The *lrd* Package: An *R* Package for Processing Lexical Response Data

Nicholas P. Maxwell & Mark J. Huff

The University of Southern Mississippi

Author Note

Correspondence concerning this article should be addressed to Nicholas P. Maxwell, School of Psychology, 118 College Dr, Hattiesburg, MS, 39406. E-mail: nicholas.maxwell@usm.edu. The source code for this package as well as all applicable data files have been made available for download at https://osf.io/admyx/. The Shiny application can be accessed at: https://npm27.shinyapps.io/lrdshiny/.

Abstract

Cued-recall tests (having participants respond to a cue with a previously studied target word) are a commonly used assessment to gauge how individuals store and retrieve learned information. Data generated from these tests can be analyzed in several ways, however, the output generated from these tests typically requires manual coding that can be time intensive and error-prone before any analyses can be conducted. To address this issue, this article introduces *lrd* (Lexical Response Data), an open-source tool for quickly and accurately processing lexical response data using *R* that can be used either from the *R* command line or using an *R Shiny* graphical user interface. We begin by providing an overview of this package and include a step-by-step user guide. We then validate this program using two methods. First, we use *lrd* to recode output from two cued-recall studies with large samples and test whether the results replicate using *lrd* scored data. We then assess the inter-rater reliability and sensitivity and specificity of the scoring algorithm using human coders for comparison. Overall, *lrd* is highly reliable and shows excellent sensitivity and specificity, indicating that recall data processed using this package is remarkably consistent with data processed by a human coder.

Word Count: 200

Keywords: Memory; Cued-Recall; Lexical Retrieval; Methodology

The *lrd* Package: An *R* Package for Processing Lexical Response Data

People are generally able to acquire new knowledge with relative ease. Much of our understanding of how individuals organize and store learned information comes from the use of recall tasks (see Polyn, Norman, & Kahana, 2009 for a review). These tasks present participants with a set of items to learn within a controlled environment, and following study, participants are asked to recall them on a later test. Recall can either be assessed via free report, in which individuals report information from memory without any cues or constraints, or by the presentation of a cue that is used to direct their retrieval. A common method that Cognitive Psychologists use to gauge cued-recall is through the use of cue-target pairs in which participants are required to retrieve a target item at test with the presence of a cue. Recall tests are common in a variety of memory research domains, including studies investigating the effectiveness of different memory strategies (e.g., deep vs. shallow encoding methods; Craik & Lockhart, 1972), survival processing (e.g., assessing memory for contaminated objects; Gretz & Huff, 2019), and metacognition (e.g., accuracy between judgments of learning and recall; Koriat & Bjork, 2005). Furthermore, because these studies often employ words as the stimuli (i.e., having participants learn a set of cue-target word pairs), much research has been conducted to explore how the lexical properties of the cue and target can influence later recall (e.g., The English Lexicon Project; Balota et al., 2007) or how the semantic relationships between concepts affect recall (e.g., how word associations affect correct recall; Nelson, McEvoy, & Schreiber, 2004). Though the research questions differ, recall studies generally employ lexical information in some capacity, either as the stimuli that participants are required to study, the dependent variable of interest, or most commonly, through a combination of the two.

Cued-recall tests often generate large amounts of lexical data. These tests are commonly used within psychological research. For example, a cursory search of Google Scholar for the keyword “cued-recall” returns approximately 18,000 publications since 2000, with these results spanning multiple subfields of psychology including neuroscience, psycholinguistics, and cognitive aging. The abundance of these studies can be attributed in part to the rise of the internet and the availability of more powerful computers. Within the past two decades, researchers have been able to access a growing number of normed databases with which to construct lexical stimuli for use within these studies (e.g., The English Lexicon Project, Balota et al., 2007; The Semantic Priming Project, Hutchison et al., 2013; The Small World of Words Project; De Deyne, Navarro, Perfors, & Brsybaert, 2019). Recently, online tools to aid researchers in selecting stimuli from the appropriate normed database have been made available (e.g., The Linguistic Annotated Bibliography; Buchanan, Valentine, & Maxwell, 2019a) and computer applications such as the *lexOPS* package for *R* (Taylor, Beith, & Sereno, 2019) have been developed to automate the stimuli selection process entirely while controlling for several types word properties. Though there has been a proliferation of datasets and tools used to aid researchers with stimuli creation, little attention has been given to developing tools that assist researchers with processing the large amounts of data that are typically generated from these studies. Since studies investigating memory typically generate large amounts of lexical data, processing the output obtained from these studies is often a time-consuming and tedious task. Furthermore, the number of participants recruited to take part in these studies has drastically increased within the past decade, resulting in a greater demand for efficient methods for processing memory data. As such, the purpose of this paper is to introduce the *lrd* (lexical response data) package, which has been designed to provide researchers with a set of simple tools that can be used to speed up the processing of lexical output.

Output from recall tests are generally scored by matching participants' responses to the various stimuli against a scoring key containing the correct set of responses. Though typed responses are unquestionably easier to process relative to handwritten responses, each response item must still be checked against the key to determine accuracy. For large datasets, this process of manually scoring data is arduous, resulting in hours of checking participant responses against an answer key. While such tasks can generally be divided amongst research assistants in a lab, this may still prove to be a time-consuming endeavor depending on the amount of data requiring processing. Furthermore, this can potentially introduce error in the coded responses, as inconsistencies amongst raters may arise if not properly controlled for (i.e., misspellings, plural vs. singular nouns, alternate tenses, etc.).

To reduce both the overall time spent processing raw output and potential inaccuracies due to coders, an alternative method is to automate the data coding processes by employing a computer application that can compare participant responses relative to a scoring key. However, simply having a program match responses does not account for participant errors in responses that may still involve a correct memory, as these may be scored as incorrect due to misspellings or the addition of an additional character either before or after a memory item such as an extra space. While a human scorer would certainly count these items as a correctly retrieved memory item, an automated program may not unless a sufficient degree of flexibility is programmed into the scoring package.

