Assignment 1: Learning and Memory PSY 306 (Winter 2023)

Name: Mudit Balooja

Roll Number: 2019258

Instructions: Please write your own responses and DO NOT copy or lift text/code from any source, including the attached paper. If you are referring to credible external sources other than the attached paper for your answers, please cite those sources (within the body of text and the provide a reference list at the end) in the APA citation format (https://www.mendeley.com/guides/apa-citation-guide). Word limits given are indicative and less than the indicated numbers may also be used.

Please download this MS word question-cum-response template to TYPE your answers and feel free to add sheets as required. Convert this document to a PDF and rename the file: name_roll no. before submitting. Please note that answers in this template only will be evaluated and hand-written or scanned answer sheets will not be evaluated. Verbatim copying of any extent and total percent similarity with other sources exceeding 20% will be deemed plagiarized and dealt as per IIITD policies.

[Strict deadline for submission: 22 Feb, 11 PM]

Q2) Please do the following for this question:

- **Register on PsyToolkit (https://www.psytoolkit.org/) & log in. References**
  Ref-1, Stoet, G. (2010). PsyToolkit - A software package for programming psychological experiments using Linux. Behavior Research Methods, 42(4), 1096-1104. Ref-2,Stoet, G. (2017). PsyToolkit: A novel web-based method for running online questionnaires and reaction-time experiments. Teaching of Psychology, 44(1), 24-31.
- **Click on ' Get from Library' on the left-hand side panel of the screen.**
- **On the central panel of the screen click on ' Official PsyToolkit experiment library' (https://www.psytoolkit.org/experiment-library/)**
- **Scroll down and click on 'N-back Task (2 back)'.**
- **Scroll down on this page (https://www.psytoolkit.org/experiment-library/nback2.html) & click under Download on 'The PsyToolkit code zip file'.**
- **Follow the instructions in this video to compile the experiment from the downloaded zip file in the previous step : https://www.youtube.com/watch?v=Vlf-UuLbi3Y&feature=youtu.be .**
- **Read the documentation of the experiment and the detailed instructions of running the experiment and the output data structure (https://www.psytoolkit.org/experiment-library/nback2.html).**
- **Run the experiment either 'in the browser' or download the compiled experiment offline by clicking on 'Download for Running Offline'**
- **PLEASE CARRY OUT ALL 75 TEST TRIALS given in three blocks of 25 trials each .**
- **At the end, a table of results will be displayed and the column headers of the results table are here** https://www.psytoolkit.org/experiment-library/nback2.html **under 'Data output file'.**
- **Download the results table (text file ('.txt')) and then answer the following…**

Insert a figure (wherever required) and paste the MATLAB/Python code for the same. The datasheet generated from the test trials may also be pasted on this sheet at appropriate places. All figures must be properly labelled and should have accompanying captions/legends to provide all information necessary to interpret the figures…

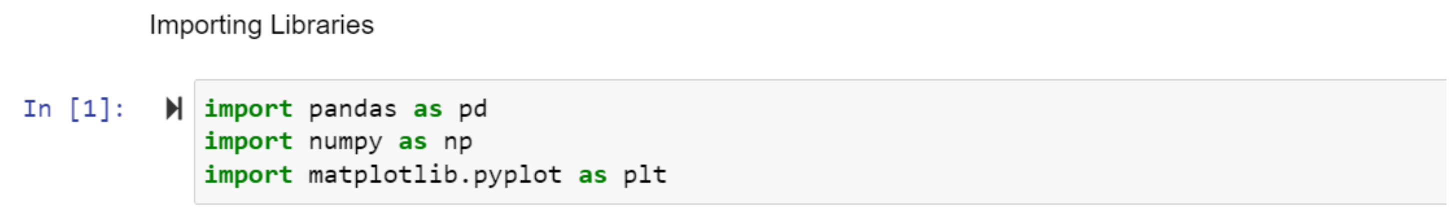
A) Which cognitive process does the test measure. Briefly explain how? [1+3]
The N-back task (2 back) measures our working memory and working memory capacity.

