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, 118 College Dr, Hattiesburg, MS, 39406. E-mail: [nicholas.maxwell@usm.edu](mailto:nicholas.maxwell@usm.edu).

Abstract

Cued-recall tasks (having participants respond to a cue with a previously studied target word) are a commonly used experimental paradigm that assess how individuals store and retrieve recently learned information. Data generated from these types of tasks are analyzed in many different ways, however, they typically require manual coding that is both time intensive and error-prone before any analyses can be conducted. To address this issue, this article introduces *lrd*, an open-source tool for quickly and accurately processing lexical response data in *R*. 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 and test whether the results of these studies replicate when the data is scored with this package. We then assess the inter-rater reliability between the results of the scoring algorithm and human coders. Overall, we show that *lrd* is highly reliable and that lexical data processed through the use of this package does not significantly differ from that of human coders.

Word Count: 178

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 ease. Much of our understanding of how individuals organize and store this knowledge comes from the use of recall tasks. These tasks present participants with a set of items to learn within a controlled environment; after study, participants are asked to recall them on a later test. Recall can either be prompted by the presentation of a studied cue item (i.e., paired-associate learning) or assessed using a free recall task in which participants are simply asked to list off as many items as they can remember. Recall tasks are commonly used across 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 lexical stimuli (e.g., word pairs), much research has been conducted to explore how the lexical properties of stimuli 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, these studies all employ lexical information in some capacity, either as the stimuli that participants are required to study (i.e., such as having participants learn a series of words through paired-associate learning), the dependent variable of interest (i.e., at test participants respond by typing the names of previously viewed items), or most commonly through a combination of the two.

Cued-recall tasks often generate large amounts of lexical data. These tasks are commonly used within psychological research. For example, a cursory search of Google Scholar for the keyword “cued-recall” returned over 6,000 publications since 2016, with these results spanning multiple subfields of psychology including neuroscience, psycholinguistics, and research on aging. The abundance of these studies can in part be attributed 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 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. Though there has been a proliferation of datasets and tools to aid researchers with stimuli creation, relatively little attention has been given to creating tools that assist researchers with processing the output that is generated from these studies. Given that studies investigating memory using cued-recall tasks typically generate large amounts of lexical data, processing the output obtained from these studies is often a time-consuming and tedious task. Thus, the goal of *lrd* is to provide researchers with a set of simple tools that can be used to speed up the processing of lexical data.

Lexical response data is generally scored by matching participants’ responses across various stimuli to an answer key containing the correct set of responses. Though typed responses are unquestionably easier to process relative to handwritten responses, each response must still be checked against the key. Often, this process occurs item by item to ensure the greatest accuracy in coding responses. For large datasets, this process of manually scoring data may result 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 to be processed. Furthermore, this can potentially introduce error in the coded responses, as inconsistencies amongst raters may arise if not properly controlled for (i.e., how are misspellings or alternate tenses handled, etc.).

To reduce both potential errors in scoring accuracy and the overall time spent processing raw output, an alternative method is to automate these processes by employing a computer application that can read in participant responses and match them to an answer key. However, simply having a program match responses does not account for participant errors in responses that may not necessarily preclude an induvial from correctly recalling an item (i.e., discrepancies between the answer and key such as misspellings that do not change a response’s overall meaning). An alternative approach is to use a scripting language such as *R* or *python* to develop a series of functions that can match participant data to a response key while also accounting for these types of response errors.

The functions within the *lrd* package have been specifically designed to address this issue. Additionally, this package prioritizes ease of use and requires minimal programming experience to operate. Though it was designed within the context of cued-recall response data, it can be applied to any experimental output that requires matching lexical responses to an answer key in order to process it. The goal of this article is to two-fold. First, we provide brief overviews of each function contained in the *lrd* package. However, though we present an overview of the package’s functionality, we note that this article is not intended to be read as a tutorial. Detailed instructions on installation and use have been made available at available http://github.com/npm27/lrd. Second, we test the accuracy and reliability of the scoring algorithm comparing output obtained from this package with traditional manually coded data. Specifically, we test this package’s reliability by using its scoring functions to rescore cued-recall data derived from three memory studies (Maxwell & Buchanan, 2020; Maxwell & Huff, under review, and an additional set of unpublished cued-recall data). We then compare data processing with *lrd* to the original datasets and test whether the original findings replicate.

The studies we test were selected due to their relevance to the topic at hand (all were memory studies that required participants to complete a cued-recall task) and their similarity in design, which allowed for easy comparison between each study. For each study, the researchers had participants study lists of paired associates and judge 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 (Maxwell & Huff, under review). Upon conclusion of the study/judgment tasks, participants completed a quick distractor task before moving on to 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., credit - ?). All recall data in the above studies was originally scored by manually checking responses against an answer key. We show that scores obtained from this package are both highly correlated with the original data and are highly reliable based on measures of inter-rater reliability.

