A streamlined approach to online linguistic surveys*

Michael Yoshitaka Erlewine and Hadas Kotek, MIT

{mitcho,hkotek}@mit.edu

Abstract

More and more researchers in linguistics use large-scale experiments to test hypotheses about the data they research, in addition to more traditional informant work. In this paper we describe a new set of free, open-source tools that allow linguists with little background in programming or experimental methods to post studies online. These tools allow for the creation of a wide range of linguistic tasks, including linguistic grammaticality surveys, sentence completion tasks, and picture-matching tasks. Our tools further help streamline the design of such experiments and assist in the extraction and analysis of the resulting data. Surveys created using the tools described in this paper can be posted on Amazon.com's Mechanical Turk, a popular crowdsourcing platform which mediates between 'Requesters' who can post surveys online and 'Workers' who complete them. This allows many linguistic surveys to be completed within hours or days and at low costs of $50 or less. Alternatively, researchers can host these randomized experiments on their own servers using a supplied server-side component.

1. Introduction

In recent years, researchers in linguistics have begun to integrate experimental components into their studies, in addition to more traditional informant work. With crowdsourcing tools such as Amazon's Mechanical Turk (henceforth: AMT), researchers can develop and test hypotheses on a wide variety of naïve speakers within days at very low cost. The purpose of this paper is to describe the process of creating an experiment, uploading it onto AMT and obtaining results using originally-developed tools that allow researchers who have little experimental experience to create linguistic surveys. These tools, as well as several example templates corresponding to different kinds of tasks, were written under a liberal, open-source license and are distributed for free online at https://github.com/mitcho/turktools.

The tools described here are designed for use with AMT, a popular online crowdsourcing website where companies or individuals (called *requesters*) can post small tasks (called Human Intelligence Tasks, or *HITs*) that cannot easily be automated, and therefore require human workers (called *workers*) for completion. HITs are generally small in nature (such as identifying the contents of an image), and generally high in quantity (it is not unusual for requesters to post thousands of tasks in a single batch). Requesters generally pay very little per HIT (e.g., several US cents) and retain the ability to Accept or Reject the results of each HIT before AMT sends

* Acknowledgements to be added

payment to the worker. HITs can be posted using an online interface (www.mturk.com), and their results are easily analyzable using mainstream statistical software (for more information, see Fort et al. 2011; Sprouse 2011; Gibson et al. 2011).

In the next section, we will briefly discuss the value of experimental methods and, in particular, crowdsourcing for theoretical linguistics. In the remainder of the paper, we describe *turktools*: a set of tools that enable easy use of online surveys for obtaining linguistic data, and a suggested workflow for their use:[1]

(1) **Skeletons**: HTML skeletons for various experiment types, including grammaticality judgment tasks with scales or yes-no choices, picture-matching tasks, tasks involving audio files, forced-choice completions.
(2) **Templater**: A Python script that creates HTML templates out of skeletons.
(3) **Lister**: A Python script that takes experimental items, including fillers, and organizes them into randomized, counterbalanced lists for presentation in the online survey.
(4) **Simulator**: A Python script that takes a template and randomized items in an AMT-style lists file and displays a version of the survey as it would appear in the online survey.
(5) **Decoder**: A Python script that takes a survey results file from AMT and decodes it into a format that can easily be analyzed by any statistical software.
(6) **Analysis.r**: A sample R script that assists in preparation and analysis of the online survey results.

While these tools were built with the AMT platform in mind, they by no means require the use of AMT. At the end of the paper we describe a server-side tool that allows for the hosting of AMT-style surveys online on a researcher's own server. For brevity, throughout the paper we describe the process of creating an experiment as if the researcher will upload the survey to AMT, but the majority of the process also applies to surveys hosted on a researcher's own server.

In addition to describing this workflow, we will touch on both practical and philosophical aspects of experimental design and analysis throughout.

2. The value of crowdsourcing in theoretical linguistics

Current research in theoretical linguistics relies heavily on *direct-informant* work – that is, obtaining linguistic data from one or several native-speaker consultants for a given language. This is often done by presenting the consultant with a sentence in a context or paired with a certain desired meaning and asking whether the sentence in appropriate in the given context

[1] At the time of writing, these tools require the use of Python 2.6.x or 2.7.x, available at http://python.org. However, the tools described here and their prerequisites and usage are subject to change. Please consult the latest information at https://github.com/mitcho/turktools before using these tools.

(Chomsky, 1965; Schütze, 1996). This method has been criticized for involving (a) relatively few participants, (b) a relatively small number of target stimuli, and (c) cognitive biases on the part of the researcher and participants (Cowart 1997; Edelman and Christiansen 2003; Featherston 2005; Ferreira 2005; Gibson and Fedorenko 2010; Marantz 2005; Myers 2009a,b; Schütze 1996; Sprouse 2009; Wasow and Arnold 2005).

Linguists have consistently offered strong arguments in response to these criticisms (e.g., Phillips and Lasnik, 2003; Marantz, 2005; Culicover and Jackendoff, 2010) and have reported formal experimental results that corroborate many informal experimental results (Featherston, 2005; Phillips, 2009). Many judgments obtained through informal methods and presented in journals and textbooks have been experimentally replicated using diverse experimental methods and tasks, thus showing that many of the criticisms cited above are unwarranted (Cable and Harris 2011, Munroe et al. 2010, Sprouse and Almeida 2010, 2012, ms., Sprouse 2011, Sprouse et al. 2012).

Following this latter work, we believe that experimental methods are not necessary in order to substantiate every linguistic claim made in the literature. Nonetheless, such methods are often useful and helpful in the process of theory-building. With crowdsourcing tools as AMT, researchers can develop and test hypotheses on a wide variety of naïve speakers within days at very low cost. Therefore, although we do not advocate the use of these methods in all cases and at all costs, we believe that it is vital for theoretical linguists to develop the skills that would allow them to use such methods when necessary.

The tools we describe here are designed specifically for use with AMT. We recognize that some concerns have been raised about the nature of the AMT participant pool in the context of linguistic experiments (see Fort et al. 2011), in particular as to the number of workers who complete most HITs that are posted on AMT. According to Fort et al. (2011), 20% of the workers complete 80% of the tasks, thus casting doubt on the diversity of the linguistic data obtained through AMT surveys. From our own experience with AMT, however, we believe that such concerns are unfounded. In all the AMT experiments conducted at the LAB between November 2010 and April 2013, a total of 4635 unique workers participated in at least one study; 643 (14%) participated in more than one study, 152 (3%) participated in more than two studies, and only 15 (0.3%) participated in more than five.[2] Thus, we believe that the diversity of our data is not jeopardized by the tendencies described in Fort et al. (2011).

