MiniJudge: Software for minimalist experimental syntax

James Myers[1]

[1]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 design. This paper shows why MiniJudge is useful, what it feels like to use it, and how it works.

1. Introduction

Linguistics is a science because linguists test hypotheses against empirical data, but this testing is done in a much more informal way than in almost any other science. Theoretical syntacticians, for example, violate protocols standard in the rest of the cognitive sciences by acting simultaneously as experimenter and subject, and by showing little concern with the issues of experimental design and quantitative analysis deemed essential in most sciences. Linguists recognize that their informally-collected data are often inconclusive; controversial native-speaker judgments are commonplace problems in both research and teaching. From my conversations with syntacticians, I get the sense that they would appreciate a tool for collecting judgments more reliably, yet this tool should be one that permits them to maintain their traditional focus on theory rather than method.

This is where MiniJudge comes in. MiniJudge (www.ccunix.ccu.edu.tw/~lngproc/MiniJudge.htm) is software to help theoretical syntacticians design, run, and analyze linguistic judgment experiments quickly and painlessly. Because MiniJudge experiments involve testing the minimum number of speakers and sentences in the shortest amount of time, all while sacrificing the least amount of statistical power, I call them "minimalist" experiments. In this paper I first argue why a tool like MiniJudge is necessary. 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. Currently the only implementation of MiniJudge is MiniJudgeJS, which is written in JavaScript, HTML, and the statistical language R (www.r-project.org).

2. Balancing speed and reliability in syntactic judgment collection

Though some readers may wonder why we should bother with judgments when we can simply analyze corpora, judgments and corpus data are actually complementary performance windows into linguistic competence, with their own strengths and weaknesses (see e.g. Penke & Rosenbach, 2004). The

question is how we can extract the maximum value out of judgments in the easiest possible way.

2.1. Experimental syntax

Phillips and Lasnik (2003:61) are entirely right to emphasize that the "[g]athering of native-speaker judgments is a trivially simple kind of experiment, one that makes it possible to obtain large numbers of highly robust empirical results in a short period of time, from a vast array of languages." Even Labov (1996:102), who generally favors corpus data, admits that "[f]or the great majority of sentences cited by linguists," native-speaker intuitions "are reliable." Yet as Phillips and Lasnik (2003:61) also point out, "it is a truism in linguistics, widely acknowledged and taken into account, that acceptability ratings can vary for many reasons independent of grammaticality." Unfortunately, in actual practice linguists don't take the distinction between "acceptability" and "grammaticality" as seriously as they know they should, and their "trivially simple methods" become merely simple-minded (Schütze, 1996).

Since the problem of detecting competence in performance is precisely the problem faced by experimental cognitive scientists every day (e.g., testing vision theories with optical illusions), a reasonable response to the syntactician's empirical challenges would be to adopt the protocols standard in the rest of the experimental cognitive sciences: multiple stimuli and subjects (naive ones rather than the bias-prone experimenters themselves), systematic controls, factorial designs, continuous response measures, filler items, counterbalancing, and statistical analysis. When judgments are collected with these more careful protocols, they often reveal hitherto unsuspected complexity. Recent examples of the growing experimental syntax literature include Sorace & Keller (2005) and Featherston (2005); Cowart (1997) is a user-friendly handbook.

2.2. Minimalist experimental syntax

Full-fledged experimental syntax is complex, forcing the researcher to spend a lot of time on work that is not theoretically very interesting. The complexity of an experiment should actually be proportional to the subtlety of the effects it is trying to detect. Very clear judgments are detectable with traditional "trivially simple" methods; very subtle judgments may require full-fledged experimental methods. But in the vast area in between, a compromise seems appropriate, where methods are powerful enough to yield statistically valid results, yet are simple enough to apply quickly: a minimalist experimental syntax (see Table 1).

Table 1. Defining characteristics of minimalist experimental syntax

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

While conducting a minimalist experiment is much simpler than conducting a full-fledged judgment experiment (an explicit guide is given in Myers 2006), some steps may still be overly complex and/or intimidating to the novice experimenter, in particular the design of the experimental sentences and the statistical analysis. The purpose of the MiniJudge software is to automate these steps.

3. Using MiniJudge

To show how MiniJudge is used, I describe a recent application of it to a morphosyntactic issue in Chinese; for another example, see MJInfo.htm#resultshelp, reachable through the MiniJudge homepage. 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.

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*. 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, with some ruling 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. Some speakers shown He's starred and non-starred examples are willing to agree with his judgments, but it's likely that the star pattern has biased them. 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 begins by choosing the experimental factors. In the case of the VO*men* claim, 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 MJInfo.htm#factorial and MJInfo.htm#interact).

