



MIT Open Access Articles

Using mechanical turk to obtain and analyze English acceptability judgments

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation	Gibson, Edward, Steve Piantadosi, and Kristina Fedorenko. "Using Mechanical Turk to Obtain and Analyze English Acceptability Judgments." <i>Language and Linguistics Compass</i> 5.8 (2011): 509–524. Web.
As Published	http://dx.doi.org/10.1111/j.1749-818X.2011.00295.x
Publisher	John Wiley & Sons, Inc.
Version	Author's final manuscript
Citable link	http://hdl.handle.net/1721.1/73969
Terms of Use	Creative Commons Attribution-Noncommercial-Share Alike 3.0
Detailed Terms	http://creativecommons.org/licenses/by-nc-sa/3.0/

**Using Mechanical Turk to Obtain and
Analyze English Acceptability Judgments**

Edward Gibson¹, Steve Piantadosi¹ and Kristina Fedorenko²

¹ Department of Brain and Cognitive Sciences, MIT

² Smith College, Northampton, MA

Send correspondence to:

Edward Gibson, MIT 46-3035, Cambridge, MA 02139, USA

Email: egibson@mit.edu

Manuscript dated April 4, 2011; comments welcome

Abstract

The prevalent method in theoretical syntax and semantics research involves obtaining a judgment of the acceptability of a sentence / meaning pair, typically by just the author of the paper, sometimes with feedback from colleagues. The weakness of the traditional non-quantitative single-sentence / single-participant methodology, along with the existence of cognitive and social biases, has the unwanted effect that claims in the syntax and semantics literature cannot be trusted. Even if most of the judgments in an arbitrary syntax / semantics paper can be substantiated with rigorous quantitative experiments, the existence of a small set of judgments that do not conform to the authors' intuitions can have a large effect on the potential theories. Whereas it is clearly desirable to quantitatively evaluate all syntactic and semantic hypotheses, it has been time-consuming in the past to find a large pool of naïve experimental participants for behavioral experiments. The advent of Amazon.com's Mechanical Turk now makes this process very simple. Mechanical Turk is a marketplace interface that can be used for collecting behavioral data over the internet quickly and inexpensively. The cost of using an interface like Mechanical Turk is minimal, and the time that it takes for the results to be returned is very short. Many linguistic surveys can be completed within a day, at a cost of less than \$50. In this paper, we provide detailed instructions for how to use our freely available software in order to (a) post linguistic acceptability surveys to Mechanical Turk; and (b) extract and analyze the resulting data.

1. Introduction

The prevalent method in theoretical syntax and semantics research involves obtaining a judgment of the acceptability of a sentence / meaning pair, typically by just the author of the paper, sometimes with feedback from colleagues. The results obtained using this method are not necessarily generalizable because of (a) the small number of experimental participants (typically one); (b) the small number of experimental stimuli (typically one); and (c) cognitive biases on the part of the researcher and participants (Schütze, 1996; Cowart, 1997; Edelman & Christiansen, 2003; Wasow & Arnold, 2005; Ferreira, 2005; Marantz, 2005; Featherston, 2007; Myers, 2009a, 2009b; Gibson & Fedorenko, 2010, in press). Whereas in some circumstances gathering non-quantitative language evidence may be warranted – e.g., when gathering data is difficult, such as when there is limited access to a single speaker of a dying language, who may only be available for testing for a brief time – in most situations, it is best to have multiple experimental materials, presented to multiple naïve experimental participants. Consequently, the above authors have argued that future research in syntax and semantics should adopt the following standards, common in all behavioral sciences, when gathering behavioral data such as acceptability judgments: (a) include many experimental participants, all of whom are blind with respect to the research questions and hypotheses; (b) include many experimental materials, in order to rule out effects due to idiosyncratic properties of individual experimental items (e.g., particular lexical items used in the critical sentences); (c) include distractor materials so that the experimental manipulations are not obvious to the participants; and (d) present counterbalanced lists of items to different experimental participants (in different random orders) to avoid effects of presentation order.

One common but fallacious argument that is sometimes given is that it would be inefficient for the field to evaluate empirical claims in a rigorous quantitative manner. For example Culicover & Jackendoff (2010, p. 234) say the following: “It would cripple linguistic investigation if it were required that all judgments of ambiguity and grammaticality be subject to statistically rigorous experiments on naive subjects...”. However, contrary to Culicover & Jackendoff’s claim, we think that the opposite is true: the field’s progress is probably slowed by *not* doing quantitative research. There are two related reasons for our position. First, many judgments in published papers turn out not to survive experimental evaluation. Consequently, theoretical claims based on such findings have no basis. And yet, some of these claims are propagated in the field for years, leading researchers on mistaken paths, and slowing real progress. We elaborate this observation below. And second, publishing papers that don’t adhere to the methodological standards of neighboring fields like cognitive science and neuroscience has the undesirable effect that the results presented in these papers will be ignored by researchers in these fields, because these results are based on non-quantitative methodology. This second point is critical for attracting new quantitatively-trained researchers into the field of language research.

Gibson & Fedorenko (in press) provide many instances of judgments in published papers that turn out not to survive experimental evaluation (see also Schütze, 1996; Wasow & Arnold, 2005; and Featherston, 2007, for additional examples like these). For example, consider multiple-wh-questions and embedded multiple-wh-clauses in English, where it has been shown experimentally that there is a subject / object asymmetry, such that the items in which the wh-subject (e.g., *who*) is clause-initial as in (1) are more acceptable than items in which the wh-object (e.g., *what*) is clause-initial as in (2) (see Kuno & Robinson, 1972, for the original claim;

see Arnon et al., 2006; Fedorenko & Gibson, submitted, for experimental evidence in support of this claim):

- (1) a. Who bought what?
- b. Mary wondered who bought what.

