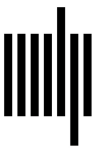
This is a section of doi:10.7551/mitpress/12200.001.0001

# The Open Handbook of Linguistic Data Management

**Edited By:** Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller, Lauren B. Collister

## Citation:

*The Open Handbook of Linguistic Data Management*
**Edited By:** Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller, Lauren B. Collister
**DOI:** 10.7551/mitpress/12200.001.0001
**ISBN (electronic):** 9780262366076
**Publisher:** The MIT Press
**Published:** 2022



The MIT Press

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

# 47 Managing Web Experiments for Psycholinguistics: An Example from Experimental Semantics/Pragmatics

**Judith Degen and Judith Tonhauser**

## 1 Introduction

This chapter reports on the organization of an experimental semantics/pragmatics project that investigated the extent to which variability in a content's projectivity is predicted by that content's at-issueness (Tonhauser, Beaver, & Degen 2018). The project included four web-based experiments in which participants adjusted sliding scales to provide projectivity and at-issueness ratings for close to three hundred items. The workflow and best practices we report here are generalizable to any sufficiently similar web-based study, as well as to in-lab studies and dependent measures that differ substantially from the reported ones, for example, online measures such as eye movements. The workflow further easily accommodates computational project components including cognitive models and corpus analyses, though the current chapter does not include these components. The chapter also describes best practices within the first author's interActive Language Processing Lab at Stanford (ALPS) at the time of writing. We close with some reflections on what we would do differently if we were we to start this project again from scratch.

## 2 Research hypothesis: The less at-issue a content is, the more it projects

The project whose organization is described in this chapter was published in Tonhauser, Beaver, and Degen (2018) and addressed the following two questions, which we explain further in this section:

1. Is there variability in the extent to which the content associated with an expression projects?
2. If there is variability, is degree of projection predicted by a content's at-issueness?

For instance, the verb *regret* is typically taken to be a factive predicate. This means that even when embedded under so-called entailment-canceling operators such as polar questions as in (1) or negation as in (2), the content of its complement—here *that Mike visited Alcatraz*—is taken to project, in other words, the speaker of (1) and (2) is taken to be committed to Mike having visited Alcatraz.

(1) Does Felipe regret that Mike visited Alcatraz?
(2) Felipe doesn't regret that Mike visited Alcatraz.

This is in contrast to (3) and (4), where the speaker is not taken to be committed to the content of the complement of the non-factive predicate *think*.

(3) Does Felipe think that Mike visited Alcatraz?
(4) Felipe doesn't think that Mike visited Alcatraz.

While intuitions appear to be clear regarding the projection of some contents, there is evidence that content varies in its projectivity (Karttunen 1971; Smith & Hall 2011; Xue & Onea 2011). Our goal was to investigate this potential projection variability systematically and test the hypothesis, based on previous work (Abrusán 2011; Beaver et al. 2017; Simons et al. 2010), that projection of content is predicted by the extent to which that content is not-at-issue, in other words, the extent to which the content is taken to be background information.

To address research questions 1 and 2, we conducted four web-based experiments: in experiments 1a and 1b, we collected projection ratings and at-issueness ratings for the projective contents of two sets of expressions. In experiments 2a and 2b, we collected at-issueness ratings for the same contents as in experiments 1a and 1b using a different at-issueness diagnostic. We found that there is variability in the projectivity of content associated with different expressions, both across and within expressions, and that this variability is systematically predicted by at-issueness: the less at-issue a particular content is, the more it projects.

While this project investigated issues at the core of formal semantics/pragmatics, the project management

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

structure described herein is general enough to extend to experimental studies in any area of linguistics.

## 3 Data management: The organizational framework

In this section we present the workflow and organization of the project and offer comments on the general workflow of any experimental project we conduct. To orient the reader to the management spirit of this and all other projects conducted within ALPS Lab: we aim for all experiments and analyses reported in papers to be directly reproducible (see also Berez-Kroeker, McDonnell, Collister, & Koller, chapter 1, this volume; Gawne & Styles, chapter 2, this volume; Mattern, chapter 5, this volume). This means that we aim to freely share as much of a project as possible and avoid using proprietary software whenever possible. For the current project, all experimental files, data files, and analysis files are freely available online. Moreover, all of the software used to generate the experiments, manage the connection between the experiment and the experiment hosting platform, and analyze the data, are open source.

