Open Source IRAP

Implicit Relational Assessment Procedure

License

Version

0.9.6 (5/5/2016)

Written in PsychoPy 1.82

NB This code is still in beta - I haven't used this in an experiment yet or had someone else do a code review.

Notes

Task fidelity

This implimentation of the IRAP has high fidelity to the procedure described in Barnes-Holmes et al. (2010: a sketch of the IRAP and REC model), and to the implimentations of the IRAP written in Visual Basic 6 by Dermot Barnes-Holmes (i.e., the IRAP "2010" etc). Most task parameters are soft coded and can be changed via the `task.xlsx` file.

- Each trial presents a "label" stimulus at the top of the screen, a "target" stimulus in the middle of the screen, and two response option labels at the bottom left and right of the screen. There are two label stimuli categories and two target stimuli categories, which when combined create four "trial types".
- You can employ an arbitrary number of stimulus exemplars per category, but all columns in the excel stimulus file must have the same number of rows (i.e., exemplars).
    - NB This could be changed by dividing the stimuli file into two seperate ones and sampling the labels at random but the targets in a counterbalanced manner. Indeed, this method of unequal selection and counterbalancing of label vs target stimuli is how the original implimentations of the IRAP in Visual Basic 6 by Dermot Barnes-Holmes were constructed (e.g., the "2010" version etc.). However, assuming that there are equal number of label and target stimuli, the current implimentation is technically supperior as it results in an equal number of presentations of all label stimuli exemplars, which the VB6 versions do not.
- The default stimulus file employs text stimuli for both labels and targets. The alternative stimuli file employs images for both. This implimentation can display text or images for either. To change this, edit the stimuli.xlsx file.
    - If using text stimuli, put `blank.png` in the img columns and

the text stimuli in text stimuli column.
    - If using image stimuli, put a single space character (i.e,) in the text stimuli column and the name of the image file (including extension) in the image stimuli column. Failure to do either of these will cause Psychopy to throw an error message.
- The location of the response options (left vs right) can be either fixed or moving randomly. Default is "false" (fixed), set this to "true" for moving.
- Number of trials per block is equal to four times the number of rows in the stimuli.xlsx file. Each block contains an equal number of each trial type.
- Participants complete pairs of blocks in which the response contingencies alternate (e.g., self-positive-true vs. self-positive-false). Each block is preceeded by a customisable responding rule (Rule A and Rule B), and followed by feedback about the median latency and % accuracy in the block.
- The order of presentation of the blocks (i.e., block order) is set for each participant in the dialogue box that appears after you run the task. The default is "a" (rule A first). Set this to "b" for Rule B first. That is, if you're counterbalancing block order, the researcher assign the block order for each participant when the task is run.
- Participant complete practice block pairs (default max 4) until they meet mastery criteria on both blocks in a pair (default median latency <= 2000ms and accuracy >= 80%), and then a fixed number of test block pairs (default 3). If mastery criteria are not bet within the max the task skips the test blocks and goes to the end screen.
- Inter trial interval is set to 400 ms.

Timing accuracy

- PsychoPy is technically capable of better timing accuracy than Visual Basic 6, depending on design choices by the researcher (see Garaizar & Vadillo, 2014).
- The current implimentation is written to be at least as accurate as the VB6 implimentations of the IRAP (i.e., accurate to within a frame or c.17ms). The stimuli to be presented within a block are generated on the preblock rule screen, and then `pop()` 'd on each trial.
- If you're looking for higher accuracy (e.g., for EEG/fMRI work) you'll want to change all timings to frames rather than seconds. You may also want to remove the presentation of images if you're not using them.
    - NB no assessment of jitter has been conducted for the current implimentation.

