To appear in *Computational Linguistics and Chinese Language Processing*
(http://www.aclclp.org.tw/journal/index.php) 2007 ms.

MiniJudge: Software for small-scale experimental syntax

James Myers
Graduate Institute of Linguistics, National Chung Cheng University, Minhsiung, Taiwan

Abstract

MiniJudge is free online open-source software to help theoretical syntacticians collect and analyze native-speaker acceptability judgments in a way that combines the speed and ease of traditional introspective methods with the power and statistical validity afforded by rigorous experimental protocols. This paper shows why MiniJudge is useful, what it feels like to use it, and how it works.

Keywords: syntax, experimental linguistics, JavaScript, R, generalized linear mixed effect modeling

1. Introduction

Every theoretical syntactician has faced the problem of native-speaker judgments that, instead of correlating neatly with the theoretical issue at hand, vary unexpectedly across sentences or speakers. This problem is generally dealt with uninsightfully, either by fiat ("assuming these judgments are correct...") or by dropping the data entirely, along with the potentially important theoretical issue they may inform. Perhaps forty years ago Chomsky (1965:19-20) was right to declare that "[t]he critical problem for grammatical theory today is not a paucity of evidence but rather the inadequacy of present theories of language to account for masses of evidence that are hardly open to serious question." However, as Schütze (1996:27) observed (ten years ago now), "the questions linguists are now addressing rely crucially on facts that are indeed 'open to serious question'."

Acceptability judgments reflect grammatical knowledge, but as data they themselves are merely a form of linguistic behavior, parallel to the accuracy rates or reaction times measured by psycholinguists (Chomsky, 1965; Penke & Rosenbach, 2004). From a cognitive science perspective, then, the ideal solution to the linguists' data woes would be for them to adopt the rigorous experimental protocols honed over the two centuries scientists have been struggling to extract information about mental structure from often messy behavioral data. When linguistic judgments are collected with such protocols, they often (though not always) reconfirm the essential validity of empirical claims made on the basis of more informal methods, but they can also go beyond simple reconfirmation (or falsification) to reveal hitherto unsuspected theoretical insights. Recent examples of the growing experimental syntax literature include Sorace & Keller (2005), Featherston (2005), and Clifton, Jr., Fanselow, & Frazier (2006); Cowart (1997) is a user-friendly handbook.

Unfortunately, full-fledged experimental syntax is complex, forcing the researcher to spend a lot of time on work that is not theoretically very interesting. Fortunately, the complexity of an experiment need only be proportional to the subtlety of the effects it is trying to detect. Most judgments are very clear (perhaps because a grammar must be shared by a speech community, and hence must be "obvious" enough to learn), and so are reliably detected even with traditional "trivially simple" methods. Very subtle or variable judgments, or hypotheses involving gradient degrees of acceptability or interactions between grammar and processing, may require full-fledged experimental methods. But in the large area in between, a compromise seems appropriate, where methods are powerful enough to yield statistically valid results, yet are simple enough to apply quickly. This is where MiniJudge comes in.

MiniJudge (Myers, 2007) is a family of software tools to help theoretical syntacticians

design, run, and analyze linguistic judgment experiments quickly and painlessly. Though MiniJudge experiments are small-scale experiments, testing the minimum number of speakers and sentences in the shortest amount of time, they use statistical techniques designed to maximize interpretive power from small data sets. In this paper I first define more precisely what makes a MiniJudge experiment small-scale. I then walk through a sample MiniJudge experiment on Chinese. Finally, I reveal MiniJudge's inner workings, which involve some underused or novel statistical techniques. The most updated implementation of MiniJudge is MiniJudgeJS, which is written in JavaScript, HTML, and the statistical language R (www.r-project.org). It has been tested most extensively in Firefox for Windows XP, but it also seems to work properly in Internet Explorer and Opera in Windows, Firefox, Opera and Safari for Macintosh, and Firefox for Linux (though line breaks are not handled properly in R for Linux). There is now also a Java implementation called MiniJudgeJava (Chen, Yang, & Myers, 2007) with somewhat different internal algorithms and interface, but which otherwise works the same as the JavaScript version described in this paper.

2. Small-scale experimental syntax

Experimental syntax (at least the type carried out in laboratories) generally adheres rather closely to conventions developed in psycholinguistics: multiple stimuli and subjects (naive ones rather than the bias-prone experimenters themselves), factorial designs (where materials represent all possible combinations of the experimental factors, to avoid confounds and make it possible to study interactions between factors), filler items (to prevent subjects from guessing which materials are the theoretically crucial ones), counterbalancing (so no subject is presented with "minimal pairs" differing only in theoretically relevant factors), continuous response measures (e.g., open-ended judgment scales, to permit the use of standard statistical techniques like the analysis of variance, or ANOVA), and statistical analysis (to determine how unlikely the obtained results were to have occurred by chance alone). Together these conventions can make the designing, running, and analysis of syntax experiments quite time-consuming and intimidating to the novice, especially if the experiment ends up merely reconfirming results already suspected from informally collected judgments.

In a small-scale judgment experiment, however, only the most essential of these conventions are maintained, as summarized in Table 1.