### 3.1 The organizational framework for collaboration

While every project starts with an idea, its organization starts with the creation of a dedicated project directory that serves to store experimental materials, data, analysis scripts, and documentation. The collaborative nature of many experimental (and computational) projects requires a framework to manage file sharing and communication between collaborators. To this end, we use GitHub,[1] Git,[2] and Slack,[3] described in the following.

As a matter of course, we immediately turn a new project directory into a GitHub repository so all materials—experimental files, data files, and analysis files—are openly shared online. GitHub is a file hosting platform that includes version control and allows people to work together on projects. We recommend the use of GitHub as a way of sharing project materials openly regardless of the number of project collaborators. The repository that accompanies this chapter can be found at https://github.com/judith-tonhauser/how-projective.

In this project (and in general for ALPS Lab projects), we used Git to keep track of the project's progress. Git is a version control system. Originally it was developed for tracking changes in software development, but it has the broader application of tracking changes in any file, which makes it useful for tracking changes in research projects generally. We use Git to locally keep track of changes we make to files, and via integration with GitHub, these changes are also visible to our collaborators and anybody with internet access at large. We additionally used Slack, a collaboration tool that is essentially a messaging system and allows for (groups of) users to create channels dedicated to specialized topics or research projects and to exchange messages about the project. We used Slack's GitHub integration feature to link the GitHub repository to our dedicated project channel, which allows Slack to follow any changes made to the GitHub repository and to automatically post as messages in Slack the commit messages created by the researcher when a change is made to the repository. This allows collaborators on the project to be notified when a change is made to the repository and to see what the nature of the change is (provided the researcher making the change has written an informative commit message). Together, Git/GitHub/Slack comprise the collaboration framework used in this project and in all of ALPS Lab's projects. Note that for the reader interested in following the structure of the project, Slack is not necessary, though we recommend it as a tool for collaborative research projects to manage communication. Similarly, for readers who have not integrated Git/GitHub into their workflow, using these tools is not necessary for viewing or emulating the project structure, though we strongly recommend integrating these tools into your own workflow as they increase ease of collaboration, increase project transparency and shareability, and assuage fears of losing data or not being able to revert to a previous version of a file.

In the following, you can (a) view the project in the browser at https://github.com/judith-tonhauser/how-projective; (b) download the project repository from that same URL and follow along locally on your machine; or (c) if you have installed Git, clone the repository from the command line and follow along locally:

```
git clone https://github.com/judith-
tonhauser/how-projective.git
```

### 3.2 Project repository structure

The project repository, like ALPS Lab repositories in general, has the following content:

1. A *README* markdown file in the root directory of the repository, which contains information about the repository's contents.

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

2. An *experiments* directory, which stores all the files required to run the experiments. This directory is further divided into sub-directories: one for each experiment, named appropriately, and a *_shared* directory that contains JavaScript libraries required for any of the experiments to run.
3. A *results* (or *analysis*) directory, which stores all the files required to analyze the collected data. This directory is further divided into sub-directories that mirror the structure of the *experiments* directory, that is, each experiment in *experiments* has its corresponding directory in *results*. It also contains files with helper functions required by all analysis scripts (in this case, aptly named *helpers.R*) as well as a separate directory for any analyses spanning multiple experiments.
4. A *paper* (or *writing*) directory, which contains the files that generate the final submitted version of the paper. When this is a more general *writing* directory, it may contain multiple writing projects, organized as separate sub-directories.

A project repository routinely contains the following additional directories:

5. A *data* directory, if the project is complex and the researchers want to keep all relevant data files in one place.
6. A *models* directory, if the project encompasses a computational cognitive modeling (separate from the statistical analysis) component, as is frequently the case in ALPS Lab.
7. A *talks* directory, where slides from talks given about the project are stored.

Some projects may also encompass corpus searches. Depending on the status of these searches—whether they constitute the main studies of the project or whether they are used to extract information to be used in the analysis of the main studies—they may either merit their own top-level directory (e.g., *corpus_analysis*) or be included in the *experiments* or *analysis* directories. Finally, each repository also contains a *.gitignore* file, which specifies files that Git should ignore (i.e., not track changes in, and not push to the online GitHub repository). We say more on this in section 4.

