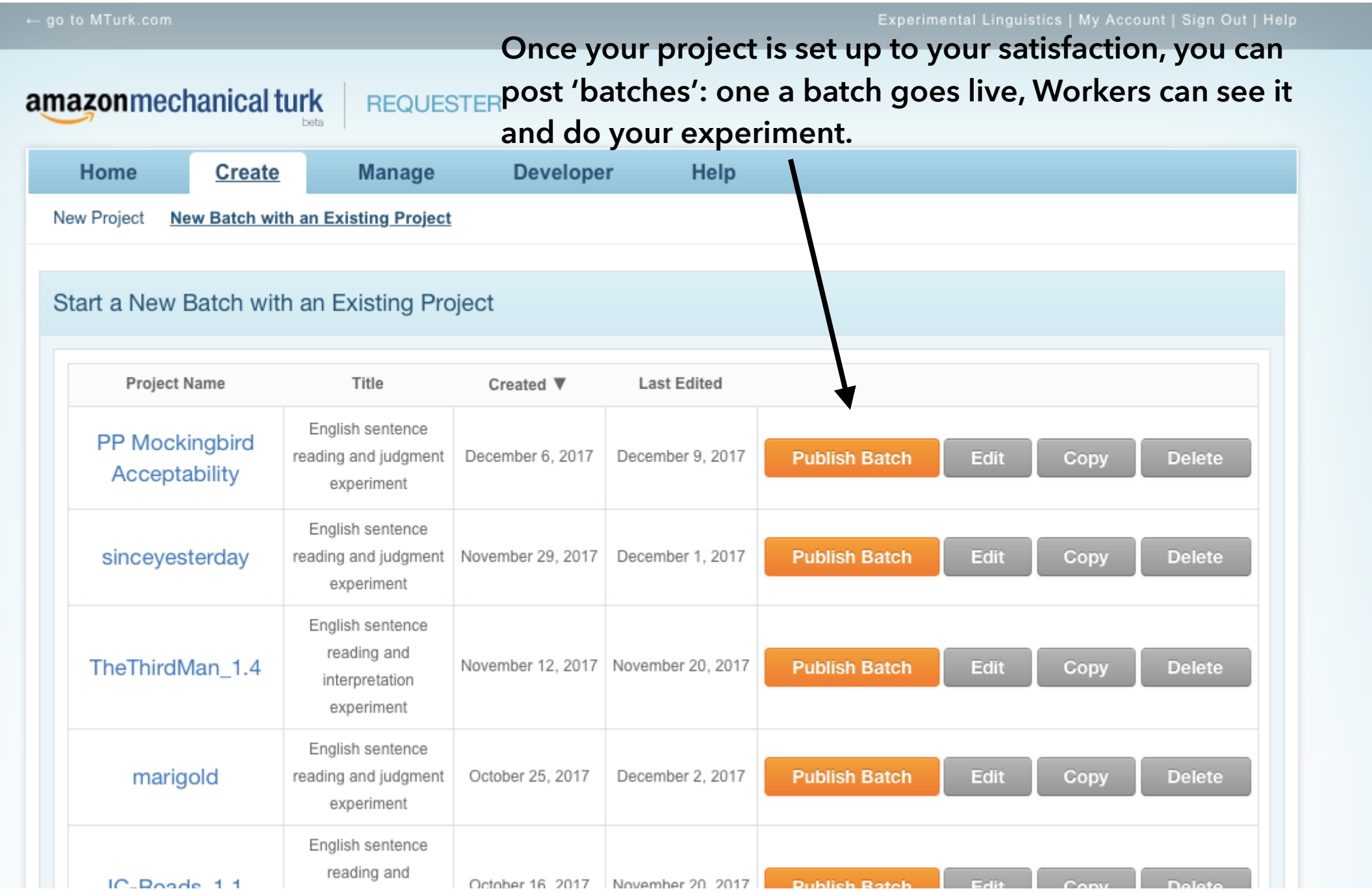


PART 3: requester.mturk.com

Qualifications can be set up to filter out Turkers. This is your first line of defense in quality control. Our general lab practice is to limit participation to users with US IP addresses, and to people with decent histories of HIT approval.