The functions comprising the *lrd* package have been specifically designed to accurately score recall data while granting increased flexibility for minor errors that may be in present in recall output. Importantly, this cost-free package has been carefully crafted to require minimal programming experience. Though *lrd* was designed within the context of cued-recall response data, it can be applied to any experimental output that requires matching lexical responses to a scoring key in order to process data for analysis. The goal of this article is two-fold. First, we provide brief overviews of each function contained in the *lrd* package along with a step-by-step guide on how to implement this package to process data. Second, we test the accuracy and reliability of the scoring algorithm by comparing output obtained from this package with human coded data using two large data sets. We test this package's reliability by using its scoring functions to recode cued-recall data derived from two recent memory studies (Maxwell & Buchanan, 2020 and Maxwell & Huff, under review). We then compare the data processed using *lrd* to the findings in the original human coded datasets and test whether the original findings reported in these studies replicate.

The tested studies were selected due to their relevance to the topic at hand (both were memory studies that required participants to complete a cued-recall test) and their similarity in design, which allowed for easy comparison between each study, and their use of a wide variety of different items to test the reliability of the *lrd* package. For each study, participants studied lists of paired associates and judged either how related the words in each pair were (Maxwell & Buchanan, 2020) or how likely they would remember the second word if cued by the first at test using a Judgment of Learning rating (Maxwell & Huff, under review). Upon conclusion of the study/judgment tasks, participants completed a distractor task followed by a cued-recall task in which the first word in each pair was presented and participants were asked to respond with the item it was originally paired with (e.g., mouse - ?). The recall data reported in both of the above studies was scored by manually checking responses against scoring key. We then rescored this output using *lrd* to illustrate that output generated automatically from this package is able to replicate the human scored results with a high degree of precision.

Overview of the *lrd* Package

lrd is an open-source package developed for the *R* environment that consists of four basic functions for scoring lexical response data and assessing the reliability of the scoring algorithm. This package's primary goal is to automate the process of scoring lexical data by matching participant responses to a list of correct responses stored in a key. Critically, this package has been designed to accomplish this task while also controlling for participant errors in responses such as misspellings or incorrect tenses. While this set of functions was developed primarily within the context of processing cued-recall responses, the scoring functions may be applied to most research designs that require participants to respond with individual words (e.g., free recall of word lists).

We begin by providing a set of general instructions for downloading and installing the *lrd* package within the *R* environment. Next, we provide a basic overview of the two scoring functions and provide a general guide on how to use the package within both the *R* environment and through the use of a graphical user interface (GUI) implemented in *Shiny* (RStudio Inc, 2020). Finally, we conclude by assessing the validity of this package by comparing the two scoring functions to process sets of cued-recall data that have been scored by human coders.

Installation and Set Up

The latest version of *lrd* (including all applicable documentation and source code for each function) can be accessed via OSF (https://osf.io/admyx/). While proficiency with *R* is not required, it is assumed that users will have some familiarity with the *R* environment and/or basic experience with object-oriented programming. Installation is relatively straightforward, but currently requires the use of the *devtools* package (Wickham, Hester, & Chang, 2019) to download and install the files from GitHub. Typing the following command, `devtools::install_github("NPM27/lrd")` will begin the installation process by downloading and installing the latest version of *lrd* along with all dependency packages from GitHub. Source code has been made available on GitHub (https://github.com/npm27/lrd/), and researchers are able to download and modify functions of this package as needed.

Function 1: Compute the Percent Match Between Response and Key

The `percent_match()` function allows for the comparison of a participant's typed response with the correct response stored in a scoring key. At a minimum, this function requires three user inputs that are then arranged in a dataframe object: A list of participant responses (generally typed responses collected through a computer based data collection tool such as *E-Prime*), a scoring key containing the correct response, and a unique identifier for each participant. The percent match function works by treating each word in the dataframe columns as a string object. This function then computes the percentage of shared characters between a participant's typed response and the corresponding correct response from the key and returns this value as a new column in the dataframe. The percentages derived by this function are computed bidirectionally to account for differences in length between the response and key that arise due to participant errors (i.e., the larger item is always the denominator when computing this percentage). For example, if the word *home* is stored in the answer key, *lrd* will compute a participant response of *hom* as a 75% percent match with the target due to the absence of 1 of 4 letters (i.e., 25%), while a response of *homme* would be computed as a match of 80%. Table 1 illustrates how output obtained using `percent_match()` is formatted.

Since `percent_match()` relies on the length of the two words being compared when computing this percentage, shorter words are more likely to have lower percent matches and are thus at a disadvantage relative to longer words. This is because typos and misspellings will have a greater negative impact on short words when this percentage is calculated. Table 1 contains an example of this. Looking at the percent match column, the participant responses for the items *home* and *windshield* each contain one typo relative to the answer key. However, because *home* is a four-letter word, the negative impact that this misspelling has on the percent match is magnified relative to when the ten-letter word *windshield* is misspelled. To account for this, `percent_match()` also includes an optional "weighted" argument, should researchers wish to control for item length when processing responses. By activating this argument (`weight.by = TRUE`), the user is able to specify a value that is then used to adjust the percent match values. The weighted percent match is then computed using the following formula:

$$W = p + \left(\frac{v}{c}\right) \tag{1}$$

In Equation 1, p represents the percentage of shared characters between the participant response and its corresponding answer key (i.e., the percent match between the two), v is a user specified weight value ranging from 0 to .99 that is specified using the `weight.by` argument, and c equals the total number of characters comprising the correct response (as stored in the answer key). Weighted match values computed from Equation 1 are then stored in a separate vector which is appended to the dataframe, rather than overwriting the initial, unweighted percent match values. The inclusion of both columns allows the user to see what improvements in scoring accuracy occur by using the weighed match method relative to using the non-weighted percent match values.

Function 2: Score Responses as Correct or Incorrect Based on Percent Match

The second function comprising *lrd* is the `score_recall()` function. This function operates by taking the saved output from `percent_match()` and using values stored in the percent match column to determine whether an item was recalled correctly. Using the `set.cutoff` argument, the user is able to specify a cutoff value that *p* must eclipse in order for the response to be marked as correct. For example, if the cutoff value is set at 0.80, then responses that are at least an 80% match (i.e., $p \geq .80$) would be marked as correct. Because the user is able to freely specify a desired cutoff point, this allows the scoring algorithm to be tuned to the dataset being processed.