We consider this to be an optimal project repository structure and have not yet encountered a project that doesn't benefit from this organizational structure. Cleanly separating the files used to generate the experiments from the analysis of the data collected in said experiments is useful because it naturally separates two aspects of the replicability pipeline: (a) the reproducibility of the *experimental methodology* (including the exact task, materials, trial structure, and instructions) and (b) the reproducibility of the *data analysis* (and thereby hopefully the replicability of the results). An external party interested in reproducing the analysis but not in re-running the experiment may therefore directly operate only on the *results* directory without having to wade through the *experiments* directory. In contrast, an external party interested only in seeing exactly what participants' task was can easily navigate to and open the .html file corresponding to the experiment of interest without having to search very hard.

In sections 4 and 5 we describe the contents of the *experiments* and the *results* directories and the associated workflows, which we consider the main components of the project.

## 4 Data collection (experiments directory)

Each experiment in the project was run via Amazon's Mechanical Turk,[4] a crowdsourcing platform that allows individuals to post so-called Human Intelligence Tasks (HITs, e.g., experiments) for workers to complete. Running web-based experiments via Mechanical Turk and similar online platforms such as Clickworker[5] and Prolific Academic[6] has many advantages over in-lab studies, including a larger and more diverse participant population than is typically available on university campuses, as well as very rapid completion times (Buhrmester, Kwang, & Gosling 2011; Mason & Suri 2012).[7] Moreover, many classic results from the cognitive psychology literature have been replicated via Mechanical Turk, suggesting that data from online participants are generally reliable (Buhrmester, Kwang, & Gosling 2011; Crump, McDonnell, & Gureckis 2013). Measures that increase data reliability and reduce dropout include paying participants reasonable rates,[8] running studies in the morning on weekdays, including attention checks, and including a progress bar so that participants have an estimate of the remaining length of the experiment.

In the following, we describe one way the workflow for a Mechanical Turk experiment can be managed; this workflow was used for the current project and is the

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

standard workflow for any web-based experiment in ALPS Lab. Its advantages include a large degree of flexibility in the choice of experimental design and task, automated infrastructure for posting experiments and retrieving data, and a high degree of reproducibility. The initial learning curve is, we believe, very much justified by the payoff.

Each experiment was programmed as an external website using HTML[9]/ JavaScript[10]/CSS[11] and posted to Mechanical Turk. While generating experiments directly using HTML/JavaScript/CSS initially involves a learning curve for the user uninitiated in web programming, we recommend it over using the Mechanical Turk templates or out-of-the box services such as Qualtrics. The reason is that programming one's own experiments allows for more flexibility and control regarding tasks and experimental design than out-of-the-box services do. While the current experiment involved a simple continuous slider rating task in a two-block design with randomization of trials within blocks, we have used the same general infrastructure to run truth-value judgment studies, response time studies, self-paced reading studies, perception studies, free and forced production studies, studies involving drag-and-drop functionality, and many others. Example templates for such studies within the same framework used for this study are provided at https://github.com/thegricean/LINGUIST245B. We encourage interested readers to use the files we have provided as the basis for their own experiments. The internet abounds with good web-programming tutorials to aid in the process of modifying the experiments.

To run the experiments, we uploaded each experiment to J.T.'s university web space to be accessed through an external URL.[12] To link the experiment website to Mechanical Turk as a HIT we used the Submiterator tool (now superseded by Supersubmiterator[13]), which provides an intuitive wrapper around the Mechanical Turk command line tools.[14] The experiment ran until all the data were collected (no longer than a few hours for any of the experiments). Again using Submiterator, the data file was downloaded from Mechanical Turk, participant worker IDs[15] anonymized, and the data reformatted in such a way that the resulting .csv file contained one data point per row with all the relevant information for statistical analysis. This data file was then copied into the corresponding data sub-directory in the results directory, which concluded the data collection process.

### 4.1 Experiment files

We find it useful to label the directories corresponding to the individual experiments that were run in ascending order (e.g., by prefacing the directory names with 1_, 2_). This serves as a reminder of the chronological order in which the experiments were run. In this case, there is only a small number of experiments so this may seem less relevant, but for projects that require a lot of piloting, this ascending labeling is extremely useful. Alternatively, we have found it useful in other projects to include *pilots* and *main* sub-directories that contain the pilot experiments and main experiments to be reported, respectively.