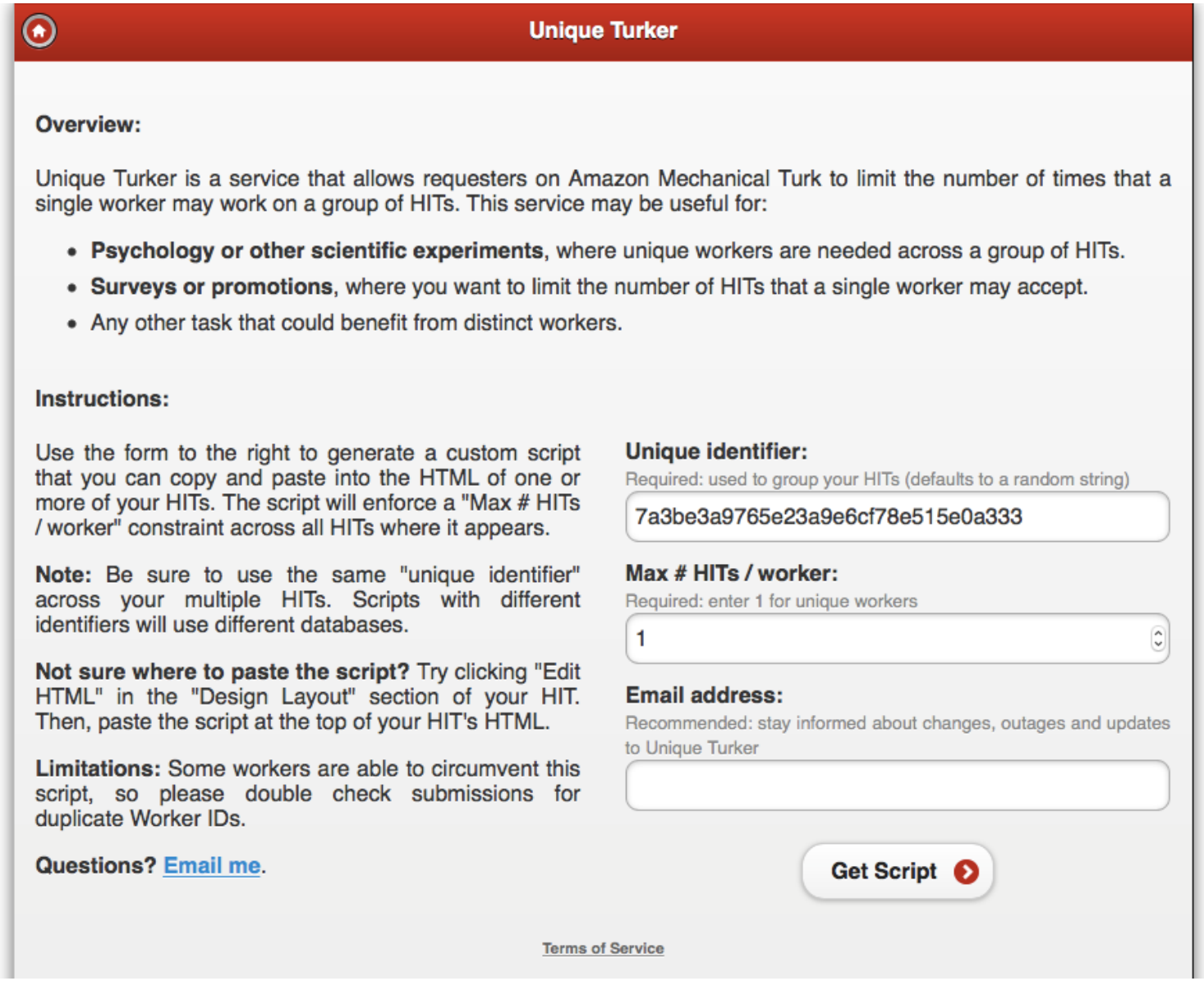


PART 3: requester.mturk.com



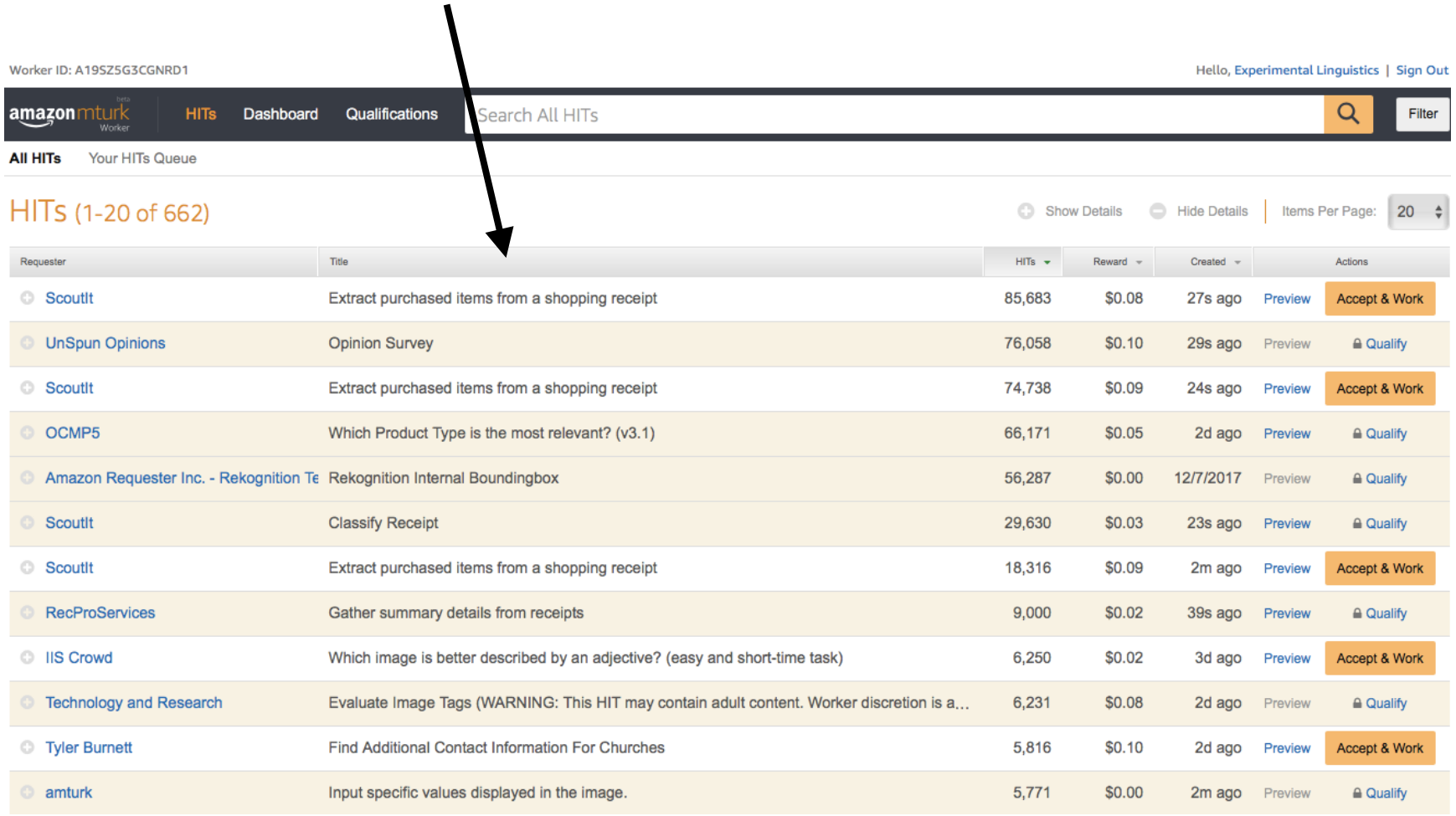
PART 3: https://uniqueturker.myleott.com