In this test, participants are presented with a series of letters (stimuli) and then the participant is asked to indicate if the current letter matches the one 2 trials back. This, therefore, requires us to remember the current as well as the last two letters in the working memory. We also need to update our working memory as letters are shown consecutively in groups of 25, and only a gap of approximately 2.5 seconds between any two letters. Our performance depends on working memory capacity (i.e., how many letters we can maintain in our memory at a time), speed (i.e., how fast letters are shown and then disappear), duration (i.e., how long the test actually takes) and N (i.e., how far back we need to maintain the letters in our working memory, 2 here). Finally, the data like number of matches, misses, reaction time, etc., is presented as a text file for further analysis of working memory and working memory capacity.

B) Plot two simple bar diagrams showing the average reaction times with their standard deviations (error bars) of the total 'match' trials and the 'false alarm' trials respectively. [3+3]

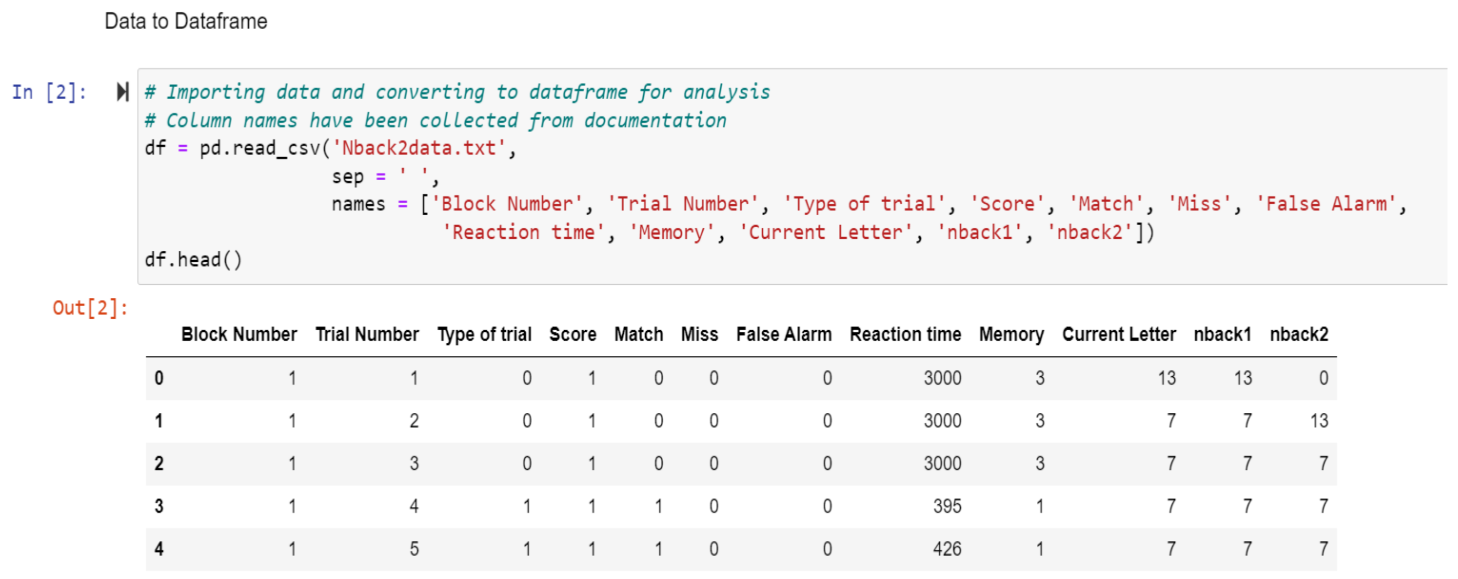
Steps taken to plot the bar diagrams of average reaction times with their standard deviations.
1. The data file from the experiment is imported and converted to a data frame for analysis.
2. From the data, reaction times of matches and false alarms (where value = 1) are extracted, and subsequently, their mean and standard deviations are calculated.
3. The above values are then printed and shown in a bar plot using matplotlib.