[2] Data collected on April 24, 2013. The vast majority of experiments were on English and restricted IP addresses of workers to within the US.

3. Formulating a research question and choosing an experimental design

Before describing in more detail the technical tools that allow one to prepare a linguistic survey and upload it on AMT, the first task the linguist must address is formulating a clear hypothesis that can be tested using experimental methods. The simplest approach, and the one that we will be demonstrating in this paper, is crossing *factors* that have two *levels* each. For simplicity, we recommend having no more than two such factors, for a 2×2 design. For the remainder of this paper, we illustrate this idea with a well-known example from the literature: *there*-constructions have been argued to show a *definiteness effect*, whereby a definite or quantified DP cannot appear in a *there*-construction, (8a-c) (Milsark 1974, 1977 and much subsequent work).

(7) a. There is a boy in the room
b. *There in the boy in the room
c. *There is every boy in the room

In what follows we restrict our attention to the contrast between (7a-b). We may formulate the hypothesis that *definite DPs are ungrammatical in a there-construction*. To test this, we will compare sentences that differ in whether they contain a definite or an indefinite DP. We will cross this *factor* with whether or not a sentence contains a *there*-construction, thus yielding a *baseline* for the sentences of interest in our survey. We recommend that at a minimum, 10 different sets of items which vary these factors be created for any linguistic survey.[3] A sample item set for a simple survey testing the grammaticality judgments reported in the literature for the *there*-construction is given in (8), with predicted grammaticality judgments indicated.

(8) a. A boy is in the room.	*–definite, –there-construction*
b. The boy is in the room.	*+definite, –there-construction*
c. There is a boy in the room.	*–definite, +there-construction*
d. *There in the boy in the room.	*+definite, +there-construction*

The tools that we provide in this paper support diverse types of experiments, some of which we describe below. For each type of experiment we describe here, we provide a *skeleton* HTML file for creating such an experiment. We will discuss skeletons in more detail in Section 4. For other types of experiments used in the psycholinguistic literature as well as recommendations for design and data analysis, see Schütze and Sprouse (to appear) and citations therein.

3.1 Acceptability judgment task

Perhaps the most commonly used experimental task using in syntax and semantics research is the *acceptability judgment task* where participants are asked to judge the *acceptability* or *naturalness* of a sentence, or of a sentence-meaning pair. The dependent measure in this task is a rating on

[3] In fact, we believe it is important for any linguist to create multiple sets of data for any phenomenon of interest that he or she might be investigating, even if they do not then create a grammaticality survey to be uploaded to AMT or otherwise crowd-sourced to obtain judgments on a large scale.

There is _____ boy in the room.
○ the ○ a

some scale, where the ends of the scale may correspond to '*acceptable*'-'*unacceptable*', '*natural*'-'*unnatural*', '*grammatical*'-'*ungrammatical*', '*good*'-'*bad*', etc. Alternatively, a fixed 'Likert' scale, often with five or seven points, may be used. A similar task known as *magnitude estimation* requires participants to compare the acceptability of target sentences to some predetermined reference sentence (Bard et al. 1996).[4]

The acceptability judgment task is perhaps the easiest language task to implement for online surveys. We provide skeletons for both binary forced-choice (e.g. 'yes' or 'no') and Likert-scale implementations. Both types of skeletons are easily adaptable to allow for different answer choices, such as '*yes*'-'*no*'-'don't know', for a different number of options on a Likert scale, and for magnitude estimation tasks.

3.2 Forced-choice completion task

Another simple task is one in which a sentence or context is presented to the participant, where the task is either to choose which word or phrase better completes a sentence, or which whole sentence is better suited to describe a given situation. An example word completion task is given below in Figure 1. The dependent measure is the rate of selection of each choice in comparison with its competitor.[5]

Figure 1. A sample word completion task

3.3 Picture-matching tasks

For tasks in which a visual scene must be presented, a *picture-matching* task is often used. In this task, participants are shown a picture and are asked to choose which among several options best describe the sentence, or alternatively they are presented with a sentence and several pictures and must choose which among the pictures best matches the description in the sentence. The dependent measure is the rate of choice of a given sentence or picture compared to its competitors. In another variant of the task, participants are presented with a picture and a sentence that describes it, and must decide whether or not the sentence is a good description of the picture. The picture-matching skeleton we provide also allows for an easy implementation of the 'Covered Box' technique, which is particularly suited for detecting less preferred interpretations of ambiguous sentences (for details, see Huang, Spelke and Snedeker, ms.; Pearson, Khan and Snedeker 2010).

3.4 Additional modifications

[4] See Gibson et al. (2011) for more on this task, including its advantages and potential criticisms.
[5] Our supplied skeletons support choices introduced with buttons below the sentence, as in Figure 1, or with a drop-down menu.

For the experiment types described above (and others that users of our tools might develop), several modifications can be easily made.

- An audio file may be added.
- A context may be added before or after the target sentence.
- A comprehension question may be added either before or after the target sentence.
- For *cognitive load* paradigms, the target task may be interrupted by a secondary task.
- Items may be presented one at a time instead of all together on the same page.
- The sequence of events with which a single item is shown can be controlled – for example, it is possible to first show a picture, then mask the picture and ask a question about the picture. This setup is also useful for *cognitive load* paradigms.
- Timing information may be logged for each item or parts of an item that have been viewed.[6]

4. *Templater*: Creating an HTML template for use on Mechanical Turk

Once a research question is formulated and an experiment type has been selected to test it, the presentation of the survey can be determined. Surveys are presented on AMT using an HTML template. We recommend creating a template in two steps: (a) choosing an appropriate *skeleton* out of the ones offered on https://github.com/mitcho/turktools and editing it to fit your study, and then (b) creating a template from the skeleton using the *Templater*.

Skeletons are recipes for different experiment types that abstract away from the number of items that will be presented in a given study. They are stripped-down versions of HTML templates which should be edited to fit each particular design or experiment. We recommend the use of a programming text editor to edit these files.[7] For each experiment type we surveyed in Section 3, we provide a corresponding skeleton, (9).