- (2) a. * What did who buy?
- b. * Mary wondered what who bought.

Whereas the contrast between (1) and (2) is supported by quantitative evidence, a further purported contrast is not. In particular, it has been claimed by Bolinger (1978) and Kayne (1983) that sentences like those in (2) become more acceptable with a third wh-phrase added at the end:

- (3) a. ?* What did who buy where?
- b. ?* Mary wondered what who bought where.

This empirical claim has resulted in several theoretical attempts to explain it, by postulating additional mechanisms and / or assumptions (e.g., Kayne, 1983; Pesetsky, 2000; Richards, 2001). It turns out, however, that when this intuition is evaluated using quantitative experimental methods, it does not hold. In particular, Clifton, Fanselow & Frazier (2006) evaluated the relative acceptability of examples like (1b), (2b) and (3b), and whereas they found a reliable difference between examples like (1b) on the one hand and (2b) and (3b) on the other, they found no difference between examples like (2b) and (3b). More recently, Fedorenko &

Gibson (in press) evaluated the same kinds of materials in supportive contexts, and replicated Clifton et al., critically finding no difference between examples like (2b) and (3b).¹

There are many examples in published syntax and semantics articles where the reported judgments do not match those of typical native speakers of the language in question. Gibson & Fedorenko (in press) argue that the reason for this disparity is a combination of the weak experimental method together with the existence of *cognitive biases* (Wason, 1960; see Nickerson, 1998, for a summary of many kinds of cognitive biases), such as (a) the experimenter will often have a confirmation bias favoring the success of the predicted result, with the consequence that he/she will tend to justify ignoring intuitions from speakers that do not fit the predicted pattern; and (b) the experimenter, and in some cases the people providing the judgments, will be unconsciously affected by the hypotheses in making their judgments about the relevant experimental materials. Furthermore, social factors may affect the evaluations of hypotheses, so that, for example, people providing the judgments may be biased because they subconsciously want to please the experimenter.

Even if most of the judgments in an arbitrary syntax / semantics paper can be substantiated with rigorous experiments, the existence of a small set of judgments that do not conform to the authors' intuitions can have a large effect on the potential theories. The problem lies in not knowing which judgments are reliable, and which are not. A typical syntax /

¹ An anonymous reviewer raises the possibility that longer sentences might be rated as less natural, which might indirectly account for the lack of contrast between (2) and (3). In particular, sentences like (3a) and (3b) might be less natural because of their length, but more syntactically grammatical, such that length-based naturalness and grammaticality cancel each other out in the ratings.

If the syntactic factor at play in this explanation – “superiority” – is the same as the one that is intended to explain the difference between (1) and (2) (as is claimed in the relevant literature), this possibility seems unlikely. In particular, Fedorenko & Gibson (in press) observed a large acceptability difference between (1b) and (2b), which differ in their grammaticality in the superiority dimension, and are matched for length. Thus, the acceptability effect associated with superiority violations is large. It would then need to be the case that the lack of difference between (3a) and (3b) is due to a large syntactic effect being canceled out by an equally large length-based naturalness effect. This would entail that adding a single word would generally make sentences greatly less acceptable, which seems unlikely to be true. In any case, it would be easy to evaluate this hypothesis in more depth on Mechanical Turk.

semantics paper that lacks quantitative evidence includes judgments for 50-100 sentences / meaning pairs, corresponding to 50-100 empirical claims. Furthermore, suppose that 90% of the judgments from such a paper are correct, which is probably a conservative estimate. This means that in a paper with 70 empirical claims, 63 of those 70 claims are correct. But which 63? There are over a billion ways to choose 63 items from 70, corresponding to over a billion different theories.²

Amazon.com's Mechanical Turk

Amazon.com's Mechanical Turk³ is a crowd-sourcing internet marketplace interface that can be used for collecting behavioral data quickly and inexpensively. Jobs ("Human Intelligence Tasks" or HITs) are posted on the Mechanical Turk website (<https://www.mturk.com/mturk/welcome>) by *requestors*. The jobs are then performed by *workers*: people anywhere in the world, who want to get paid to do the posted jobs. Currently, the payment is through U.S.-based credit cards, so most of the workers and requestors are in the United States. We gathered demographic information on 519 Turk users for a recent experiment from September & October 2010 (see Piantadosi, Tenenbaum, Goodman, in prep.). Of these

² Recently, Sprouse & Almeida (2010) have provided evidence suggesting that 98% of the intuitive judgments from a current textbook (Adger, 2003) are supported by quantitative evaluation on Mechanical Turk. Whereas testing materials from a linguistics textbook is certainly useful, the high validity of the judgments there may be overestimated compared to typical research in syntax and semantics. In particular, the judgments in a textbook are likely the best ones from the literature because the people writing the text can filter the less good ones (they don't have the same cognitive biases as the authors of the sources). Thus, in order to get an estimate of the validity of judgments in typical syntax and semantics papers, one has to sample the judgments in the source articles. In addition, the fact that Sprouse & Almeida were able to gather judgments on so many materials so quickly demonstrates the ease of using Mechanical Turk as a way to gather linguistic data (Sprouse, 2011).

