Visual Memory N-back Tests
For web browsers
(model-view-controller)

Programming Manual

8-12-2011
Olympia Colizoli
University of Amsterdam
olympia.colizoli@gmail.com

N-back tests
N-back tests are designed to measure working memory, since the person taking the test is required to remember what object was presented 'n' items back. Difficulty increases with n.

Test schema:
Object memory (32 tests)
=> Categories
1. Orientation
2. Shapes
3. Colors
4. Faces
5. Spatial
6. Global vs. Local
7. Verbal vs. Visual

=> Matching condition
1. Identity (Nback = 2)
2. Partial (feature specific, Nback=2)
3. RT (not a memory test, Nback=0)

Procedure:
Tests are given within a Dojo StackContainer in blocks of 4. The first few content panes are general instructions. Before each test, a practice round of 10 trials is given. The real test has 100 trials. Specific instructions are given before each practice round and again before the actual test. Scores are given at the end of each block, in order to promote finishing the whole block.

The instructions are a variation of "Press 'Match' when the current item matches the one presented N trials back." This depends on the exact category tested (see Appendix B).

For the 0-back reaction time tests, the instructions are "Press Match each time you see a 'target goes here'. An alert pops up with the target number. The task will start when you press OK".

The Match button is located directly below the square where the images are presented.

When ‘OK’ is clicked, the test begins. The first image shows up 1000 ms after they click OK.

An image is shown for a certain duration (pic_time – isi_time = 1000 ms) with a blank screen in between images for isi_duration (1000 ms).
This is repeated until max_trials has been reached.
The number of targets = 33% of the max_trials.

A response is a button press (Match button)
total_responses are recorded.
A hit is a response during a target image.
total_hits are recorded.
A miss is an absence of a response during a target image.
total_misses are recorded.
An incorrect is a response during a non-target image.
total_incorrect are recorded.
Reaction time = time image presented – response time.
If there was no response, then reaction time = 0.

The response window = stimulus_time + *isi_time* (2000 ms). As soon as a new image is presented, the response window for the previous image has ended.

The final score is calculated:
Score = hits/ (hits + incorrect + misses)
This gives a score out of 100, i.e. if incorrect = 0, misses = 0, then, hits/hits = 1.

Database:

exp1_subject_info

1. subject_id
2. email
3. language
4. age
5. sex
6. education
7. glasses
8. color_blind
9. colored_sequences
10. colored_sound
11. spatial_form
12. netherlands
13. participation
14. informed_consent

nback_blocks_exp1_eng (English version)

1. nback_id
2. exp_id
3. block_name
4. factor
5. object_category
6. nback
7. number_trials
8. number_stimuli
9. stimulus_time
10. isi_time
11. matching
12. duration
13. stimuli_names
14. language
15. instructions1
16. instructions2
17. example (for an illustrated example picture)

A. nback_trials

1. trial_id
2. subject_id
3. nback_id = id of each individual test in nback_blocks
4. completed = 0 or 1
5. targets = number of targets
6. responses = number of responses
7. hits = number of hits
8. misses = number of misses
9. incorrect = number of incorrect responses
10. score = hits/hits+misses+incorrect
11. rt_correct = average reaction time for hits (ms)
12. rt_incorrect = average reaction time for incorrect responses (ms)

13. sequence = the algorithm for each test
14. image_order = the order of the stimulus array after being shuffled
15. meta_data = trial by trial data
    i. trial_number
    ii. position in sequence (starts with position = 0)
    iii. name_stimuli
    iv. response = 0 for no, 1 for yes
    v. hit = 0 for no, 1 for yes
    vi. miss = 0 for no, 1 for yes
    vii. rt_correct
    viii. rt_incorrect
16. start_time
17. time_stamp = updated automatically

The data is sent to the server at the beginning of the test and every *post_unit* number of trials, as well as the end of the test:

1) Beginning of test
2) Every *post_unit*
3) End of test

Currently *post_unit* = 10

Dependencies:

stack_container.css – *style sheet for nback tests*

exp1_common_code.js – *code for all 3 versions of nback matching*
exp1_identity.js – *code specifically for identity matching*
exp1_partial.js – *code specifically for partial matching*
exp1_rt.js – *code specifically for reaction time (0-back)*

shuffle.js – *shuffles an array*
sumElements.js – *sums the elements in an array*
contains.js – *tells if a certain element is in an array*

...dojo/resources/dojo.css - *dojo style sheet*
...dijit/themes/tundra/tundra.css - *dijit style sheet*
...dojo/dojo.js - *link to dojo "isDebug: true, parseOnLoad: true"*

```
dojo.require("dojo.parser");
dojo.require("dijit.layout.ContentPane");
dojo.require("dijit.layout.StackContainer");
dojo.require("dijit.form.Button");
```

Functions:

View*: general_instructions.php*

check_email() – Checks the form field to make sure an email has been submitted; calls subject_info();
subject_info() – Tthe very first page, checks to see if the email is in database; returns 1 if new subject, 0 if old subject; returns list of tests from block_tracker() in exp1 controller; calls tests();
check_info() – If the subject is new, they will take the questionnaire; this function makes sure that all fields are answered; calls question_info().
question_info() – Sends the subject information to the database; sends the stackcontainer forward.
tests() – Sorts the tests which come from the block_tracker() function in exp1 controller, returned with the subject_info() function; puts the tests in the necessary arrays for processing. Calls number_of_tests();
number_of_tests() – Adjusts stackcontainer in case there are less than 4 tests; inserts instructions in innerHTML; removes questionnaire if subject is old; sends the stackcontainer forward.
start_experiment() – Checks to see if it is practice or the real thing, setting the number of trials accordingly; sets the type of matching (identity, partial, reaction_time); starts the experiment.
show_scores() – Shows the scores of the 4 tests in the current block, with picture example of stimuli.
get_scores() – Gets all the scores of completed tests; returns scores and nback_ids in JSON format; calls write_scores()
write_scores() – Averages all scores; shows how many tests have been completed; lists all scores of completed tests, with picture example of stimuli
check() – writes the value from the radio buttons to global variables for questionnaire

exp1_common_code.js

send_data() – dojo.xhrPost
preload() – Preloads the images in *imgNames*
setStartTime_RT() – Time when the image is presented
getElapsedTime() – Returns a difference between 2 dates in milliseconds
set_initial_positions() – Constructs an array with all positions = true
rand() – Returns a random number between a lower and upper bound
indicate_response() – Turns the border around the objects back to black (it turns blue)
indicate_response_correct() – Turns the border around the objects back to green
indicate_response_incorrect() – Turns the border around the objects back to red
reset() – Resets response, hit, miss and rt back to 0 for the next trial
finished() – Saves scores; changes practice condition; increases block_tracker; resets variables; moves the stackcontainer forward.
blank_screen() – Sets the initial image to the white screen; a blank screen to go in between objects.
match_func() – When the Match button is pushed, changes the color of the box around the objects, gets reaction time, and indicates there is a response.

practice_match_func() – When the Match button is pushed, changes the color of the box around the objects, gets reaction time, and indicates there is a response.
calculate_responses() – total hits, misses, responses, incorrect, average rt, score
ResponseManager() – An object keeps arrays of hits, misses, responses, rt, and meta_data.
theRM() = new ResponseManager() – Copies the data previously in it and adds new data
reset_everything() – Resets all global variables for the new test (when block_tracker increases)

exp1_rt.js, exp1_partial.js, exp1_identity.js

These are specific for the type of matching:
define_trials() – Determines the sequence of the test (the most important function)
change_picture() – The animation, keeps track of trials, calls response manager

