Learning and Memory

My data table:

:

	Blocknumber	Trial number	Type of trial	Score	Match	Miss	False Alarm	Reaction Time	Memory	Current letter	nback1	nback2
0	1	2	0	1	0	0	0	3000	2	10	10	5
1	1	3	1	1	1	0	0	507	1	5	5	10
2	1	4	0	1	0	0	0	3000	3	6	6	5
3	1	5	0	1	0	0	0	3000	2	11	11	6
4	1	6	0	1	0	0	0	3000	2	12	12	11
...	...	...	...	...	...	...	...	...	...	...	...	...
69	3	21	0	0	0	0	1	1098	2	12	12	14
70	3	22	1	1	1	0	0	483	1	14	14	12
71	3	23	0	0	0	0	1	874	3	11	11	14
72	3	24	0	0	0	0	1	776	3	15	15	11
73	3	25	0	0	0	0	1	864	2	7	7	15

74 rows × 12 columns

Q1. 2-back-task measure this process. This is used for only short-term memory systems only as it manipulates information. This process presents a sequence of numbers, alphabets, etc. For asking the participants whether the current numbers alphabet matches the alphabets numbers that occurred two trials ago.

Working of 2-back-task

1. At the start, it will present a set of sequence numbers or alphabets. We must press some key for these characters that appeared two trials ago.

2. Participants will be required to retain much memory in this process.

3. Participants must immediately press the button when the same number or alphabet appears.

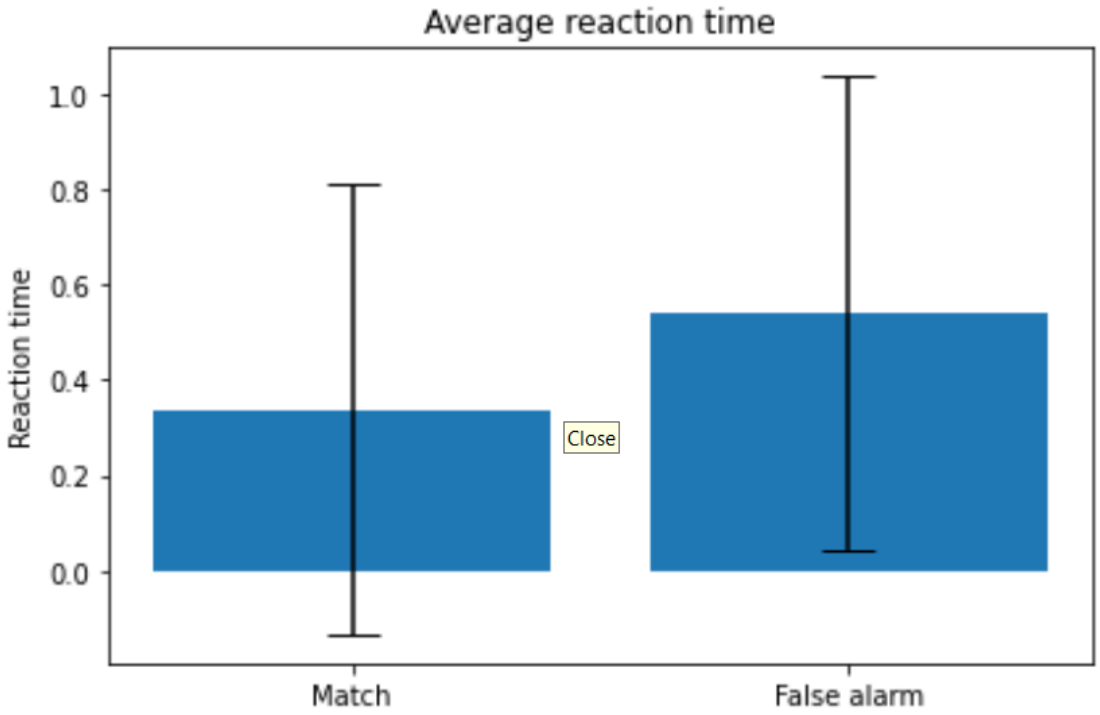
4. This task continues for 50-75 trials and scores the participant's accuracy and response time.