³ Wikipedia explains the source of the name "Mechanical Turk" as follows: *The name Mechanical Turk comes from "The Turk" a chess-playing automaton of the 18th century, which was made by Wolfgang von Kempelen. It toured Europe beating the likes of Napoleon Bonaparte and Benjamin Franklin. It was later revealed that this "machine" was not an automaton at all but was in fact a chess master hidden in a special compartment controlling its operations. Likewise, the Mechanical Turk web service allows humans to help the machines of today perform tasks they aren't suited for.*

participants, 58% were female. Users of Turk tend to be generally educated: only 2% do not have a high school degree, 40% have only a high school degree and 41% completed up to a college degree. The remaining 17% report completing a graduate degree. In addition, most users spend less than 8 hours on Turk each week: 35% reported spending 0-4 hours on Mechanical Turk each week, 23% reported 4-8 hours, 17% reported 8-12 hours, and the remaining 25% spent more than 12 hours a week. The distribution of ages was largely unimodal, with a mean age of 34 and a range of 18 to 77 in our sample (participants younger than 18 were told they would not be able to participate). The form of the jobs to be performed on Mechanical Turk is typically anything that is difficult to do automatically, so that human labor is preferred, such as “identifying objects in a photo or video, performing data de-duplication, transcribing audio recordings, or researching data details” (<https://www.mturk.com/mturk/help?helpPage=overview>). Mechanical Turk is therefore perfect for testing hypotheses in behavioral psychology, including language (Sprouse, 2011).

What makes Mechanical Turk particularly attractive for gathering linguistic data is the easy access of a large participant pool. Previous internet-based research (e.g., WebExp, Keller et al. 1998; Corley & Scheepers, 2002; Keller et al., 2009; and MiniJudge, Myers, 2009a, 2009b) does not have this feature.

Acceptability Judgment Surveys

Perhaps the most commonly used method in syntax and semantics research is the *acceptability judgment task* where participants are asked to judge the acceptability / naturalness of a sentence / meaning pair. The dependent measure in this task is a rating on some scale where the ends of the

scale correspond to “acceptable” / “not acceptable”, “natural” / “unnatural”, “grammatical” / “ungrammatical”, “good” / “bad”, etc. In a quantitative version of this task, a fixed scale – a “Likert scale” – with five or seven points is typically used. Alternatively, a geometric scale is sometimes used where the acceptability of each target sentence is compared to a reference sentence. This latter method is known as *magnitude estimation* (Bard et al., 1996).⁴

Although some researchers have criticized the acceptability judgment method because it requires participants to be aware of language as an object of evaluation, rather than simply as a means of communication (Edelman & Christiansen, 2003), there are two major advantages to this method: (1) it is a simple task, which people can do quickly and reliably; and (2) results from acceptability judgment experiments are highly systematic across speakers, and correlate with other dependent measures, presumably because the same factors affect participants’ responses across different measures (Schütze, 1996).

Because of the simplicity of the acceptability judgment task, it is perhaps the easiest language task to implement for use on Mechanical Turk. The software that we provide here implements a Likert scale with five points. However, it is straightforward to adapt the html template that appears on Mechanical Turk in order to use a different scale (e.g., either a different number of fixed points on a Likert scale, or a magnitude estimation scale).

In the remainder of this paper, we describe four pieces of software – available at <http://tedlab.mit.edu/software/> – which enable easy use of Mechanical Turk for obtaining linguistic acceptability judgment data:

⁴ Although some researchers have hypothesized that magnitude estimation allows detecting more fine-grained distinctions than Likert scales (Bard et al., 1996; Keller, 2000; Featherston, 2005), controlled experiments using both Likert scales and magnitude estimation suggest that the two methods are equally sensitive (Weskott & Fanselow, 2009, in press; Fedorenko & Gibson, submitted).

- (1) `turkolizer.py`: A python program that takes lists of experimental and filler items, and organizes them in randomized, counterbalanced lists for presentation on Mechanical Turk.
- (2) `tedlab-turk-survey1.html` / `tedlab-turk-survey2.html`: Html templates for a 150-item survey on Mechanical Turk for items with one / two comprehension questions each
- (3) `tedlab-turk-survey1-format.R` / `tedlab-turk-survey2-format.R`: R programs to format an output file from Mechanical Turk into an analyzable format assuming that there is one / two comprehensions question per experimental trial.
- (4) `tedlab-turk-survey-analysis.R`: An R program to analyze the results from the survey using linear mixed effects regression methods.

2. Formatting items for presentation on Mechanical Turk

The *turkolizer* program takes an input file in *turkolizer* format (described below), and returns (a) a file to be uploaded on Mechanical Turk and (b) two files to help in decoding the acceptability judgment results that are returned from Mechanical Turk. A *turkolizer* file includes materials for any number of experiments, usually including one with the special name *filler*.

A fully-crossed factorial experimental design consists of an $n_1 * n_2 * \dots * n_m$ set of conditions, crossing the first factor's n_1 conditions by the second factor's n_2 conditions ... by the m th factor's n_m conditions (for some elaboration on how to construct factorial experimental designs, see Schutze, 1996; Cowart, 1997; and Myers, 2009b). Consider a two-factor example, examining the acceptability of wh-questions with two or three wh-phrases as discussed above,

where two factors are manipulated, and each factor has two conditions: subject-object order (subject-object, object-subject) and number of wh-phrases (3wh, 2wh) (Fedorenko & Gibson, in press). So, in this case, $n_1 = 2$ and $n_2 = 2$, resulting in a 2x2 design. The example in (4) presents one item in this experimental design, with all four conditions:

(4)

- a. subject-object, 3wh: Peter was trying to remember who carried what when.
- b. object-subject, 3wh: Peter was trying to remember what who carried when.
- c. subject-object, 2wh: Peter was trying to remember who carried what.
- d. object-subject, 2wh: Peter was trying to remember what who carried.

An experiment in turkolizer format⁵ consists of a set of *sub-experiments*, each of which consists of a set of *trials*. One of the sub-experiments is typically designated as *filler*, with one condition, also typically labeled *filler*. This sub-experiment contains distractor items for the other sub-experiments being run. Typically, there are at least twice as many filler items as items in a sub-experiment.