Table 1. Key characteristics of small-scale experimental syntax

Very few sentence sets (about 10)	No fillers
Very few (naive) speakers (about 10-20)	No counterbalancing of sentence lists
Maximum of two binary factors	Random sentence order
Binary *yes*/*no* judgments	Order treated as a factor in the statistics

The very small number of sentence sets and speakers (in comparison with the typical psycholinguistics experiment) means that experiments can be designed and conducted quite quickly. Statistical power need not be sacrificed, since as explained below, the statistical analysis uses all of the raw data; hence an experiment with ten speakers judging ten sentence pairs yields 200 distinct observations. The restriction to two binary factors also speeds up experimental design, and reflects quite well the sorts of designs implicit in most actual syntactic research; an example demonstrating this is given below. Binary *yes/no* judgments are inherently less information-rich than judgments on a continuous scale, but they are generally easier for naive subjects to provide (see, e.g., Snyder, 2000); unclear cases can simply be responded to with an arbitrary guess (which may feel random, but rarely is). Though binary judgments are the default when judgments are collected informally, they are often avoided when experimenters intend to analyze their results statistically, one reason being that the most

familiar statistical techniques (like ANOVA) are designed for continuous data. Rather than adjusting the judgment conventions to suit the statistics, MiniJudge adjusts the statistics to suit the judgment conventions of actual practicing syntacticians, adopting a recently developed method designed specifically for binary response measures collected across both subjects and materials (see 4-2-1).

The lack of fillers and counterbalancing means that subjects have more opportunities to guess the purpose of the experiment than is typically tolerated in psycholinguistics, but the effect of any biases that may result is limited due to the treatment sentence order. First, as is standard in psycholinguistic experiments, materials are presented in random order, since by far the most powerful (hence annoying) bias on linguistic responses is memory for recently processed forms. Second, going beyond standard practice, MiniJudge is capable of factoring out in the statistics any lingering order effects (see 4-2-2). As I demonstrate below, this feature is sometimes essential to bring particularly subtle and sensitive judgment patterns up to the level of statistical significance.

For further justification of the built-in restrictions of MiniJudge, see MJInfo.htm#minexp and MJFAQ.htm, both reachable through the MiniJudge homepage.

3. Using MiniJudge

To show how MiniJudge is used, I describe a recent application of it to a morphosyntactic issue in Chinese (see Myers, in press, for discussion of the linguistic background). MiniJudge has also been used to run syntax experiments on English and Taiwan Sign Language, as well as to run pilots for larger studies and to help teach basic concepts in experimental design. MiniJudge can also be used for judgments experiments in pragmatics, semantics, and phonology.

3-1 Goal of the experiment

He (2004) presents an interesting observation regarding the interaction of compound-internal phrase structure and affixation of the plural marker *men* in Chinese. Part of his paradigm is shown in Table 2, where V = verb and O = object (based on his (2) & (4), pp. 2-3).

Table 2. The VO*men* paradigm of He (2004)

	[+men]	[-men]
[+VO]	*zhizao yaoyan zhe men *make rumor person* PLURAL	zhizao yaoyan zhe *make rumor person*
[-VO]	yaoyan zhizao zhe men *rumor make person* PLURAL	yaoyan zhizao zhe *rumor make person*

He's analysis is not relevant here; the question is simply whether or not his observation about the judgment pattern in Table 2 is empirically correct. As a non-native speaker of Chinese, I have no intuitions myself. When I have informally asked colleagues and students to double-check the judgments, I have received a mixed response. Some looking at He's paper seem to be influenced more by the printed star pattern than the examples themselves. Others rule out *men* or VO entirely, but this misses the point, since He's claim concerns the ungrammaticality of the VO*men* form relative to all the others. It may also be that He's generalization works for the few examples he cites, but fails in general. My goal, then, was to use MiniJudge to generate more examples to test systematically on native speakers.

3-2 The MiniJudgeJS interface

MiniJudgeJS is simply a JavaScript-enabled HTML form. Input and output are handled entirely by text areas; generated text includes code to run statistical analyses in R. Like the rest

of the MiniJudge family, MiniJudgeJS divides the experimental process into the steps listed in Table 3.

Table 3. The steps used by MiniJudge

I. Design experiment	**II. Run experiment**	**III. Analyze experiment**
Choose experimental factors	Choose number of speakers	Download and install R
Choose set of prototype sentences	Write instructions for speakers	Enter raw results
Choose number of sentence sets	Print or email survey forms	Generate data file
Segment prototype set (optional)	Save schematic survey file	Save data file
Replace segments (optional)		Generate R code
Save master list of test sentences		Paste R command code into R

3-3 Designing the experiment

A MiniJudge experiment is defined by its experimental factors. Thus the paradigm in Table 2 is derived via two binary factors: [±VO] (VO vs. OV) and [±*men*] (with or without *men* suffixation). As noted above, He's observation doesn't relate to each factor separately, but rather to an interaction: the combination of the factor values [+VO] and [+*men*] is claimed to result in lower acceptability, relative to overall judgments for [+VO] and for [+*men*].

The next step is to enter the prototype set of sentences (a pair if one factor, a quartet if two factors). Similar to the example sets shown in syntax papers and presentations, the prototype set serves multiple purposes. Most fundamentally, it helps to make the logic of factorial experimental design intuitive for novice experimenters. Syntacticians are not always aware of the importance of contrasting sentences that differ *only* in theoretically relevant factors, or of the central role played by interactions in many syntactic claims (for further discussion of the relevance of factors and interactions in syntax experiments, see Cowart 1997, as well as MJInfo.htm#factorial and MJInfo.htm#interact, available through the MiniJudge main page).

Another purpose of the prototype set is that it can be used to help generate further sentence sets that maintain the same factorial contrasts but vary in irrelevant lexical properties. In the case of the present experiment, the claim made in He (2004) says nothing about the particular verb, object, or head that is used. Thus the judgment pattern claimed for Table 2 above should also hold for the sets shown in Table 4 below, regardless of any additional influences from pragmatics, frequency, suffixlikeness (*zhe* vs. the others), or freeness (*ren* vs. the others); the stars here represent what He should predict (lexical content for the new sets was chosen with the help of Ko Yu-guang and Zhang Ning).