Using `score_recall()` returns a .csv file saved to the working directory. The first three columns of this file contain the three initial inputs used when computing the percent match (i.e., participant number, participant response, and answer key). The remaining columns denote the percentage of characters shared between the response and the key, the weighted version of this percentage (if applicable), whether the item was recalled correctly. Scores are represented as a series of 0's and 1's denoting whether the item was correctly recalled (i.e., 0 = incorrect, 1 = correct). Table 1 illustrates the structure of the output file.

Scoring Functions Example

In this section, we provide a guide to using *lrd* to score lexical response data. This example uses a set of simulated response data that were designed to mimic output that might be obtained in a cued-recall study. While this dataset is smaller than what is typically generated from psychological experiments, we note that it is sufficient for our purpose of illustrating how *lrd* scores participant responses. We begin this section by detailing the creation of this dataset before providing a step-by-step walkthrough of the *lrd* package's scoring functions.

Materials and Dataset Creation

To simulate a set of cued-recall data, forty words were randomly generated using *LexOPS* (Taylor et al., 2019) to serve as target items (i.e., the scoring key containing correct responses). To simplify the stimuli selection process, we followed the general example provided by Taylor et al. by controlling for word prevalence and concreteness when generating this set of items. First, only highly concrete words were included (concreteness ≥ 4; Brysbaert, Warriner, & Kuperman, 2014). Pairs were then evenly split based on word prevalence (e.g., the proportion of individuals who are familiar with a word; Brysbaert, Mandera, McCormick & Keuleers, 2019). Thus, the final stimuli consisted of 20 concrete, high prevalence words (i.e., prevalence ≥ 4) and 20 concrete, low prevalence words (i.e., prevalence ≤ 2).

We next simulated five sets of participant responses to these items. These response simulations varied in their degree of accuracy so as to cover a broad spectrum of potential participant responses, including no response errors (Participant 1), minor misspellings (Participants 2 and 3), and major errors in responses (e.g., blank responses, incorrect answers, misspellings of more than two letters, subjects 4 and 5). For Participant 1, all responses matched the key so as to simulate a situation in which a participant correctly recalls all items. Data for Participants 2 and 3 was manipulated to simulate situations in which participants make minor mistakes at recall that don't necessarily preclude them from being counted as correct (e.g., misspellings where it is evident what the intended word is). These were generated by removing, replacing, or doubling specific letters. As such, the letter "e" was removed from all responses for subject 2 (e.g., "hey" becomes "hy"). For Participant 3, the letter "i" was removed from all pairs, all instances of the letter "e" were replaced with "a", and "y" was replaced with "yy" (e.g., "you" becomes "yyou"). This allowed us to simulate a range of common participant errors such as omitting a letter, typing the wrong letter, or double pressing a key by mistake. Finally, data for

Participants 4 and 5 were manipulated to simulate situations in which participants make major mistakes at recall (e.g., responding at test with an incorrect word). To simulate this type of response error for subject 4, five responses from the answer key were randomly changed to a different but conceptually similar word (e.g., *fuel* becomes *gas*). The simulated data for subject 5 increased the number of incorrect responses and added three instances of missing data. The sample dataset (test_data.csv) and the code used to generate it are available for download at https://osf.io/admyx.

Formatting and Loading a Dataset

The *lrd* package requires that the initial input data is formatted as .csv with a header row. This file will need to be arranged in long format and contain the following three columns: A unique identifier for each participant, an answer key containing the correct responses, and a list of participant responses. The input data may contain additional columns, but they will not be processed by *lrd.* Because the scoring functions are case sensitive, the response and answer key columns will need to be checked to ensure that there are not discrepancies in case. For simplicity, we suggest converting both the answer key and response columns to lowercase before scoring the data. Finally, all missing responses will need to be converted from NAs to blanks.

[R CODE]

```
## set up
library(lrd)
dat = read.csv("test_data.csv")
dat = dat[ , -1] #remove index
summary(dat)
# make sure everything is lowercase
dat$Response = tolower(dat$Response)
# replace response NAs with blanks
```

```
dat$Response[is.na(dat$Response)] = ""
```

Scoring a Dataset

Scoring the data is a relatively straightforward process. To begin, run `percent_match()` and save the output as a new object (see code below for an example). When running percent match, you will need to specify the columns containing the participant responses, the answer key, and the subject number. This function returns a dataframe object containing the three input columns and a new column that denotes the percentage of characters shared between the participant response and the answer key. Recall can then be scored by running the `score_recall()` function on the stored output. This function requires specifying the cutoff score for percent match (for this example, we used a cutoff of 75%). The output of this function is saved to the working directory as a .csv file named "output.csv." This file contains the three input columns, the percent match column, and a column denoting whether an item was correctly recalled. An example of the output file has been made available on our OSF page.

[R CODE]

```
# Compute percent match
matched = percent_match(dat$Response, key = dat$key, id = dat$subID)
# Now score the output using a 75% match to compute scores
# Note that score_recall automatically stores output in a .csv file
score_recall(matched, set.cutoff = .75)
```

R Shiny Application

While *lrd* was originally designed to be used from the *R* command line, we have also developed a *Shiny* application that provides researchers with a programming free alternative to using this tool that can be operated with basic Excel skills. The application can be accessed at https://npm27.shinyapps.io/lrdshiny. The input data needs to take the form of a .csv file or tab delimited .txt file with at least three columns that are arranged in the following order: A unique participant identifier, a scoring key, and a set of participant responses. Any additional columns (e.g., those denoting experimental conditions) must be placed starting with the fourth column. To begin the file upload process, the input settings must first be selected based on the type of file (e.g., .txt or .csv file and the type of separator used). Next, the scoring criteria must be specified. The strictest option available is a 95% match criteria, with options to decrease this down to a 55% match (all 5% increments between 95% and 55% are available). After all options are selected, the file can be uploaded. Once the upload is complete, the scored data will appear below to the left of the inputs, and the scored data can be saved as a .csv file using the download button at the bottom left of the screen. See Figures 1 and 2 for illustrations of this process.