In each sub-experiment, the *trials* have the format shown in Figure 1. The turkolizer format for the items in (4) is presented in Figure 2. A *trial* consists of three parts, each starting on a new line:

- (a) The trial header. The trial header consists of four components, separated by spaces:
 - (i) the symbol "#";
 - (ii) the name of the experiment (e.g., “expt-name-1” in the example in Figure 1;

⁵ The turkolizer format is very similar to the format in the program Linger, written by Doug Rohde, which can be downloaded at: <http://tedlab.mit.edu/~dr/Linger/>,

“2WH3WH” in the example in Figure 2);

(iii) the item number within the experiment (a number from 1 to the number of items);

(iv) the condition name (e.g., “condition-1-a” in the example in Figure 1; “3wh_SO”, “3wh_OS”, “2wh_SO”, and “2wh_OS” in the example in Figure 2). The condition names are not constrained in any way, but it is most perspicuous to use labels that reflect the experimental design.

(b) The trial body. The trial body starts on the line following the trial header. It consists of any number of lines up until the trial question(s), or the next trial header (if there are no trial questions). Carriage returns are treated as carriage returns to be displayed as part of the item.

(c) The trial questions (optional for the purposes of turkolizer, but highly recommended from the standpoint of experimental design, as discussed below). The trial questions consist of a (possibly empty) set of questions about the trial body. The format for each trial question is as follows:

? Question? Answer

Each question is initiated and terminated by a question mark. Following the question is the correct answer value. The exact form of this answer needs to be present among the possible answers to be presented to participants.

We will present examples with Yes/No questions, so that the possible answers are always “Yes” or “No”. As discussed in Section 4 below, the format of the answers needs to

match the format of the answers in the Mechanical Turk presentation template, so “Yes”/“No” cannot be replaced with “Y”/“N” or “yes”/“no” for the template that we provide.

Every trial within and across sub-experiments that are being run together in one experiment must include the same number of trial questions.

Because the goal of Mechanical Turk participants is to finish the work as quickly and efficiently as possible (because they are paid by the job, not the hour), it is wise to include at least one comprehension question for each trial to ensure that participants read and understand the materials.

```
# expt-name-1 item-num condition-1-a
An experimental item
which can be over multiple lines
like this
? Question 1? Yes
? Question 2? No
```

Figure 1: The turkolizer format for a trial in an experiment.

```
# 2WH3WH 1 3wh_SO
Peter tried to remember who carried what when.
? Did Peter try to carry something? No

# 2WH3WH 1 3wh_OS
Peter tried to remember what who carried when.
? Did Peter try to carry something? No

# 2WH3WH 1 2wh_SO
Peter tried to remember who carried what.
? Did Peter try to carry something? No

# 2WH3WH 1 2wh_OS
Peter tried to remember what who carried.
```

```
? Did Peter try to carry something? No
```

Figure 2: The turkolizer format for the four conditions in example (1).

3. Creating lists of experimental items to be uploaded on Mechanical Turk

To run *turkolizer*, simply include the program `turkolizer.py` in the current directory, and type `"python turkolizer.py"` in a UNIX shell.⁶ The program will then ask for some information, as in the following example:

```
egibson$ python turkolizer.py

Please enter the name of the text file: myexperiment.txt
Please enter the desired number of lists: 80
Please enter the desired number of in-between trials: 1
Please enter the desired number of fillers in the beginning of
each list: 3

Processing the text file...

Number of experiments: 2

Experiment: filler
  - 40 items
  - 1 conditions
  - 40 trials
  - number of questions: 1
  - conditions:
['filler']

Experiment: 2WH3WH
  - 20 items
  - 4 conditions
  - 80 trials
  - number of questions: 1
  - conditions:
['2wh_OS', '2wh_SO', '3wh_OS', '3wh_SO']

Performing a check of the parameters...
Creating a latin square...
```

⁶ If you don't use mac or linux, Cygwin (<http://www.cygwin.com/>) allows you to run unix shells on windows.

```

Creating LCM ( 4 ) lists...
Creating 80 lists...
Randomizing each list...
Creating two csv files...
----- DONE! -----
egibson$

```

Figure 3: An example run of *turkolizer*.

The program will then ask for the name of the *turkolizer* input file, which needs to be located in the current directory. In the above example, this file is named `myexperiment.txt`. The *turkolizer* program will then ask for the number of lists that the user wants to generate. This is the number of participants that the user wants to run in the experiment. Each presentation list that the program generates will be a different randomized order of the experimental items, across the different sub-experiments. The program takes one version of each item in a sub-experiment, such that there is the same number of items from each condition in a sub-experiment in each list that is generated, using a Latin Square design. The details of how the Latin Square is computed are often not relevant to many experimental designs. But sometimes one wants to balance other properties of the materials across items, without creating a new factor. In such circumstances, one needs to know the details of how the Latin Square is computed. Assuming that there are k items in a sub-experiment with i conditions, with j items in each condition (such that $i * j = k$), the first list in the Latin Square is generated by taking the first j items in condition 1, the next j items in condition 2, ... and the last j items in condition i . The next list in the Latin Square is generated by taking the second j items in condition 1, the next j items in condition 2, ... and the first j items in condition i , and so on for the rest of the lists in the Latin Square.

The number of presentation lists that the user should request should generally be a multiple of the least common multiple (LCM) of the number of conditions in each of the

experiments, so that each experiment will be run on a balanced number of participants in each of the lists in its Latin Square. For example, an experiment consisting of one sub-experiment with six conditions, another with four conditions, and a third with five conditions, should be run on some multiple of $\text{LCM}(6, 4, 5) = 60$ participants. In the simpler example above, there is a single sub-experiment with four conditions (along with a filler sub-experiment, which has only one condition). Thus, the number of lists that were run in this experiment was a multiple of four (80, in this case, thus giving 20 participants for each of the sets of items in the Latin Square for the experiment).

Next, the program asks the user to enter the number of items from other experiments and fillers that the user wants to have between experimental items in any given sub-experiment. We often enter “1” here, so that no two items from the same sub-experiment appear back-to-back. Finally, the user is asked to enter the number of filler items that they would like to initiate each presentation list.

