The N-back Test

Jackm

December 20, 2019

Contents

1 Introduction

In 1958, Wayne Kirchner invented the n-back test [2]. The n-back test is a visuospatial task that has been shown to improve working memory and attentional skills [1]. The basic mechanisms of the test involve the presentation of continuous stimulis in terms of letters or pictures – for every stimulus presented, the participant has to indicate whether it matches a stimulus that was presented n stimuli ago [4]. There are different types of n-back tests known as loads: 3-back test, 2-back test and 1-back test [3].

2 Hypothesis

Our hypothesis was that participants would have a more challenging time remembering things initially which would be reflected in a longer reaction time to congruent stimulis in the 2-back test compared to the reaction time of a 1-back test. However, as n-back tests are shown to improve working and short term memory [5], we expect participants to get better at remembering, reflected in shorter reaction times in responding to congruent stimulis.

3 Materials/Methods

3.1 Information from Pelegrina et. Al (2015)

From [6]

3.1.1 R code

```
table4 <- read.csv("./dataFromPaper/csvfpsyg-06-01544.csv")
```

		nil	Age group						
		nil	7	8	9	10	11	12	
		nil	(n = 193)	(n = 285)	(n = 310)	(n = 297)	(n = 315)	(n = 253)	(
Total	M	nil	14.45	16.71	17.63	20.23	21.94	22.5	
	SD	nil	8.49	9.2	9.17	9.44	9.9	9.66	
	Percentile	5	3	3	4	4	4	4	
		25	7	10	10	13	14	14	
		50	13	15	16	21	24	24	
		75	22	25	25	28	30	30	
		95	28	32	32	35	35	37	
1-back	M	nil	8.05	8.82	9.14	9.72	10.01	10.2	
	SD	nil	3.04	3.09	2.82	2.96	2.98	2.96	
	Percentile	5	3	3	4	4	4	4	
		25	6	7	7	8	8	8	
		50	8	9	9	10	11	11	
		75	10	11	11	12	12	13	
		95	13	13	13	14	14	14	
2-back	M	nil	4.1	5.02	5.23	6.29	6.96	7.26	
	SD	nil	3.64	3.67	3.81	3.78	4.08	3.91	
	Percentile	5	–	–	–	–	–	–	
		25	–	2	2	3	4	5	
		50	4	5	5	6	7	8	
		75	7	8	8	9	10	11	
		95	11	11	12	12	13	13	
3-back	M	nil	2.3	2.87	3.26	4.23	4.97	5.04	
	SD	nil	3.41	3.81	3.91	4.15	4.27	4.12	
	Percentile	5	–	–	–	–	–	–	
		25	–	–	–	–	–	–	
		50	–	–	–	4	6	6	
		75	5	6	7	8	9	8	
		95	9	10	11	11	11	11	

3.2 Python Code For

3.3 Inline usage

4 Results

4.1 Table

4.2 Simple summary statistics

4.3 2 plots

5 Discussion

6 Bibliography

need to add the fpsyg-06-01544 citation!

References

[1] Roberto Colom, Francisco J. Román, Francisco J. Abad, Pei Chun Shih, Jesús Privado, Manuel Froufe, Sergio Escorial, Kenia Martínez, Miguel Burgaleta, M.A. Quiroga, Sherif Karama, Richard J. Haier, Paul M. Thompson, and Susanne M. Jaeggi. Adaptive n-back training does not improve fluid intelligence at the construct level: Gains on individual tests suggest that training may enhance visuospatial processing. *Intelligence*, 41(5):712 – 727, 2013.

[2] Carina Coulacoglou and Donald H. Saklofske. Chapter 5 - executive function, theory of mind, and adaptive behavior. In Carina Coulacoglou and Donald H. Saklofske, editors, *Psychometrics and Psychological Assessment*, pages 91 – 130. Academic Press, San Diego, 2017.

[3] Joan Forns Guzman, Mikel Esnaola, Mónica López-Vicente, Elisabet Suades González, Mar Alvarez-Pedrerol, Jordi Julvez, James Grellier, Nuria Sebastian Galles, and Jordi Sunyer. The n-back test and the attentional network task as measures of child neuropsychological development in epidemiological studies. *Neuropsychology*, 28, 05 2014.

[4] Michael Kane and Andrew Conway. The invention of n-back: An extremely brief history. *The Winnower*, 06 2016.

[5] Umberto León-Domínguez, Juan Francisco Martín-Rodríguez, and José León-Carrión. Executive n-back tasks for the neuropsychological assessment of working memory. *Behavioural Brain Research*, 292:167 – 173, 2015.

[6] Santiago Pelegrina, M. Teresa Lechuga, Juan A. Garcia-Madruga, M. Rosa Elosua, Pedro Macizo, Manuel Careiras, Luis J Fuentes, and M Teresa Bajo. Normal data on the n-back task for children and young adolescents. *Frontiers in Psychology*, 10 2015.