Scoring Functions Validation

In the next section, we report the results of two sets of analyses in which we tested the scoring accuracy of *lrd*. Each analysis serves as an additional check to ensure that *lrd* can consistently produce accurate scoring across different sets of stimuli. First, we use *lrd* to score the datasets used for each set of analyses. These data were derived from two sources: Maxwell and Buchanan (2020) and Maxwell and Huff (under review). We then conducted three sets of analyses to test the reliability of this package. First, we tested whether the results of these studies would significantly differ from the original findings after the raw data was processed and scored using *lrd*, allowing us to test the accuracy of this package at the participant level. Finally, we computed Cohen's κ to assess reliability between the different coding sources.

We begin this section by providing details for each dataset, including participant and stimuli characteristics for reach study. We then discuss the selection criteria for the percent match value and detail the results of a set of sensitivity and specificity analyses that were used to test potential cutoff values and provide a step-by-step walkthrough of the scoring process. Finally, we conclude this section by detailing each of the analyses described above.

Participants and Materials

Each dataset was collected separately across two different experimental settings. The first set of participants was originally reported in Maxwell and Buchanan (2020; dataset available at https://osf.io/y8h7v/). This dataset consists of 222 participants who were recruited online via Amazon's Mechanical Turk, a site which allows researchers to access a large pool of participants who complete surveys in exchange for small sums of money (Buhrmester, Kwang, & Gosling, 2011). Next, Maxwell and Huff's (under review; submitted manuscript and dataset available at https://osf.io/hvdma/) data consists of 112 undergraduate students who were recruited from The University of Southern Mississippi's psychology research pool and were tested in lab. These participants completed the study in exchange for partial course credit and were recruited to take part in one of four experiments. For purposes of this paper, we collapsed across experiment so as to include all 112 subjects in one dataset. Combining datasets across both studies resulted in 31,301 recall entries generated from 334 participants.

Datasets were selected due to their similarity in design. Each study presented participants with paired associate study lists and later had them complete cued-recall tasks. Furthermore, each study contained moderately sized samples (all *n*s > 90) and presented participants with at least 60 item pairs to study, providing us with a sufficient number of observations with which to test the reliability of this package. Each study presented participants with a set of cue-target paired associates (e.g., credit – card). Participants were asked to study each pair before making a judgment of either the pair's relatedness or their ability to recall the pair at test. After completing the study and judgment tasks, participants then complete a cued-recall test. While participant judgments were collected in each experiment, they are not included in the following analyses as we are only interested in analyzing the accuracy of *lrd* in scoring recall responses.

First, Maxwell and Buchanan (2020) used 63-word pairs that were selected using the Buchanan et al. (2013) semantic feature overlap norms. The stimuli pairs used in this study were selected based on the strength of their semantic relatedness as measured by cosine overlap (See Buchanan, Valentine, and Maxwell (2019b) for a review of cosine overlap) while also controlling for association strength and thematic similarity. Next, the Maxwell and Huff (under review) dataset used 180 study pairs selected from the University of South Florida Free Association norms (USF norms, Nelson et al., 2004). Stimuli pairs used in this study were originally selected based on their levels of forward associative strength (FAS) and backward associative strength (BAS).

Each of these studies assessed participant recall using the same method. After conclusion of the study tasks, participants completed a cued-recall test in which the first item of each study pair was presented with the second item removed (e.g., *mouse - ?*). Participants in each study were informed that they would not be penalized for guessing or incorrect spellings of answers.

Determining the Optimal Percent Match Cutoff Value

Because the *lrd* package's scoring functions work by computing the number of characters that are shared between two strings (i.e., the percent of characters that are the same within two words), we first needed to determine the optimal cutoff value for the percent match function that would maximize the number of correct hits (e.g., true positives) while also minimizing the number of false positives and false negatives. To determine this value, we conducted a set of sensitivity and specificity analyses for each dataset (see Altman & Bland, 1994, for review). Within the context of this study, sensitivity refers to the proportion of true positives that *lrd* correctly identifies (i.e., a participant correctly responds to the target and the program correctly identifies it), while sensitivity refers to the proportion of true negatives identified by the program (i.e., the program correctly identifies that a participant missed an item on the recall test).

Sensitivity and specificity analyses were computed in *R* using the *caret* package (Kuhn, 2008). Because computing sensitivity and specificity requires that all tested cutoff points be selected a priori, we selected ten percentage values from 55% to 100% at 5% intervals to serve as sample cutoff values (see Tables 2 and 3 for the selected percentages). We note, however, that percent matches below 50% were not included because at a match rate of less than 50%, the majority of characters within each pair would be incorrect, and furthermore, any matching characters would likely be due to chance.

Tables 2 and 3 report sensitivity and specificity percentages for each dataset computed across of the ten tested cutoff points. Overall, both datasets displayed a consistent pattern of results: Sensitivity and specificity were each maximized when the percent match cutoff value was set to 75%, suggesting that this value allowed the scoring algorithm to achieve maximum accuracy. As such, we suggest that 75% provides the optimal cutoff value for minimizing false positives and negatives; however, the program allows researchers to increase or decrease the cutoff value as desired.

Data Processing and Scoring

To assess the reliability of the *lrd* package, we next used its two primary scoring functions to process and score the two cued-recall datasets introduced above. We then compared output obtained through this scoring process to the original, manually coded output originally reported in these studies and tested whether the original findings would replicate.

Prior to running the scoring algorithm, .csv files consisting of the participant responses, answer key, and unique identifiers for each subject were created for each of the three datasets. Data from each study were then scored using the `percent_match()` and `score_recall()` functions. Scoring was an iterative process which used the top four cutoff values from each dataset that maximized sensitivity and specificity as determined by the analyses above. Thus, each dataset was scored 11 times using each percent match cutoff value. This allowed us to track how changing the percent match criteria affected scoring accuracy.