(9) **Supplied skeleton files:**

a.	`binary.skeleton.html`	grammaticality judgment yes-no task
b.	`binary-image.skeleton.html`	picture matching yes-no task
c.	`completion.skeleton.html`	word-completion task
d.	`image-choice.skeleton.html`	picture matching task
e.	`likert.skeleton.html`	grammaticality judgment scale task
f.	`sentence-choice.skeleton.html`	sentence-completion task
g.	`completion-menu.skeleton.html`	completion task with drop-down menu

[6] This might require custom JavaScript programming in the template, and furthermore such timing information may not be very reliable. Our own templates utilize the jQuery JavaScript library (http://api.jquery.com/), and we recommend its use for such custom programming. See Ibex, "Internet Based Experiments", http://spellout.net/ibexfarm/ for a tool specifically designed for conducting timing-sensitive experiments online.
[7] Good, free options include *Notepad++* for Windows and *TextWrangler* for Mac.

A skeleton is itself an HTML file, but contains *substitution tags* which will be filled in by the Templater. These tags are all wrapped in double curly braces, i.e. `{{…}}`.

The skeletons that we provide all share the same basic structure, illustrated in Figure 2. The very top of the skeleton has a comment to remind users that a template should be created out of the skeleton to be uploaded on AMT.[8] Each experiment will be assigned a code, is used to instruct workers to only complete one survey of a certain type.[9] Next are the instructions for the survey, including any practice items, as well as a consent statement and contact information for the experimenters.[10] Finally, there is text requesting workers to turn on JavaScript in their browser. This enables a counter that is included in all the skeletons which helps workers make sure that they have answered all the questions in the survey.

[8] Comments, which will be completely ignored and removed by the Templater, are tags with an exclamation point, as in `{{!...}}`.

[9] This is often desired in linguistic experiments, since collecting multiple judgments from the same language consultant may result in skewed results of a study, if the researcher mistakenly treats these observations as independent.

[10] Approval for your proposed experiments from the Institutional Review Board (IRB) at your institution may be required before you run them on AMT. Please consult your institution's IRB.

{! This is a template skeleton; use templater.py! }}

urvey Code: {{code}}

LEASE COMPLETE AT MOST ONE {{code}} SURVEY. YOU WILL NOT BE PAID OR COMPLETING MORE THAN ONE SURVEY WITH THIS CODE.

nstructions

his questionnaire presents {{total_number}} English sentences. Each sentence contains a gap nd there are two options below the sentence for what should go in that gap. Choose the option hat sounds more natural to you. Here is an example:

There is ___ boy in the room.
○ the ○ a

n this example...

onsent Statement: ...

f your browser has JavaScript turned on, a counter will be displayed at the bottom of the page ndicating how many questions have been answered. **It is highly recommended that you turn on avaScript and use this tool before submitting to ensure that all questions have been nswered and you can receive payment.**

{#items}}
{number}}. {{field_1}} ___ {{field_2}}.
○ {{field_3}} ○ {{field_4}}

{/items}}

Demographic questions, questions about native language.

0 questions (out of {{total_number}} total) have been answered. If you submit now, you will not be paid.

After submitting this HIT, do NOT submit another HIT with survey code {{code}}. You will not be paid for completing more than one survey with this code.

Figure 2. An example skeleton file

Notice the substitution tags in this skeleton. The templater will replace `{{code}}` with the experiment's unique code, and `{{total_number}}` with the number of items presented in the experiment.

The main body of the skeleton is the items block, beginning with `{{#items}}` and ending with `{{/items}}`. The items block contains one sample item of the shape that all items in the survey will take. When a template is created out of the skeleton, this block will be duplicated as many times as there are items in your survey. The item contains the tag `{{number}}`, which will be replaced with the item number in the experiment, as well as a number of `{{field_`*n*`}}` tags. In this case, each item is made up of four fields: fields 1 and 2 are the parts of the sentence before and after the gap and fields 3 and 4 correspond to possible answer options. Each of these `{{field_`*n*`}}` tags will be replaced by a different "field" in the items file. The makeup of the items file will be described in the following section.

Below the items block are some demographic questions. We always ask participants to indicate their native language and any other languages they speak, although we clarify that payment is not contingent on their answers to these questions. Researchers may wish to add questions about gender, age, origin, etc. Finally, at the bottom of the skeleton is a counter to help workers ensure that they reply to all questions before submitting their survey. Below the counter is text warning participants not to accept a second HIT with the same code after submitting their survey.

Note that all parts of the skeleton can be edited, moved around or removed to suit the preferences of the researcher and the study. We recommend saving the changes to the skeleton in a dedicated folder that will contain all files pertaining to the same experiment. At least some changes will only have to be made once – for example, the consent statement should be the same in all surveys conducted under the same IRB approval.

After choosing a skeleton and editing it, a template must be created out of the skeleton using the *Templater*. The script *templater.py* should be saved in the same folder as the skeleton. The *templater.py* script and all other Python scripts described in this paper can be run in the following manner:

(10) **Executing a Python script (here *templater.py*):**

a. UNIX shell: move (`cd`) to the directory that contains the script and execute `python templater.py`.

b. Mac OS X: Right-click on the file in the Finder and choose Open With… > Python Launcher.

c. Windows: Double-click on the file.[11]

[11] For Mac OS X and Windows, these instructions assume that Python was installed using the standard Python binary packages provided at http://python.org. If not, use the UNIX shell instructions via the Terminal on Mac, and open using the `python` application on Windows.

The *Templater* will ask for three pieces of information, (11). Once these are provided, a template file will be created. The template file will be located in the same folder as the skeleton file and the *templater.py* script.

(11) a. the skeleton file name: one of the skeletons provided here or others you create yourself.
b. the number of items in your survey: including all experimental and filler items but not practice items, which are coded in the skeleton.
c. a survey code: any letter-number combination you choose.

Note that the *Templater* requires an items file that is formatted in a way that matches the skeleton you chose. That is, each item must have at least as many fields as there are fields in the sample item in the skeleton that you have chosen.[12] Thus, the choice of design and skeleton will dictate the format of your items file. We discuss the items file in Section 5.

5. Creating an items file

Once a design and a template have been decided on for your survey, items need to be created. For a simple 2×2 experiment crossing two *factors* (here: *definiteness*, *there-construction*) with two *levels* each (here: +/– *definite*, +/– *there-construction*), each target item set will contain four different *conditions*. A sample target *item set* for a sentence judgment task will look as in (12).

(12) **A sample item set:**

	Stimulus:	*Condition name:*	*Factor values:*
a.	A boy is in the room.	`indef-no.there`	*–definite, –there-condition*
b.	The boy is in the room.	`def-no.there`	*+definite, –there-condition*
c.	There is a boy in the room.	`indef-there`	*–definite, +there-condition*
d.	There is the boy in the room.	`def-there`	*+definite, +there-condition*

We recommend a naming convention whereby the *condition name* specifies the chosen value for each of the factors, separated by hyphens (13). For our *there*-construction experiment, this results in conditions names as in the middle column of example (12), where the settings of the two factors are shown in the third column and the first column shows the actual sentence that we would like participants to rate.

