WORD ASSOCIATION DATA PROCESSOR 0.3
A SHORT USER GUIDE

Andreas Buerki
buerkiA@cardiff.ac.uk

CONTENTS

1 INTRODUCTION

The Word Association Data Processor (WA DATA PROCESSOR) is an open-source, free software package which automates key aspects of the processing of word association data gathered from respondents in word association tests. Its user base is expected to be linguists and others working with word association data and employing a methodology similar to that presented in Fitzpatrick et. al. (2015). This brief guide introduces the reader to what the WA DATA PROCESSOR is and how to use it. It assumes basic familiarity with a UNIX-like operating system and a rudimentary understanding of how to run software using a command line interface.[1] This guide is structured as follows: the next section lists the key features of the WA DATA PROCESSOR and is followed by a section with installation information. A tutorial-style introduction to basic operations is presented in section 4. More advanced features and their operation are introduced in section 5. The guide then features a number of expected FAQs (section 6) and a list of features that might be introduced in future releases of the software (section 7). It concludes with some final comments and information on the software license under which the WA DATA PROCESSOR is made available and an appendix describing file formats.

2 KEY FEATURES

The principal goal of the WA DATA PROCESSOR is to facilitate the efficient and consistent categorisation of responses to word association tasks. The WADP comprises of three individual programmes with the following key features:

`categoriser.sh`:

[1] There are a number of introductory texts that cover these topics, including Sobell and Seebach (2006) or Muster (2003).

- a tidy and efficient interface for the manual categorisation of word association responses
- automatic categorisation of responses in cases where categorisations for the relevant cue-response pairs are found in a database of past category ratings
- automatic storage of all new ratings in the database
- tracking of respondent IDs and rater IDs (if provided) in all in- and output files

`reporter.sh`:

- automatic creation of individual response profiles
- automatic creation of cue profiles

`administrator.sh`:

- automatic comparison and reporting of differences between categorisation databases
-
- support for inter-rater agreement procedures through interactive resolution of differences between two database files
- automatic combination of two database files

3 INSTALLATION

The WA DATA PROCESSOR is written in the bash shell scripting language and as such can be run on wide range of operating systems, including Linux, OS X and, if Cygwin is installed, Windows. All recent versions of these systems include bash version 3.2 or later which is minimally required to run the WA DATA PROCESSOR. The WA DATA PROCESSOR was tested and confirmed working on MacOS 10.8, Xubuntu Linux 14.02 and Cygwin 1.7 on Windows 7 and 8.

Under Windows, before WA DATA PROCESSOR is installed, the Cygwin environment needs to be installed by following these steps:

1) Download and then open the application `setup-x86.exe` from `http://cygwin.com/setup-x86.exe`

2) Follow the on-screen instructions. The default installation settings can be used, except in the following: 1) on the 'Choose Installation Directory'-screen, under 'Install For', the option `'Just Me'` should be chosen; 2) on the 'Select Packages' screen, the packages `diffutils` and `ncurses` (both under the `utils` section) need to be installed in addition to the default packages.

3) When the installation completes, a shortcut called `Cygwin Terminal` should appear on the desktop. Also take a moment to locate the directory named *cygwin* (if necessary using the search function). Knowing the location of this directory could be useful later on.

To install the WA DATA PROCESSOR, simply navigate into the directory `WADP` and copy all files ending in `.sh` into the directory `/usr/local/bin`. This can be accomplished by following these basic steps:

1) Open the application `Terminal` (or `Cygwin Terminal` if under Windows)
Under OS X, this is found in `Applications/Utilities`, under Linux typically via menu `Applications>Accessories>Terminal`, under Windows, double click on the '`Cygwin Terminal`' shortcut on the desktop.

2) type `mkdir /usr/local/bin` (it may say 'File exists', that's fine)

3) type `echo $PATH` . If you can see `/usr/local/bin` somewhere in the output, move to step 8, if not carry on with the next step.

4) type `cd $HOME` , type `cp .profile .profile.bkup` (mind the dots). If it says there is no such file, that's fine.