Table 4. Extending the VO*men* paradigm of He (2004)

	[+men]	[-men]
[+VO]	*chuanbo bingdu yuan men *spread virus person PLURAL*	chuanbo bingdu yuan *spread virus person*
[-VO]	bingdu chuanbo yuan men *virus spread person PLURAL*	bingdu chuanbo yuan *virus spread person*
[+VO]	*sheji shipin ren men *design ornaments person PLURAL*	sheji shipin ren *design ornaments person*
[-VO]	shipin sheji ren men *ornaments design person PLURAL*	shipin sheji ren *ornaments design person*

MiniJudge partly automates the process of creating new sentence sets by dividing up the

prototype sentences into the largest repeating segments and replacing them with user-chosen substitutes. The prototype segments for Table 2 are shown in the first row of Table 5. The user only has to find parallel substitutes for four segments, rather than having to construct whole new sentences consistent with the factorial design (Table 5 shows the segments needed to generate the new sets in Table 4). The segmentation and set generation algorithms (see 4-1) are designed to work equally well in English-like and Chinese-like orthographies. Of course, since MiniJudge knows no human language, it sometimes makes strange errors, so users are allowed to correct its output, or even to generate new sets manually.

Table 5. Prototype segments and new segments for the VO*men* experiment

Set 1 (prototype) segments:	zhizao	yaoyan	zhe	men
Set 2 segments:	chuanbo	bingdu	yuan	men
Set 3 segments:	sheji	shipin	ren	men

After the user has corrected and approved the master list of sentences, it can be saved to a file for use in reports. In the present experiment, the master list contained 48 sentences (12 sets of 4 sentences each). This is an unusually large number of sentences for a MiniJudge experiment; significant results have been found with experiments with as few as 10 sentences.

3-4 Running the experiment

In order to run a MiniJudge experiment, the user must make three decisions. The first concerns the maximum number of speakers to test. It is possible to get significant results with as few as 7 speakers, but in the present experiment, I generated 30 surveys. As it turned out, only 18 surveys were returned.

The second decision concerns whether surveys will be distributed by printed form or by email. In MiniJudgeJS, printing surveys involves saving the them from a text area and printing them with a word processor. MiniJudgeJS cannot send email automatically, so emailed surveys must be individually copied and pasted. In the present experiment, I emailed thirty students, former students, or faculty of my linguistics department who did not know the purpose of the experiment.

The final decision concerns the instructions, which the user may edit from a default. MiniJudgeJS requires that judgments be entered as 1 (*yes*) vs. 0 (*no*). Chinese instructions for the VO*men* experiment were written with the help of Ko Yu-guang.

Surveys themselves are randomized individually to prevent order confounds, as is standard in psycholinguistics. The randomization algorithm, taken from Cowart (1997:101), results in every sentence having an equal chance to appear at any point in the experiment (by randomization of blocks), while simultaneously distributing sentence types evenly and randomly.

Each survey starts with the instructions, followed by a speaker ID number (e.g., "##02"), and finally the survey itself, with each sentence numbered in the order seen by the speaker. Because the speakers' surveys intentionally hide the factorial design, the experimenter must save this information separately in a schematic survey file. This file is meant to be read only by MiniJudgeJS; as an example, the first line of the schematic survey file for the present experiment is explained in Table 6.

Table 6. The structure of the schematic survey information file for the VO*men* experiment

File line:	`01`	`20`	`05`	`01`	`-VO`	`-men`
Explanation:	speaker ID number	sentence ID number	set ID number	order in survey	value of first factor	value of second factor

After completed surveys have been returned, the experimenter pastes them into a text area in any order (as long as each survey still contains its ID number), and pastes the schematic survey information back into another text window. MiniJudgeJS extracts judgments from the surveys and creates a data file in which each row represents a single observation, with IDs for speakers, sentences, and sets, presentation order of sentences, factor values (1 for [+] and -1 for [-]), and judgments. As an example, the first three lines of the data file for the VO*men* experiment are shown in Table 7.

Table 7. First three lines of data file for the VO*men* experiment

Speaker	Sentence	Set	Order	VO	men	Judgment
1	20	5	1	-1	-1	1
1	45	12	2	1	1	0

3-5 Analyzing the results

For novice experimenters, the most intimidating aspect of psycholinguistic research is statistical analysis. MiniJudge employs quite complex statistical methods that are unfamiliar even to most psycholinguists, yet hides them behind a relatively user-friendly interface. Data from a MiniJudge experiment are both categorical and repeated-measures (grouped within speakers). Currently the best available statistical model for repeated-measures categorical data is generalized linear mixed effect modeling (GLMM), which can be thought of as an extension of logistic regression (see e.g. Agresti et al., 2000).

GLMM poses serious programming challenges, so MiniJudgeJS passes the job to R, the world's foremost free statistical package (R Development Core Team, 2007). R is an open-source near clone of the proprietary program S (Chambers & Hastie, 1993), and like S, is a full-featured programming language. Its syntax is somewhere between C++ and Matlab, and of course it has a wide variety of built-in statistical functions, including many user-written packages. The specific R package for GLMM used by MiniJudgeJS is `lme4` and its prerequisite packages (Bates & Sarkar, 2007).