Use the 'Unique Turker' service to create a unique identifier for a study or set of studies where you do not want a worker to do your HIT(s) more than once:

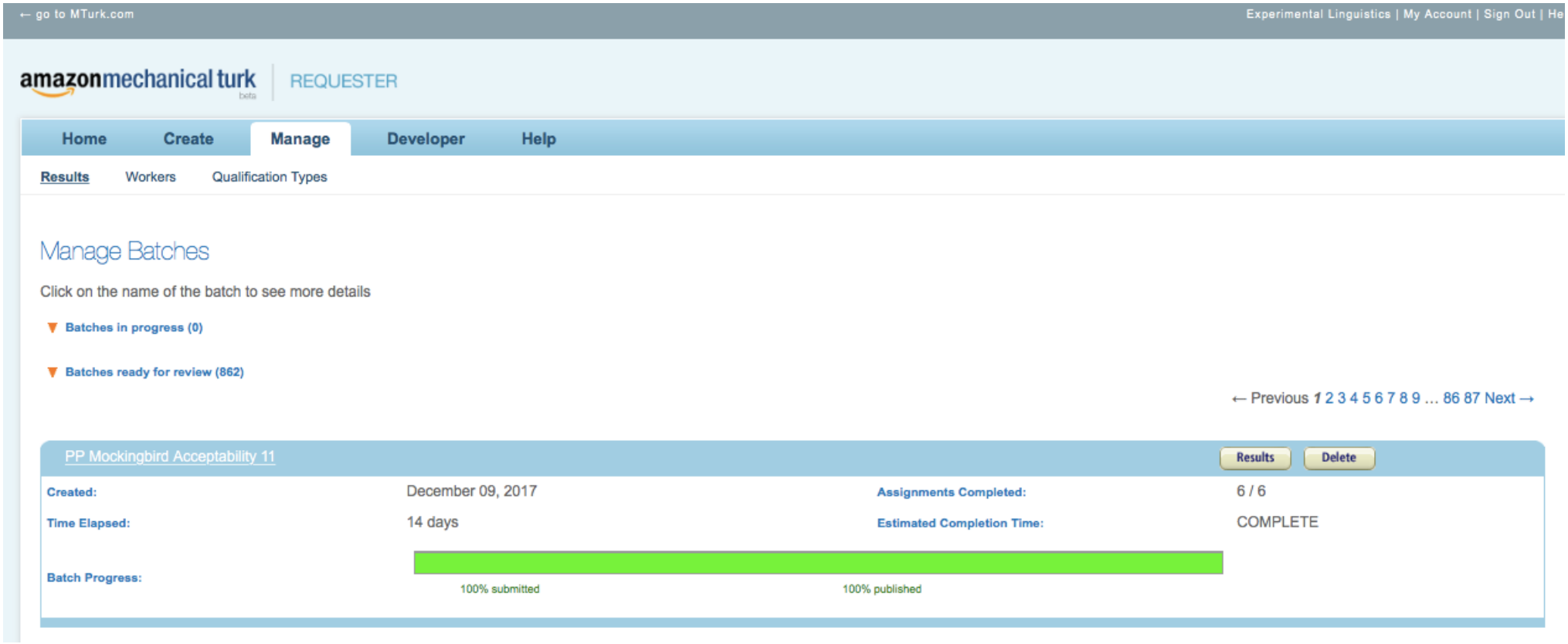


PART 3: requester.mturk.com

Here's what it looks like from a Worker's POV! Important info: your Requester name, and the Title of your HIT:



PART 3: requester.mturk.com



PART 3: requester.mturk.com

Manage Batches > **Batch Details**

PP Mockingbird Acceptability 11

View the latest status of this batch, make changes, or get results.

Native speakers of Mainstream American English read sentences and answer questions about them

Status — Delete

Status: Pending Review

100% submitted — 100% published

Assignments Completed: 6 / 6
Creation Time: December 09, 2017 2:06 PM PST
Average Time per Assignment: 30 minutes 23 seconds
Completion Time: December 09, 2017 4:18 PM PST

Settings

PP Mockingbird Acceptability

Description: Native speakers of Mainstream American English read sentences and answer questions about them
Keywords: English, reading, questionnaire, experiment, linguistics, psychology
HIT Approval Rate (%) for all Requesters' HITs greater than or equal to 95
Qualification Requirement: Number of HITs Approved greater than 100
Location is US

Number of Assignments per HIT: 6
Reward per Assignment: $2.50

HIT expired on: December 23, 2017 2:06 PM PST
Assignment duration: 1 hour 30 minutes
Auto Approval Delay: 2 days

Results — Results

Assignments pending review: 0
Assignments approved: 6
Assignments rejected: 0

Cost Summary

Estimated Total Reward: $15.00
Estimated Fees to Mechanical Turk: $3.00 (fee details)
Estimated Total Cost: $18.00
These costs are only an estimate until all of the assignments have been submitted and reviewed.

PART 3: What if I don't want to work on English?

MECHANICAL TURK: works great for English, Spanish, and (plausibly) south Indian languages. There exist good crowdsourcing sites for French (le RISC) and languages spoken in Mainland China (Zhubajie Wang).

But what if you want to target a different population for an internet-based experiment? Say: Romanian speakers? Assuming you're working with Ibex, you have (at least) two technical challenges to overcome:

- Find a way to target population: social media, class mailing lists, online fora.
- Find a way to compensate your speakers

Again: IRB/ethics approval precedes all of this! This can be especially important, depending on the specific population you are targeting.