Controller: *exp1.php*

index() – general_instructions.php
scores() – Will show all the possible scores
check_email() – Is the subject new? Gets/sets $subject_id and $new_subject session data
send_subject_info() – Inserts questionnaire data into exp1_subject_info
block_tracker() – chooses the tests to be presented in each block, by checking to see what the subject has already done.
subject_scores() – Gets all the scores and nback_ids for completed tests.
send_nback_data() – validates the post, splits into two arrays – one for sessions and one for trials, and then calls the two functions in the model for these

Model: *exp1_model.php*

email() – Checks exp1_subject_info for existing emails
get_id() – Selects $subject_id from exp1_subject_info
new_subject() – Inserts new email into exp1_subject_info
questionnaire() – Inserts subject info into exp1_subject_info
nback_stimuli4() – Selects information for nback tests when 4 tests are presented
nback_stimuli3() – Selects information for nback tests when 3 tests are presented
nback_stimuli2() – Selects information for nback tests when 2 tests are presented
nback_stimuli1() – Selects information for nback tests when 1 test is presented
sessions() – inserts session_data into nback_sessions table
trials() – inserts trial_data into nback_trials table
check_tests() – See what tests the subject has completed
check_test_scores() – Gets the $session_id of completed tests, in order to get the scores
get_all_tests() – All possible tests to take
get_scores() – Returns the scores and nback_ids using completed session data
count_completed() – How many completed tests have been done by a single subject

Difference in algorithms between Identity, Partial and Zero-Back code

Identity:
define_trials();
For i->33%, Pick a random number out of the image names and set it as a target.
Place this number 'nback' trials before.
Fill in the rest of the sequence with image names that do not generate another target (Either +nback or –nback).
Count the number of targets. There are usually more than 33%.
If so, then shuffle the positions of targets and take as many extra targets as the difference between actual targets and 33% targets.
Count the number of targets again.

match_func();
stimuli_names[counter] === stimuli_names[counter-nback]

Partial:
define_trials();
For i->33%, Pick a random number out of the image names and set it as a target. This number is the index of the image names array (because there are multiple images in each index).
Pick another number that choses the image within the index of the target number so that it is not the same exact image.
Place one image 'nback' trials before, and the other at the current spot.
Fill in the rest of the sequence with image names that do not generate another target (Either +nback or –nback).
Count the number of targets. There are usually around 37-38, so I added code that subtracts 3 targets.

match_func();
contains(partial_order[counter-nback], imgNames[trial_order[counter]])
&& partial_order[counter] != partial_order[counter-nback])
An image and itself is not a correct match!

Zero-back:
define_trials();
For i->33%, Pick a random number out of the image names and set it as a target.
Fill in the rest of the sequence with image names other than the target.
Count the number of targets.

match_func();
stimuli_names[counter] === t

Appendix

Instructions:
1. Orientation: You will see a series of images. Click "MATCH" each time the object is in the same orientation as it was 2 trials back.
2. Shape: You will see a series of images. Click "MATCH" each time an object is identical to one which appeared 2 trials back.
3. Color: You will see a series of images. Click "MATCH" each time the color of the object is the same as it was 2 trials back.
4. Faces: You will see a series of images. Click "MATCH" each time a face is identical to one which appeared 2 trials back.
5. Spatial: You will see a series of images. Click "MATCH" each time the object is in the same position as it was 2 trials back.
6. Global: You will see a series of images. Click "MATCH" each time the global (biggest) image is the same as it was 2 trials back.
7. Local: You will see a series of images. Click "MATCH" each time the local (smallest) image is the same as it was 2 trials back.
8. Perceptual: You will see a series of object words. Click "MATCH" each time the shape of that object matches the shape of the object named 2 trials back.
9. Semantic: You will see a series of words. Click "MATCH" each time the meaning of that object matches the meaning of the object named 2 trials back.
10. Letters: You will see a series of letters. Click "MATCH" each time you see a letter identical to one which appeared 2 trials back.
11. Rhyme: You will see a series of words. Click "MATCH" each time the current word rhymes with the word which appeared 2 trials back.
12. Reaction Time: You will see a series of numbers. Click "MATCH" each time you see your target number. For this test, you only need to remember your target number.