However, since R is a command-line program, and its outputs can be unintelligible without statistical training, MiniJudgeJS handles the interface with it. The user merely enters the name of the data file, decides whether or not to test for syntactic satiation (explained below in section 3-5-2), and pastes the code generated by MiniJudgeJS into the R window. After the last line has been processed by R, the code will either generate a warning (that the file was not found or was not formatted correctly), or if all went well, display a simple interpretive summary report. A much more detailed technical report is also saved automatically; this report is explained, step by step for the novice user, in MJInfo.htm#resultshelp.

3-5-1 A null result?

When the data file containing the 18 completed surveys in the VO*men* experiment was analyzed using the R code generated by MiniJudgeJS, the summary report in Figure 1 was produced, along with the bar graph in Figure 2. The summary report has three parts: a table showing the number of *yes* judgments for each category (shown graphically in Figure 2), a listing of significant patterns (if any), and a statement about whether there was any significant confound between items and factors (explained more fully in section 4-2-5).

```
Number of YES judgments for each category:
            [+V]    [-V]    Total         V = VO
     [+m]    23      74      97           m = men
     [-m]    89      163     252
     Total   112     237     349

Significance summary (p < .05):

     The factor VO had a significant negative effect.
     The factor men had a significant negative effect.
     Order had a significant negative effect.
     There were no other significant effects.

     Items and factors were significantly confounded, so the above results
     take cross-item variability into account.
```

Figure 1. Default results summary generated by MiniJudgeJS for the VO*men* experiment

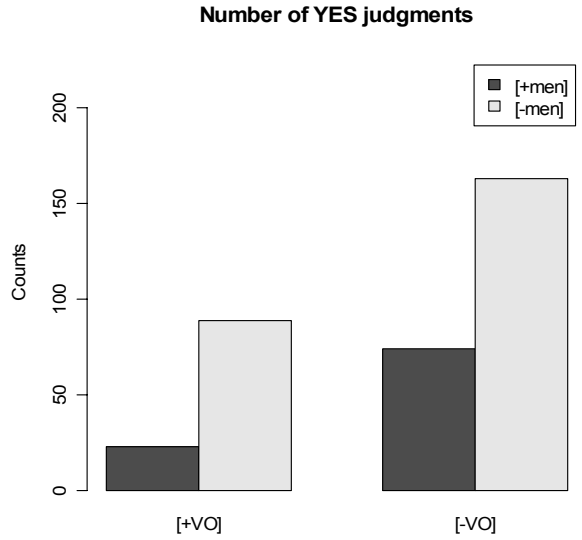

Figure 2. Default graph generated by MiniJudgeJS for the VO*men* experiment

The negative effects of the [VO] and [men] factors mean that items containing VO or *men* were judged worse, on average. These patterns are also clear from the table and graph. However, as noted in 3-1, these patterns are not what the empirical claim of He (2004) is concerned with. What we expected to see was a significant interaction between [VO] and [men], but this was not found. Instead, inspection of the technical results file shows that the *p* value for the interaction was .89 for the by-speaker-only analysis and .73 for the by-speaker-and-sentence analysis, clearly non-significant ($p > .05$).

However, this is not a refutation of He's claim, but merely a null result. Indeed, the number of *yes* judgments trends in the predicted direction: for VO forms, non-*men* forms were judged better than *men* forms by a ratio of almost 4:1 (89/23 = 3.87), about twice as high as the ratio for OV forms (163/74 = 2.20). That is, it was worse to affix *men* to VO forms than to OV forms, just as He claims.

One possible cause of a null result is a confound with a nuisance variable. A clue to what this nuisance variable might be here is the significant negative effect of order, which means that judgments got worse as the experiment progressed (i.e., there was a rising probability of judging a form as unacceptable). This shift in judgments suggests that further analysis may be advisable, as described next.

3-5-2 Syntactic satiation

Though MiniJudge factors out raw order effects in its default analysis, it is possible that order also *interacts* with one or more factors. Testing for interactions with continuous variables without a specific theoretical reason may make it more difficult to interpret main effects (see e.g. Bernhardt & Jung, 1979), but MiniJudge offers the option to test for interactions with order because it helps in the detection of syntactic satiation. Satiation is the phenomenon (known informally as "linguist's disease") in which linguistic intuitions are dulled by repeated testing, making it harder to be confident in one's judgments. MiniJudge tests for satiation by looking for interactions with order: early on, the effect of some factor is strong, but later it's weak.

Snyder (2000) argues that satiation could provide a new window into grammar and/or processing, since different types of syntactic violations differ in whether or not they satiate. Snyder suggests two possible reasons for such differences. On the one hand, satiation may be caused by processing, not grammar, thus providing a diagnostic for performance influences on acceptability (a position taken by Goodall, 2004). On the other hand, satiability may differ due to differences between the components of competence itself, thus permitting a new grammatical classification tool (a position taken by Hiramatsu, 2000).

Although He (2004) makes no predictions relating to satiation, the unexpected null result noted in section 3-5-1 suggests that it may be worthwhile trying out a more complex analysis that includes interactions with order. Running this analysis simply involves telling MiniJudgeJS that we want to test for satiation by clicking a check box, and then pasting the newly generated code into R.

Since we chose to test satiation, MiniJudge changes the format of the graph to a line graph, as in Figure 3, which makes the overall order effect quite clear. A satiation trend is also visible in the graph, since the lines not only drop over time, but also get closer together, meaning that discrimination between sentence types weakened over the course of the experiment. Unfortunately, factoring out satiation doesn't result in any change in the main report, which comes out the same as the earlier one shown in Figure 1: no significant interaction between [VO] and [men].