Usage

- The task is written in PsychoPy, a free and open source python library for delivering psychology experiments.
- To run the task, download a copy of PsychoPy and open either the IRAP.psyexp or IRAP.py file in PsychoPy. PsychoPy runs locally on Windows, Mac, and Linux.
- You can run either the `.psyexp` file or the `.py` file inside PsychoPy. The `.py` file should have greater cross platform support; if you run into errors with the `.psyexp` file use the py instead.
- The escape key quits the task at any time. The return key ends the task properly once it’s complete.
- All stimuli and instructions can be altered by editing the `.xlsx` files. - PsychoPy has unicode support, so translating the task into other languages only requires changes to these excel files. All text presented in the task are set via the excel files.
    - NB a poorly documented bug is that if you zip and unzip

excel files using archieve utility on Mac OS X, unicode characters are no longer correctly displayed and will throw an ASCII error in PsychoPy. Make new excel files to correct the issue.

- When you run the task, an autoresponse 'monkey' can be invoked by setting "UseMonkey" in the dialogue box to "y" or "yes". This will simulate keypresses throughout the task to that you can test your script without you having to hit D and K interminably. NB you may have to lower your accuracy criterion to below 50% (i.e., just put it to 0) for the task to run through entirely, as the monkey simply simulates the D key and then the K key, in that order, on every trial.

Data output

- `.psydat`, `.csv` and `.log` files are produced for each participant. The `.csv` file alone is sufficient to most analyses (e.g., calculation of D scores).
- To my understanding, the format of the `.csv` output files are Tidy Data compliant (Wickam, 2014) and therefore easy to analyse in R with little to no processing needed.
    - NB If a participant gets 100% of trials correct throughout the task then the incorrect response RT column will not be created for that participant. This is a) extremely unlikely, and b) not a problem if you process data files based on column header matching (e.g., most R methods, including the bundled script). However, it can be problematic if your data processing workflow relies on column order rather than column header name (e.g., a SPSS script using a GET command).

Data processing R script

- The included `data processing.r` R script produces accuracy and latency summary data and *D*1 scores for each participant (including "overall" *D*1 scores, *D*1 scores for each trial-type, and split-half overall *D*1 scores).
- Very little familiarity with R/Rstudio is needed to use this script: simply change the set working directory line to the location of your data (e.g., `setwd("~/git/IRAP/data`), and the save output line to your chosen directory (e.g., `write.csv(all_tasks_df, file = '~/git/IRAP/data processing/IRAP_data.csv', row.names=FALSE)`), and then run the script.
- Some important notes on the methods included in this script are included below.

*D*1 scoring method

D scores are (Greenwald et al., 2003) are a variant of Cohen's *d*, and are used to quantify the effect size difference between two response patterns (e.g., rts on block As vs. block Bs in an IRAP). They differ from Cohen's *d* in how standard deviations are calculated, subvariants differ in their exclusion criteria and the presence/absence of an error penalty. *D*1 scores are have been employed in the majority of published IRAP research to date (although see next heading). The generic steps in calculating *D*1 scores are as follows:

1. Participants with >10% of (test block) trials <300ms are exlcuded.
2. All rts > 10000 ms are excluded.
3. *D*1 = (mean rt block B - mean rt block A) / SD of all trials in blocks A and B.

As noted above, one key step in calculating *D*1 scores is excluding participants who produce >10% rts < 300ms. The current R script outputs the variable `exclude_based_on_fast_trials` which indicates that a participant should be excluded if `TRUE` .

- These exclusions must be done by the researcher, and are not automatically done by the script, in order to allow the auto response monkey functions correctly and quickly.
- Researchers may not be that familar with this step as it is not produced by Dermot Barnes-Holmes' "IRAP 2010" program and variants, written in VB6.

D-IRAP vs *D*1 nomenclature

Many published articles refer to the "*D*-IRAP" score rather than the "*D*1" score, as it was originally referred to by Greenwald et al. (2003). This was on the rationale that there are differences between the two, e.g., when applied to the IRAP scores are often calculated for each trial type rather than one overall score. However, *D*1 refers only to the general strategy of [difference between means/SD of all items, with some exclusion criteria]. Indeed, even when applied to the IAT, "pure" *D*1s are not typically calculated; rather, one is typically caculated for blocks 3&6 and a second for 4&7 and the two are then averaged. As such, to seperate the generic effect size score from the analytic strategy employed in a given experiment (e.g., overall D scores, trial type D scores, etc), this script refers to *D*1 scores throughout.