(13) **The structure of a condition name:**
`<factor 1 value>-<factor 2 value>-<factor 3 value>-…`

[12] It is possible to have more fields in the items file than in the template, but not the other way around. The use of *hidden fields* will be discussed in Section 5.

An experiment is composed of different *sections*, where normally at least one section will be used as *fillers*, and the others will be *targets*. This designation is made when using the Lister (section 6). Each *item* in the items file has two parts, each beginning on a new line:[13]

(14) a. **The item header**. The item header consists of four components, separated by spaces.
   (i) the symbol `#`;
   (ii) the name of the *section* (e.g., `target` in the example in Figure 3);
   (iii) the item set number within the *section* (we recommend 1, 2, 3, …);
   (iv) the *condition* name (e.g., `indef-no.there`, `def-no.there`, `indef-there`, `def-there` in our example). The condition names are not constrained in any way, but it is most helpful to use labels that reflect the experimental design, and in particular to describe the setting of the *factors* of interest for that item.

   b. **The item body**. The item body consists of *fields*, each on its own line.[14] Line *n* corresponds to the text that will be substituted for the text `{{field_n}}` in the skeleton file. A field may specify a sentence to be judged, choices for completions, a picture, an audio file, a context, a comprehension question, etc.

It is possible to add *hidden fields* to the body of an item by adding extra lines that do not have corresponding substitution tags in the HTML template. This is a way to specify additional information for a given item that is useful for the researcher but should not be available to the participants. For example, if a skeleton refers to `{{field_1}}` through `{{field_4}}` but not `{{field_5}}`, then anything that occurs in the 5th line (and onward) of an item's body will not be displayed on the screen and participants will not have access to this information. However, the data will still be part of the results file created by AMT after the experiment is completed and can be used as part of the data analysis. For example, the correct answers to comprehension questions or expected answers to filler items may be specified in this way.[15]

[13] This structure is very similar to that used for items files in Linger, written by Doug Rohde (http://tedlab.mit.edu/~dr/Linger/), and Gibson et al.'s (2011) Turkolizer. Note that our terminology differs slightly: Gibson et al refers to a group of *trials* together forming an *item*, which corresponds to our notions of individual *items* which are grouped together into *item sets*.
Furthermore, the structure of the item body in our *Lister* format is much more flexible, allowing for an arbitrary number of fields. Unlike in Linger and Gibson et al.'s Turkolizer, comprehension questions are not given a special status but are instead treated like any other field that could be added to an item. If a comprehension question is given or if some filler items are expected to have correct answers, these values should be specified as *hidden fields* in the items file; see discussion following (14).

[14] If a line break is required within a single text field, use the HTML code `<br/>` .

[15] If some items have more than one hidden field, it is important to make sure that each field is exclusively used for one purpose (e.g. field 5 for the correct answer to the item and field 6 for the expected answer to a comprehension question). It is possible to leave a field empty if a given item does not have a value associated with some field (e.g., no expected correct answer to an item, but there is an expected correct answer to the comprehension question – leave field 5 empty, as a blank line, and input the expected correct answer to the comprehension question in field 6).

```
# target 1 indef-no.there
a boy is in the room.
http://DOMAIN/experimentpictures/definiteness-effect/picture1.png

# target 1 def-no.there
The boy is in the room.
http://DOMAIN/experimentpictures/definiteness-effect/picture17.png

# target 1 indef.there
There is a boy in the room
http://DOMAIN/experimentpictures/definiteness-effect/picture1.png

# target 1 def-there
There in the boy in the room
http://DOMAIN/experimentpictures/definiteness-effect/picture1.png
```

Figure 3. Sample item set with four conditions in *Lister*-compatible format

The format for a sample item set in a *there*-construction study is given in Figure 3. For illustration, the experiment here pairs each sentence with a picture.[16] Each item consists of two fields, the first specifying the sentence and the second, a picture.[17] This corresponds to two substitution tags in the binary-image skeleton: `{{field_1}}` stands in for the sentence, and `{{field_2}}` for the URL in an `<img>` tag, specifying the picture to be shown.

6. *Lister*: Creating randomized lists of items to be uploaded on Mechanical Turk

The *Lister* takes an items file as input and returns a comma-separated-value (.csv) format file with randomized lists of items that can be uploaded to AMT. When you run *lister.py* the program will maximally ask for the following information:[18]

(15) a. the name of your items file (described in the previous section)
b. which sections should be treated as fillers
c. how many filler items you would like placed between each target item

[16] Notice that these items illustrated in Figure 3 are not compatible with the skeleton illustrated above in Figure 2. The items illustrated here are designed for use with a skeleton such as the supplied `binary-image.skeleton.html`.

[17] Resources such as images and audio or video files must be hosted separately, rather than included with the AMT experiment. We recommend hosting such resources on your own web space, for example in a hidden directory on your website, and including links to those resources in the items.

[18] We recommend that relevant scripts be copied into the same folder as your items file.

d. how many filler items you would like placed in the beginning and end of the experiment
e. how many lists you would like to create
f. whether or not you would like the reverse of each list to also be created, to help reduce any ordering effects that may occur

After this information is given, the *Lister* will create as many lists as requested, using a Latin Square design. That is, for each item set in a given section, an item for only one of the specified conditions for that item set will be included in a list. If the number of conditions does not match for all item sets in a section, an error will be given. Note that if all the item sets in a particular section have only one item, they will all be included, regardless of the condition names given.[19]

It is possible to dictate at least how many filler items must occur between each two target items and at the beginning and end of each list, if there are enough filler items. The algorithm creates counterbalanced and randomized lists for each section individually (including the fillers), and then spreads fillers between target items to minimally comply with the constraints given to the *Lister*. If additional filler items are left after this process, they are randomly distributed across the list. If there are not enough filler items to comply with the constraints given to the *Lister*, an error message will appear. Note that it is possible to have less filler items than target items, but in that case it is not possible to impose restrictions on their position relative to target items. It is also possible to have no sections designated as fillers. In this case, all the items in the file will be randomized first within their sections and then across other sections, but no other conditions can be imposed on the ordering.

If some items have more fields than others (for example, if some items have expected correct answers but others don't), the *Lister* will produce a message notifying the user of the discrepancy. The user may choose to go back and check that the file is correct or to continue with the current file. Assuming no such issues, the *Lister* will output a randomized lists file in CSV format (`xxxxxxx.turk.csv`) that can be uploaded onto AMT. For example, if the input to the *Lister* is the items file `definiteness-effect.txt`, the output will be `definiteness-effect.turk.csv`. The randomized lists file contains a row for each randomized list, indicating the actual text that the workers will see, with a header line at the top. AMT will substitute these strings into the HTML template for the participants to see.