PART 3: What if I don't want to work on English?

MECHANICAL TURK: works great for English, Spanish, and (plausibly) south Indian languages. There exist good crowdsourcing sites for French (le RISC) and languages spoken in Mainland China (Zhubajie Wang).

But what if you want to target a different population for an internet-based experiment? Say: Romanian speakers? Assuming you're working with Ibex, you have (at least) two technical challenges to overcome:

- **Find a way to target population: social media, class mailing lists, online fora.**
- Find a way to compensate your speakers

Again: IRB/ethics approval precedes all of this! This can be especially important, depending on the specific population you are targeting.

PART 3: What if I don't want to work on English?

MECHANICAL TURK: works great for English, Spanish, and (plausibly) south Indian languages. There exist good crowdsourcing sites for French (le RISC) and languages spoken in Mainland China (Zhubajie Wang).

But what if you want to target a different population for an internet-based experiment? Say: Romanian speakers? Assuming you're working with Ibex, you have (at least) two technical challenges to overcome:

- Find a way to target population: social media, class mailing lists, online fora.
- **Find a way to compensate your speakers**

Again: IRB/ethics approval precedes all of this! This can be especially important, depending on the specific population you are targeting.

PART 3: What if I don't want to work on English?

LOTTERY SYSTEM*:* Mechanical Turk streamlines paying participants. This is a non-trivial part of what the service does for you. Taking this on yourself can be non-trivial. Our lab's solution to this is to adopt a lottery system. Basic idea:

- Every participant gets a random code for participating in the experiment.
- They are then directed to an external site to input in their code as part of a lottery.
- After the experiment is complete, choose *n* lottery winners.
- Award each lottery winner their prize.

PART 3: What if I don't want to work on English?

EXAMPLE: Self-paced reading experiment on Romanian. Participants contacted through courses at the university of Bucharest. Target N = 40 participants for a 15-minute experiment.

Normally, each participant would receive 2$ USD; total experiment costs would be $80. In the lottery, we distribute this amount in *n* prizes:

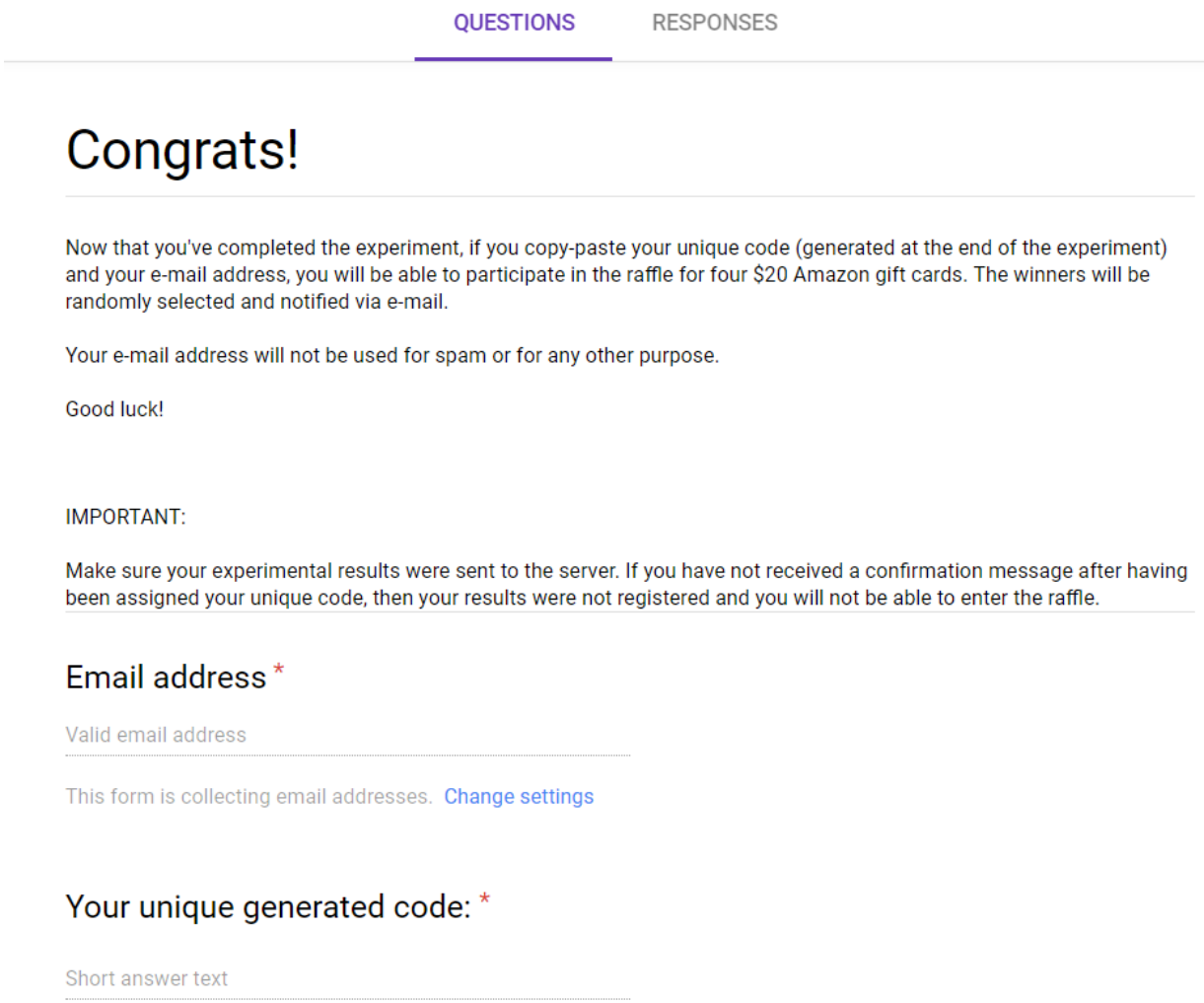
n = *2*, prize value = $40
n = 4, prize value = $20
n = 8, prize value = $10

PAYMENT OPTIONS: Vary depending on population, but we use gift cards that can be mailed to an email address:

- **Amazon.com gift cards**
- **Pre-paid Visa gift cards (***https://www.giftcards.com/visa-gift-cards*)

PART 3: What if I don't want to work on English?