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 might 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 while keeping track of the factorial design (Table 5 also shows the segments needed to generate the new sets in Table 4). The segmentation and set generation processes 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 (as I am doing here). 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*); in the current version, if surveys are to be collected electronically, these judgments must be typed before each sentence ID number. 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 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, 2005). 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 a mixture of 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 used by MiniJudgeJS for GLMM is `lme4` (and its prerequisite package `Matrix`), authored by Douglas Bates and Deepayan Sarkar, and maintained by Douglas Bates. R has a simple GUI interface, and by default, the Windows version nativizes (e.g., in Chinese Windows, menus and basic messages are in Chinese).

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 either will generate a warning (that the file was not found or was not formatted correctly), or if all went well, will 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. There are three parts: a table showing the number of *yes* judgments for each category, a listing of significant patterns (if any), and a statement about whether there was any significant confound between items and factors (discussion of this last point is reserved for section 4.3.5).

```
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.

     The above results do not take cross-item variability into account because
     no confound between items and factors was detected (p > .2).
```

Figure 1. Default results summary 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 showing the number of *yes* judgments (in the total row and total column, respectively). However, as discussed 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 0.89, clearly non-significant.

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 claimed.

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 (i.e., there was a rising probability of judging a form as unacceptable) as the experiment progressed. 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. This 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. Following the logic proposed in Myers (2006), MiniJudge tests for satiation by looking for negative interactions with order: early on, the [±F] contrast 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 effects (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 checkbox), and then pasting the generated code into R. Doing this with the VO*men* data resulted in the two new lines in Figure 2 being added to the significance summary.

```
The interaction between VO and men had a significant positive effect.
The interaction of VO * men with Order had a significant negative effect
  (satiation).
```

Figure 2. New lines in results summary when satiation was tested in the VO*men* experiment

As hoped, factoring out the interactions with order revealed a significant interaction between the factors [VO] and [men]. This shows that the ratio difference seen in Figure 1 is indeed statistically reliable (the detailed report file shows $p = 0.02$), thus vindicating He's empirical claim. This new analysis also detected satiation in the VO*men* effect; it was this interaction with order that had obscured evidence for the VO*men* effect in the default analysis.

This experiment thus not only provided reliable evidence in favor of the empirical claim made by He (2004), but 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 using traditional informal methods. Despite this power, the MiniJudge experiment was designed, run, and analyzed within a matter of days, rather than the weeks required for full-fledged experimental syntax.

4. The inner workings

MiniJudgeJS, as with all future versions in the MiniJudge family, is free and open source. The JavaScript and R code can be modified freely by downloading the HTML file and opening it in a text editor, 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),

characters are treated as basic units. 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 consisted 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. This is done using JavaScript's built-in regular expression functions, which only became available with Netscape 4 and Internet Explorer 4.

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.

4.3.1 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. Moreover, since each observation is treated as a separate data point, GLMM is usually not affected much by missing data, but only if 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 reliable only if the number of observations is greater than 50 or so, but in actual practice, 50 judgments are trivial to collect (e.g., 5 speakers judging 10 sentences each). 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 perfectly; 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 minimalist 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; 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 3 are always run, and then compared with another R function described below in 4.3.5.

```
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 3. R commands for computing GLMM when testing satiation in a two-factor experiment

4.3.2 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.

4.3.3 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. Unfortunately, the necessary statistical programming is quite formidable.

Instead, 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, whose command-line interface is quite intimidating for novice users, and whose online help leaves a lot to be desired (cf. Fox, 2005).

Of course, using JavaScript as an interface has its 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 are currently exploring other versions of MiniJudge. One that has made some progress is MiniJudgeJava, written by Chen Tsung-ying in Java using its own platform-independent GUI tools. Interfacing with R is likely to remain tricky, however, unless we create something like MiniJudgeR, written in R itself, or figure out how to program GLMM directly in JavaScript.

4.3.4 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)       -0.0810381   0.2330613    -0.3477     0.728057
Factor1           -0.8090969   0.0886143    -9.1305     < 2.2e-16
Factor2           -0.9741367   0.0891447    -10.9276    < 2.2e-16
Order             -0.0192680   0.0059976    -3.2126     0.001315
Factor1:Factor2   0.0119932    0.0877194    0.1367      0.891250
```

Figure 4. GLMM output generated by `lmer` for the VO*men* experiment without testing for satiation

Unfortunately, the output of the `lmer` function is a list object, containing only the parameters used to compute the estimates and *p* values, not the values themselves. Thus the R code generated by MiniJudgeJS "sinks" `lmer`'s displayed output to an offline file, and then reads this file back in as a

string (the offline file becomes 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 this, 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 Figures 1 and 2 above. The R code generates the summary table counting the number of *yes* judgments for each category (see Figure 1) directly from the data file itself.