**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) and the reliability functions may be used to compute reliability between any two raters, regardless of context. Detailed descriptions of each function are available in a “read me” file (available at https://osf.io/admyx/[ADD EXTENSION TO FILE]).

We begin by providing a set of general instructions for downloading and installing the *lrd* package in the *R* environment. Next, we provide a basic overview of the two scoring functions before providing a general guide on how to use the package We conclude by assessing the validity of this package by using the two scoring functions to process three sets of cued-recall data that have also 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. 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 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. Researchers are welcome to download and modify all functions of this package as they see fit.

**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 an answer key. At a minimum, this function requires three user inputs that are then arranged in a dataframe object: A list of participant responses, an answer key containing the correct response, and a unique identifier for each subject. 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. Using the target item *home* as an example, *lrd* will compute a participant response of *hom* as a 75% percent match with the target, while a response of *homme* would be computed as a match of 80%. Table 1 illustrates how output obtained using percent\_match()is formatted.

Because 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 this percentage is calculated. Table 1 contains an example of this. Looking at the percent match column, the participant responses for the target items *home* and *windshield* each contain one typo relative to the answer key. However, because *home* is a four-letter word, the negative impact 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 the optional “weighted” argument, should researchers wish to control for item length processing responses. By turning on the “weighted” argument, the user is able to select a value (weight.by = ?) which is then used to adjust values in the percent match column. The weighted argument uses the following formula to compute the weighted percent match:

(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 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 is the score\_recall() function. This function works 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* ≥ .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 customized based on 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 step-by-step guide to using *lrd* to score lexical response data. This example uses a set of simulated response data that was designed to mimic output that might be obtained from 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 key containing correct responses). To simplify the stimuli selection process, we followed the general example provided by Taylor et al., 2019 by controlling for word prevalence and concreteness when generating this stimuli set. First, only highly concrete words were included (concreteness ≥ 4; Brysbaert, Warriner, & Kuperman, 2014). Pairs were then evenly split word based on 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, including no response errors (subject 1), minor misspellings (subjects 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 the subject 1, all responses matched the key to simulate a situation in which a participant correctly recalls all items. Data for subjects 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 subject 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 mistakes such as leaving off a letter, typing the wrong letter, or double pressing a key by mistake. Finally, data for subjects 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/[EXTENSION].

**Formatting and Loading a Dataset**

The *lrd* package requires that the 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 not 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 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)

**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 determine the optimal cutoff value 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). The original data may be viewed as (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, the Maxwell & Huff (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. 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 observations 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 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.

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., *credit - ?*). 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 next conducted a set of sensitivity and specificity analyses for each dataset (see Altman & Bland, 1994 for a 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 nine percentage values between 50% and 100% 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, each of the three datasets displayed a consistent pattern of results: Sensitivity and specificity were each maximized when the percent match cutoff value was set between 0.60 and 0.70, suggesting that this range allowed the scoring algorithm to achieve maximum accuracy. Thus, when conducting the analyses in the following sections, we include a set of *lrd* scored responses that were generated using a cutoff value of 0.65 (i.e., 65% of response characters must match the answer key to be scored as correct). We note, however, that the selected values should not be considered as the only viable options. Rather, researchers are encouraged to use several iterations of scoring at various cutoff values to select a cutoff criterion that best fits their dataset.

**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, scoring was conducted four times for each dataset, 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 manually coded and automatically scored data. Each dataset was analyzed individually, providing us with three 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 three percent match cutoff values: 50%, 65%, and 100%. We selected 50% because it represents the minimum acceptable cutoff value, 65% was chosen based on the sensitivity and specificity analyses, and 100% was used as it is the strictest scoring criteria and leaves no room for participant errors. Next, two one-way Analysis of Variance models (ANOVA) were used on each dataset to test whether recall rates differed between the four scoring types (human coded vs *lrd* 100 vs *lrd* 65 vs *lrd* 50). Means, 95% *CI*’s and Cohen’s *d* effect size indices for all comparisons are reported in Table 4.