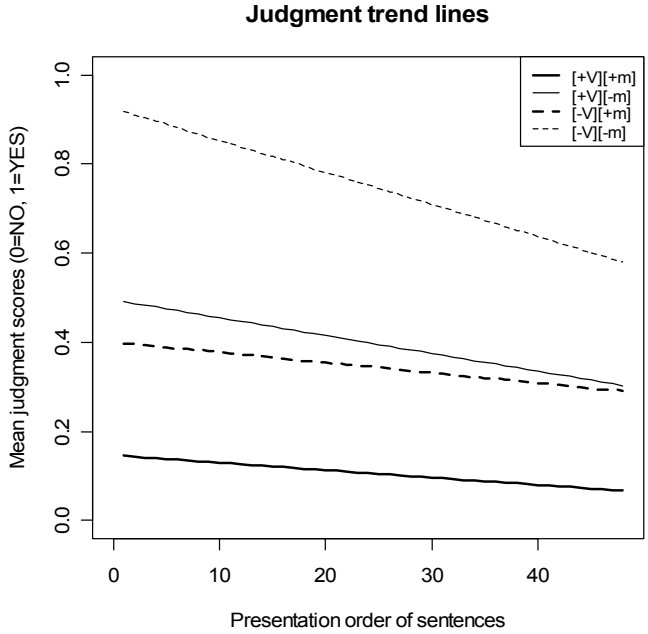

Figure 3. Graph generated by MiniJudgeJS when testing for satiation

The technical report for this analysis, automatically saved under a different name from the earlier one, clarifies what happened. The summary report gives the analysis that takes both

cross-speaker and cross-sentence variability into account, since this model had statistically better coverage of the data, as indicated by the statement in the summary report that "items and factors were significantly confounded." This analysis only shows marginally significant satiation of the VO*men* effect ($p = .08$), and the VO*men* effect itself shows $p = .17$. However, in the less stringent, but still meaningful, by-speakers-only analysis, factoring out satiation did make the interaction between the factors [VO] and [men] significant ($p = .023$), and this analysis also shows a three-way interaction between [VO], [men] and order ($p = .016$), that is, satiation of the VO*men* effect.

This experiment thus not only provided reliable evidence in favor of the empirical claim made by He (2004), though only in the less stringent by-speakers analysis. It also revealed three additional patterns not reported by He: overall lower acceptability for VO forms relative to OV forms, overall lower acceptability of *men* forms, and the satiability of the VO*men* effect. Detecting satiation, and the VO*men* effect it obscured, depended crucially on the use of careful experimental design and statistical analysis, and would have been impossible to confirm using traditional informal methods. Despite this power, the MiniJudge experiment was designed, run, and analyzed within a matter of days (with most of the delay due to tardy subject replies), rather than the weeks required for full-fledged experimental syntax.

4 The inner workings

MiniJudgeJS, as with MiniJudgeJava and all future versions in the MiniJudge family, is free and open source. The JavaScript and R code can be modified freely, and both are heavily commented to make them easier to follow. In this section I give overviews of the programming relating to material generation and statistical analysis.

4-1 Material generation

As described in section 3-3, MiniJudgeJS can assist with the generation of additional sentence sets. This involves two major phases: segmenting the prototype sentences into the largest repeated substrings, and substituting new segments for old segments in the new sentence sets.

The first step is to determine whether the prototype sentences contain any spaces. If they do, words are treated as basic units, and capitalization is removed from the initial word and any sentence-final punctuation mark is also set aside (for adding again later). If there are no spaces (as in Chinese, or in a phonology or morphology experiment involving single words), characters are treated as basic units and there is no capitalization adjustment. Next, the boundaries between prototype sentences are demarcated to indicate that cross-sentence strings can never be segments. The algorithm for determining other segment boundaries requires the creation of a lexicon containing all unique words (or characters) in the prototype corpus. If the algorithm detects that items from the corpus and from the lexicon match only if one of the items is lowercase, this item is recapitalized. Versions of the prototype sentences with "word-based" capitalization is later used when old segments are replaced by new ones.

The most crucial step in the segmentation algorithm is to check each word (or character) in the lexicon to determine whether or not it has at least two neighbors on the same side in the corpus. For example, suppose the prototype set consists of the sentences "A dog loves the cat. The cat loves a dog." The lexical item "loves" has two neighbors on the left: "dog" and "cat". Thus a segment boundary should be inserted to the left of "loves" in the corpus. Similarly, the right neighbor of "loves" is sometimes "the" and sometimes "a"; hence "loves" will be treated as a whole segment. By contrast, the lexical item "cat" always has the same item to its left (once sentence-initial capitalization is removed): "the". Similarly, the right neighbor of "the" is always "cat". Thus "the cat" will be treated as a segment, and the same logic applies to "a dog". The prototype segments are thus "a dog", "loves", "the cat".

The final phase involves substituting the user-chosen new segments for the prototype

segments using JavaScript's built-in regular expression functions.

4-2 Statistical analysis

The statistical analyses conducted by MiniJudgeJS involve several innovations: the use of GLMM, the inclusion of order and interactions with order as factors, the use of JavaScript to communicate with R, the use of R code to extract key values from R's technical output so that a simple report can be generated, and the use of R code to compare by-subject and by-subject-and-item analyses to decide whether the latter is really necessary. In this section I describe each of these innovations in turn.