Analyses and Results

After determining the optimal range of cutoff values to use with the scoring functions, we now turn to a set of analyses that test whether the data scored using *lrd* can successfully reproduce the results from each of the original manually scored datasets. We begin this section by providing descriptive statistics of recall rates for both the original and rescored datasets and then test whether these recall rates differ as a function of coder. Finally, we compute the inter-rater reliability between the human coded and *lrd* scored data. Each dataset was analyzed individually, providing us with two separate tests of the *lrd* package's scoring accuracy. For all analyses, significance was set at the $p < .05$ level.

Replication of Cued-Recall Studies

First, each dataset was scored with *lrd* using 10 percent match cutoff values between 55% and 100% that were used in the sensitivity and specificity analyses. Additionally, we included 50% because it represents the minimum acceptable cutoff value, bringing the total number of cutoff values tested to 11. Next, two one-way Analysis of Variance (ANOVA) models were used on each dataset to test whether recall rates differed between the 12 scoring types (the 11 *lrd* scoring criteria plus the human coded data). For completeness, means, 95% *CI*'s and Cohen's *d* effect size indices for all comparisons are reported in Tables 4 and 5.

Starting with the Maxwell and Buchanan (2020) dataset, no significant differences were detected between the human coded data or the *lrd* scored data at any of the percent match cutoff values, *F*(11, 2652) = 0.63, *MSE* = 733.76, *p* = .80. For the Maxwell and Huff (under review) dataset, a significant effect of scoring type was detected, *F*(11, 1332) = 2.67, *MSE* = 187.21, *p* = .002, η_p^2 = .02. However, post-hoc analyses revealed that this effect was largely driven by the 50% *lrd* scoring condition, as mean recall in this condition (47.70) significantly differed from all other conditions (*t*s ≥ 2.04, ds ≥ 0.27), with the exception of the 60% (44.91) and 55% (45.28) cutoff criteria (ts ≤ 1.53). Additionally, the 50% scoring condition was the only set of *lrd* scored data which significantly differed from the human scored data, *t*(221) = 2.03, *SEM* = 1.85, *p* = .04, *d* = 0.27. Recall rates did not differ between the human coded data (43.96) and any of the *lrd* cutoff points. Thus, using *lrd* to score participant responses did not result in significant changes in outcome across any of the experiments, particularly when the cutoff was optimized via sensitivity and specificity analyses. As such, these findings suggest that this package is able to code lexical data at a level similar to that of human coders.

Inter-Rater Reliability

To test the inter-rater reliability between the original data and the rescored data, we computed *κ* values for all data sets at the individual trial level. These values were computed in *R* using the *psych* package (Revelle, 2019). The *κ* statistic ranges from -1 to 1, and inter-rater reliability is considered strong if *κ* exceeded .80 (Cohen, 1960).

Beginning with the Maxwell and Buchanan (2020) data, a strong agreement was detected between the human coded data and each of the 11 *lrd* scored response sets, *κ*s ≥ .90. The

Maxwell and Huff (under review) dataset showed a similar pattern of agreement between coding methods, κs $\geq .89$. Table 6 reports individual κ statistics for all comparisons within each dataset. Across both datasets, reliability between human and *lrd* scored data was highest when a percent match of 75% was used, and lowest when a percent match of 50% was used. As such, these results provide further evidence that using *lrd* to score lexical responses results in output that is highly consistent with what is produced by human coders.

Summary and Conclusion

Although cued-recall tests are widely used in psychology, no open access tools currently exist to quickly process the large amounts of lexical data that these studies generate. The *lrd* package provides researchers with a means of automating this process so as to both save time and reduce coding errors. This package allows researchers to quickly and accurately score large amounts of lexical output, while also being able to control for minor errors in participant responses. As such, we show that data scored using *lrd* accurately reproduces manually coded data by using this package to replicate the results of two cued-recall studies and by testing the reliability of its output relative to hand coded data. We hope that *lrd* will both drastically reduce the amount of time spent coding lexical data and assist the reproducibility measures being adopted by the field by providing researchers with a standardized, open-source method for processing lexical output across psychological studies.

Open Practices Statement

The data for all experiments have been made available at https://osf.io/admyx/ and none of the experiments were preregistered.

References

Altman, D. G., & Bland, J. M. (1994). Diagnostic tests. 1: Sensitivity and specificity. *BMJ (Clinical research ed.)*, *308*(6943), 1552.

Balota, D. A., Yap, M. J., Hutchison, K. A., Cortese, M. J., Kessler, B., Loftis, B., Neely, J. H., Nelson, D. L., Simpson, G. B., & Treiman, R. (2007). The English lexicon project. *Behavior Research Methods 39*, 445-459.

Brysbaert, M., Mandera, P., McCormick, S. F., & Keuleers, E. (2019). Word prevalaence norms for 62,000 English lemmas. *Behavior Research Methods, 51*, 467-479.

Brysbaert, M., Warriner, A. B., & Kuperman, V. (2014). Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods, 46*, 904-911.

Buchanan, E. M., Holmes, J. L., Teasley, M. L., & Hutchison, K. A. (2013). English semantic word-pair norms and a searchable web portal for experimental stimulus creation. *Behavior Research Methods, 45*, 746-757.

Buchanan, E. M., Valentine, K. D., & Maxwell, N. P. (2019a) LAB: Linguistic Annotated Bibliography – a searchable portal for normed database information. *Behavior Research Methods, 51*, 1878-1888.

Buchanan, E. M., Valentine, K. D., & Maxwell, N. P. (2019b) English semantic feature production norms: An extended database of 4436 concepts. *Behavior Research Methods. 51*, 1849-1863.

Burhmester, M., Kwang, T., & Gosling, S. D. (2011). Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality data? *Perspectives on Psychological Science, 6*(1), 3-5.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Education and Psychological Measurement, 20,* 37-46.

Craik, F. I. M. & Lockhart, R. S. (1972). Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior, 11*(6), 671-684.