For data security purposes, you should not store identifying information (email adds) with your data. Third party websites (on a different serve) can be used to coordinate collection of email addresses and participation codes.



PART 4: Quality control

INTERNET BASED STUDIES:

Pros:

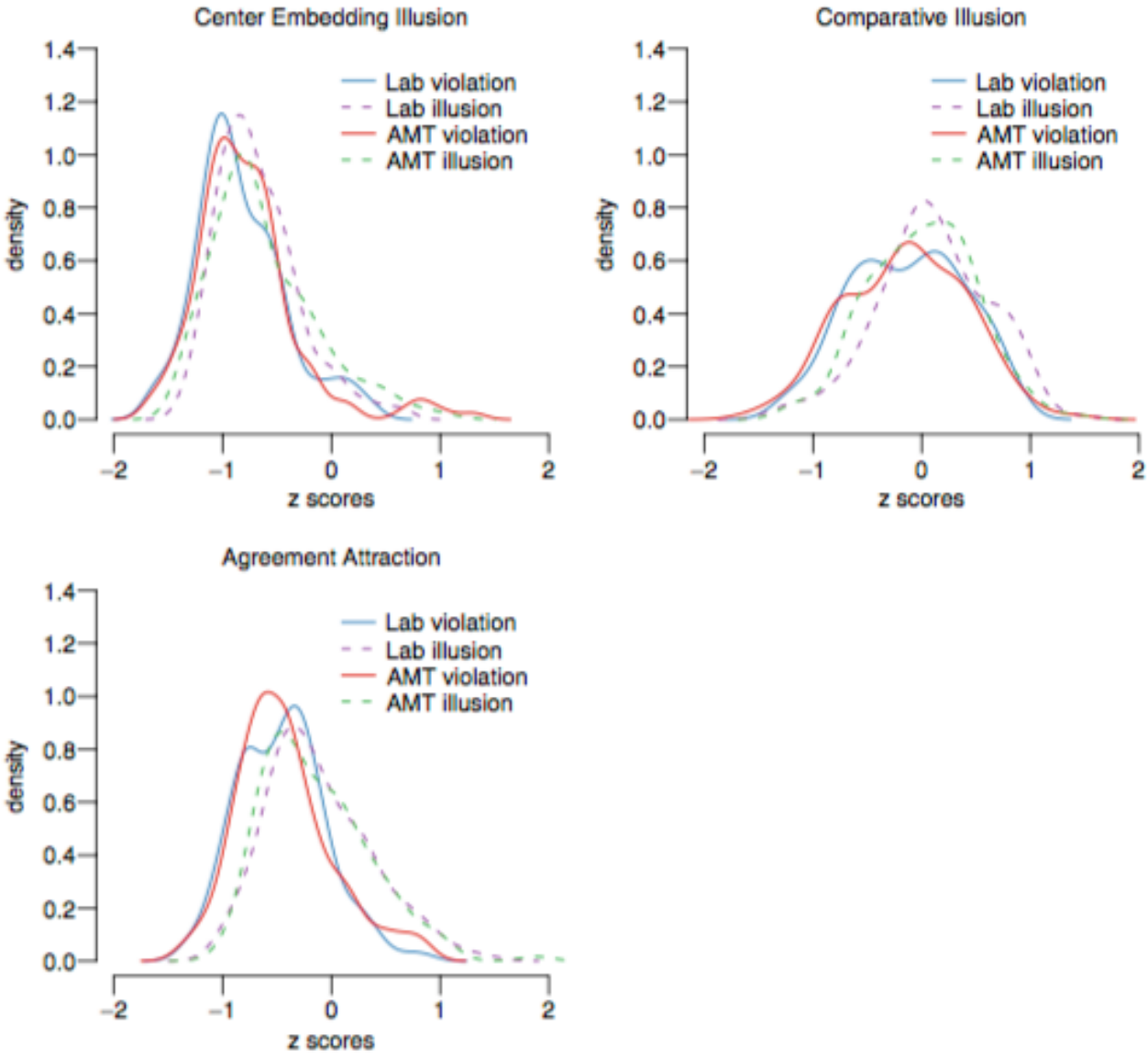
- Easy and fast to collect participants
- More diverse subject pool than university subject pools

Cons:

- Usually costs more than in-lab experiments
- **More quality control is needed to weed out 'non-compliant' participants**
- Less control can be exercised over subject population