4.3.5 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-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. Thus 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. 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 ($p = 1$). This is unsurprising, given that the materials were almost perfectly matched, and that the by-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 only in the form of MiniJudgeJS, is software for theoretical syntacticians without any experimental training who want to collect and interpret judgments quickly and reliably. Though MiniJudgeJS is limited in some ways, in particular in how it interfaces with R, it is still quite easy to use, as testing by my students has demonstrated. Moreover, it is unique, offering syntacticians power that they cannot obtain any other way. Behind this power are original programming and statistical techniques. Finally, MiniJudgeJS is an entirely free, open-source program (as will be all future versions). Anyone interested is invited to try it out, save it for use offline, and contribute to its further development.

6. 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 and Chen Tsung-yin. 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. Of course I am solely responsible for any mistakes.

7. References

1. A. Agresti, J. G. Booth, J. P. Hobert, & B. Caffo, "Random-effects Modeling of Categorical Response Data," *Sociological Methodology*, Vol. 30, pp. 27-80, 2000.
2. R. H. Baayen, "Statistics in Psycholinguistics: A Critique of Some Current Gold standards," *Mental Lexicon Working Papers*, Vol. 1, University of Alberta, Canada, 2004, pp. 1-45. www.mpi.nl/world/persons/private/baayen/submitted/statistics.pdf
3. I. Bernhardt & B. S. Jung, "The Interpretation of Least Squares Regression with Interaction or Polynomial Terms," *The Review of Economics and Statistics*, Vol. 61, No. 3, 1979, pp. 481-483.
4. J. M. Chambers & T. J. Hastie, *Statistical Models in S*, Chapman & Hall, 1993.
5. H. Clark, "The Language-as-fixed-effect Fallacy: A Critique of Language Statistics in Psychological Research," *Journal of Verbal Learning and Verbal Behavior*, Vol. 12, pp. 335-359, 1973.
6. W. Cowart, *Experimental Syntax: Applying Objective Methods to Sentence Judgments*. Sage Publications, London, 1997.
7. S. Featherston, "Magnitude Estimation and What It Can Do for Your Syntax: Some wh-constraints in German," *Lingua*, Vol. 115, No. 11, pp. 1525-1550, 2005.
8. J. Fox, "The R Commander: A Basic-statistics Graphical User Interface to R," *Journal of Statistical Software*, Vol. 14, No. 9, 2005.
9. G. Goodall, "On the Syntax and Processing of wh-questions in Spanish," in *WCCFL 23 Proceedings*, B. Schmeiser, V. Chand, A. Kelleher, & A. Rodriguez (eds), Cascadilla Press, Somerville, MA, 2004, pp. 101-114.
10. Y. He, "The Words-and-rules Theory: Evidence from Chinese Morphology," *Taiwan Journal of Linguistics*, Vol. 2, No. 2, pp. 1-26, 2004.
11. K. Hiramatsu, *Accessing Linguistic Competence: Evidence from Children's and Adults' Acceptability Judgements*. Doctoral dissertation, University of Connecticut, Storrs, 2000.
12. W. Labov, "When Intuitions Fail," in *CLS 32: Papers from the Parasession on Theory and Data in Linguistics*, L. McNair (ed), University of Chicago, pp. 77-105, 1996.
13. N. Mendoza-Denton, J. Hay, & S. Jannedy, "Probabilistic Sociolinguistics: Beyond Variable Rules," in *Probabilistic linguistics*, R. Bod, J. Hay, & S. Jannedy (eds), MIT Press, Cambridge, MA, pp. 97-138, 2003.

14. J. Myers, "An Experiment in Minimalist Experimental Syntax," National Chung Cheng University ms. Submitted, 2006.
15. M. Penke & A. Rosenbach, "What Counts as Evidence in Linguistics? An Introduction," *Studies in Language*, Vol. 28, No. 3, pp. 480-526, 2004.
16. C. Phillips& H. Lasnik, "Linguistics and Empirical Evidence: Reply to Edelman and Christiansen," *Trends in Cognitive Science*, Vol. 7, No. 2, pp. 61-62, 2003.
17. J. C. Pinheiro & D. M. Bates, *Mixed-Effects Models in S and S-Plus*. Springer, Berlin, 2000.
18. R Development Core Team. "R: A Language and Environment for Statistical Computing," R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, 2005, URL http://www.R-project.org.
19. J. G. W. Raaijmakers, J. M. C. Schrijnemakers, & F. Gremmen, "How to Deal with 'the Language-as-fixed-effect Fallacy': Common Misconceptions and Alternative Solutions," *Journal of Memory and Language*, Vol. 41, pp. 416-426, 1999.
20. C. T. Schütze, *The Empirical Base of Linguistics: Grammaticality Judgments and Linguistic Methodology*. University of Chicago Press, Chicago, 1996.
21. W. Snyder, "An Experimental Investigation of Syntactic Satiation Effects," *Linguistic Inquiry*, Vol. 31, pp. 575-582, 2000.
22. A. Sorace & F. Keller, "Gradience in Linguistic Data," *Lingua*, Vol. 115, pp. 1497-1524, 2005.