De Deyne, S., Navarro, D. J., Perfors, A., Brysbaert, M., & Storms, G. (2019). The “Small World of Words” English word association norms for over 12,000 cue words. *Behavior Research Methods, 51*, 987-1006.

Gretz, M. R. & Huff, M. J. (2019). Did you wash your hands? Evaluating memory for objects touched by healthy individuals and individuals with contagious and noncontagious diseases. *Applied Cognitive Psychology, 33*(6), 1271-1278.

Hutchison, K. A., Balota, D. A., Neely, J. H., Cortese, M. J., Cohen-Shikora, E. R., Tse, C.-S., … Buchanan, E. M. (2013). The semantic priming project. *Behavior Research Methods, 45*(4), 1099–1114.

Koriat, A, & Bjork, R. A. (2005). Illusions of competence in monitoring one’s knowledge during study. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 31*(2), 187-194.

Kuhn, M. (2008). Building predictive models in R using the caret package. *Journal of Statistical Software, 28*(5), 1-26.

Maxwell, N. P. & Buchanan, E. M. (2020). Investigating the interaction of direct and indirect relation on memory judgments and retrieval. *Cognitive Processing, 21*, 41-53.

Maxwell, N. P. & Huff, M. J. (under review). The deceptive nature of associative word pairs: The effects of associative direction on judgments of learning. Submitted for review, *Psychological Research.*

Nelson, D. L., McEvoy, C. L., & Schreiber, T. A. (2004). The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers, 36*, 402-407.

Polyn, S. M., Norman, K. A., & Kahana, M. J. (2009). A context maintenance and retrieval model of organizational processes in free recall. *Psychological Review, 116*(1), 129-156.

Revelle W (2019). psych: Procedures for Psychological, Psychometric, and Personality Research. Northwestern University, Evanston, Illinois. R package version 1.9.12, Retrieved from https://CRAN.R-project.org/package=psych.

RStudio Inc (2015). Easy web applications in R. [computer software manual]. (http://www.rstudio.com/shiny/).

Taylor, J. E., Beith, A., & Sereno, S. C. (2019). LexOPS: An R Package and user interface for the controlled generation of word stimuli. Retrieved from https://github.com/JackEdTaylor/LexOPS

Wickham, H., Hester, J., & Chang, W. (2019). Devtools: Tools to make developing R packages easier. R package version 2.2.1 Retrieved from https://CRAN.R-project.org/package=devtools

Table 1

Sample output obtained using the percent_match() function

Participant	Response	Key	Percent Match	Weighted Match
1	hom	home	75%	0.88
1	windsheld	windshield	90%	0.95
1	pepper	pepper	100%	1.00
2	homme	home	80%	0.93
2	windsheild	windshield	100%	1.00
2	pepper	pepper	100%	1.00

Notes. This example uses a weighting criteria of 0.5. Because *lrd* computes percent match values based on matching characters, the position of the characters within the string does not matter. Thus, *windsheild* and *windshield* will each be counted as 100% matches.

Table 2

Sensitivity and specificity results for Maxwell and Buchanan (2020)

Cutoff Criteria	% Sensitivity	% Specificity
100% Match	99.99	94.52
95% Match	99.99	94.52
90% Match	99.99	94.72
85% Match	99.99	95.22
80% Match	99.59	96.00
75% Match	99.17	96.64
70% Match	98.58	96.67
65% Match	98.23	96.79
60% Match	96.80	96.82
55% Match	96.71	96.81

Note. Percent matches of 50% or lower were excluded from this set of analysis.

Table 3

Sensitivity and specificity results for Maxwell and Huff (under review)

Cutoff Criteria	% Sensitivity	% Specificity
100% Match	99.69	92.61
95% Match	99.69	92.61
90% Match	99.69	92.65
85% Match	99.65	93.29
80% Match	98.90	95.29
75% Match	98.61	96.18
70% Match	98.27	96.47
65% Match	97.55	96.98
60% Match	96.33	97.56
55% Match	95.84	97.78

Note. Percent matches of 50% or lower were excluded from this set of analysis.

Table 4

Comparison of mean correct recall percentages across all associative direction groups for Maxwell & Buchanan (2020).

Group	*M*	HC	100%	95%	90%	85%	80%	75%	70%	65%	60%	55%
Hand Coded	54.14 (3.47)	--										
lrd 100%	50.81 (3.60)	0.12	--									
lrd 95%	50.81 (3.60)	0.12	0.00	--								
lrd 90%	50.93 (3.60)	0.12	0.00	0.00	--							
lrd 85%	51.19 (3.58)	0.11	0.01	0.01	0.00	--						
lrd 80%	51.74 (3.55)	0.09	0.02	0.02	0.01	0.02	--					
lrd 75%	52.27 (3.58)	0.07	0.03	0.03	0.05	0.04	0.02	--				
lrd 70%	52.56 (3.59)	0.07	0.06	0.06	0.06	0.05	0.03	0.01	--			
lrd 65%	52.78 (3.59)	0.05	0.07	0.07	0.07	0.06	0.04	0.02	0.00	--		
lrd 60%	53.45 (3.56)	0.03	0.10	0.10	0.10	0.08	0.06	0.04	0.03	0.02	--	
lrd 55%	53.49 (3.55)	0.02	0.10	0.10	0.09	0.08	0.06	0.05	0.03	0.03	0.00	--
lrd 50%	55.28 (3.49)	0.04	0.17	0.17	0.16	0.15	0.13	0.11	0.10	0.09	0.07	0.07

Note. Mean recall rates for each scoring condition. *95% CI*'s are in parentheses. HC = Hand coded data. HC and percentage columns indicate Cohen's *d* effect sizes for post-hoc comparisons, * = $p < .05$.

Table 5

Comparison of mean correct recall percentages across all associative direction groups for Maxwell & Huff (under review).

