Analyses for iABC results (version 1.2.11)

1) **read_iABC_results**: This script requires you to select a **.json** file to process as this has been downloaded from the iABC admin site (you can select multiple files too). As the .json file contains the results of all the project tasks, this script will split the file into one .mat file per task. If the user has run a task multiple times (e.g. a multiple session task or they stopped and re-run the task), then there will be multiple .mat files for this task. The suffix of these files will be incremental based on their execution time (e.g. _1, _2, etc). During script execution, the script will give you a printout with some information per task in the Matlab command window. Once this script is executed, .json files will no longer been needed for analysis (but should be stored somewhere safely) as the rest of the scripts use only the .mat files.

   Note: If a task is not completed (i.e. stopped or abandoned), you will get a prompt whether you want its results to be saved or not. This decision is up to the experimenter and will depend on the specifics of the project.

2) The subsequent analysis is **task specific** from here on. Please select the right script for the task in question.
   a) **run_QN_analysis**: This script processes the Questionnaire results (_qn suffix). It only allows you to select one .mat file at a time and will give you a printout of the answer to each question.
   b) **run_IGT_analysis**: This script processes the Iowa Gambling results (_iowa suffix). It only allows you to select one .mat file at a time and will give you a printout of the analysis. The script will save the pre-processed data into a *processed_data* variable and the results from the analysis into a *results* variable. The *results* variable stores the following outputs: (i) **nonrisky choices** (number of times a "non-risky" deck is selected), (ii) **risky choices** (number of times a "risky" deck is selected), (iii) **total points earned** (i.e. *points_net)*, (iv) **net score** (number of non-risky choices minus risky choices; *net*) and (v) **change in net score** (net score of the last 20 trials minus net score in the first 20 trials; *net_change*). We recommend using *net* and *net_change* for statistical analyses.
   c) **run_WCS_analysis**: This script processes the Wisconsin Card Sorting results (_wisconsin suffix). It only allows you to select one .mat file at a time and will give you a printout of the analysis. The script will save the pre-processed data into a *processed_data* variable and the results from the analysis into a *results* variable. The *results* variable stores the following outputs: (i) **number of perseverative errors** per rule (trials until finding the correct rule after a change) and (ii) **number of non-perseverative errors** per rule (errors after finding the correct rule).
   d) **run_WCS_analysis_v2**: This is a variant of the above script for Wisconsin Card Sorting results (_wisconsin suffix). It only allows you to select one .mat file at a time and will give you a printout of the analysis. The script will save the pre-processed data

into a *processed_data* variable and the results from the analysis into a *results* variable. The *results* variable stores the following outputs: (i) **number of perseverative errors** per rule (trials persisting with the old rule after a change), (ii) **number of efficient errors** per rule (trials AFTER changing the response from the old rule and until finding the correct new rule), and (iii) **number of non-perseverative errors** per rule (errors after finding the correct rule). Note that the number of perseverative errors plus the number of efficient errors equal the number of perseverative errors in the other script.

e) **run_PAL_analysis**: This script processes the Paired Associates Learning results (_PAL suffix). It only allows you to select one .mat file at a time and will give you a printout of the analysis. The script will save the pre-processed data into a *processed_data* variable and the results from the analysis into a *results* variable. The *results* variable stores the following outputs: (i) **number of missed responses** per condition, (ii) **number of correct responses** per condition, (iii) **number of trials** (number of presentations required to locate all patterns correctly across conditions), (iv) **errors** (total number of errors across conditions) and (v) **memory score** across conditions (number of correct responses in the first presentation of a condition, summed across conditions). We recommend using *num_trials*, *errors* and *memory_score* for statistical analyses.