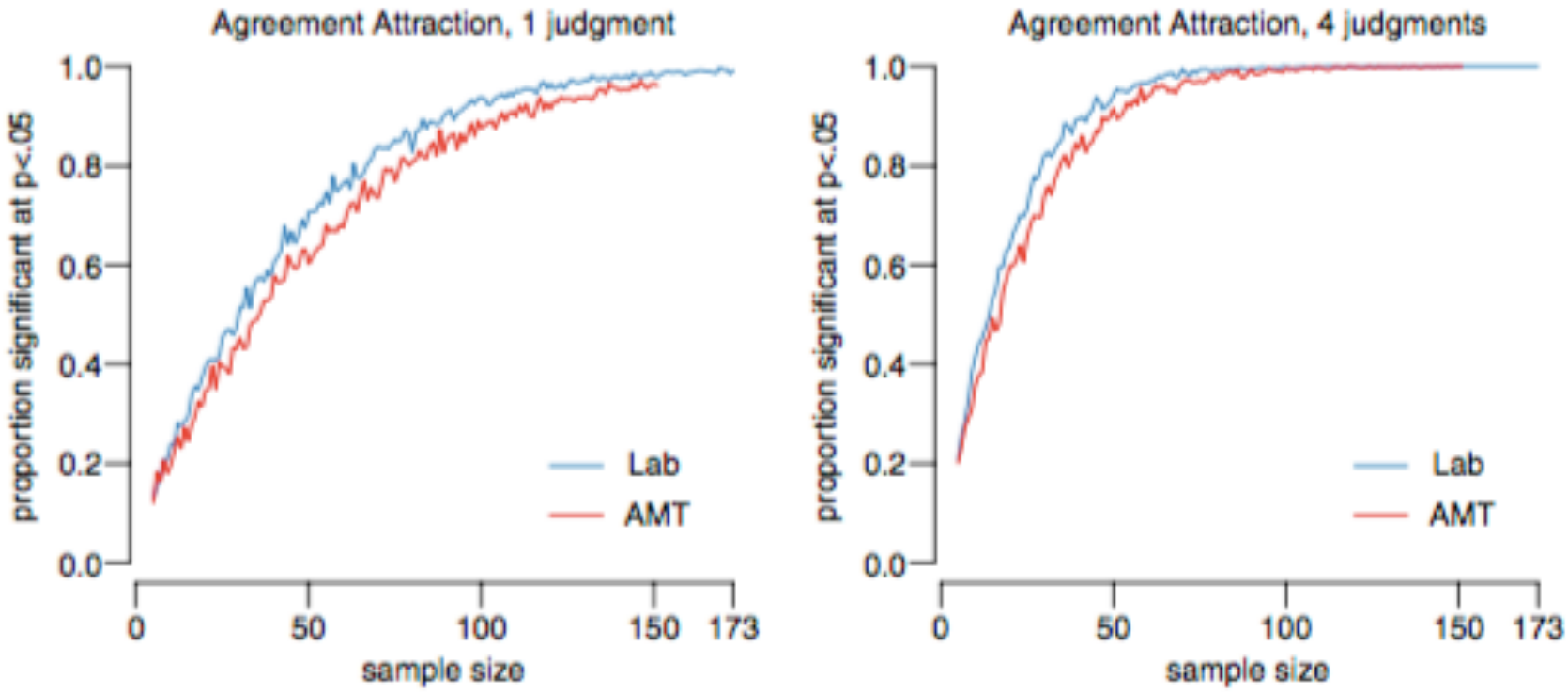
PART 4: Quality control

Fig. 6 The distributions of judgments for the extremely weak effects: Density curves for each condition of the extremely weak effects. The *x*-axis represents the judgments after a *z*-score transformation. The *y*-axis is density. The control violations are plotted as solid lines, and the illusion conditions are plotted as dashed lines. The laboratory sample is plotted in blue and purple, respectively, and the AMT sample is in red and green, respectively.



Figures from Sprouse, 2011; see also Enochson & Culbertson, 2015 on SPR measures.

PART 4: Quality control



Figures from Sprouse, 2011; see also Enochson & Culbertson, 2015 on SPR measures.

PART 4: Quality control

After participant rejection, data quality is comparable to in-lab data for both judgment studies (Sprouse, 2011) and SPR studies (Enochson & Culbertson, 2015). However, there are many more **non-cooperative** participants in internet-based studies (Häussler & Juzek, 2017). Häussler & Juzek's proposed typology of non-cooperative participants:

- **Simple spammers:** Participants who just rapidly 'click through' to get the code at the end, generating useless data along the way. Characterized by low variability in responses, implausibly fast reaction times.
- **Clever spammers:** Participants who seek to just click through, but know that simple strategies like pressing 1 for every trial will draw attention. A clever spammer offsets this by attempting to spoof or emulate true behavior; for example, they may pause every so often on a question to drive up mean response time.
- **Inattentive participants:** Participants who are attempting the task but not paying attention (due to multi-tasking, for instance). Their data may not bear on question under investigation, but it is hard to distinguish from 'real' data.

PART 4: Quality control

- **Simple spammers:** Participants who just rapidly 'click through' to get the code at the end, generating useless data along the way. Characterized by low variability in responses, implausibly fast reaction times.
- **Clever spammers:** Participants who seek to just click through, but know that simple strategies like pressing 1 for every trial will draw attention. A clever spammer offsets this by attempting to spoof or emulate true behavior; for example, they may pause every so often on a question to drive up mean response time.
- **Inattentive participants:** Participants who are attempting the task but not paying attention (due to multi-tasking, for instance). Their data may not bear on question under investigation, but it is hard to distinguish from 'real' data.

Tools in your QC toolkit:

- Analysis of reaction times
- Analysis of response variability
- Analysis of 'catch trials'