4-2-1 GLMM

As explained in section 3-5, generalized linear mixed effect modeling (GLMM) is conceptually akin to logistic regression, which is at the core of the sociolinguistic variable-rule analyzing program VARBRUL and its descendants (Mendoza-Denton et al. 2003), but unlike logistic regression, GLMM regression equations also include random variables (e.g., the speakers); see Agresti et al. (2000). One major advantage of a regression-based approach is that no data are thrown away as they are when by-subject and by-item averages are analyzed in separate ANOVAs, as is standard in psycholinguistics (see 4-2-5). Moreover, since each observation is treated as a separate data point, GLMM is usually not affected much by missing data as long as they are missing non-systematically (this is why participants in MiniJudge experiments are requested to judge *all* sentences, guessing if they're not sure).

Though GLMM is the best statistical model currently available for repeated-measures categorical data, it does have some limitations. First, R's implementation of GLMM tests significance using *z* scores, which are most reliable if the number of observations is greater than 100 or so, but in actual practice, 100 judgments are trivial to collect (e.g., 5 speakers judging 10 sentence pairs). Second, like regression in general, GLMM assumes that the correlation between the dependent and independent variables is not perfect, so it is paradoxically unable to confirm the significance of perfect correlations. Third, like logistic regression (but unlike ANOVA or ordinary regression), it is impossible to calculate GLMM coefficients and *p* values exactly; they can only be estimated. Unfortunately, the best way to estimate GLMM values is extremely complicated and slow, so R uses "simpler" yet less accurate estimation methods. Currently, R provides two options for estimating GLMM coefficients: the faster but less accurate penalized quasi-likelihood approximation, and the slower but more accurate Laplacian approximation. MiniJudgeJS uses the latter.

The function in the `lme4/Matrix` packages used for GLMM is `lmer`, which can also handle linear mixed-effect modeling (i.e., repeated-measures linear regression). The syntax is illustrated in Figure 3, which shows the commands used to run the final analyses described above in section 3-5-2. "Factor1" and "Factor2" are variables whose values are set in the R code to represent the actual factors. The use of categorical data is signaled by setting the distribution family to "binomial". The name of the loaded data file is arbitrarily called "minexp" (for MiniJudge experiment). The first function treats only subjects as random, while the second function treats both subjects and items as random. The choice to test for satiation or not is determined by the user, and based on this choice, JavaScript generates different versions of the R code. The choice to run one-factor or two-factor analyses is determined by the R code itself by counting the number of factors in the data file. Both analyses in Figure 4 are always run, then compared with another R function described in 4-2-5.

```
glmm1 = lmer(Judgment ~ Factor1 * Factor2 * Order + (1|Speaker), data = minexp,
  family = "binomial", method = "Laplace")
glmm2 = lmer(Judgment ~ Factor1 * Factor2 * Order + (1|Speaker) + (1|Sentence),
  data = minexp, family = "binomial", method = "Laplace")
```

Figure 4. R commands for GLMM when testing satiation in a two-factor experiment

4-2-2 Order as a factor

MiniJudgeJS includes order as a factor whether or not the user tests for satiation, to compensate for the fact that MiniJudge experiments use no counterbalanced lists of sentences across subgroups of speakers. List counterbalancing is used in full-fledged experimental syntax so that speakers don't use an explicit comparison strategy when judging sentences from the same set (a comparison strategy may create an illusory contrast or have other undesirable consequences). However, comparison can only occur when the second sentence of a matched pair is encountered. If roughly half of the speakers get sentence type [+F] first and half get [-F] first, then on average, judgments for [+F] vs. [-F] are only partially influenced by a comparison strategy. The comparison strategy (if any) will be realized as an order effect: early judgments (when comparison is impossible) will be different from later judgments. Thus factoring out order effects in the statistics serves roughly the same purpose as counterbalanced lists.

4-2-3 JavaScript as an R interface

JavaScript is much more powerful than many programmers realize. In fact, a key inspiration for MiniJudgeJS was the Logistic Regression Calculating Page (http://statpages.org/logistic.html), a JavaScript-enabled HTML file written by John C. Pezzullo. Using only basic platform-universal JavaScript, the page collects data, reformats it, estimates logistic regression coefficients via a highly efficient maximum likelihood estimation algorithm, and generates chi-square values and *p* values. Thus a JavaScript-only version of MiniJudgeJS is conceivable, without any need to pass work over to R.

Currently, however, in MiniJudgeJS the role of JavaScript in the statistical analysis is mainly as a user-friendly GUI. Since the statistics needed for a MiniJudge experiment is highly standardized, very little input is needed from the user, but the potential to use JavaScript to interface with R in more flexible ways is there. This would help fix a major limitation with R, which has a command-line interface quite intimidating for novice users and online help that leaves a lot to be desired (cf. Fox, 2005).

Of course, JavaScript has its own limitations, the most notable of which are the built-in security constraints that prevent JavaScript from being able to read or write to files, or to communicate directly with other programs. For example, it's impossible to have JavaScript run R in the background, to save users the bother of copying and pasting in R code. This is why we developed MiniJudgeJava as well, though in its current version it still requires the user to interface with R by pasting in code.

4-2-4 R code to simplify output

GLMM is a high-powered statistical tool, unlikely to be used by people who don't already have a strong background in statistics, and so the outputs generated by R are not understandable without such a background. Since MiniJudge is intended for statistical novices, extra programming is needed to translate R output into plain language. For MiniJudgeJS, the most crucial portion of R's output for GLMM is the matrix containing the regression coefficient estimates and *p* values, like that shown in Figure 4 (from the VO*men* experiment, without testing for satiation). The trick is to extract the estimates (the signs of which provide information about the nature of the pattern) and the *p* values (which indicate significance) in

order to generate a simple summary containing no numbers at all.

```
                   Estimate    Std. Error  z value    Pr(>|z|)
(Intercept)        -0.198279   0.392239    -0.506     0.6132
Factor1            -1.125097   0.252556    -4.455     8.40e-06
Factor2            -1.420005   0.253253    -5.607     2.06e-08
Order              -0.019289   0.007518    -2.566     0.0103
Factor1:Factor2    0.087524    0.251891    0.347      0.7282
```

Figure 5. Output generated by `lmer` for the VO*men* experiment without testing for satiation

The current version of MiniJudgeJS extracts values from the `lmer` output by "sinking" `lmer`'s displayed output to an offline file, and then reading this file back in as a string (the offline file becoming the permanent record of the detailed analysis). The string is then searched for the string "(Intercept)" which always appears at the upper left of the value matrix. The coefficient is the first value to the right of the left-most column, and the *p* value is the fourth value (skipping "<", if any).

