

Open Source IRAP

Implicit Relational Assessment Procedure

License

Copyright (c) Ian Hussey 2015

(ian.hussey@ugent.be)

Released under the GPLv3+ open source license.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

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.

Notes

Task fidelity

This implementation 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 implementations 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".
- Participants complete pairs of blocks of trials in which the response contingencies alternate (e.g., self-positive-true vs. self-positive-false). Each block is preceded 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 must 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 $\leq 2000\text{ms}$ and accuracy $\geq 80\%$), and then a fixed number of test block pairs (default 3). If mastery criteria are not met 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 implementation is written to be at least as accurate as the VB6 implementations 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 pre-block 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 implementation.

Usage

1. Running

- 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 `Open Source IRAP.psyexp` Or `Open Source IRAP.py` file in PsychoPy. PsychoPy runs locally on Windows, Mac, and Linux. It's not possible to run PsychoPy scripts online.
- You can run either the `Open Source IRAP.psyexp` file or the `Open Source IRAP.py` file inside PsychoPy. The `Open Source IRAP.py` file should have greater cross platform support; if you run into errors with the `Open Source IRAP.psyexp` file use `Open Source IRAP.py` instead.
- The escape key quits the task at any time. The return key ends

the task properly once it's complete.

2. Localisation and customisation

- All stimuli and instructions within the task are set via the `stimuli.xlsx` and `task.xlsx` files.
- PsychoPy has Unicode support, so translating the task into other languages (Spanish, Polish, Japanese, etc.) only requires changes to these excel files.
 - NB a poorly documented bug is that if you zip and unzip excel files using archive 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.

3. Stimuli

- Label and target stimuli can be either text or image stimuli. The default `stimulus.xlsx` file employs text stimuli for both labels and targets. The alternative file in the `alternative stimuli files` folder employs image stimuli for both labels and targets, but you can mix and match as you like, even use some text and some image stimuli within a category.
 - The file that will be used is that which is placed in the same folder as the `Open Source IRAP.psyexp` / `Open Source IRAP.py` that is run. All image stimuli should be placed in the same folder. Remember to reduce your image files to as small as possible so as to minimise load and rendering time.
 - If using text stimuli, put `blank.png` in the image columns (e.g., `labelB_image_stimuli`) and the text stimuli in text stimuli column (e.g., `labelB_stimuli`). The task presents images on every trial either way; this sets it to present a black square that is effectively invisible. As such, the `blank.png`

file must always be left in the same folder as the `Open Source IRAP.psyexp` / `Open Source IRAP.py` files.

- 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 this or the above step will cause PsychoPy to throw an error message.
- 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). The number of trials per block is a function of the number of exemplars (see task parameters below).
 - NB This could be changed by dividing the `stimulus.xlsx` file into two separate ones and changing the code to sample a) the labels at random and b) the targets in a counterbalanced manner. Indeed, this method of unequal selection and counterbalancing of label vs. target stimuli is how the original implementations of the IRAP were constructed (i.e., those in Visual Basic 6 by Dermot Barnes-Holmes, such as the "2010" version etc.). However, assuming that there are equal number of label and target stimuli, the current implementation is technically superior as it results in an equal number of presentations of all label stimuli exemplars, which the VB6 versions do not.
- The pre block rules (e.g., "respond AS IF...") and response options (i.e., "similar" and "different") are specified in the `task.xlsx` file.

4. Task parameters

- The number of trials per block is equal to (the number of rows in the `stimuli.xlsx` file) * (4 [the number of trial types]).
 - For example, IRAPs frequently employ four exemplars per stimulus category, and 32 trials per block. The default

stimulus file accomplishes this by including the four exemplars twice each (on separate rows) in the `stimuli.xlsx` file. 8 rows(4 exemplars *2 rows each)*4(trial types) = 32 trials per block.