5) type `vi .profile`

6) move to an empty line and press the 'i' key, then enter the following: `PATH=/usr/local/bin:$PATH`

7) now press the `ESC` key, then type `:wq!` and press `ENTER`.

8) type `cd`, followed by a space

9) Drag the directory `WADP` anywhere into the terminal window. The path to WA DATA PROCESSOR's directory now appears on the command line. Now press `ENTER`. We have now moved into the `WADP` directory.[2]

10) now type `cp *.sh /usr/local/bin/` and press `ENTER`.

Installation is now complete.

4 BASIC OPERATION

To illustrate the basic operation of the WA DATA PROCESSOR, the example files found in the `test_data` directory inside the `WADP` directory will be used. To prepare for the following tutorial sections, start the application `Terminal` (or `Cygwin Terminal` if under Windows) and move into the mentioned `test_data` directory by typing `cd`, followed by a space, and then dragging the `test_data` directory into the terminal window. Then press `ENTER`.

The programme categoriser.sh

The main programme within the WA DATA PROCESSOR, `categoriser.sh`, helps categorise word association data. Let us assume we wish to categorise a set of responses to a word association task. Let us further assume that we have, at this moment, no database of previous category assignments available to automate category assignment. To start the process of assigning categories to cue-response pairs, `categoriser.sh` is called in the following manner:

- `categoriser.sh IN-FILE.csv`

where `IN-FILE.csv` is a .csv file containing the word association data to be categorised. This file needs to be formatted according to the formatting specification for in-files given in the appendix (page 13). Essentially, the file must be a .csv file and contain a header (first

[2] In rare cases this can fail on Windows because there may be directories with spaces in the filename. In that case, it is best to move the `WADP` directory into the `cygwin` directory (see box above) and repeat the procedure.

row), listing the cues, followed by any number of rows of responses to those cues, each row consisting of one respondent's responses. A file of this type might be constructed using a spreadsheet application (such as Excel or Open Office) and exported as a csv file. The `test_data` directory contains an in-file of the proper format named `five.csv`. Type the following command to start the process of categorising the data in `five.csv`:

```
categoriser.sh five.csv
```

`categoriser.sh` will first recognise that no database file was provided and asks for confirmation to continue. Press `y` and then ENTER (answers to prompts are generally case-insensitive).[3] Next, `categoriser.sh` asks for the present rater's ID. This is to make it possible to associate categorisations with raters if this is desired. E-mail addresses generally make useful IDs because they are globally unique. Enter an ID (or none) and press ENTER. `categoriser.sh` will now present cue-response pairs for categorisation. By default, it is assumed that categories according to Fitzpatrick et. al. (2015:18-9) will be assigned. Table 1 lists the categories and examples that are also displayed on screen. Additional categories or an entirely different set of categories can be specified if desired (see section 5).

Category	Explanation	Example
(A)	Affix manipulation	irony -> ironic
(CR)	Cue-Response collocation	fence -> post
(CRRC)	Cue-Response & Response Cue collocation	rock -> hard
(E)	Erratic	wolf -> and
(F)	similar in Form only	fence -> hence
(I)	two-step association	weak -> Monday, via week
(L)	Lexical set	bean -> vegetable, bean -> pea
(LCR)	Lexical set & Cue-Response collocation	gold -> silver
(LRC)	Lexical set & Response-Cue collocation	cheese -> bread
(OC)	Other Conceptual	fence -> field
(OCCR)	Other Conceptual & Cue-Response colloc.	long -> corridor
(OCRC)	Other Conceptual & Response-Cue colloc.	attack -> knife
(RC)	Response-Cue collocation	fence -> electric
(S)	Synonym	delay -> impede
(SCR)	Synonym & Cue-Response collocation	torch -> light
(SRC)	Synonym & Response-Cue collocation	shove -> push
(SS)	Synonym plural	variety -> choices

Table 1: Default categories, based on Fitzpatrick et.al. (2015:18-9)

The first cue-response pair that needs to be rated is 'irony -> IRONIC'. Regardless of original case, the response will always be displayed in block capitals. To categorise this pair, choose an appropriate category and press ENTER. Note that it does not matter whether lower or upper case letters are entered. If no rating is entered (i.e. if merely ENTER is pressed), this accepted as a valid response and will appear as an empty field in the output file. However, if that same pair is to be rated again in the future, the programme will ask for a rating again, treating it as though no rating had ever been given. Rate a few more pairs, but no

[3] The default answer, indicated by an upper-case letter, can be given by simply pressing ENTER.