PART 4: Quality control: Reaction times

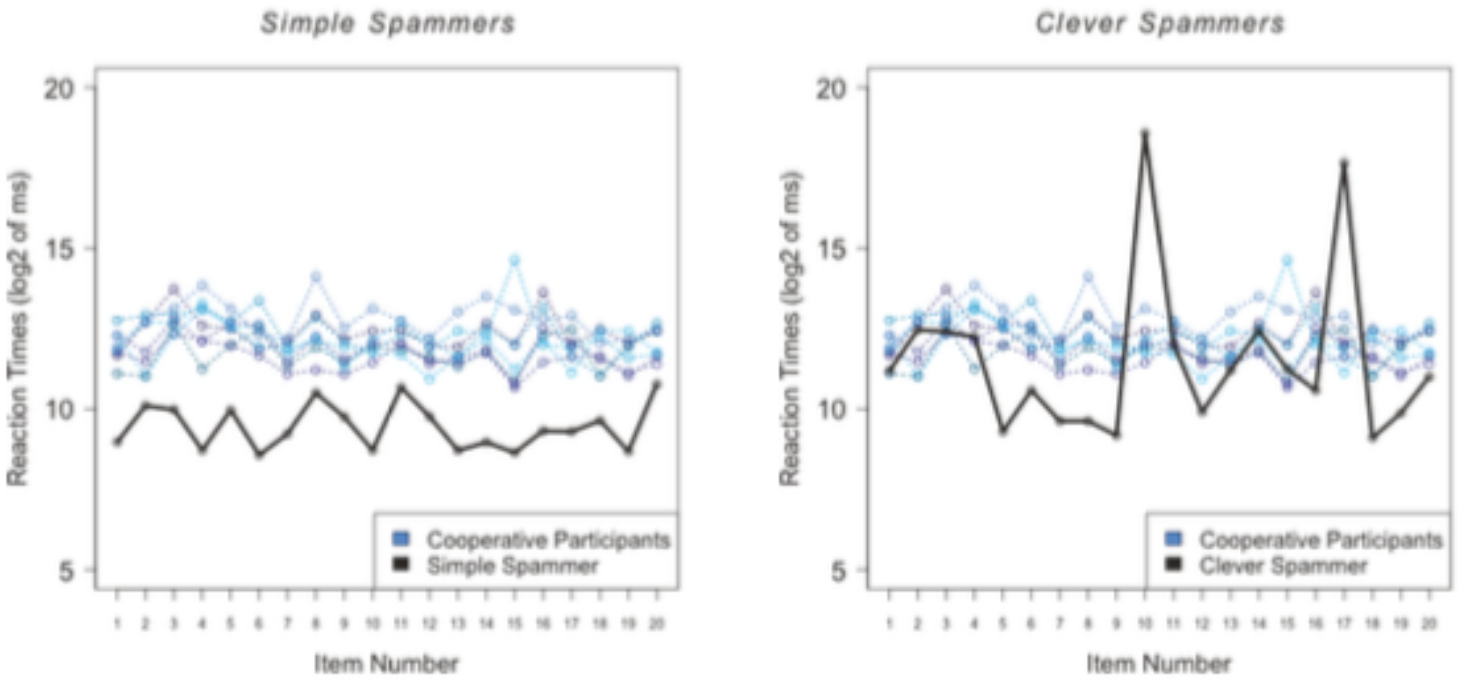


Figure 1. An illustration of the response times produced by cooperative participants (dashed gray lines) and two types of non-cooperative participants (solid black lines) – simple spammers (left) and clever spammers (right)

figure from Häussler & Juzek, 2017

PART 4: Quality control: Reaction times

Table 2. Selected response times (in ms) for the first seven participants in Experiment 2a, Part 1

	Item							
Participant	**1**	**2**	**3**	**4**	**5**	**6**	**Mean**	**Median**
1	1,710	1,890	5,090	2,310	2,980	2,070	2,675	2,190
2	2,210	2,800	11,280	7,030	4,260	3,200	5,130	3,730
3	3,340	2,170	4,470	3,060	2,790	3,020	3,192	3,040
4	3,390	5,050	8,040	5,460	5,520	5,720	5,530	5,490
5	1,900	1,920	5,820	4,710	3,460	5,560	3,895	4,085
6	750	460	490	740	**390,850**	550	**65,640**	**645**
7	380	460	990	870	420	800	**653**	**630**

figure from Häussler & Juzek, 2017

PART 4: Quality control: Reaction times

Typically, outlier rejection based on reaction times is concerned with trimming or removing overlong response times (e.g. Ratcliff, 1993). Detecting spammers requires you to detect *too fast* response times. For judgment experiments, this can be tricky. Two options:

1) **Cutoff-based approach**. Define a mean/median rating time below which a participant will be rejected.

   Pro: Easy to deploy; will work no matter how many spammers are in your sample. Con: Difficult to establish a satisfactory cutoff a priori.

2) **Distributional approach**. Define a cutoff based on the distribution of mean/median rating time below which a participant will be rejected. For example, the fastest 5% of participants will be rejected.

   Pro: Does not depend on arbitrary cutoff, sensitive to distribution of the data; Con: Not reliable if spammers constitute a substantial proportion of your sample..

PART 4: Quality control: Reaction times

Häussler & Juzek (2017) recommend using **1.5 times the standard deviation of participant means or medians** to define the lower RT threshold.

Table 3. Lower thresholds for part 1 of Experiment 2a, applying different methods for detecting spammers

Measure	Mean	SD	θ_{lower}	Participant 6	Participant 7
Mean RTs	5,467	2,266	2,068	7,749	883
Median RTs	3,974	1,418	1,848	1,490	745

$$\theta_{lower} = \frac{1}{n}\sum_{i=1}^{n} x_i - 1.5\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{x} - x_i)^2} \quad (7)$$

Table 2. Selected response times (in ms) for the first seven participants in Experiment 2a, Part 1

	Item							
Participant	1	2	3	4	5	6	Mean	Median
1	1,710	1,890	5,090	2,310	2,980	2,070	2,675	2,190
2	2,210	2,800	11,280	7,030	4,260	3,200	5,130	3,730
3	3,340	2,170	4,470	3,060	2,790	3,020	3,192	3,040
4	3,390	5,050	8,040	5,460	5,520	5,720	5,530	5,490
5	1,900	1,920	5,820	4,710	3,460	5,560	3,895	4,085
6	750	460	490	740	**390,850**	550	**65,640**	**645**
7	380	460	990	870	420	800	**653**	**630**