- Each block contains an equal number of each trial type. This is determined by the `block_layout.xlsx` file, which specifies how label and target stimuli are combined to make trial types. This file should not be altered.
- Task parameters such as the practice block mastery accuracy and median latency criteria, maximum number of practice block pairs, number of test block pairs, location of the response options (left vs. right), and on-screen location of all stimuli are specified in the `task.xlsx` file.
 - Response options locations can be either fixed or moving randomly. Default is "False" (fixed), set this to "True" for moving.
 - All screen locations are in PsychoPy's [normalised units](#).
- The values provided in the included file are representative of commonly used task parameter values among published studies.
E.g.:
 - $\geq 80\%$ correct
 - median latency $\leq 2000\text{ms}$
 - Max 4 practice block pairs
 - 3 test block pairs

5. Auto-response "monkey" for piloting the task

- When you run the task, an auto-response 'monkey' can be invoked by setting "UseMonkey" in the dialogue box to "y" or "yes". This will simulate key presses throughout the task so 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 (Wickham, 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 *D1* scores for each participant (including "overall" *D1* scores, *D1* scores for each trial-type, and split-half overall *D1* 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/Open Source IRAP/data")`), and the save output line to your chosen directory (e.g.,
`write.csv(all_tasks_df, file = '~/git/Open Source IRAP/data_processing/processed_IRAP_data.csv',`

`row.names=FALSE)`), and then run the script.

- Some important notes on the methods included in this script are included below.

1. *D1* 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, sub variants differ in their exclusion criteria and the presence/absence of an error penalty. *D1* scores have been employed in the majority of published IRAP research to date (although see next heading). The generic steps in calculating *D1* scores are as follows:

1. Participants with >10% of (test block) trials <300ms are excluded.
2. All rts > 10000 ms are excluded.
3. $D1 = (\text{mean rt block B} - \text{mean rt block A}) / \text{SD of all trials in blocks A and B}$.

As noted above, one key step in calculating *D1* 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 familiar with this step as it is not produced by Dermot Barnes-Holmes' "IRAP 2010" program and variants.

2. *D-IRAP* vs *D1* nomenclature

Many published articles refer to the "*D-IRAP*" score rather than the "

D1 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, *D1* 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" *D1*s are not typically calculated; rather, one is typically calculated for blocks 3&6 and a second for 4&7 and the two are then averaged. As such, to separate 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 *D1* scores throughout.

3. Block-pair *D1* scores vs. All-task *D1* scores

Some background for this point is required: Dermot Barnes-Holmes' "IRAP 2010" program (and variants) both delivers the task and calculates *D1* 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 *D1* scores in the IRAP 2010 program (and variants), which has been detailed in several published articles (e.g., Barnes-Holmes, Barnes-Holmes, Stewart & Boles, 2010), notes that four *D1* scores are calculated for each test block pair, one for each trial type.

Given that most IRAP studies deliver three pairs of test blocks, these *D1* scores are then averaged across the three block pairs to leave four trial-type *D1* scores. One "overall" *D1* score is then often calculated by averaging these four trial-type *D1* 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 calculate *D1* scores in the R script here, whereby the number of point estimation effect sizes is purposefully minimised, so that each is calculated using

the maximum number of data points. Specifically, an "overall" *D1* score (simply called *D1*) is calculated from all the test block reaction times at once (split only by which half of a block pair they occurred in). Trial type *D1* scores are then calculated by splitting the reaction times up by trial type and recalculating *D1* scores, but again pooling across all test blocks. This is arguably statistically more appropriate.

I've compared *D1* 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 *D1* score or absolute values of *D1* 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., "*D1* 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 *D1* 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 argument 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. Calculating "whole task" *D1* scores is

both in line with this and precludes the possibility of cherry picking a method post hoc.

Known issues & to do list

1. Why is a call to `disctint()` needed in the R file, where do the duplicated rows come from? Ideally this should be removed, or accidentally duplicated participant codes will be treated as a single instance.
2. Do a final check of the interplay between manipulating the task parameters and the R processing script.

Changelog

0.9.6

1. 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 codebase.

0.9.5

1. Added option for block order selection via the dialogue box.
2. Added auto response monkey
3. Added option for image stimuli.
4. Added option for moving response options.
5. Tidied up post block text and locations.
6. Added warning level logging
7. Separated the block layout and stimuli excel files. This allows for

an arbitrary number of exemplars for each stimulus category.