Each of the sub-directories in the experiments has the same structure and consists of the following:

1. An *experiment.html* file that specifies the elements of the experiment and can be opened in a browser to view the experiment.[16]
2. A *js* directory that contains an *experiment.js* JavaScript file that determines the logical flow of the experiment.
3. A *css* directory that contains .css style files that determine stylistic elements of the experiment.

### 4.2 Interfacing with Mechanical Turk

Once the experiments were ready to be deployed, we used Submiterator, a python program originally written by Dan Lassiter and extended by Erin Bennett, to post the experiment to Mechanical Turk, subsequently retrieve the data, and anonymize participants' worker IDs. As noted, this tool has since been superseded by Supersubmiterator, written by Sebastian Schuster, which has the following convenient features; features 2 and 3 were not yet available with Submiterator:

1. One simple command each for (a) posting the experiment, (b) retrieving the results, and (c) reformatting the Mechanical Turk results file (which comes back as one participant's data per row) into a .csv file with one data point per row for easy analysis in R with already anonymized worker IDs.
2. No interaction with the no longer supported Mechanical Turk Command Line Tools is necessary. Simply specify a .config file with all relevant information about the HIT (see the Supersubmiterator documentation for more details, and see figure 47.1 for an example .config file for Experiment 1a).
3. Built-in batch support that makes sure that the total number of assignments (participants) is spread over

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

```
{
"liveHIT":"yes",
"title":"Language study",
"description":"You will rate short dialogs.",
"experimentURL":"https://web.stanford.edu/~jdegen/exp1a/",
"keywords":"language research fun cognitive science linguistics university",
"USonly?":"yes",
"minPercentPreviousHITsApproved":"95",
"frameheight":"650",
"reward":"1.00",
"numberofassignments":"250",
"assignmentduration":"1800",
"hitlifetime":"2592000",
"autoapprovaldelay":"60000"
}
```

**Figure 47.1**
Example .config file for posting experiments to Mechanical Turk with Supersubmiterator.

batches of no more than nine each. This is relevant because, at the time of writing, Amazon charges a 40% fee for HITs with more than nine assignments, but only 20% for HITs with up to nine assignments. The reformatting command automatically generates one merged data file for analysis.

Running Supersubmiterator requires only installing the tool, whereas Submiterator additionally required installing the no longer supported Mechanical Turk Command Line Tools (highly discouraged, given the lack of future support). Additionally, any requirements associated with Mechanical Turk studies generally—including having an Amazon Payments account and a Mechanical Turk requester account—are necessary.

Because using Supersubmiterator means that multiple HITs with nine participants are generated, the issue of how to prevent participants from taking the experiment multiple times is relevant: the Unique Turker[17] service provides that functionality. Setup notes are provided on the first author's LINGUIST 245B "Methods in Psycholinguistics" GitHub repository[18] along with setup notes for the rest of the web-based experiment pipeline used in ALPS Lab and useful Mechanical Turk–related tips.

For the current project, the steps involved in running each experiment were:[19]

1. Program experiment as external website and copy it to web space.
2. Create *mturk* directory inside the *experiments* directory where all Mechanical Turk–related files are stored and from where the Submiterator command will be called.
3. Create *experiment.config* file (see example in figure 47.1) in *mturk*.
4. Run from command line to post experiment to Mechanical Turk:

   ```
   submiterator posthit experiment
   ```

5. Run from command line to retrieve data from Mechanical Turk:

   ```
   submiterator getresults experiment
   ```

6. Run from command line to reformat data for worker ID anonymization and easy analysis in R:

   ```
   submiterator reformat experiment
   ```

7. Copy generated data file *experiment.csv* to its corresponding *data* directory in *results* directory.

Step 7 is necessary because the project repository's *.gitignore* file specifies that all files inside of directories called *mturk* are to be ignored. This is a safety measure to ensure that deanonymized participant data does not end up visible to the public on the internet. It does introduce one step of potential human error if the wrong file is copied or is copied to the wrong place. We consider this preferable to the potential for exposing deanonymized data. A record of participant worker IDs is thus only kept on the local machine of the researcher who ran the experiment.