Starting with the Maxwell and Buchanan dataset, no significant differences were detected between the human coded data or the *lrd* scored data at any of the three percent match cutoff values, *F*(3, 884) = 1.13, *MSE* = 722.9, *p* = .33. Mean recall for the human coded data (54.14) did not differ from the *lrd* data when the cutoff criteria was set to 100% (50.81), 65% (52.78), or 50% (55.28). For the Maxwell and Huff dataset, a significant effect of scoring type was detected, *F*(3, 444) = 4.63, *MSE* = 188.06, *p* = .003, *η*p2 = .03. However, post-hoc analyses revealed that this effect was largely driven by differences between the 50% *lrd* scoring condition and the three other coding conditions. Recall rates did not differ between the human coded data (43.96) and the *lrd* data at the 100% cutoff criteria (40.89), or the 65% cutoff criteria (43.99), though we note that the difference was between the human coded and 50% *lrd* data was significant (47.70, *t*(221) = 2.03, *SEM* = 1.85, *p* = .04, *d* = 0.27). Thus, using *lrd* to score participant responses did not result in significant changes in outcome across any of the experiments, particularly when the cutoff criteria 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 data, a strong agreement was detected between the human coded data and each of the three response sets coded with *lrd* (e.g., 50%, 65%, and 100% cutoff criteria), *κ*s ≥ .90. The Maxwell and Huff dataset showed a similar pattern of agreement between coding methods, *κ*s ≥ .89. Tables 5 and 6 report 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 65% was used, and lowest when a percent match of 50% was used. Overall, these results provide further evidence that using *lrd* to score lexical responses results in output that is consistent with what is produced by human coders.

**Conclusion**

Although cued-recall tasks 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 while 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.

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.

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.

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.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subject | Response | Key | Percent Match | Weighted Match |
| 1 | hom | home | 0.75 | 0.88 |
| 1 | windsheld | windshield | .80 | 0.85 |
| 1 | pepper | pepper | 1.00 | 1.00 |
| 2 | homme | home | 0.80 | 0.93 |
| 2 | windsheild | windshield | 1.00 | 1.00 |
| 2 | pepper | pepper | 1.00 | 1.00 |

*Sample output obtained using the percent\_match() function*

*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

|  |  |  |
| --- | --- | --- |
| Cutoff Criteria | % Sensitivity | % Specificity |
| 100 % Match | 99.9 | 94.5 |
| 90 % Match | 99.9 | 94.7 |
| 85 % Match | 99.9 | 95.2 |
| 80 % Match | 99.6 | 96.0 |
| 75 % Match | 99.2 | 96.6 |
| 70 % Match  65 % Match  60 % Match  55 % Match | 98.6  98.2  96.8  96.7 | 96.7  96.8  96.8  96.8 |

*Sensitivity and specificity results for Maxwell and Buchanan (2020)*

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

Table 3

|  |  |  |
| --- | --- | --- |
| Cutoff Criteria | % Sensitivity | % Specificity |
| 100 % Match | 99.7 | 92.6 |
| 90 % Match | 99.7 | 92.7 |
| 85 % Match | 99.7 | 93.3 |
| 80 % Match | 98.9 | 95.3 |
| 75 % Match | 98.6 | 96.2 |
| 70 % Match  65 % Match  60 % Match  55 % Match | 98.2  97.6  96.3  95.8 | 96.4  97.0  97.6  97.8 |

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

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

Table 4

*Comparison of mean JOL ratings and correct recall percentages across all associative direction groups for each experimental manipulation and pooled analysis.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment | Group | *M* | *95% CI* | HC | 100% | 65% |
| Maxwell & | Hand Coded | 54.14 | 3.47 |  |  |  |
| Buchanan, 2020 | *lrd* 100% | 50.81 | 3.60 | 0.12 |  |  |
|  | *lrd* 65% | 52.78 | 3.58 | 0.05 | 0.07 |  |
|  | *lrd* 50% | 55.28 | 3.49 | 0.04 | 0.08 | 0.09 |
|  |  |  |  |  |  |  |
| Maxwell & Huff, | Hand Coded | 43.96 | 2.61 |  |  |  |
| (Under Review) | *lrd* 100% | 40.89 | 2.51 | 0.22 |  |  |
|  | *lrd* 65% | 43.99 | 2.53 | < .001 | 0.23 |  |
|  | *lrd* 50% | 47.70 | 2.50 | 0.27\* | 0.50\* | 0.27\* |

*Note.* Mean recall rates for scoring condition across each experiment. The three right-most columns indicate Cohen’s *d* effect sizes for post-hoc comparisons, \* = *p* < .05. HC = Hand coded data.

Table 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Group | HC | 100% | 65% | 50% |
| Hand Coded | -- |  |  |  |
| lrd 100% | 0.94 | -- |  |  |
| lrd 65% | 0.95 | 0.96 | -- |  |
| lrd 50% | 0.90 | 0.91 | 0.95 | -- |

*Inter-rater reliability statistics for Maxwell & Buchanan (2020)*

*Notes.* All values are Cohen’s *κ*.

Table 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Group | HC | 100% | 65% | 50% |
| Hand Coded | -- |  |  |  |
| lrd 100% | 0.93 | -- |  |  |
| lrd 65% | 0.95 | 0.94 | -- |  |
| lrd 50% | 0.89 | 0.86 | 0.92 | -- |

*Inter-rater reliability for Maxwell & Huff (Under Review)*

*Notes.* All values are Cohen’s *κ*.