7. Simulator: Verifying that the survey works properly

[19] In a filler section where each item set is made up of one item each, it may be helpful to use different condition names even though all items will be displayed. This may help make the analysis script easier to create. For example, some fillers may be designated as 'catch' items and accuracy can be calculated for those items only and then used to discard participants from the analysis.

With your items randomized into lists and your HTML template prepared, you are in principle ready to upload your survey onto AMT. However, before doing so, we recommend pre-testing your study using the *simulator.py* script provided here. The *Simulator* takes as input a randomized lists file (`xxxxxxx.turk.csv`) and a template and creates a version of the survey based on one of the lists in the items file (you can choose which list to simulate). The *Simulator* allows researchers to bypass the rather cumbersome process of uploading a survey onto AMT, or using AMT's *Sandbox* to simulate an experiment.[20] The output of the *Simulator* is an HTML file that can be opened in any web browser.

It is beneficial for researchers to complete their own study, for several reasons. Doing so helps spot any typos or errors in the rendering of the survey. If your items file includes hidden fields, you should make sure that all hidden fields are not displayed and all other fields are displayed as expected. Additionally, it is important to get a feeling for how hard the experiment is and how long it takes to complete it. This will help you determine payment for your HITs. Furthermore, because of the existence of *satisficing* behavior on AMT – the behavior of some workers to put in minimal effort and still get paid – it is useful to make sure that there are no easy strategies to complete the study without doing the linguistic task of interest. Strategies for dealing with such behavior include the addition of comprehension questions to items, the inclusion of 'catch' filler items that should have clear correct answers, and the use of diverse types of fillers that cannot all be solved in the same way.

8. Uploading surveys onto Mechanical Turk

After verifying that the survey is ready, it can be uploaded onto AMT. To do so, it is necessary to create an account and log in to AMT (https://www.mturk.com/mturk/welcome) as a *requester*. You can create a new project through the *Create* tab. The creation of a new project involves three steps: in the first screen, choose a name and description for your project and decide on number of workers and payment.[21] At this point, certain restrictions can be placed on the workers who will be offered this HIT. For example, it is often useful to restrict IP addresses of workers to a certain country or region or to request workers with a high approval rate. In the second screen, the template is uploaded. Click on 'Edit HTML Source', then open your template file in a text editor and copy all of its contents into the edit screen. Save this change, preview the resulting

[20] Amazon Mechanical Turk's Sandbox allows researchers to upload "dummy" versions of their experiment, much like using the *Simulator*. However, to use the Sandbox it is necessary to again go through the process of uploading an experiment that is also necessary on AMT itself. That is, one must go through all the steps described in Section 8, and re-do them all again in case any mistake is found. Then one must go through all the same steps a second time in order to create the actual survey on AMT; there is no way to transfer a survey from one platform to the other.

[21] The number of workers is specified *per HIT*, which in our terms means *per randomized list*. For example, suppose our target item sets have four conditions each, and *Lister* generated eight different randomized lists of items (four original lists and their reverse). The Latin Square design ensures that each individual target item will be presented in just two of the eight lists. If we then request 10 workers per HIT, we will be requesting a total of 80 workers to participate in our study, and each individual target item will be seen by 20 workers.

template in the third screen and finish. After creating a new project, you must upload your randomized lists file as a 'New Batch.' After you click the button, a window will open that will allow you to choose your `xxxxxxxx.turk.csv` file. Make sure that the surveys render properly and that the payment is appropriate, and post your study.

9. Downloading the results of surveys from Mechanical Turk

The results file is obtained through the *Manage* tab on AMT. You should visit the *Manage* tab while your survey is in progress to check on its progress, and approve or reject the results of the workers who have participated in your study. Before doing so, we recommend first downloading the results file (by clicking the 'Download CSV' button) in order to scrutinize your participants' behavior more closely. The results of a AMT survey are saved in the form of a .csv file with a name like '`Batch_999999_result.csv`'. This raw results file consists of a header row with the names of each of its columns, followed by one row for each participant's values for each of the columns. In addition to participants' responses to the experimental items, the raw results file also contains information about the submission such as time of submission, duration of the experiment, time of approval, approval/rejection rate, etc.

In some cases it may be necessary to reject submissions from workers who have completed more than one survey (see footnote 8 above). It may also be desirable to reject workers with low accuracy on comprehension questions and 'catch' filler items and those workers who failed to complete at least 80% of the study. To quickly identify workers who completed more than one survey, open the file in software such as Excel and sort by 'WorkerId.' For suggestions for additional exclusion criteria and how to identify workers who meet them using R, see Section 11 below. Once you have decided which workers to approve and which to reject, return to the *Manage* tab in your AMT webpage and approve or reject your workers there. In order to maintain a positive reputation as a requester on AMT, it is advisable to approve submissions in a timely fashion and to maintain a low rejection rate.

10. Decoder: Creating an output file in analyzable format

After your survey is completed, download a final copy of the raw results file from the AMT *Manage* tab. We recommend re-naming this file, e.g. `definiteness-effect.results.csv` (to continue using the same example as above), and saving it in the same folder as the other documents produced in the course of preparing your survey. Use the *decoder.py* script, which should be in the same folder. The *Decoder* will ask for the name of the raw results file (here: `definiteness-effect.results.csv`) and return a decoded file, here named `definiteness-effect.results.decoded.csv`. This file can be analyzed in any commonly used statistical software.

Unlike the raw results file, the decoded results file contains one row for each *item* in each list in the survey. For each item, the decoded file specifies information about the `Section`, `Condition`, `Item`, `List`, `PresentationOrder`, and all fields and participant responses associated with that item, including hidden fields. In addition, the decoder adds a number of `Factor` columns, corresponding to different factor values. These factor values are constructed by simply splitting the `Condition` value up by hyphens. For example, in our example where an item may have `Condition` value '`indef-no.there`', `Factor1` would be set to '`indef`' and `Factor2` would be set to '`no.there`.' This processing of condition names using the schema in (13) facilitates the analysis of the data later.

Metadata about a submission, including the `AssignmentId` (unique for each worker-survey pair), `SubmissionStatus`, `WorkerId`, `WorkTimeInSeconds` (for the entire survey) and answers to demographic questions, is duplicated for each item for a given worker in the file. Therefore, for each participant who worked on your survey there will be as many rows in the file as there were items in the survey,[22] and the meta-information about the submission will be the same in all rows from that submission.