## 5 Data analysis (results directory)

For this project, and in ALPS Lab generally, we used the open-source statistical software package R (R Core Team 2017) in conjunction with the integrated development environment RStudio (RStudio Team 2016) to analyze the data. Reasons why RStudio is highly recommended include the ease of managing multiple active analysis scripts and plots, workspace visualization, syntax highlighting and prediction, and seamless integration with R Markdown[20] and knitr.[21]

Useful packages we used in this project (1–4 we use routinely) include:

1. *lme4* (Bates, Maechler, & Dai 2009) for conducting mixed-effects analyses.

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

2. *brms* (Bürkner 2017) for conducting Bayesian mixed-effects analyses.
3. *tidyverse* (Wickham 2017) for tidy data wrangling in R.
4. *ggplot2* (Wickham 2016) for data visualization.
5. *lsmeans* (superseded by *emmeans*; Lenth 2016) for computing pairwise comparisons.

The directory structure of *results* roughly mirrors that of *experiments*, in that each experiment receives its own directory in *results*. Each experiment directory further contains sub-directories *data*, *graphs*, and *rscripts*, which cleanly separates data files (e.g., the *experiment.csv* file generated by Submiterator reformat), R analysis files,[22] and graphs generated in the analysis process.

The *rscripts* directory contains separate R scripts for preprocessing the raw data from Mechanical Turk (mostly for the purpose of performing data exclusions, *preprocessing.R*), creating useful visualizations that are saved to the *graphs* sub-directory (*graphs.R*), and analyzing the data using mixed-effects models (*analysis.R*). All scripts should contain enough comments to allow the reader to reproduce the analyses reported in the paper and to follow what was done at each step. This particular way of separating the scripts is not necessary, though it can be helpful to separate preprocessing, visualization, and analysis.

## 6 Concluding remarks

We believe the reported organizational framework for web-based studies is a useful one: it cleanly separates relevant components of the reproducibility pipeline and openly stores all components for others to reproduce and build off of, with the exception of Mechanical Turk–related files that contain confidential participant information.

### 6.1 Generalization to other types of studies

The organizational framework we reported in this chapter generalizes to any project using web-based experiments. Experience suggests that it also easily generalizes to projects that use computational cognitive modeling or corpus searches in addition to (or instead of) behavioral experiments.[23] It also generalizes to lab-based experiments and experiments conducted in the field—storing experimental scripts is a general property of the framework and is not specific to web-based or even more specifically Mechanical Turk experiments. An issue arises when proprietary software, such as E-Prime, is used. In this case, less material can be shared. We consider this a reason to use open-source software whenever possible.

A different problem arises for studies that require storing very large data sets, as is often the case with eye-tracking or ERP studies. GitHub has limits on how much data can be stored within one repository: at the time of writing, no one file can be larger than one hundred megabytes; repositories have a hard limit of one hundred gigabytes; and it is recommended that repositories be kept under one gigabyte in size. While large file storage solutions are constantly being developed, an option for those wanting to remain within the GitHub universe is to use Git Large File Storage (Git-LFS).[24] Another option is to upload downsampled or otherwise compressed data files and keep the raw data files backed up locally.

### 6.2 What would we change?

In general, we consider this an instance of a well-organized and easily shareable project that we recommend as a model to others. However, given the rapid pace at which discussions regarding best practices in open and transparent science are progressing, we have already identified one key issue that we would handle differently, were we to start from scratch: preregistration. The Open Science Framework (OSF) makes it easy to preregister hypotheses, experimental design, and analyses (Foster & Deardorff 2017). The literature suggests that preregistration reduces the potential for many pitfalls in scientific practice, including avoiding problematic researcher degrees of freedom in fiddling with exclusion criteria, determining stopping criteria for running participants, and running exploratory analyses that are framed as planned analyses in the resulting papers (Roettger 2019; Simmons, Nelson, & Simonsohn 2011). Indeed, in current follow-ups to this project we have preregistered our hypotheses (see OSF preregistration at https://osf.io/hn8px/), and the policy we have implemented in ALPS Lab is that any hypothesis-driven experiment must be preregistered. This excludes experiments that only serve norming purposes or are acknowledged to be exploratory experiments that serve a subsequent hypothesis-generation purpose, though we have begun to preregister even those simply to note in an official place what the point of the experiment is. This is not to say that we see preregistration as the solution to all replicability problems, nor that there is anything wrong with conducting exploratory analyses—indeed, we are fans of fully