Group	*M*	HC	100%	95%	90%	85%	80%	75%	70%	65%	60%	55%
Hand Coded	43.96 (2.61)	--										
lrd 100%	40.89 (2.51)	0.22	--									
lrd 95%	40.89 (2.51)	0.22	0.00	--								
lrd 90%	40.90 (2.51)	0.22	0.00	0.00	--							
lrd 85%	41.20 (2.51)	0.20	0.02	0.02	0.02	--						
lrd 80%	42.50 (2.54)	0.11	0.12	0.12	0.11	0.10	--					
lrd 75%	43.06 (2.54)	0.06	0.16	0.16	0.16	0.14	0.04	--				
lrd 70%	43.37 (2.55)	0.04	0.18	0.18	0.18	0.16	0.06	0.02	--			
lrd 65%	43.99 (2.54)	0.00	0.23	0.23	0.23	0.20	0.11	0.07	0.05	--		
lrd 60%	44.91 (2.55)	0.07	0.29*	0.29*	0.29*	0.27*	0.18	0.13	0.11	0.07	--	
lrd 55%	45.28 (2.54)	0.09	0.32*	0.32*	0.32*	0.30*	0.20	0.16	0.14	0.09	0.03	--
lrd 50%	47.70 (2.50)	0.27*	0.50*	0.50*	0.50*	0.48*	0.38*	0.34*	0.32*	0.27*	0.20	0.18

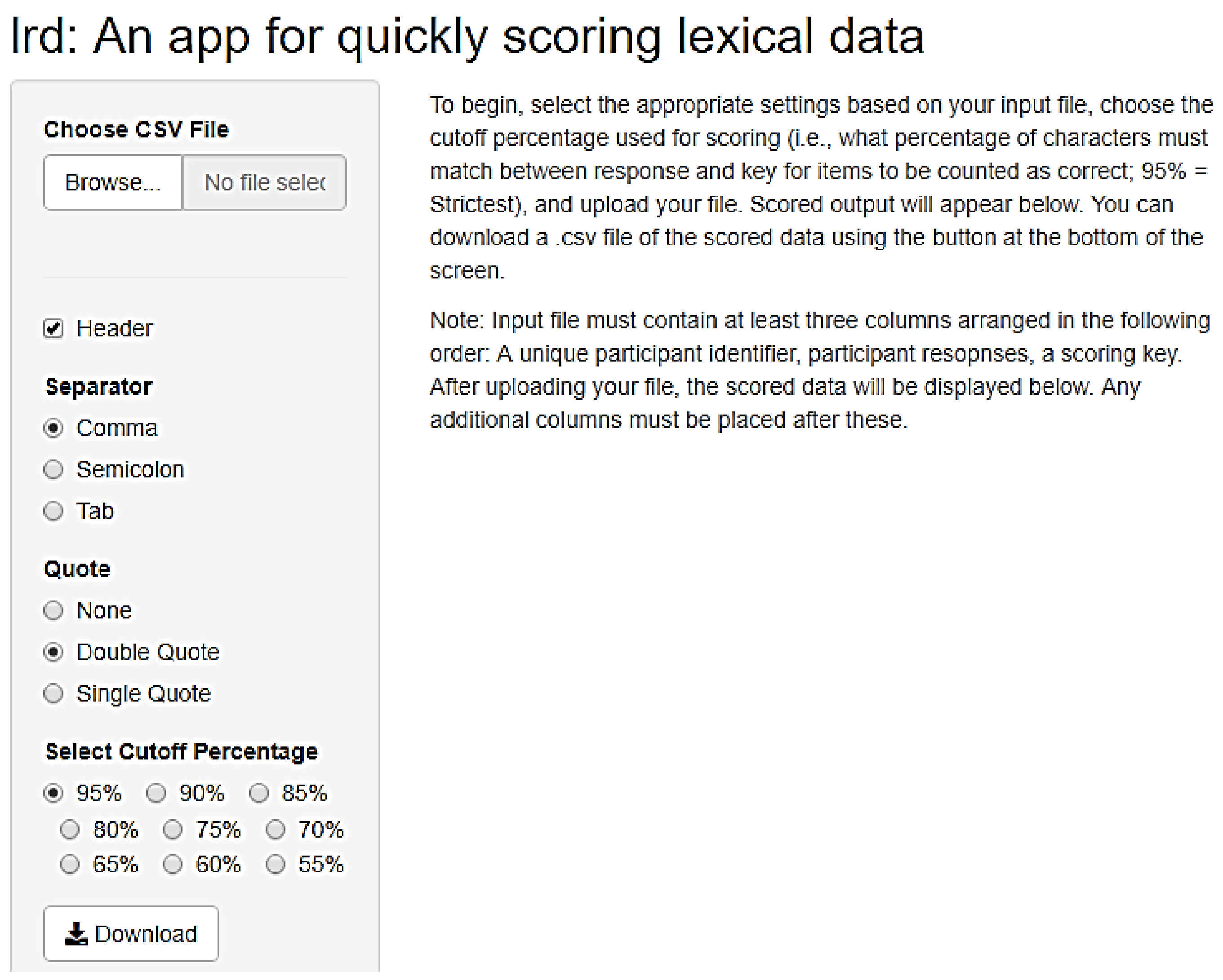
Note. Mean recall rates for each scoring condition. *95% CI*'s are in parentheses. HC = Hand coded data. HC and percentage columns indicate Cohen's *d* effect sizes for post-hoc comparisons, * = $p < .05$.