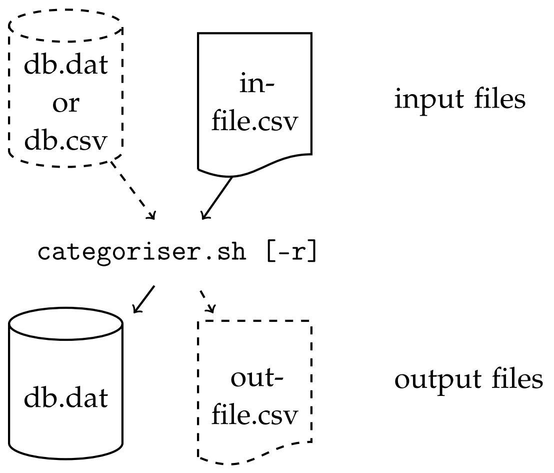

Figure 1: In- and output files of categoriser.sh.
Note. Dashed files are optional; the -r option is explained in section 5

more than ten at this stage. Regardless of whether all pairs in the in-file have been rated or not, it is always possible to interrupt the rating process by inputting 'X' and pressing `ENTER`. Do so now. Ratings up to this point will be saved and the programme will state that the rating is not yet complete and ask whether the user would like an output file for the part that is complete. Enter `Y`, or just press `ENTER`.

Take a look at the `test_data` directory in the `WADP` directory. You will notice that two new files have appeared:

- `categorised-five.csv`[4]
  This file contains the list of rated responses. If you rated less than five cue-response pairs, only the header will be written, otherwise all complete rows will be listed (incomplete rows do not appear in the output file, but the ratings have been saved, see below). To look at the file, you may wish to open it in a spreadsheet application.

- `db-DATE.dat`
  Where `DATE` is today's date. This file is a database of the category assignments you have just made. It can be used to automatically assign a category to identical cue-response pairs in the future. The file itself is in WA DATA PROCESSOR's own database format, but since this is a plain text format, it can be examined in a plain text editor, though there usually is no need to do so.[5]

Note that the columns in the output .csv file are ordered alphabetically by cue. Rows are in identical order to the in-file. `categoriser.sh` will always produce a `db-DATE.dat` file, but it will only produce a `categorised-XXX.csv` file if either the rating is complete, or the user answers 'yes' when prompted.

At this point, it may be worth setting up the terminal window so that it is convenient for working on it for extended periods of time. Exact settings depend on the operating system used, but on all of them, the size of the terminal window, the background colour as well as colour, type and size of font can be adjusted. Typically, a light background with dark font of at least size 12 or 13 pt is recommended.

Let us now assume we wish to continue rating the set of word association data. Note that since the output file `categorised-five.csv` only contains the data that has been rated,

[4] Under some operating systems, the .csv and .dat extensions might remain invisible.
[5] See section 5 for more information on how to edit a database file.

we need to provide `categoriser.sh` with the original in-file in order to carry on with the rating process. Unlike the first time around, however, we now have a database of rated responses that can be used to automatically categorise responses for which we have assigned a category in the past. This means that, as more and more cue-response pairs are categorised, there will be less and less need for manual categorisation work as the correct category of any previously rated pair is automatically assigned through look-up in the database. When supplying `categoriser.sh` with a database, the database must be supplied as the first argument and the in-file to be categorised as the second. The generic command syntax is as follows:

- `categoriser.sh DATABASE.dat IN-FILE.csv`

Type the following command (replacing `DATE` with the appropriate value), and press `ENTER` to continue rating the input file `five.csv`:

```
categoriser.sh db-DATE.dat five.csv
```