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

**Table 47.1**
Overview of tools used

| Use | Tool | More information |
|---|---|---|
| Communication | Slack | https://slack.com |
| Version control | Git<br>GitHub | https://git-scm.com<br>https://github.com |
| Experiment development | HTML/JavaScript/CSS | https://www.w3schools.com/html/default.asp |
| Mechanical Turk interface | Supersubmiterator | https://github.com/sebschu/Submiterator |
| Data analysis | R<br>RStudio | https://www.r-project.org<br>https://rstudio.com |
| Preregistration | Open Science Framework | https://osf.io |

exploring and becoming intimate with one's data sets. Preregistering hypotheses simply allows for a clear separation between confirmatory and exploratory analyses (for discussion, see Nicenboim et al. 2018; Roettger 2019).

A second thing we might do differently is to use R Markdown files instead of simple R files for running and documenting analyses. R Markdown allows for weaving together text written in the Markdown language with R source code to generate a coherent document that contains both the output of R code that is run (e.g., visualizations or tables of model coefficients) as well as prose descriptions of the analysis. The resulting document can be compiled into multiple output formats, including HTML and PDF. It can thus function as an internal lab notebook or even as a full paper.

### 6.3 Adopting the organizational framework

We are aware of the steep learning curve involved in adopting the workflow outlined in this chapter for those who have never used any of the described tools, but we believe it is worth it in the long run for all the reasons discussed in this chapter. The reader need not adopt all components of the framework at once. Table 47.1 provides an overview of the tools we discussed. You can add these tools to your own workflow incrementally, which is what we did over the years. Each addition yielded an improvement in transparency and organization of our projects, and we hope the same will be true for you.

#### Notes

1. https://github.com.

2. https://git-scm.com.

3. https://slack.com.

4. https://www.mturk.com.

5. https://www.clickworker.com.

6. https://www.prolific.co.

7. We routinely complete studies that require three hundred participants within a few hours of posting.

8. ALPS Lab typically aims to pay participants at a rate of $12–$14 per hour. At the time of writing, US federal minimum wage is $7.25 and California minimum wage is $12. Workers typically get paid about $3 per hour on Mechanical Turk, which has been documented as a serious and systematic case of labor exploitation (Pittman & Sheehan 2016).

9. https://www.w3schools.com/html/.

10. https://www.w3schools.com/js/.

11. https://www.w3schools.com/css/.

12. The experiments have since been taken off of J.T.'s Ohio State university web space. We provide a lasting example of Experiment 1a at https://web.stanford.edu/~jdegen/exp1a/experiment.html—this is the same experiment that is viewable by the reader by opening the file experiments/exp1a/experiment.html in a browser on your local machine if you have cloned the repository as described in section 3.

13. https://github.com/sebschu/Submiterator.

14. Other tools or entire frameworks that provide interfaces with Mechanical Turk and other crowdsourcing platforms and include some functionality for experiment development directly include psiTurk (http://psiturk.org/) and _magpie (https://magpie-ea.github.io/magpie-site/), among many others.

15. A Mechanical Turk worker ID is a unique identifier associated with a participant, linked to their e-mail address, and thus provides personal identifying information. These IDs must therefore never be posted online.

16. To view the HTML and JavaScript code, open any of the files in your favorite editor. We like Sublime Text (https://www

Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022

.sublimetext.com) and recommend avoiding built-in editors such as Notepad.

17. https://uniqueturker.myleott.com.

18. https://github.com/thegrican/LINGUIST245B.

19. The experiments for this project were run before the existence of Supersubmiterator, so these commands assume the use of Submiterator.

20. https://rmarkdown.rstudio.com.

21. https://yihui.name/knitr/.

22. Researchers may use different statistical software packages for data analysis. For example, for someone who uses python wanting to replicate our workflow, the only difference should be in the *rscripts* directory. We have no recommendations for researchers using SPSS or other drop-down menu software packages except to switch to R or python, for the simple reason that they offer more control, better shareability, automation and consequently increased reproducibility of analysis steps, comparatively easy identification of analysis errors, better online documentation, and, thanks to its large developer base, many more methods than are implemented in SPSS.

23. Examples of recent lab repositories with computational components: https://github.com/thegrican/RE_production.

24. https://git-lfs.github.com/.