If there are inconsistencies between the user’s desired experimental set-up and the sub-experiments that are in the turkolizer input file (e.g., not enough fillers to suit the desires of the user; or an incorrectly formatted input file), then error messages will appear. Otherwise, the turkolizer program will now print some information about each of the sub-experiments to the screen, and will output three files to the current directory. If the input file was named `myexperiment.txt`, then the output files will be named as follows:

`myexperiment.turk.csv`

`myexperiment.decode.csv`

`myexperiment.correct.csv`

The file `myexperiment.turk.csv` is a comma-separated-value (csv) file containing a line for each of the randomized lists of trials, with a header line at the top.

The file `myexperiment.decode.csv` is a csv file containing information indicating which item in each experimental list is in which condition. The file `myexperiment.correct.csv` is a csv file that contains the correct answers to the comprehension questions. The last two files are used in decoding the responses from participants on Mechanical Turk (see Section 7).

4. Creating a presentation template on Mechanical Turk

Before you can upload your experiment onto Mechanical Turk, you will need to log in to <https://www.mturk.com/mturk/welcome> as a *requestor*. Once you have set up an account for yourself and are logged in, you will need to design a template for your “HIT” (Human Intelligence Task). We have set up two templates at <http://tedlab.mit.edu/software/> for users to download and edit for linguistic acceptability tasks. These are templates for surveys with 150 trials with one / two comprehension questions each. The beginning of the first of these templates – `tedlab-turk-survey1.html` – is provided in Figure 4. Each of these templates is initiated by a consent statement which has been approved by MIT’s Institutional Review Board. (If you represent a university or institution, then you need to obtain approval for your proposed experiments before you run them on Mechanical Turk.) There are two questions following the consent statement: (1) “What country are you from?” followed by a list of five English-speaking countries; and (2) “Is English your first language?”. We generally restrict the location of the IP addresses of the respondents to our surveys to be from the US, so that the population is fairly

uniform in their dialect (American English). People are then paid for their participation, as long as they can answer the comprehension questions following each item. We then generally restrict analyses to native English speakers from the US. Note that because we pay people no matter whether they indicate that they are native English speakers or not, participants have no motivation to pretend to be native English speakers if they are not.

Choose the survey template that you desire, then edit out the items that you do not need, and upload the resulting template at the “Design” section of the Mechanical Turk website (https://requester.mturk.com/bulk/hit_templates). The current templates are set up with five possible acceptability rating values: “Extremely unnatural”, “Somewhat unnatural”, “Possible”, “Somewhat natural”, and “Extremely natural”. In analysis, these ratings are translated to the numbers 1 (Extremely unnatural) – 5 (Extremely natural).

Note that the templates that we provide require that the only possible answer choices given to the participants are “Yes” or “No” (not “Y” and “N”, or “yes” and “no”, etc.). Thus, if other answers are required, then the templates need to be edited accordingly.

Sentence understanding

Consent Statement
 By answering the following questions, you are participating in a study being performed by cognitive scientists in the MIT Department of Brain and Cognitive Science. If you have questions about this research, please contact Edward Gibson at egibson@mit.edu. Your participation in this research is voluntary. You may decline to answer any or all of the following questions. You may decline further participation, at any time, without adverse consequences. Your anonymity is assured; the researchers who have requested your participation will not receive any personal information about you.

What country are you from? USA Canada UK Australia / New Zealand India Other

Is English your first language? Yes No

Please read each sentence, then answer the question immediately following, and provide the requested rating.

Please note that there is a correct answer for each question.

Because some Mechanical Turk users answer questions randomly, we will reject users with error rates of 25% or larger. Consequently, if you cannot answer 75% of the questions correctly, please do not fill out the survey.

Note: **Please read the sentence** before answering the question and giving the rating!

Sentence: \${trial_1}

Question: \${question_1_1}

Yes No

Sentence rating:

Extremely unnatural Somewhat unnatural Possible Somewhat natural Extremely natural

Figure 4. A consent statement and instructions in a Mechanical Turk linguistic acceptability survey.

5. Uploading the materials on Mechanical Turk

Once the template is designed and saved, click on the “Publish” button on the Mechanical Turk website (<https://requester.mturk.com/bulk/batches/new>). Now select the appropriate template for use, and follow the Mechanical Turk instructions to upload your experimental item lists in the file `myexperiment.turk.csv` created in Section 3. Select the appropriate rate of payment, and publish the survey on Mechanical Turk.

Depending on your rate of pay (anything over \$2 / hour will probably generate very high demand for your survey), you should get your results within a few hours, or perhaps a day.

6. Downloading the results of your experiments from Mechanical Turk

Click on the “Manage” button on the Mechanical Turk website to see the progress of your survey. Once the survey is complete, click on the “Results” button on the website. You can approve or reject the results from each participant at this point, but we recommend first downloading the results, in order to see how the participants did in answering your comprehension questions, and approving / rejecting them based on their performance. Download the results of your survey by clicking the “Download CSV” button.

Once you have analyzed the response accuracy rates of participants (see Section 7), return to the “Manage” section of your Mechanical Turk webpage. Click on the results of the survey again. We recommend accepting results from each participant who completed at least 80% of the survey, and who answered the comprehension questions correctly at least 75% of the time. Because there are no memory demands on these rating surveys, this accuracy rate should be easy for conscientious participants to achieve. Few participants should be below 75% accuracy. If there are a substantial number, then there may be something wrong with the answers to your survey.

7. Transforming your Mechanical Turk output into an easily analyzable format

The output from Mechanical Turk consists of a csv file with a name like “Batch_999999_result.csv”. This file consists of a header row consisting of names for each of its columns, followed by one row for each participant’s values for each of the columns.