Block-pair *D*1 scores vs. All-task *D*1 scores

Some background for this point is required: Dermot Barnes-Holmes' "IRAP 2010" program (and variants, written in VB6) both delivers the task and calculates *D*1 scores (however, it is both closed source and its output is difficult to work with: hence the motivation for the Open Source IRAP). The method employed to calculate *D*1 scores in the VB6 IRAP programs, which has been is detailed in several published articles (e.g., Barnes-Holmes, Barnes-Holmes, Stewart & Boles, 2010), notes that four *D*1 scores are caculated for each test block pair, one for each trial type.

Given that most IRAP studies deliver three pairs of test blocks, these *D*1 scores are then averaged across the three block pairs to leave four trial-type *D*1 scores. One "overall" *D*1 score is then often caculated by averaging these four trial- type *D*1 scores. As such, this method involves the calculation and averaging of a large number of point estimation effect sizes.

An alternative "whole task" method is employed to caculated *D*1 scores in the R script here, whereby the number of point estimation effect sizes is purposefully minimised, so that each is calcualted using the maximum number of data points. Specifically, an "overall" *D*1 score (simply called *D*1) is calculated from all the test block reaction times at once (split only by which half of a block pair they occured in). Trial type *D*1 scores are then calculated by splitting the reaction times up by trial type and recalculating *D*1 scores, but again pooling across all test blocks. This is arguably statistically more appropriate.

I've compared *D*1 scores produced by the two methods from a real dataset of typical size (n = 61), and correlations between the two methods are extremely high ($r > .99$), and means and SDs are equivalent. Additionally, difference scores between the two are not correlated with *D*1 score or absolute values of *D*1 scores.

The *take home point* here is that the two methods are generally comparable, so choice of method should not affect publication etc. However, the "whole task" method employed here is:

a. Arguably more statistically appropriate.
b. Requires fewer steps and is therefore easier to explain in a manuscript. E.g., " *D*1 scores (Greenwald et al., 2003) were calculated from the test block data"
c. As such, it is therefore also easier to interpret.
d. Finally, by calculating all-task *D*1 scores, this method constrains the degrees of experimenter freedom regarding how to conduct

exclusions of test block data. Several articles have employed the method used by Nicholson & Barnes-Holmes (2012) which excludes single test block pairs and averages the remaining ones. However, De Schryver, Hughes, De Houwer & Rosseel (On the Interpretation of Reliability in the Context of Implicit Cognition, in prep) make a persuasive arguement for treating the data produced by a given instance of a measure (e.g., an IRAP) as a single analytic unit. Personally, I plan to make test block exclusions based on performance at the whole task level rather than the block pair level in future. Calcualting "whole task" *D*1 scores is both in line with this and precludes the possiblity of cherry picking a method post hoc.

Known issues & to do list

1. R script for processing D scores, development and proofing.
2. Needs a block length multiplier, given that many people will use only 3 or 4 stimulus exemplars. A simple multiplier variable in the task.xlsx file could make this easier.

Changelog

0.9.6

- Made the code that generates and selects the stimuli for each trial in a block more transparent by writing it as a function and calling it rather than repeating code. No functional difference for the user, but better code transparency and consistency across the IAT/RRT/IRAP code.

0.9.5

1. Added option for block order selection via the dialogue box.
2. Added autoresponse monkey
3. Added option for image stimuli.
4. Added Colored text for pre block rules.
5. Added option for moving response options.
6. Tidied up post block text and locations.
7. Added warning level logging
8. Separated the block layout and stimuli excel files. This allows for an arbitrary number of exemplars for each stimulus category.
9. Commented out the saving of median latency and % accuracy data to the csv file, as it was causing problems in reading the file into R.