## References


Abrusán, M. 2011. Predicting the presuppositions of soft triggers. *Linguistics and Philosophy* 34 (6): 491–535. https://doi.org/10.1007/s10988-012-9108-y.

Bates, D., M. Maechler, and B. Dai. 2009. lme4: Linear mixed-effects models using S4 classes. R package version 0.999375–31.

Beaver, D. I., C. Roberts, M. Simons, and J. Tonhauser. 2017. Questions under discussion: Where information structure meets projective content. *Annual Review of Linguistics* 3: 265–284. https://doi.org/10.1146/annurev-linguistics-011516-033952.

Buhrmester, M., T. Kwang, and S. D. Gosling. 2011. Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science* 6 (1): 3–5. https://doi.org/10.1177/1745691610393980.

Bürkner, P.-C. 2017. brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software* 80 (1): 1–28. https://doi.org/10.18637/jss.v080.i01.

Crump, M. J. C., J. V. McDonnell, and T. M. Gureckis. 2013. Evaluating Amazon's Mechanical Turk as a tool for experimental behavioral research. *PloS One* 8 (3): e57410. https://doi.org/10.1371/journal.pone.0057410.

Foster, E. D., and A. Deardorff. 2017. Open Science Framework (OSF). *Journal of the Medical Library Association* 105 (2): 203–206. https://doi.org/10.5195/JMLA.2017.88.

Karttunen, L. 1971. Implicative verbs. *Language* 47 (2): 340–358.

Lenth, R. V. 2016. Least-squares means: The R package lsmeans. *Journal of Statistical Software* 69 (1): 1–33. https://doi.org/10.18637/jss.v069.i01.

Mason, W., and S. Suri. 2012. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior Research Methods* 44 (1): 1–23. https://doi.org/10.3758/s13428-011-0124-6.

Nicenboim, B., S. Vasishth, F. Engelmann, and K. Suckow. 2018. Exploratory and confirmatory analyses in sentence processing: A case study of number interference in German. *Cognitive Science* 42:1075–1100. https://doi.org/10.1111/cogs.12589.

Pittman, M., and K. Sheehan. 2016. Amazon's Mechanical Turk a digital sweatshop? Transparency and accountability in crowdsourced online research. *Journal of Media Ethics: Exploring Questions of Media Morality* 31 (4): 260–262. https://doi.org/10.1080/23736992.2016.1228811.

R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. https://www.r-project.org.

Roettger, T. B. 2019. Researcher degrees of freedom in phonetic research. *Laboratory Phonology: Journal of the Association for Laboratory Phonology* 10 (1): 1. https://doi.org/10.5334/labphon.147.

RStudio Team. 2016. *RStudio: Integrated Development ENvironment for R*. https://rstudio.com.

Simmons, J. P., L. D. Nelson, and U. Simonsohn. 2011. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science* 22 (11) 1359–1366. https://doi.org/10.1177/0956797611417632.

Simons, M., J. Tonhauser, D. Beaver, and C. Roberts. 2010. What projects and why. *Semantics and Linguistic Theory* 20:309–327. https://journals.linguisticsociety.org/proceedings/index.php/SALT/article/viewFile/2584/2332.

Smith, E. A., and K. C. Hall. 2011. Projection diversity: Experimental evidence. In *Proceedings of the ESSLLI 2011 Workshop on Projective Meaning*, 156–170. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.646.290&rep=rep1&type=pdf.

Tonhauser, J., D. I. Beaver, and J. Degen. 2018. How projective is projective content? Gradience in projectivity and at-issueness. *Journal of Semantics* 35 (3): 495–542. https://doi.org/10.1093/jos/ffy007.

Wickham, H. 2016. *ggplot2: Elegant Graphics for Data Analysis*. New York: Springer-Verlag. https://ggplot2.tidyverse.org.

Wickham, H. 2017. *tidyverse: Easily Install and Load the "Tidyverse."* https://CRAN.R-project.org/package=tidyverse.

Xue, J., and E. Onea. 2011. Correlation between presupposition projection and at-issueness: An empirical study. In *Proceedings of the ESSLLI 2011 Workshop on Projective Meaning*, 171–184. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.646.290&rep=rep1&type=pdf.


Downloaded from http://direct.mit.edu/books/chapter-pdf/1980031/c042800_9780262366076.pdf by guest on 05 July 2022