`categoriser.sh` will check the database provided and make any automatic category assignments it can. When it encounters a cue-response pair that has not yet been rated, it prompts the rater to assign a category as before. Wait until prompted, and then enter another few category assignments. During manual rating, `categoriser.sh` features the possibility of going back to the previous rating (perhaps because a typo was made, or a rater changes their mind) by typing `B` instead of a rating and pressing `ENTER`. It is only possible, however, to go back to ONE previous rating. If ratings before that need changing, the database file needs to be edited (see page 9). After rating a few more cue-response pairs, exit (or, if continuing to the end, `categoriser.sh` will advise that rating is completed). Again, two new output files will have appeared in the `test_data` directory: a new version of the database, updated with any category assignments made, and a new version of the list of categorised responses. One may wish to discard the old versions (or archive them somewhere), but there is no need to do so, as output files have incremental numbering attached to differentiate them. Typically, only the categorised output list produced after all the cue-response pairs of the in-file have been categorised is worth retaining in the long-term since all previous ones only contain the output up to the time when they were produced. Naturally, the most up-to-date database file should also be retained. Output files can be renamed if necessary, but any database files in WA DATA PROCESSOR's dat format must contain within them the '`.dat`' extension in order to be recognised as database files by `categoriser.sh`. Figure 1 shows schematic summary of how the `categoriser.sh` programme operates.

The programme reporter.sh

To create reports from categorised word-association data, `reporter.sh` is used. Two kinds of report can be produced:

- Individual response profiles: instances of each category summed *by respondent*. A schematic example is shown in table 2. Respondent IDs are only listed if present in the input file.

- Cue profiles: instances of each category summed *by cue*. A schematic example is shown in table 3.

`reporter.sh` requires the following pattern:

respondent ID	CAT1	CAT2	CAT3
respondent 1	3	21	1
respondent 2	0	16	33
respondent 3	24	5	10

Table 2: Individual response profiles

cues	CAT1	CAT2	CAT3
cue 1	89	234	1
cue 2	143	259	0
cue 3	6	272	478

Table 3: Cue profiles

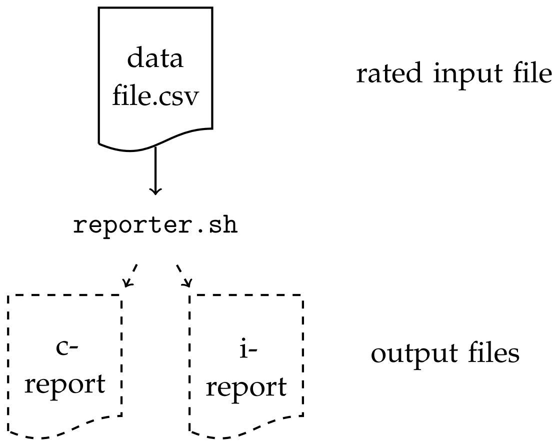

Figure 2: In- and output files of reporter.sh.
Note. Dashed files depend on processing options chosen.

- `reporter.sh FILE.csv`

where FILE.csv is a csv of the format produced as output by `categoriser.sh`. The following command will produce reports for our earlier output file:

```
reporter.sh categorised_five-1.csv
```

The programme will ask what types of report(s) should be produced. Type `I`, `C`, or `IC` and press `ENTER`. All being well, the reports will appear in the working directory. The individual response profile report output is prefixed `i-report_`, the cue profile report is prefixed `c-report_`. These are csv files that can be viewed in a spreadsheet application. Note that categories are listed in alphabetical order and only categories to which assignments were made appear in the report.

The programme administrator.sh

`administrator.sh` performs a number of administrative tasks in relation to database files. It has three basic functions:

1. Produce a report listing all differences between two database files
2. Resolve differences between two database files interactively
3. Combine two database files automatically

The first two of these functions can be useful, for example, when wishing to check inter-rater agreement for a data set that was rated (i.e. had categories assigned) by two different raters. One may first wish to simply produce a report listing any differences, and later, when one is certain of the correct category assignment for each instance where raters differ, one can then use the resolution function to resolve differences (the programme will ask which rating should be used for all instances where ratings differ) and end up with an agreed database file that can be used to assign categories to the same or new input files. Refer to the FAQs (p. 10) for a suggested inter-rater agreement procedure. There are a number of other scenarios in which the mentioned functions will prove useful, such as for the exchange of database files between researchers.