7 Appendix

7.1 Python Code for n-back test

```
from psychopy import visual, event, core
import pandas as pd
import random
import time as systime

#########
# setup #
#############################

#############
# Make lists / define functions #
#############


def makeMatches(in_list, trials=5,
                threshold=0, n_back=2,
                keep_list_stats=True, verbose=False):
    '''Creates the matches in a given list.if a random number is greater than threshold,
    then match the letters at positions [idx] and [idx-n_back]
    in_list: list of letters, strings, etc
    trials: how many trials to run
    threshold: type(float) in range(0,1)ld
    keep_stats: Bool: will output a list with information on
    the matches (position, character) and their frequency
    verbose: Bool: prints information about the lists for immediate viewing
    '''

    # done this way to avoid changing original list, confirm necessity?
    out_list = [i for i in in_list]
    list_stats = []  # list holding the character and positions it was matched at
    num_matches = 0
    for idx, char in enumerate(in_list):
        if idx > 1:
            if (random.random() > threshold):
                out_list[idx] = in_list[idx-n_back]
                list_stats.append([(idx, idx-2), char]
                                    ) if keep_list_stats else None
                    num_matches += 1

                    real_match_rate = num_matches / (len(in_list) - 2)
                    # show _stats or not
                    if verbose:  # switch this out of a print statement for final thing so it do
                        print(
                            f"{num_matches} of {len(in_list)-2} possible matches: {real_match_ra
                        print(f"in_list\n", in_list, "\nmatched list\n", out_list)
                    else:
                        pass

                    if keep_list_stats:
                        list_stats.insert(0, [(num_matches), "number of matches"])
                        list_stats.insert(0, [(real_match_rate), "actual match rate"])
            return(out_list, list_stats)
        else:
            return(out_list)


#####################
# create trial list #
#####################

n_trials = 15
# need to think of this inverted with how the code is currently written
match_frequency_threshold = 0.5
alphabet = [i for i in "ABCDEFGHIJKLMNOPQRSTUVWXYZ"]
initial_letters = [random.choice(alphabet) for i in range(n_trials)]

trial_list = makeMatches(initial_letters, trials=n_trials,
                         threshold=match_frequency_threshold, keep_list_stats=False)
ptt = 1.2
# ptt is the amount of time between trials, stands for "per time trial"

######################
# Window setup below #
######################
mywin = visual.Window(fullscr=True, screen=0, allowGUI=False, allowStencil=False,
                      monitor='testMonitor', color=[0, 0, 0], colorSpace='rgb')

clock = core.Clock()  # this is a clock

press_times = []  # List records the data
```

```
##############################

intro = True

if intro:
    # TODO  Find out how to display the last sentence in text_string
    text_string = f"This is an N-Back task.  This task is a test of working memory.  You wil
    textList = text_string.split("  ")
    for msg in textList:
        displayMsg = visual.TextStim(
            mywin, text=msg, pos=(0.5, 0))
        mywin.flip()
        displayMsg.draw()
        core.wait(3.5)

    countdownMessage = visual.TextStim(
        mywin, text='The task will begin after this countdown.', pos=(0.5, 0))
    countdownMessage.autoDraw = True
    mywin.flip()
    core.wait(3.5)
    countdownMessage.text = ' '
    mywin.flip()
    core.wait(0.5)



countdownString = "5,4,3,2,1"
countdown = countdownString.split(',')
# ct is the countdown timer

for num in countdown:
    txtDisplay = visual.TextStim(
        mywin, text = num , alignHoriz='left', alignVert='center', pos=(0, 0))
    mywin.flip()
    txtDisplay.draw()
    core.wait(1.0)


###################
# display letters #
###################

trialTime = core.Clock()

for idx, char in enumerate(trial_list):
    trialLength = core.CountdownTimer()
    keys = event.getKeys(keyList=["space"], timeStamped = trialLength)
    txtDisplay.text = char
    mywin.flip()
    txtDisplay.draw()
    print(keys, trialLength.getTime(), txtDisplay.text)
    press_times.append([keys, trialLength.getTime(), txtDisplay.text])
    core.wait(ptt)
    txtDisplay.text = "+"
    mywin.flip()
    txtDisplay.draw()
    core.wait(ptt)
    trialLength.reset()
    # currently appending in tuple form list_stats = []  # list holding the character and po

endMessage = visual.TextStim(
    mywin, text = ' ', pos=(0.5, 0))
endMessage.autoDraw=True
mywin.flip()
core.wait(1.5)
endMessage.text = 'You have completed the N-Back task. Thank you!'
mywin.flip()
core.wait(3.0)

print(press_times)

ts = systime.localtime()
timestamp = str(systime.strftime("Y%yM%mD%dH%HM%MS%S",ts))
datafile = open(f"datafile_{timestamp}.txt", "w+")

################
# writing file #
################
for line in press_times:
    datafile.write(str(line))
    datafile.write("\n")
    datafile.close()

# #not sure needed
# for line in n_list:
#     datafile.write(line,)
#     datafile.write("\n")

# for line in stats:
#     datafile.write(line)
#     datafile.write("\n")
```

7.2 Data from Our Python Code