11. Data preparation: Eliminating data from bad submissions

Before analyzing the results of your experiment, it is useful to prepare the data for analysis. Below are suggestions for steps that may be useful to follow, implemented in R. R is free, open-source statistics software available from http://www.r-project.org/. We recommend using RStudio, freely available at http://www.rstudio.com/ide/download/desktop, rather than the R console itself. You will need to have R installed on your computer to use RStudio. We recommend using the packages 'languageR' for simple analyses and 'lattice' for data visualization; these packages are available from CRAN (http://cran.r-project.org/), the Comprehensive R Archive Network. Packages can be installed using the R command `install.packages` or through the command Install Packages in the Tools menu in RStudio. To read more about this and any other functions in R, type a question mark before the function name into the R console, e.g. `?install.packages`, or go to the Help tab in the lower right-hand side of your screen in RStudio.

Once RStudio is running, you can create or edit an analysis script for your experiment. We have provided as part of the tools in this paper a simple analysis script `analysis.r` that you can modify for your needs. We recommend keeping the script in the same folder as the other scripts

[22] But note that if a worker completed two surveys, for example, there will be twice as many rows. All the rows corresponding to the first survey completed by this worker will share meta-data pertaining to the first submission and all rows corresponding to the second survey will share meta-data pertaining to the second submission. The AssignmentIds will be distinct, but the WorkerId will be the same across these rows. This allows us to easily identify such workers who completed multiple surveys. At the time of writing, AMT cannot itself enforce that each worker only complete one survey per logical experiment.

and data for your experiment. We further recommend having separate directories for each experiment that you create. It is better to edit analysis scripts rather than typing commands directly into the R-Studio (or R) console, because having a script ensures replicability and creates a record of your work.

After saving your analysis script in the same folder as the decoded results file, set the Working Directory to the source file location. This can be done from the Tools menu in RStudio or by using the command `setwd('xxxxxx')`, where `xxxxxx` is the full path to the directory where your results file is located on your computer.

Start by reading in the decoded results file into a variable, here results.

```
results <- read.csv('definiteness-effect.results.decoded.csv', header=TRUE)
```

We recommend cleaning up this file in at least four ways, by excluding the following types of participants: (a) non-native speakers, (b) participants whose submission you rejected on AMT, (c) participants who did not complete the survey and participants who completed more than one survey, (d) participants with low accuracy rates on 'catch' filler items and comprehension questions. Depending on the study, you may choose to use only some of the suggestions in (a)-(d) and perhaps to use other additional exclusion criteria.[23] Normally, we find that few participants need to be filtered from an experiment—usually fewer than 10%.

Non-native speakers can be excluded by applying the subset command to the results file and keeping only participants who said that they are native speakers (in this case, of English). We will use the answer given to one of the demographic questions in our skeletons, stored in the `english` column. The `subset` command will take a data frame and a criterion and return only those lines in the data frame that meet the criterion.

```
results <- subset(results, english == 1)
```

To reject participants whose submission you rejected on AMT, subset the file to return only those participants whose assignment status is not 'Rejected.'

```
results <- subset(results, AssignmentStatus != 'Rejected')
```

It is up to the researcher to decide how many missed items are too many and whether or not to reject participants who completed more than one survey. Below is a suggestion for rejecting anyone who completed more than one study or completed less than 80% of the study. Other values for `notEnough` can be used by replacing `0.8*surveyLength` with any other setting.

```
results$isNA <- ifelse(is.na(results$Choice) | results$Choice == '', 0, 1)
```

[23] For example, participants with demographic information that you are not interested in – e.g. participants from countries or states other than your focus, participants who are too young/old, or participants who speak a second language.

```
howMany <- aggregate(results$isNA, list(results$WorkerId), sum)

surveyLength <- as.numeric(aggregate(results$AssignmentId,
  by=list(results$AssignmentId), length)$x[1])
notEnough <- 0.8*surveyLength

didTooMany <- subset(howMany, x > surveyLength)$Group.1
didntDoEnough <- subset(howMany, x < notEnough)$Group.1

results <- subset(results, !(WorkerId %in% didTooMany))
results <- subset(results, !(WorkerId %in% didntDoEnough))
```

Finally, compute accuracy on comprehension questions and filler items that had expected values. Following the recommendations in Gibson et al. (2011), we recommend only accepting results from participants whose accuracy on such items is at least 75%, unless there is a substantial number of participants whose performance was below this rate. Because there are no memory demands in most linguistic surveys, a 75% accuracy rate should be easy for conscientious native speaker participants to achieve. If there are many participants who did not achieve this success rate, however, it is possible that there was something wrong with the answers to your survey and we recommend that you inspect your items before deciding to reject workers on AMT based on their accuracy.

If expected correct answers were included in hidden fields in your items file, they can easily be compared to the answers provided by participants in your study. For example, if expected answers to fillers were specified in the fifth line of their item bodies, it will be stored in the decoded results file in column `field_5`. The following code uses this to compute accuracy rates per participant and reject those whose accuracy rate is below 75%. You should inspect `accuracy` to see if any filler item had low accuracy rates and, if so, exclude it from the accuracy calculation.

```
results$isCorrect <- ifelse(is.na(results$field_5), NA,
  ifelse(results$field_5 == results$Choice, 1, 0))
accuracy <- aggregate(results$isCorrect, by=list(results$WorkerId), mean,
  na.rm = T)
disqualifiedWorkerIds <- subset(accuracy, x < 0.75)$WorkerId
results <- subset(results, !(WorkerId %in% disqualifiedWorkerIds))
```

12. Data analysis: basic suggestions

A full discussion of R and statistical analysis methods is beyond the scope of this paper, as we believe that a statistical analysis must fit the design of a given experiment and its research question. Therefore, suggesting an analysis in the absence of a specific experiment could lead to

the application of incorrect methods by researchers. Below we present some suggestions for simple first steps of looking at your data in R. For more detailed reading on data analysis, see DeGroot et al. (1986) for statistics; Gelman and Hill (2007) for regression; Baayen et al. (2008), Bates and Maechler (2009) and Barr et al. (2013) for mixed effects regressions; and Venables and Ripley (2000) and Baayen (2008) for R. See also Gibson et al. (2011) for some suggestions for preliminary analysis steps specifically for AMT surveys.