The third function simply combines two database files automatically. While this generally assumes that there are no conflicting ratings in the two databases, `administrator.sh` carefully checks for conflicts and resolves them by giving priority to the first named database file. Any conflicts will also be logged and can then be reviewed by consulting the logfile produced. `administrator.sh` is called in the following manner:

- `administrator.sh db.dat db.dat`

Thus exactly two .dat-formatted database files need to be supplied. The order of the files is significant for the third function where the file that is supplied first overrides in case conflicting category assignments are found. It is also significant for the second function in that, where database files agree, the rater IDs present in the first file are transfered to the resolved file. To try out `administrator.sh`, type the following:

```
administrator.sh db-DATE.dat-1 db-22-10-2014.dat
```

Where `dab-DATE.dat-1` is the latest database file produced earlier and `db-22-10-2014.dat-1` is a database file with nonsense categories for some cues and responses. `administrator.sh` will now prompt the user about which function to carry out. Typing `P` and pressing `ENTER` will produce a list of differences between the two input database files. The differences will be displayed on screen and also saved to a file called `difference_report.txt` in the current directory. Typing `R` will start the process of interactive difference-resolution. The user is prompted to enter their rater ID because this ID will be associated with the decisions made in the difference-resolution process.[6] The user is alerted to differences in cues and responses and a menu is presented for differences in ratings of responses. When all differences have been resolved, a new database file named `resolved-db-DATE.dat` is created and placed in the present working directory. Optionally, a log file can be created that details the differences and decisions made to resolve them. Finally, typing `C` will combine the two database files into a single new database file named `combined-db-DATE.dat`. Any conflicts in ratings will be displayed on screen and saved in a text file called `administrator-log.txt`. Figure 3 shows a schematic overview of input and outfiles of `administrator.sh`.

5 MORE ADVANCED OPTIONS

Rater IDs in categoriser.sh

`categoriser.sh` can optionally show the rater IDs in output files. This includes the rater IDs of category assignments stored in the database. To produce output that includes rater IDs, the `-r` option needs to be passed to `categoriser.sh` like this:

[6] The resolved output file will contain the rater IDs of the first supplied database file where the two raters agree, and the rater ID supplied to `administrator.sh` where they do not.

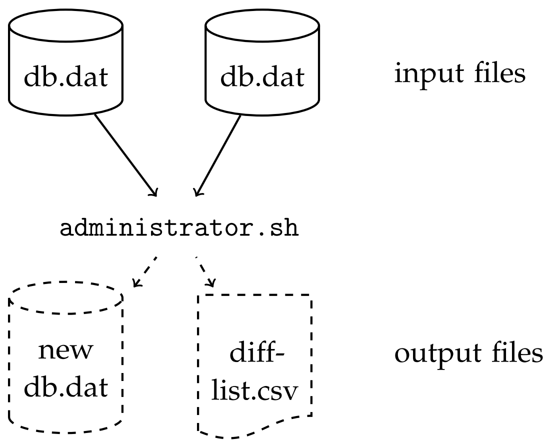

Figure 3: In- and output files of administrator.sh.
Note. Dashed output files depend on options selected; diff-list is a list of differences.

- `categoriser.sh -r DATABASE.dat IN-FILE.csv`

The resulting output file contains a new column called 'rated by' after each category column, indicating the rater IDs of the raters that originally assigned the category in question (if that rater provided a rater IDs at the time of rating). Rater IDs are shown regardless of whether a particular category was assigned in the current session or looked up from the database.