This task is more demanding than the 1-back-task memory as it requires participants to maintain a large amount of memory as the sequence of numbers, alphabets, etc., that occurred two trials ago.

Q2.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("nback2.data.2023-02-22--08-
```

```
00.txt",delim_whitespace=True)
data.columns = ['Blocknumber','Trial number','Type of
trial','Score','Match','Miss','False Alarm','Reaction
Time','Memory','Current letter','nback1','nback2']

average_Reaction_time = sum(data['Reaction Time']) / len(data['Reaction
Time'])
matchTrails_mean = np.mean(data['Match'])
falseAlarm_mean = np.mean(data['False Alarm'])

match_trails_std = np.std(data['Match'])
false_alarm_std = np.std(data['False Alarm'])

fig, ax = plt.subplots()
ax.bar(['Match', 'False alarm'], [matchTrails_mean, falseAlarm_mean],
yerr=[match_trails_std, false_alarm_std], capsize=10)
ax.set_ylabel('Reaction time')
ax.set_title('Average reaction time')
plt.show()
```



Q3.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statistics as stat

data = pd.read_csv("nback2.data.2023-02-22--08-
00.txt",delim_whitespace=True)
data.columns = ['Blocknumber','Trial number','Type of
trial','Score','Match','Miss','False Alarm','Reaction
```

```
Time','Memory','Current letter','nback1','nback2']

typeTrainMean = stat.mean(data['Type of trial'])
matchMean = stat.mean(data['Match'])
missMean = stat.mean(data['Miss'])
falseAlarmMean = stat.mean(data['False Alarm'])

meanOfAllTrails = (typeTrainMean + matchMean + missMean + falseAlarmMean)/4

meanReactionTime = stat.mean(data['Reaction Time'])

#Less than M
m1 = data[data['Reaction Time'] < meanReactionTime]

#Greater or equal to m
m2 = data[data['Reaction Time'] >= meanReactionTime]

# for data m1
fal = m1[m1['False Alarm'] ==1]
missedFalseAlarm = fal['False Alarm']
totalMissedFalse = sum(missedFalseAlarm)
totalMissedFalse

miss = m1[m1['Miss'] ==1]
missPar = miss['Miss']
totalMissPar = sum(missPar)
totalMissPar

matchMiss = m1[m1['Match'] ==0]
totalMatchMissed =0;
for i in matchMiss['Match']:
    if i==0:
        totalMatchMissed+=1
totalMatchMissed

totalTrails = len(m1['Trial number'])
totalTrails

percentOFMissFalseAlarm = (totalMissedFalse/totalTrails)*100

print("Percentage of Missed False Alarm to the totalTrails is {}
percent".format(percentOFMissFalseAlarm))
```

```
Percentage of Missed False Alarm to the totalTrails is 61.66666666666667
percent
```

```


percentOFMatchMissed = (totalMatchMissed/totalTrails)*100

print("Percentage of Match Missed to the total trails is {}
percent".format(percentOFMatchMissed ))
```

```
Percentage of Match Missed to the total trails is 61.66666666666667 percent

percentageofMiss = (totalMissPar/totalTrails)*100

print("Percentage of Missed False Alarm to the totalTrails is {}
percent".format(percentageofMiss))
```