If the *p* value associated with a factor or interaction is less than 0.05, a summary line is generated that gives the actual factor name and the sign of the estimate, as in Figure 1 above. The R code generates the summary table and bar graph counting the number of *yes* judgments for each category (see Figures 1 and 2) directly from the data file itself. When satiation is tested, the line graph (as in Figure 3) is created by computing the mean judgment values (i.e. proportion of 1 judgments) across speakers with each order value (i.e. 1, 2, ...), separately for each item type (as defined by the experimental factors), and then plotting linear regression lines for each item type.

4-2-5 By-subject and by-item analyses

MiniJudgeJS runs both by-subject and by-subject-and-item analyses, but it reports only the first in the main summary unless it finds that the more complex analysis is really necessary. This approach differs from standard psycholinguistic practice, where both by-subject and by-item analyses are always run. A commonly cited reason for always running a by-item analysis is that it is required to test for generality across items, just as a by-subject analysis tests for generality across subjects. However, this logic is based on a misinterpretation of Clark (1973), the paper usually cited as justification.

First, it is wrong to think that by-item analyses check to see if any item behaves atypically (i.e., is an outlier). For parametric models like ANOVA, it is quite possible for a single outlier to cause an illusory significant result, even in a by-item analysis (categorical data analyses like GLMM don't have this weakness). To test for outliers, there's no substitute for checking the individual by-item results manually. MiniJudge helps with this by reporting the by-sentence rates of *yes* judgments in a table saved as part of the offline analysis file; items with unusually low or high acceptability relative to others of their type stand out clearly. In the case of the VO*men* experiment, this table did not seem to show any outliers.

The second problem with the standard justification for performing obligatory by-item analyses, as Raaijmakers et al. (1999) emphasize, is that the advice given in Clark (1973) actually applies only to experiments without matched items, such as an experiment comparing a random set of sentences with transitive verbs ("eat" etc) with a random set of sentences with unrelated intransitive verbs ("sleep" etc). Such sentences will differ in more than just the crucial factor (transitive vs. intransitive), so even if a difference in judgments is found, it may actually relate to uninteresting confounded properties (e.g., the lexical frequency of the verbs). However, if lexically matched items are used, as in the VO*men* experiment, there is no such confound, since items within each set differ only in terms of the experimental factor(s). If items are sufficiently well matched, taking cross-item variation into account won't make any

difference in the analysis (except to make it much more complicated), but if they are not well matched, ignoring the cross-item variation will result in misleadingly low *p* values.

Nevertheless, if we only computed models that take cross-item variation into account, we might lose useful information. After all, a high *p* value does not necessarily mean that there is no pattern at all, only that we have failed to detect it. Thus it may be useful to know if a by-speaker analysis is significant even if the by-speaker-and-sentence analysis is not. Such an outcome could mean that the significant by-speaker result is an illusion due to an uninteresting lexical confound, but it could instead mean that if we do a better job matching the items in our next experiment, we will be able to demonstrate the validity of our theoretically interesting factor. Moreover, it is quite difficult to compute GLMM models with two random variables, making such models somewhat less reliable than those with only one random variable. Just in the last year, the `lme4` package in R has been upgraded, so that the `lmer` function now gives different results for by-subjects-and-items analyses than it did when MiniJudge was first developed. Due to concerns like these, MiniJudge runs both types of analyses, and only chooses the by-subjects-and-items analysis for the main report if a statistically significant confound between factors and items is detected. The full results of both analyses are saved in an off-line file, along with the results of the statistical comparison of them.

The R language makes it quite easy to perform this comparison, since the model in which only speakers are treated as random is a special case of the model in which both speakers and sentences are treated as random. This means the two GLMM models can be compared by a likelihood ratio test using ANOVA (see Pinheiro & Bates, 2000). As with the output of the `lmer` function, the output of the `lme4` package's `anova` function makes it difficult to extract *p* values, so again the output is "sunk" to the offline analysis file to be read back in as a string. Only if the *p* value is below 0.05 is the more complex model taken as significantly better. If the *p* value is above 0.2, MiniJudgeJS assumes that items and factors are not confounded and reports only the by-subjects-only analysis in the main summary. Nevertheless, MiniJudgeJS, erring on the side of caution, gives a warning if $0.2 > p > 0.05$. In any case, both GLMM analyses are available for inspection in the offline analysis file. Each analysis also includes additional information, generated by `lmer`, that may help determine which one is really more reliable, including variance of the random variables and the estimated scale (compared with 1); these details are explained in MJInfo.htm#resultshelp.

In the case of the VO*men* experiment, the comparison of the two models showed that the by-subjects-only model was sufficient, unsurprisingly, given that the materials were almost perfectly matched, and that the items table showed no outliers among the sentence judgments.