An important first step is to visualize the data. This helps identify unexpected behavior in the course of the experiment or during the conversion of data to the correct format in the R script. It also gives you an understanding of the general patterns in the data and therefore of how the statistics should come out. We use the library 'lattice' for visualization purposes in R. After it is installed, load it using the command `library(lattice)`. With this library, certain types of data like simple ratings and distribution of choices over a binary variable can be visualized for each condition by using the command:[24]

```
histogram(~ Choice | Factor1 * Factor2, data=results)
```

A second useful step is to aggregate, or compute the mean of each Choice made by participants according to certain criteria. For example, the first line of code below will return a mean of the Choice by participant, and the second line will return a mean by participant and two factors. These results will help detect interesting patterns in the data.

```
aggregate(results$Choice, by=list(results$WorkerId), mean)
aggregate(results$Choice, by=list(results$WorkerId, results$Factor1,
  results$Factor2), mean)
```

The analysis script provided with this paper also contains similar simple suggestions for dealing with completions data, which is somewhat different than forced-choice and Likert scale ratings. For additional suggestions for data visualization and basic analysis of linguistic grammaticality surveys using Likert scale ratings, see Gibson et al. (2011). Gibson et al's discussion is also applicable to similar designs using a binary decision (e.g. 'yes'-'no' or 'natural'-'unnatural'). See additionally the literature suggested there and in this paper for more detailed reading on the theory and implementation of more sophisticated statistical tests which go beyond the scope of this paper.

13. Dealing with unexpected results

[24] We recommend renaming the Factor columns to give each factor a meaningful name. This can be done easily using the `plyr` package using code such as the following, e.g. for our example:

```
library(plyr)
results <- rename(results, c("Factor1" = "Definiteness", "Factor2" = "There")
```

After getting an impression of the results of your study and performing a statistical analysis, it is sometimes the case that the results are not exactly as had been expected prior to the experiment. This result can be due to many factors, not least of which is that the hypothesis that the predictions were based on is incorrect for the data used in the experiment. However, before an experiment can be used to contest existing theories and put forward new ones, it is important to check what other factors may have led to the unexpected results.

As a first step, when filtering based on the accuracy of filler items or comprehension questions, it is important to make sure that the items are generally understood by participants as expected. For that purpose, it is useful to look at the mean accuracy for each item that had an expected value. See Section 11 for R code that can produce mean accuracy values per item. It is also useful to similarly treat target items: for all items that are expected to yield a certain result, aggregate the mean accuracy for those items and search for any exceptions that may be introducing noise into the analysis. It may be the case that certain items should be re-written and the experiment re-run, in order to avoid such effects of individual items.

Moreover, it is possible that there are different sub-groups of items that exhibit different patterns of judgments, only some of which were expected. In that case, we recommend trying to identify the factors that define these different groupings. Once this is done, the status of these grouping factors must be examined: it is possible that such factors should be considered relevant to the behavior at hand and hence added to the experiment as a factor that co-varies systematically with the other factors of interest. On the other hand, it is possible that the factor is not theoretically relevant, and in that case the noise should be controlled for by keeping just one kind of item as opposed to items that vary in their behavior with regard to this theoretically irrelevant factor.

After looking at individual items, it is also important to look at the behavior of individual participants in the experiment to identify any unusual behavior. Participants who were not paying attention to the task should have already been excluded from the analysis by means of filler accuracy and comprehension questions. However, it is possible that participants who have not been excluded nonetheless performed poorly in one or more conditions, or after a certain point in the study. It may be possible that some participants should be excluded, despite meeting the basic accuracy criteria, for consistently performing poorly in certain cases in an idiosyncratic manner not common among other participants. To identify such subjects, we recommend examining the accuracy of target and filler items by participant, to detect any unusual pattern in the data.

It is also possible that for some linguistic phenomena, participants who speak a second language behave differently than mono-lingual speakers. To test this possibility, we recommend comparing the results yielded by analyzing data from all native-speakers to data from only those

speakers who reported that they are native speakers who do not speak any foreign language.[25] This can be done as follows, using the value stored in the foreign column by our skeletons:

```
results <- subset(results, foreign == 1)
```

Lastly, we recommend splitting participants according to the speed with which they finished the study. For some tasks, it is reasonable to assume that participants who were very fast performed the task using a different strategy than slower participants. As a first step, it is possible to split the participant population by the average experiment completion time, classifying all participants who were faster than the mean as 'fast' and all the others as 'slow.' After this classification is done, for example using the code below, it is advisable to re-do the analysis that was previously performed on the entire population on each subgroup separately, to see whether there is variation among the groups.

```
meanTime <- mean(targets$WorkTimeInSeconds)

fastWorkers <- subset(targets, targets$WorkTimeInSeconds < MeanTime)

slowWorkers <- subset(targets, targets$WorkTimeInSeconds >= MeanTime)
```

More generally, it is often advisable not to rely on the results of the initial version of an experiment. Rather, those initial results should be considered *pilot* data. This data should then be scrutinized, any items that did not yield expected results should be corrected, and the experiment should be re-run to confirm the original findings. It is often the case that after more consideration has been given to the data and analysis, the original results will change or be refined.

14. Hosting AMT-style experiments on your own server with turkserver

In addition to posting an experiment on AMT, the tools we have provided with this paper can be used to generate surveys to be hosted on a researcher's own server. In order to do so, we have developed a program called *turkserver*, freely available at https://github.com/mitcho/turkserver under a liberal open-source license. *Turkserver* requires a server capable of serving PHP scripts, and is designed for Apache web servers. See the documentation at https://github.com/mitcho/turkserver for detailed instructions on installation and usage.

To use this option, construct the experiment as described in Sections 3-7. An experiment is set up with *turkserver* by uploading the template and item lists file (.turk.csv). A URL for the experiment is generated, and this URL can then be shared with potential participants. *Turkserver*

[25] We have found that the exclusion of speakers of multiple languages is a good way to control for the possibility that non-native speakers self-report themselves to be native speakers of a language, despite the fact that our surveys make it clear that payment is not contingent on the answer to this question. Generally, we have found that people are proud to report that they speak a second language, and will not see a need to hide this fact or lie about it (at least, not to report that they do not speak another language when in fact they do). Hence, using data only from mono-lingual speakers allows for a more conservative exclusion criterion and helps eliminate some noise in the data.

handles the random assignment of participants to different lists, displaying the survey using the chosen list of items, and recording the results in a format compatible with AMT's results files. The results can then be analyzed as in Sections 10-13.

There are some differences between running an experiment on AMT and on *turkserver*. AMT comes with many potential participants registered on the system, and therefore acts as a subject recruitment tool as well. On the other hand, the potential participant pool is limited to those people who have chosen to sign up to be a worker on AMT. With *turkserver* it is up to the experimenter to recruit subjects and send them to the experiment URL. Similarly, unlike AMT, *turkserver* does not handle payment to participants. *Turkserver* does, however, attempt to produce a unique AssignmentId for each submission, as AMT does, and to produce a WorkerId for each participant that is stable across experiments.