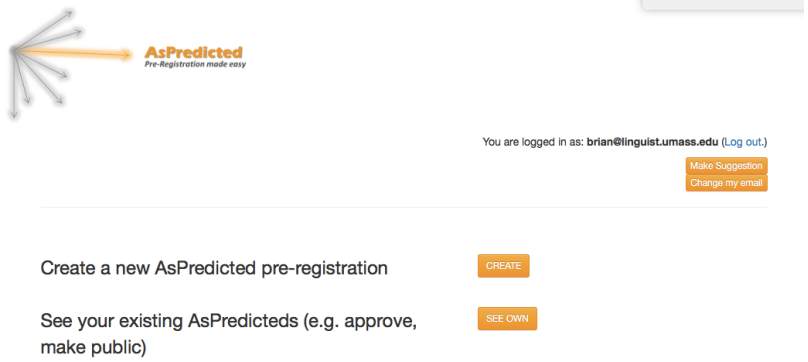
figure from Häussler & Juzek, 2017

PART 4: Quality control: Reaction times

Your mileage may vary, and there is no one size fits all solution. Our lab's practice is to:

i) pilot experiments, and qualitatively investigate RT patterns to identify best strategies for rejecting participants and

ii) **pre-register** rejection choices ahead of time to keep yourself honest and to avoid unintentional cherry-picking and "p-hacking"



PART 4: Quality control: Catch trials

Last, you can include in your fillers trials designed to filter out people who are not paying attention or otherwise not giving usable data. Broadly speaking, these are called 'catch trials'. They come in different types:

– '**Attention check**' trials look like normal experimental trials, but instruct the participant to do something very specific. For example, in a rating task, you might give someone the sentence 'This sentence is an attention check, so please do not rate its acceptability and instead just click number 4.'

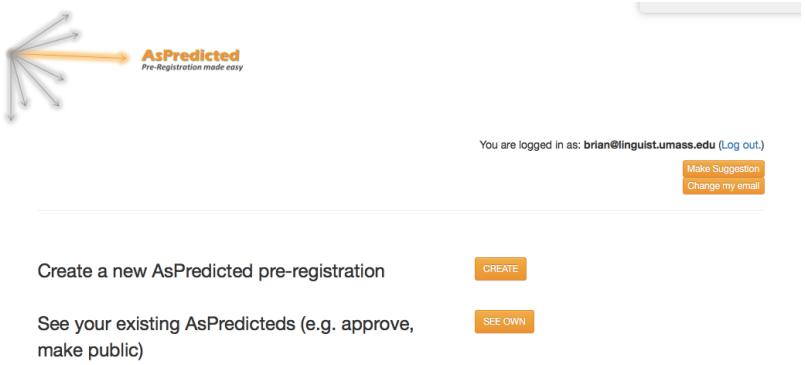
- **Catch trials** more generally are trials that you can use to objectively determine whether someone is paying attention. In some experiments, these might be trials with clear, objective answers, where inaccuracy indicates a lack of engagement with the task. In experimental syntax, this is largely unclear, because there are not 'right' and 'wrong 'answers.

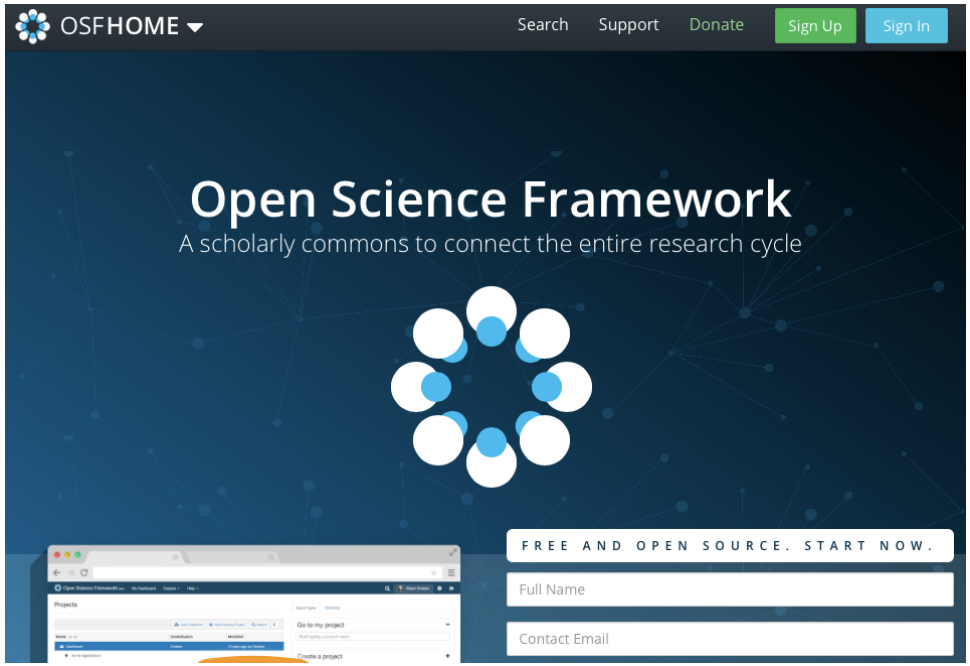
    One solution: set aside fillers for sentences whose ratings you are confident that most native speakers will agree upon, and determine a threshold for acceptable performance a prior (e.g. participants must rate sentences with correct agreement above incorrect agreement).

PART 4: Quality control

It is incredibly important to ensure data quality with data collected over the internet, and scanning for spammers using RTs and performance on catch trials is critical. However, down this road lies peril: it can be very tempting to find 'excuses' to throw out participants whose data seems problematic. You can convince yourself of lots of things post hoc.

We highly recommend that you **pre-registration** as a solution for this.





PART 4: Quality control

Now it's your turn.

- **Access sample.results.csv from the Dropbox.**
- **Open it in Excel or R, as you prefer.**
- **Inspect the performance of each of the four participants in the experiment.**
- **Determine which participants you think were taking the experiment in good faith, and which were spammers. Justify your decision.**