You are now ready to process the results into a format that can be used for statistical analysis. You perform this preprocessing step using the R analysis program `tedlab-turk-survey1-format.R`, or `tedlab-turk-survey2-format.R`. Use `tedlab-turk-survey1-format.R` if you have one comprehension question per experimental trial. Use `tedlab-turk-survey2-format.R` if you have two comprehension questions per experimental trial.

Rename the relevant file to suit your experiment (e.g., `myexperiment-turk-survey-format.R`) and start R (e.g., by double-clicking on the program in your finder window on a Mac). You will need to have installed R on your computer to use this program. R is freely available from <http://www.r-project.org/>. We generally use a package called "lattice" for visualization, and a package "languageR" for analyses; both are available from CRAN (<http://cran.r-project.org/>), the Comprehensive R Archive Network. You can generally install packages using the R command `install.packages`. You can read about this and other functions in R by typing a question mark before the function name into the R console:

```
?install.packages.
```

Once R is running, you can edit the file `myexperiment-turk-survey-format.R`. Three lines in this file need to be edited:

```
turk.output.file <- "Batch_XXXXXX_result.csv"
decode.file <- "xxx.decode.csv"
correct.file <- "xxx.correct.csv"
```

First, edit the `turk.output.file` name "`Batch_XXXXXX_result.csv`" so that it matches your turk output file name. Second, edit the `decode.file` name so that it is the name of the file for item decoding from Section 3 (e.g., `myexperiment.decode.csv` from Section 3). Third,

edit the correct.file name so that it is the name of the file with the correct answers for comprehension questions from Section 3 (e.g., myexperiment.correct.csv from Section 3).

Now load the myexperiment-turk-survey-format.R into the R shell. In the R shell, type “participant.summary.info”, which is the name of a variable which includes summary information about the participants in the experiment: (a) Answer.country: what country they are from; (b) Answer.English: whether English is their native language; (c) Accuracy: their accuracy rate across the comprehension questions in the experiment; (d) NAPercent: their rate of blank answers; and (e) ResponseCount: the number of responses that this participant was presented with. An artificial example is provided in Figure 5.

```
> participant.summary.info
```

	Participant	Accuracy	NAPercent	ResponseCount	Answer.country	Answer.English
1	A10GDZ6LSVPO9C	0.9306931	0.009803922	102	USA	yes
2	A1338BOIT0IEBT	0.9019608	0.000000000	102	USA	yes
3	A149N3D2483WXS	0.9215686	0.000000000	102	USA	yes
4	A14X7JN1CL4I5K	0.9405941	0.009803922	102	USA	yes
5	A16JTXYL3MQ77P	0.8811881	0.009803922	102	USA	yes
6	A16Q77JGW17NA	0.8910891	0.009803922	102	USA	yes
7	A1APGSHXCIUAM5	0.7647059	0.000000000	102	CAN	yes
8	A1DU3I57BN5079	0.9509804	0.000000000	102	USA	yes
9	A1DXXMYU2HRU6N	0.9313725	0.000000000	102	USA	yes
10	A1GUGP8HBVU668	0.9019608	0.000000000	102	USA	yes
11	A1JRBA1EMMGWRJ	0.9313725	0.000000000	102	USA	yes
12	A1KL8HZHJ6AWC9	0.7128713	0.009803922	102	USA	yes
13	A1MRHIA5YGGBY4	0.9300000	0.019607843	102	USA	yes
14	A1NUA58PEFFN45	0.9702970	0.009803922	102	USA	no
15	A1OK4SKO6L7OND	0.9285714	0.588235294	102	USA	yes
16	A1OLRUT93TXWEP	0.9411765	0.000000000	102	USA	yes
17	A1QRJ6ZGZNMTRC	0.9117647	0.000000000	102	USA	yes
18	A1RWNYJA5X25YH	0.9215686	0.000000000	102	USA	yes
19	A1T4AT8PX9GNY5	0.9117647	0.000000000	102	USA	yes
20	A1X86TQB8KO6OR	0.9509804	0.000000000	204	USA	yes

Figure 5. Sample participant.summary.info. The participant ID number is provided by Mechanical Turk.

The final line in the R formatting code filters (a) participants that are not from the USA, such as participant 7 from Canada in Figure 5; (b) participants who are not native English speakers, such as participant 14 in Figure 5; (c) participants with accuracy rates below 75%, such as participant 12 in Figure 5; (d) participants with an NAPercent rate of 20% or higher, such as participant 15 in Figure 5; and (e) participants who filled out more than the total number of responses on one survey, such as participant 20 in Figure 5, who has filled out the survey twice (which had 102 ratings), resulting in twice as many possible answers.

In practice, we find that few participants need to be filtered from an experiment, usually fewer than 5% or 10%. (Some of the participants were artificially created in the above example in order to illustrate each filtering step.)

8. Analyzing your results using regression analyses in R

Whereas a full discussion of R and statistical analysis methods is beyond the scope of this paper, we will present preliminary first steps in analyzing this type of data in R. For more details of analysis, see Gelman & Hill (2007) for regression; DeGroot & Schervish (1986) for statistics; and Venables & Ripley (2000) and Baayen (2008) for R.