Use of existing database files in csv-format with categoriser.sh

`categoriser.sh` features the ability to import existing csv files with categorised wa-data for use as databases. To make use of this feature, a csv file conforming to the specification for csv-databases (see appendix, page 14) can be provided in lieu of a database in WA DATA PROCESSOR's own dat format. The appropriate command syntax is accordingly the following:

- `categoriser.sh DATABASE.csv IN-FILE.csv`

Note that `categoriser.sh` will produce the usual two types of output files only. The .dat suffixed database will include items supplied via the csv-database.

Editing a database file

If ratings in the database need to be changed for any reason, this can be achieved by editing the respective .dat file in a text editor (by performing a 'find and replace' action, for example). This should be seen as a last resort and a backup copy of the database file to be edited should first be created. This is because of the possibility of inadvertently introducing formatting (such as extra spaces, tabs or line breaks) that can cause errors in automatic processing. Changes should therefore be kept to the necessary minimum. It is essential that after any changes, the file is saved in plain text format. For a brief explanation of the .dat file format, see the appendix (page 13).

Changing the default set of categories used by categoriser.sh

If changes should be made to the categories that are displayed during the rating process, this can be accomplished by editing the file `categoriser.sh` itself in a plain text editor. It may be saver to make a copy of `categoriser.sh` before making any changes, then open it in a text editor. The section starting around line 9 of the file (entitled 'the following section can be adjusted') and ending with the comment 'end of user-adjustable section' hold the information used by `categoriser.sh` to display and handle category assignments. Both the keys and the list of allowed categories should be changed at the same time. Only the information INSIDE of `key='...'` and `allowed_categories='...'` should be adjusted and the single quotes are essential. When editing is finished, the file must be saved in plain text format. Note, however, that `categoriser.sh` accepts any and no category during manual rating (even categories not displayed as possible categories). This is to allow flexibility if needed.

Brief help, version, copyright and licensing information

To display a brief help message, any of the programmes in the WORD ASSOCIATION DATA PROCESSOR can be run with only `-h` after their names (e.g. `categoriser.sh -h`). Similarly, to display version, copyright and licensing information use `-V` after the name of the programme (note the capital letter).

6 FREQUENTLY ASKED QUESTIONS

Q1 I produced an output file without rater IDs. How can I convert it into one with rater IDs?
Answer: Use the most up-to-data database file, together with the unrated version of the in-file that needs to have categories and rater IDs added, and run `categoriser.sh` with the `-r` option active. There should be no need for any manual category assignments since they should all be stored in the database. `categoriser.sh` will produce an output list with the desired rater IDs listed.

Q2 How can I use the WORD ASSOCIATION DATA PROCESSOR in a scenario where inter-rater reliability needs to be checked?
Answer: One suggestion for a procedure would be following:

(a) Prepare the word-association data to be rated in a csv file according to the specifications in the appendix for in-files.

(b) Supply two raters with a) the latest version of a .dat database file and the in-file with data to be rated.

(c) Each rater independently assigns categories to the in-file data using `categoriser.sh`. They return the updated .dat database files.

(d) `administrator.sh` can then be used to first produce a list of inter-rater differences and then to resolve those differences.

(e) The resulting resolved-db.dat file can then be supplied to `categoriser.sh`, together with the unrated in-file to produce a rated output file that includes the resolved ratings. This output file can then be further processed with `reporter.sh`.

(f) The combined database file can then be used as the basis for future ratings. If, while the two raters rated the in-file, the .dat file was changed in some way (such as being used by a third rater to rate more data), the two most recent versions can again be combined into a unified, up-to-date database file using `administrator.sh`.

Q3 I somehow ended up with several database files that all contain valid category assignments not contained in the others. What do I do?
Answer: Use `administrator.sh` to combine the databases pairwise until a combined database with data from all the various database files results.

Q4 I am unsure which database file is the most up-to-date one. How do I find out?
Answer: It is sometimes possible to check the creation or modification date of files (or order them in a window according to any of those dates. If that cannot be done or database files have been modified after creation and do not provide reliable indications, it will be easiest to simply use `administrator.sh` to combine the files that are thought to be the most recent into a single combined database file which will then contain the most up-to-date data.