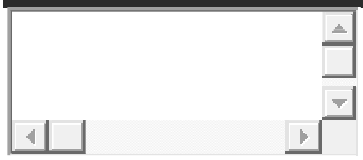
```


# for m2
```

```
fal = m2[m2['False Alarm'] ==1]
missedFalseAlarm = fal['False Alarm']
totalMissedFalse = sum(missedFalseAlarm)
totalMissedFalse

miss = m2[m2['Miss'] ==1]
missPar = miss['Miss']
totalMissPar = sum(missPar)
totalMissPar

matchMiss = m2[m2['Match'] ==0]
totalMatchMissed =0;
for i in matchMiss['Match']:
    if i==0:
        totalMatchMissed+=1
totalMatchMissed

totalTrails = len(m2['Trial number'])
totalTrails

percentOFMissFalseAlarm = (totalMissedFalse/totalTrails)*100

print("Percentage of Missed False Alarm to the totalTrails is {}
percent".format(percentOFMissFalseAlarm))
```

```
Percentage of Missed False Alarm to the totalTrails is 21.428571428571427
percent
```

```


percentOFMatchMissed = (totalMatchMissed/totalTrails)*100

print("Percentage of Match Missed to the total trails is {}
percent".format(percentOFMatchMissed ))
```

```
Percentage of Match Missed to the total trails is 85.71428571428571 percent
```



```
percentageofMiss = (totalMissPar/totalTrails)*100

print("Percentage of Missed False Alarm to the totalTrails is {}
percent".format(percentageofMiss))
Percentage of Missed False Alarm to the totalTrails is 7.142857142857142
percent
```

M1 table:

Out[35]:

	Blocknumber	Trial number	Type of trial	Score	Match	Miss	False Alarm	Reaction Time	Memory	Current letter	nback1	nback2
1	1	3	1	1	1	0	0	507	1	5	5	10
6	1	8	0	0	0	0	1	607	2	11	11	9
7	1	9	1	1	1	0	0	644	1	9	9	11
10	1	12	1	1	1	0	0	504	1	13	13	11
11	1	13	0	0	0	0	1	993	2	10	10	13
12	1	14	1	1	1	0	0	391	1	13	13	10
13	1	15	1	1	1	0	0	471	1	10	10	13
14	1	16	0	0	0	0	1	466	3	9	9	10
15	1	17	0	0	0	0	1	459	3	6	6	9
16	1	18	0	0	0	0	1	466	2	5	5	6
17	1	19	1	1	1	0	0	476	1	6	6	5
18	1	20	0	0	0	0	1	496	3	9	9	6
19	1	21	0	0	0	0	1	527	2	12	12	9
20	1	22	0	0	0	0	1	469	2	8	8	12
21	1	23	1	1	1	0	0	906	1	12	12	8
22	1	24	0	0	0	0	1	570	2	6	6	12
23	1	25	1	1	1	0	0	499	1	12	12	6
25	2	2	0	0	0	0	1	381	1	7	7	2
26	2	3	0	0	0	0	1	330	2	12	12	7
28	2	5	0	0	0	0	1	300	3	13	13	7
29	2	6	0	0	0	0	1	376	3	1	1	13

M2 table:

:

	Blocknumber	Trial number	Type of trial	Score	Match	Miss	False Alarm	Reaction Time	Memory	Current letter	nback1	nback2
0	1	2	0	1	0	0	0	3000	2	10	10	5
2	1	4	0	1	0	0	0	3000	3	6	6	5
3	1	5	0	1	0	0	0	3000	2	11	11	6
4	1	6	0	1	0	0	0	3000	2	12	12	11
5	1	7	0	1	0	0	0	3000	3	9	9	12
8	1	10	0	1	0	0	0	3000	3	13	13	9
9	1	11	0	1	0	0	0	3000	2	11	11	13
24	2	1	0	1	0	0	0	3000	3	2	2	12
27	2	4	1	0	0	1	0	3000	1	7	7	12
41	2	18	0	0	0	0	1	1051	3	8	8	2
44	2	21	1	1	1	0	0	1169	1	9	9	8
45	2	22	0	0	0	0	1	1144	2	14	14	9
52	3	4	1	1	1	0	0	1133	1	7	7	10
69	3	21	0	0	0	0	1	1098	2	12	12	14

The conclusion we get between response accuracy and reaction time is that participants must submit as fast as possible for a simple task. Here faster reaction will be directly proportional to reaction time.

But in the Stroop task, the faster reaction will be inversely proportional to reaction time.

So, for different task response accuracy and reaction time will differ.