Before you analyze your data, you need to edit the file `tedlab-turk-survey-analysis.R`. We recommend putting all analysis scripts and data in a separate directory for each experiment, and editing analysis scripts rather than just typing into the R console. Having a script ensures replicability and creates a record of what you have done. In `tedlab-turk-survey-analysis.R`, we first define the appropriate factors for each experiment as fields in the data-frame for analysis. This is necessary because the `turkolizer` script requires factor names

and values in a single field (“Condition”), but it is more convenient – for analysis purposes – to have one column for each factor, with different values for each factor value. This is done by editing the R code lines in Figure 6.

```

### this example set of factors assumes two factors: Factor1 and Factor2
### The values for Factor1 are Factor1_Value1 and Factor1_Value2
### The values for Factor2 are Factor2_Value1, Factor2_Value2, and Factor2_Value3

mydata$Factor1 <- NA
mydata[mydata$Condition == "Factor1_Value1_Factor2_Value1",]$Factor1 <- "Factor1_Value1"
mydata[mydata$Condition == "Factor1_Value1_Factor2_Value2",]$Factor1 <- "Factor1_Value1"
mydata[mydata$Condition == "Factor1_Value1_Factor2_Value3",]$Factor1 <- "Factor1_Value1"

mydata[mydata$Condition == "Factor1_Value2_Factor2_Value1",]$Factor1 <- "Factor1_Value2"
mydata[mydata$Condition == "Factor1_Value2_Factor2_Value2",]$Factor1 <- "Factor1_Value2"
mydata[mydata$Condition == "Factor1_Value2_Factor2_Value3",]$Factor1 <- "Factor1_Value2"

# set the order of factors for the regression (first is the baseline)
mydata$Factor1 <- factor(as.character(mydata$Factor1), levels=c("Factor1_Value1",
"Factor1_Value1"))

```

Figure 6. R code to be edited in order to define the appropriate factors to be analyzed in each experiment.

In order to analyze each sub-experiment using a regression, we first provide factor names for each condition in the sub-experiment. Suppose that sub-experiment 1 has two factors, Factor1 and Factor2. Furthermore, suppose that Factor1 has two values Factor1_Value1 and Factor1_Value2, while Factor2 has three values Factor2_Value1, Factor2_Value2 and Factor2_Value3. The code in Figure 6 creates one factor for each of the conditions in the sub-experiment, and associates the appropriate element in the data-frame with each value for that factor. All the user has to do here is replace the factor names and value names provided in the code in `tedlab-turk-survey-analysis.R` with the appropriate factor and value names for the conditions in their sub-experiments.

Once this step has been completed, the first – also, most important – step is to visualize the data. This helps to ensure that unexpected things have not happened in the course of the experiment (such as participants answering the same way for all ratings) or during the conversion of data to the correct format. It also gives you an understanding of what the general patterns in

the data look like: how the statistics should come out. As mentioned above, we use the “lattice” library for visualization. With this library, simple ratings can be visualized for each condition by entering:

```
histogram( ~ Answer.rating | Factor1 * Factor2, data=mydata )
```

This command plots a histogram of Answer.rating for each combination of Factor1 and Factor2. To visualize the data without separating by both factors you can use:

```
histogram( ~ Answer.rating | Factor1 , data=mydata )
```

to collapse over Factor2, but not Factor 1, and

```
histogram( ~ Answer.rating, data=mydata )
```

to collapse over both factors.

A second useful command, `aggregate`, computes the mean rating for every value of any column of `mydata`. For instance to see the mean ratings by participant you could compute

```
aggregate(mydata$Answer.rating, by=list(mydata$WorkerId), mean)
```

This command returns a new data frame with one line for each value of `WorkerId`, corresponding to the mean of `Answer.rating`. To compute the mean by participant and condition, additional columns can be added to the “by” part of this command, as in

```
aggregate(mydata$Answer.rating, by=list(mydata$WorkerId,
mydata$Factor1, mydata$Factor2), mean)
```

Note that the result of `aggregate` can be saved into a new variable, forming a new data frame that can be manipulated further.

The most basic and conservative statistical tests you can do are nonparametric, meaning that they do not make assumptions about the types of distributions the data come from. Many of these are built in to R: one simple one is the Wilcoxon test, which can be called in R using `wilcox.test`, and is roughly used to compare means between conditions. However, we generally find that parametric tests – in particular, mixed effect regressions – work reasonably well on rating data, even though the rating data does not strictly meet all of the assumptions of the regression. These regressions can be called in R by first loading the “languageR” package (via “`library(languageR)`”) and then running the following command:

```
l <- lmer( Answer.Rating ~ Factor1 * Factor2 + (1 | Participant) + (1
| Item), data=mydata)
```

This command creates a regression predicting the rating from the two factors and random intercepts for participants and items.⁷ The p values for this regression should be computed using

```
pvals.fnc(1)
```

which will return estimated p values for this regression.

Use of these methods should be done with care and require an understanding of the logic of regressions and ways of checking to be sure that the analysis does the “right” thing. Particular care must be used on rating data because it does not strictly satisfy the assumptions of the regression. For instance, the ratings are bounded, which contradicts the assumptions underlying the use of normal linear regression. There do exist more sophisticated regression models such as *ordinal regression* for these kinds of data, but as long as ratings are not near an extreme of the scale, linear regression works well (see Gelman & Hill, 2007, for an excellent introduction to regression models).

9. Concluding remarks

It is our hope that this software will enable syntax and semantics researchers to be able to test their hypotheses using the acceptability judgment task on Mechanical Turk quickly and easily. In our experience, the largest remaining bottle-neck in doing syntax and semantics research is constructing multiple instances of the relevant construction, which can admittedly be time-consuming, especially if they have to be normed for various factors like frequencies of

⁷ You can also include random slopes for each factor by participant by replacing the string “(1 | Participant)” with “(1 + Factor1 * Factor2 | Participant)”. See Gelman & Hill (2007) and Baayen (2008) for discussion.

critical words or constructions or plausibility. But of course Mechanical Turk can be used to perform plausibility norming and syntactic frequency norming (by means of a sentence completion task). So even material construction is greatly speeded up by the existence of Mechanical Turk. The advent of Amazon.com's Mechanical Turk allows many empirical claims in syntax and semantics to be tested easily. We hope that it will further the adoption of quantitative methods, advancing the empirical standards of linguistics to those found in other psychological sciences.