Table 6

Inter-rater reliability statistics for Maxwell & Buchanan (2020) and Maxwell & Huff (under review)

Experiment	*lrd* 100%	*lrd* 95%	*lrd* 90%	*lrd* 85%	*lrd* 80%	*lrd* 75%	*lrd* 70%	*lrd* 65%	*lrd* 60%	*lrd* 55%	*lrd* 50%
MB	0.94	0.94	0.94	0.95	0.95	0.96	0.95	0.95	0.94	0.93	0.90
MH	0.93	0.93	0.93	0.94	0.95	0.95	0.95	0.95	0.94	0.93	0.89

Note. MB = Maxwell & Buchanan, 2020; MH = Maxwell & Huff (under review). Percent columns indicate criteria used when computing percent match. All values are Cohen's κ between human scored data and data scored at each *lrd* percentage.


lrd: An app for quickly scoring lexical data

Choose CSV File

[✓ Header]

Separator
[✓ Comma]
Semicolon
Tab

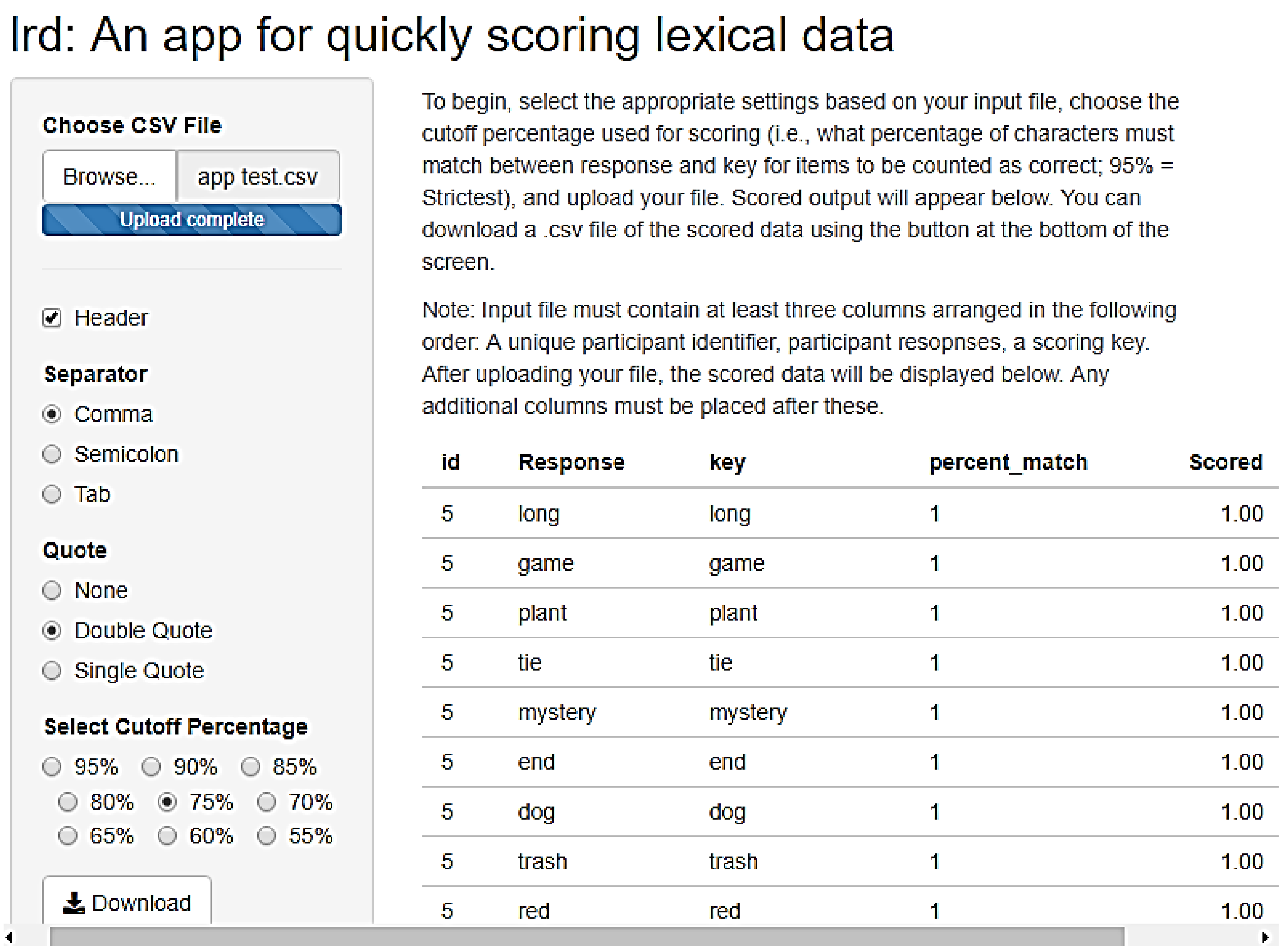
Quote
None
[✓ Double Quote]
Single Quote

Select Cutoff Percentage
[✓ 95%] 90% 85% 80% 75% 70% 65% 60% 55%

To begin, select the appropriate settings based on your input file, choose the cutoff percentage used for scoring (i.e., what percentage of characters must match between response and key for items to be counted as correct; 95% = Strictest), and upload your file. Scored output will appear below. You can download a .csv file of the scored data using the button at the bottom of the screen.

Note: Input file must contain at least three columns arranged in the following order: A unique participant identifier, participant resopnses, a scoring key. After uploading your file, the scored data will be displayed below. Any additional columns must be placed after these.

Figure 1. Illustration of the *lrd Shiny* application before uploading a dataset.



lrd: An app for quickly scoring lexical data

Choose CSV File

Browse... app test.csv

Upload complete

[✓] Header

Separator

● Comma
○ Semicolon
○ Tab

Quote

○ None
● Double Quote
○ Single Quote

Select Cutoff Percentage

○ 95% ○ 90% ○ 85%
○ 80% ● 75% ○ 70%
○ 65% ○ 60% ○ 55%

Download

To begin, select the appropriate settings based on your input file, choose the cutoff percentage used for scoring (i.e., what percentage of characters must match between response and key for items to be counted as correct; 95% = Strictest), and upload your file. Scored output will appear below. You can download a .csv file of the scored data using the button at the bottom of the screen.

Note: Input file must contain at least three columns arranged in the following order: A unique participant identifier, participant resopnses, a scoring key. After uploading your file, the scored data will be displayed below. Any additional columns must be placed after these.

id	Response	key	percent_match	Scored
5	long	long	1	1.00
5	game	game	1	1.00
5	plant	plant	1	1.00
5	tie	tie	1	1.00
5	mystery	mystery	1	1.00
5	end	end	1	1.00
5	dog	dog	1	1.00
5	trash	trash	1	1.00
5	red	red	1	1.00

Figure 2. Illustration of the *lrd Shiny* application after uploading a dataset. Data in this is example is scored using a 75% cutoff criteria.