Running AMT-style surveys on *turkserver* can be useful for situations where the AMT subject pool does not include many speakers of a particular language, but such speakers can be recruited separately. It is also ideal for when the experimenter wishes to recruit participants from a particular group, such as a vetted subject pool, a class, or among colleagues. Furthermore, the ability to generate experiments that run both with and without AMT offers great flexibility. For example, one could use the exact same experimental materials (template and item list files) and conduct the experiment with the general public on AMT, and also on recruited non-naïve (working linguist) participants via *turkserver*, and compare the results in behavior.

15. Conclusion

We hope that the tools we have developed will allow linguists to quickly and efficiently test diverse hypotheses stemming from their work using linguistic surveys on Amazon's Mechanical Turk, following an initiative led by Gibson et al. (2011). The ease and speed with which linguistic hypotheses can be tested on AMT makes this platform a useful tool for theoretical linguists, and we hope that more linguists will learn to appreciate this tool and develop the skills necessary to use it. We hope that the tools we have developed will make experimental methods more accessible to researchers with less background in the use of such techniques and will thus make experimental methods more accessible to a wider range of working linguists.

References

Baayen, R. H. 2008. Analyzing linguistic data: a practical introduction to statistics using R. Cambridge, UK: Cambridge University Press.

Baayen, R.H., Davidson, D.J., Bates, D.M. 2008. Mixed-effects modeling with crossed random effects for subjects and items. Journal of Memory and Language, 59, 390-412.

Bard, E. G., Robertson, D. and Sorace, A. 1996. Magnitude estimation of linguistic acceptability. Language 72: 107–50.

Bates, D.M. and Maechler, M. 2009. lme4: Linear mixed-effects models using S4 classes. R package version 0.999375-32.

Barr, D.J., Levy, R., Scheepers, C., and Tily, H.J,. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. Journal of Memory and Language 68(3):255-278.

Cable, S. and Harris, J. 2011. On the Grammatical Status of PP-Pied-Piping in English: Results from Sentence-Rating Experiments. Ms, ling.auf.net/lingbuzz/001302/current.pdf.

Chomsky, N. 1965. Aspects of the theory of syntax. Cambridge: MIT Press.

Cowart, W. 1997. Experimental syntax: applying objective methods to sentence judgments. Thousand Oaks, CA: Sage Publications.

Culicover, P. W., and Jackendoff, R. 2010. Quantitative methods alone are not enough: Response to Gibson and Fedorenko. Trends in Cognitive Sciences, 14(6), 234-235

DeGroot, M. H., Schervish, M. J., Fang, X., Lu, L., and Li, D. 1986. Probability and statistics. Reading, MA: Addison-Wesley.

Edelman, S., and Christiansen, M. 2003. How seriously should we take Minimalist syntax? Trends in Cognitive Sciences 7. 60–1.

Featherston, S. 2005. Magnitude estimation and what it can do for your syntax: some wh-constraints in German. Lingua 115. 1525–50.

Ferreira, F. 2005. Psycholinguistics, formal grammars, and cognitive science. The Linguistic Review 22. 365–80.

Fort, K., Adda, G., and Cohen, K.B. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine?. Computational Linguistics 37, No. 2, Pages 413-20.

Gelman, A., and Hill, J. 2007. Data analysis using regression and multilevel／hierarchical models. Cambridge, UK: Cambridge University Press.

Gibson, E., and Fedorenko, E. 2010. Weak quantitative standards in linguistics research. Trends in Cognitive Science 14. 233–4.

Gibson, E., Piantadosi, S., and Fedorenko, K. 2011. Using Mechanical Turk to Obtain and Analyze English Acceptability Judgments. Language and Linguistics Compass 5/8: 509–524.

Huang, Y.T., Spelke, E. and Snedeker, J. ms. What exactly do numbers mean?

Marantz, A. 2005. Generative linguistics within the cognitive neuroscience of language. The Linguistic Review 22. 429–45.

Milsark, G. 1974. Existential sentences in English. Doctoral dissertation, MIT.

Milsark, G. 1977. Toward an explanation of certain peculiarities of the existential construction in English. Linguistic Analysis 3, 1- 29.

Munro, R., Bethard, S., Kuperman V., Tzuyin Lai, V., Melnick, R., Potts, C., Schnoebelen, T., and Tily, H. 2010. Crowdsourcing and language studies: the new generation of linguistic data. Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, Los Angeles, CA.

Myers, J. 2009a. Syntactic judgment experiments. Language and Linguistics Compass 3. 406–23.

Myers, J. 2009b. The design and analysis of small-scale syntactic judgment experiments. Lingua 119. 425–44.

Pearson, H., Khan, M. and Snedeker, J. 2010. Even more evidence for the emptiness of plurality: An experimental investigation of plural interpretation as a species of implicature. Proceedings of SALT, 20, 489-508.

Phillips, C., and Lasnik, H. 2003. Linguistics and empirical evidence: Reply to Edelman and Christiansen. Trends in Cognitive Sciences, 7, 61-62.

Schütze, C. 1996. The empirical base of linguistics: Grammaticality judgments and linguistic methodology. Chicago: University of Chicago Press.

Schütze, C. and Sprouse, J. To appear. *Judgment Data*. *Research Methods in Linguistics*. Edited by Devyani Sharma and Rob Podesva.

Sprouse, J. 2009. Revisiting satiation: Evidence for an equalization response strategy. Linguistic Inquiry, 40, 329–341.

Sprouse, J. 2011. A validation of Amazon Mechanical Turk for the collection of acceptability judgments in linguistic theory. Behavior Research Methods 43.

Sprouse, J and Almeida, D. 2010. A quantitative defense of linguistic methodology. Irvine: University of California, ms.

Sprouse, J and Almeida, D. To appear. *The empirical status of data in syntax: A reply to Gibson and Fedorenko. Language and Cognitive Processes.*

Sprouse, J and Almeida, D. 2012. *Assessing the reliability of textbook data in syntax: Adger's Core Syntax. Journal of Linguistics* 48: 609-652.

Sprouse, J., Schütze, C., and Almeida, D. Ms. A comparison of informal and formal acceptability judgments using a random sample from Linguistic Inquiry 2001-2010.

Wasow, T., and Arnold, J. 2005. Intuitions in linguistic argumentation. Lingua 115. 1481–96.