The final problem with the standard justification for automatic by-item analyses is one that even Raaijmakers et al. (1999) fail to point out. Namely, since repeated-measures regression models make it possible to take cross-speaker and cross-sentence variation into account at the same time, without throwing away any data, they are superior to standard models like ANOVA. To learn more about how advances in statistics have made some psycholinguistic traditions obsolete, see Baayen (2004).

5. Conclusions

MiniJudge, currently implemented as MiniJudgeJS and MiniJudgeJava, is software for theoretical syntacticians who want a reliable and easy way to collect and interpret judgments consistent with the key methodological principles of experimental cognitive science. MiniJudge is limited in some ways, in particular in how it interfaces with R, though in ongoing work we are developing efficient code to compute GLMM within JavaScript or Java itself. Nevertheless, even in its current version, MiniJudge is quite easy to use, as testing by my students has demonstrated, and powerful enough to detect theoretically interesting patterns with very little data. Behind this power are original programming and statistical techniques.

Finally, MiniJudge is an entirely free, open-source program (as will be all future versions). Anyone interested is invited to try it out and contribute to its further development.

Acknowledgements

This research was supported by National Science Council (Taiwan) grant NSC 94-2411-H-194-018. MiniJudgeJS is co-copyrighted by National Chung Cheng University. Experimental or programming help came from my research assistants Ko Yu-guang, Chen Tsung-yin, and Yang Chen-Tsung. The students in my spring 2006 class *Competence & Performance* helped test MiniJudgeJS and made useful suggestions. John C. Pezzullo and Harald Baayen also provided helpful information on programming and statistical matters. An earlier version was presented at ROCLING 18 (Hsinchu, Taiwan, September, 2006), and I thank conference reviewers and audience members for their comments. Of course I am solely responsible for any mistakes.

References

Agresti, A., Booth, J. G., Hobert, J. P., & Caffo, B. (2000). Random-effects modeling of categorical response data. *Sociological Methodolo*gy, 30, 27-80.

Baayen, R. H. (2004). Statistics in psycholinguistics: A critique of some current gold standards. *Mental Lexicon Working Paper*s, 1, 1-45. University of Alberta, Canada. www.mpi.nl/world/persons/private/baayen/submitted/statistics.pdf

Bates, D., & Sarkar, D. (2007). lme4: Linear mixed-effects models using S4 classes [Computer software]. R package.

Bernhardt, I., & Jung, B. S. (1979). The interpretation of least squares regression with interaction or polynomial terms. *The Review of Economics and Statistics, 61* (3), 481-483.

Chambers, J. M., & Hastie, T. J. (1993). *Statistical models in S*. Chapman & Hall.

Chen, T.-Y., Yang, C.-T., & Myers, J. (2007). MiniJudgeJava (Version 0.9.9) [Computer software]. http://www.ccunix.ccu.edu.tw/~lngproc/MiniJudge.htm.

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Clark, H. (1973). The language-as-fixed-effect fallacy: A critique of language statistics in psychological research. *Journal of Verbal Learning and Verbal Behavior, 12*, 335-359.

Clifton, Jr., C., Fanselow, G., & Frazier, L. (2006). Amnestying superiority violations: Processing multiple questions. *Linguistic Inquiry, 37* (1), 51–68.

Cowart, W. (1997). *Experimental syntax: Applying objective methods to sentence judgments*. London: Sage Publications.

Featherston, S. (2005). *That*-trace in German. *Lingua, 115* (9), 1277-1302.

Fox, J. (2005). The R Commander: A basic-statistics graphical user interface to R. *Journal of Statistical Software, 14* (9).

Goodall, G. (2004). On the syntax and processing of wh-questions in Spanish. In B. Schmeiser, V. Chand, A. Kelleher, & A. Rodriguez (Eds.), *WCCFL 23 Proceedings* (pp. 101-114). Somerville, MA: Cascadilla Press.

He, Y. (2004). The words-and-rules theory: Evidence from Chinese morphology. *Taiwan Journal of Linguistics, 2* (2), 1-26.

Hiramatsu, K. (2000). *Accessing linguistic competence: Evidence from children's and adults' acceptability judgements*. Doctoral dissertation, University of Connecticut, Storrs.

Mendoza-Denton, N., Hay, J., & Jannedy, S. (2003). Probabilistic sociolinguistics: Beyond variable rules. In R. Bod, J. Hay, & S. Jannedy (Eds.), *Probabilistic linguistics* (pp. 97-138). Cambridge, MA: MIT Press.

Myers, J. (2007). MiniJudgeJS (Version 1.0) [Computer software]. http://www.ccunix.ccu.edu.tw/~lngproc/MiniJudgeJS.htm.

Myers, J. (in press). Generative morphology as psycholinguistics. In G. Libben, & G. Jarema (Eds.), *The mental lexicon: Core perspectives*. Elsevier.

Penke, M., & Rosenbach, A. (2004). What counts as evidence in linguistics? An introduction. *Studies in Language*, 28 (3), 480-526.

Pinheiro, J. C., & Bates, D. M. (2000). *Mixed-effects models in S and S-Plus*. Berlin: Springer.

R Development Core Team (2007). R: A language and environment for statistical computing [Computer software]. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. http://www.R-project.org.

Raaijmakers, J. G. W., Schrijnemakers, J. M. C., & Gremmen, F. (1999). How to deal with 'the language-as-fixed-effect fallacy': Common misconceptions and alternative solutions. *Journal of Memory and Language*, 41, 416-426.

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

Snyder, W. (2000). An experimental investigation of syntactic satiation effects. *Linguistic Inquiry*, 31, 575-582.

Sorace, A., & Keller, F. (2005). Gradience in linguistic data. *Lingua, 115*, 1497-1524.