References

- Arnon, I., Snider, N., Hofmeister, P. Jaeger, T.F. & Sag, I.A. (2007) Cross-Linguistic Variation in a Processing Account: The Case of Multiple Wh-Questions. *Proceedings of Berkeley Linguistics Society*, 32.
- Baayen, R.H. (2008). *Analyzing linguistic data: A practical introduction to statistics using R*. Cambridge University Press, Cambridge, UK.
- Bard, E. G., Robertson, D. and Sorace, A. (1996). Magnitude Estimation of Linguistic Acceptability. *Language*, 72.
- Bates, D., Maechler, M. and Dai, B. (2008). lme4: Linear mixed-effects models using S4 classes. R package version 0.999375-27. <http://lme4.r-forge.r-project.org/>
- Bolinger, D. (1978). Asking more than one thing at a time. In Hiz (Ed.), *Questions*. Dordrecht: D. Reidel.
- Clifton, C., Jr., Fanselow, G., & Frazier, L. (2006). Amnestying superiority violations: Processing multiple questions. *Linguistic Inquiry*, 37, 51-68.
- Corley, M.M.B. & Scheepers C. (2002). Syntactic priming in English sentence production: Categorical and latency evidence from an internet-based study. *Psychonomic Bulletin and Review* 9, 126-131.
- Cowart, W. (1997). *Experimental syntax: Applying objective methods to sentence judgments*. Thousand Oaks, CA: Sage Publications.
- DeGroot, M.H., Schervish, M.J., Fang, X., Lu, L. & Li, D. (1986). *Probability and statistics*. Addison-Wesley, Reading, MA.
- Edelman, S. & M. Christiansen. (2003). How seriously should we take Minimalist syntax? *Trends in Cognitive Sciences* 7: 60-61.
- Featherston, S. (2005). Magnitude estimation and what it can do for your syntax: Some wh-constraints in German. *Lingua*, 115, 11, 1525–1550.
- Fedorenko, E. & Gibson, E. (In press). Adding a third wh-phrase does not increase the acceptability of object-initial multiple-wh-questions. *Syntax*.
- Fedorenko, E. & Gibson, E. (Submitted). Syntactic parallelism as an account of superiority effects: Empirical investigations in English and Russian.
- Ferreira, F. (2005). Psycholinguistics, formal grammars, and cognitive science. *The Linguistic Review*, 22, 365-380.

- Gelman, A. & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, Cambridge, UK.
- Gibson, E. & Fedorenko, E. (2010). Weak quantitative standards in linguistics research. *Trends in Cognitive Science*, 14, 233-234.
- Gibson, E. & Fedorenko, E. (In press). The need for quantitative methods in syntax and semantics research. *Language and Cognitive Processes*.
- Kayne, R. (1983). Connectedness. *Linguistic Inquiry*, 14, 223-249.
- Keller, F. (2000). Gradience in Grammar: Experimental and Computational Aspects of Degrees of Grammaticality. PhD Thesis, University of Edinburgh.
- Keller, F., Corley, M., Corley, S., Konieczny, L., & Todorascu, A. (1998). *WebExp: A Java toolbox for web-based psychological experiments* (Technical Report No. HCRC/TR-99). Human Communication Research Centre, University of Edinburgh.
- Keller, F., Gunasekharan, S., Mayo, N. & Corley, M. (2009). Timing Accuracy of Web Experiments: A Case Study Using the WebExp Software Package. *Behavior Research Methods* 41:1, 1-12
- Kuno, S. and Robinson, J. (1972). Multiple wh-questions. *Linguistic Inquiry*, 3, 463-487.
- Marantz, A. (2005). Generative linguistics within the cognitive neuroscience of language. *The Linguistic Review*, 22, 429-445.
- Myers, J. (2009a). Syntactic judgment experiments. *Language and Linguistics Compass* 3, 406–423.
- Myers, J. (2009b). The design and analysis of small-scale syntactic judgment experiments. *Lingua* 119, 425–444.
- Nickerson, R.S. (1998). Confirmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, 2, 175-220.
- Piantadosi, S.T., Tenenbaum, J.B, Goodman, N. (in preparation). Representation languages for complex compositional concepts. Department of Brain and Cognitive Sciences, MIT.
- Pesetsky, D. (2000). *Phrasal movement and its kin*. Cambridge, MA: MIT Press.
- R Core Development Team, (2008). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. <http://www.R-project.org>
- Richards, N. (2001). *Movement in language*. Oxford: Oxford University Press.

- Schütze, C. (1996). *The empirical base of linguistics: Grammaticality judgments and linguistic methodology*. Chicago: University of Chicago Press.
- Sprouse, J. (2011). A validation of Amazon Mechanical Turk for the collection of acceptability judgments in linguistic theory. *Behavior Research Methods*, 43.
- Sprouse, J. & Almeida, D. (2010). The 469 data points that form the empirical foundation of generative syntactic theory are at least 98% replicable using formal experiments. Manuscript, University of California, Irvine.
- Venables, W.N. & Ripley, B.D. (2000). *S programming*. Springer Verlag.
- Wason, P.C. (1960). On the failure to eliminate hypotheses in a conceptual task. *Quarterly Journal of Experimental Psychology*, 12, 129-140.
- Wasow, T. & Arnold, J. (2005). Intuitions in linguistic argumentation. *Lingua*, 115, 1481-1496.
- Weskott, T. & Fanselow, G. (2009). Scaling issues in the measurement of linguistic acceptability. In: Featherston, S. & Winkler, S. (eds.): *The Fruits of Empirical Linguistics. Vol.1: Process*. Berlin, New York: Mouton de Gruyter, pp. 229-245
- Weskott, T. & Fanselow, G. (in press). On the Informativity of Different Measures of Linguistic Acceptability. *Language*.