f) **run_ProbRev_analysis**: This script processes the Probabilistic Reversal results (_reversal suffix). It only allows you to select one .mat file at a time and will give you a printout of the analysis. The script will ask you to accept or define a learning criterion, as the number of consecutive correct responses within a phase to consider rule as learned (default value was defined as the minimum number of trials that will definitely contain a trap trial for 80/20 probabilities). Then, it will save the pre-processed data into a *processed_data* variable (trials separated by phase) and the results from the analysis into a *results* variable. The *results* variable stores the following outputs: (i) **perseverance** in old rule after rule reversal, (ii) **number of response switches** after negative feedback (i.e. *probswitch*), (iii) **number of incorrect responses** per phase (i.e. *errors*), (iv) **number of response switches** after negative feedback per phase (i.e. *probswitch*), (v) **number of correct responses for trap trials** per phase (i.e. *proberror*), (vi) **probability matching score** per phase (i.e. *probmatch*; as *probswitch*/*proberror*), (vii) **number of trials until learning criterion** is met per phase (i.e. *trials_to_crit*; NaN if it is not met), (viii) **number of errors after learning criterion** is met per phase (i.e. *postcrit_errors*; NaN if it is not met) and (ix) **maintenance failure score** per phase (i.e. *maintenance_failure*; as *postcrit_errors/#remaining_trials*). We recommend using *perseverance* and *probswitch* for statistical analysis (*maintenance_failure* and *probmatch* might be good candidates too).

g) **run_SL_analysis**: This script processes the Sequence Learning results (_SL suffix). It will ask you to select one or more .mat files to process. Please only select multiple files if you want them processed as one training (e.g. multiple sessions of the same subject). The script will give you a printout for each phase and will ask you if you want to plot the two behavioural indices (*PI* and *strategy*). If you do, the figures will be saved as **_PI.fig* and **_Strategy.fig*, respectively. The script will also save the pre-processed

data into a *processed_data* variable (separated by phase) and the results from the analysis into a *results* variable (separated by phase). The *results* variable stores the following outputs: (i) **average response time** per phase and run (in seconds), (ii) **absolute performance index** per phase and run (i.e. *pi_abs*), (iii) **baseline performance index for random response** per phase and run (i.e. *pi_rand*), (iv) **relative performance index** per phase and run (i.e. *pi_rel*), (v) **strategy choice** per phase and run (i.e. *strategy*), (vi) **exact matching strategy** per phase and run (i.e. *exactMatch*), (vii) **exact maximization strategy** per phase and run (i.e. *exactMaximize*) and (viii) **strategy index** per phase (i.e. *strategy_icd*; it is not defined if there is only one run in this phase). We recommend using *pi_rel* (last two minus first two runs) and *strategy_icd* for statistical analysis.

h) **run_SRT_analysis**: This script processes the Serial Reaction Time results (_SRT suffix). It will ask you to select one or more .mat files to process. Please select **all** files to be processed as one training program (e.g. multiple sessions of the same subject). The script will give you a printout for each phase. The script will also save the pre-processed data into a *processed_data* variable (separated by phase) and the results from the analysis into a *results* variable (separated by phase). The *results* variable stores the following outputs: (i) **response time** for high-probability vs for low-probability transitions (in seconds). This script can still be improved with a more sophisticated analysis.

i) **run_ProgMat_analysis**: This script processes the Progressive Matrices results (_matrices suffix). It only allows you to select one .mat file at a time and will give you a printout of the analysis. The script will save the pre-processed data into a *processed_data* variable and the results from the analysis into a *results* variable. The *results* variable stores the following outputs: (i) **duration of the task** (in seconds), (ii) **average response time** (in seconds) and (iii) **percentage of correct responses** (as *#correct/#trials*).

j) **run_Associates_analysis**: This script processes the Remote Associates results (_associates suffix). It only allows you to select one .mat file at a time and will give you a printout of the analysis. The script will save the pre-processed data into a *processed_data* variable and the results from the analysis into a *results* variable. The *results* variable stores the following outputs: (i) **duration of the task** (in seconds), (ii) **average response time** (in seconds) and (iii) **percentage of correct responses** (as *#correct/#trials*).