7 TO DO

The following features are planned for future releases of the WORD ASSOCIATION DATA PROCESSOR:
`categoriser.sh`:

- Option to output rated data with header and column order identical to the in-file (rather than columns being alphabetically ordered by cues in the output file)
- Change the order of manual ratings such that all responses to a cue are rated together, rather than the present mixed order

`reporter.sh`:

- Automatic creation of stereotypy ratings

`administrator.sh`:

- Function to strip a fully rated file of all category assignments and rater IDs

8 COMMENTS, LICENCE AND DISCLAIMER

Comments

Although the WA DATA PROCESSOR has been extensively tested, results should always be checked for rough plausibility at the very least. In its current implementation, the WA DATA PROCESSOR lacks many desirable features and others are implemented in less than elegant ways which is why the software is offered as a beta release. As the software is designated *beta*, certain features of the user interface may change in future releases. Users are advised to read release notes where any such changes will be discussed.

Concerning the licence

The WA DATA PROCESSOR is licensed under the EUPL v.1.1. The EUPL is an open-source license which allows licencees to use, reproduce, modify, distribute and sub-license the software. See the attached `EUPL.pdf` file or https://joinup.ec.europa.eu/software/page/eupl/licence-eupl for the full licence. The copyright holder is Andreas Buerki.

A APPENDIX: FORMAT DESCRIPTIONS

This appendix lists file format specifications used by the WA DATA PROCESSOR.

in-files (files containing word association data to be categorised)

FILE FORMAT csv (comma separated values)

HEADER ROW The first row of the file must be a header row. It must be formatted as in one of the following examples: (the respondent ID field is optional)

respondent ID	cheese	fence	rock	...

or

respondent ID	cue 1: cheese	cue 2: fence	cue 3: rock	...

or

respondent ID	1. cheese	2. fence	3. rock	...

If there are no respondent IDs, the first field is left out:

1. cheese	2. fence	3. rock	4. shove	...

The field labelled 'respondent ID' can in fact contain any text at all as long as the word 'ID' is included. In the cue fields, the second pattern can be used with any text preceding the colon, so 'any text at all: cheese' would be fine.

REMAINING ROWS Depending on whether a 'respondent ID' row was declared in the header, these rows take either the format

respondent 1	moon	barbed wire	hard	...
respondent 2	goat		and roll	...

or

moon	barbed wire	hard	shovel	...
goat		and roll	push	...

There may be empty fields, such as when no response was given by a respondent. Respondent IDs can be anything, but must be unique for each respondent.

dat-database files

FILE FORMAT dat (content is in plain text)

HEADER ROW none

REMAINING LINES Each line of the document consists of a cue word, followed by a tab, followed by a response in uppercase, followed by a bar (|), followed by a category, followed by a semi-colon, followed by the ID of the rater that entered this particular rating. This is followed by a tab and the next response, etc. All lines end in a tab, followed by a line break. Here are the beginnings of two lines of an imaginary database file:

fence	POST|M;buerkiA@cf.ac.uk	BARBED_WIRE|M;buerkiA@cf.ac.uk
irony	IRONIC|M;buerkiA@cf.ac.uk	NOT_FUNNY|M;buerkiA@cf.ac.uk

FILE FORMAT csv (comma separated values)

HEADER ROW The first row of the file must be a header row. It must be formatted as in the following example:

WA001	cheese	WA002	rock	...

Note that a field with 'WA' immediately followed by one or more digits, needs to alternate with fields containing cue words.

REMAINING ROWS As in the following example:

moon	OC	barbed wire	RC	...
goat	RC	and roll	CR	...

Fitzpatrick, T., Playfoot, D., Wray, A., & Wright, . J. (2015). Establishing the reliability of word association data for investigating individual and group differences. Applied Linguistics. Accessible at http://applij.oxfordjournals.org/content/36/1/23.full.pdf+html

Muster, J. (2003). *Introduction to Unix and Linux*. New York: McGraw-Hill.

Sobell, M. G., & Seebach, P. (2006). *Practical Guide to Unix for MAC OS X Users*. Upper Saddle River, NJ: Prentice Hall.