Below are the code snippets with appropriate comments (The code is written in Jupyter Notebook):



Importing Libraries

```
In [1]:  import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
```



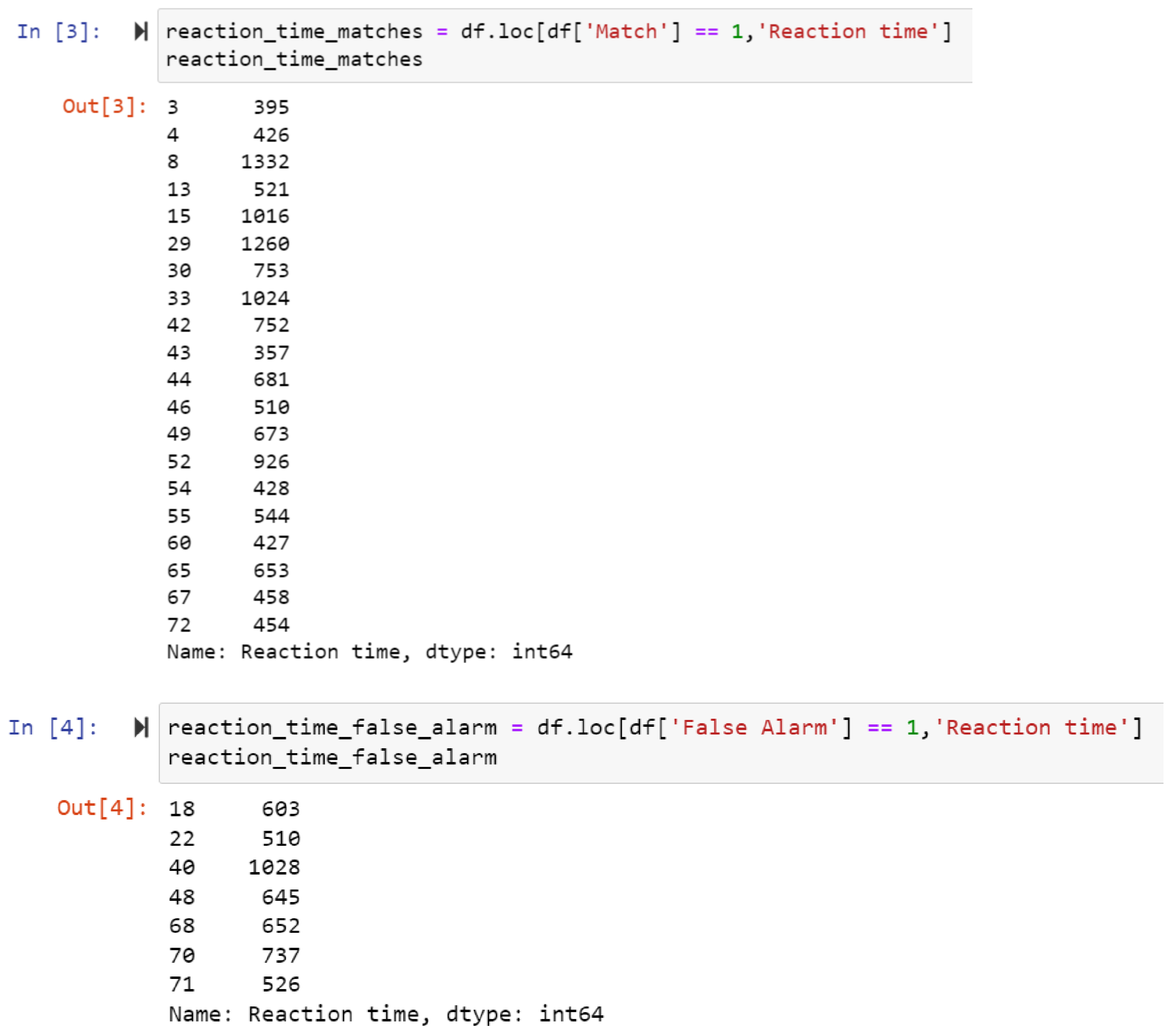
Data to Dataframe

```
In [2]:  # Importing data and converting to dataframe for analysis
         # Column names have been collected from documentation
         df = pd.read_csv('Nback2data.txt',
                          sep = ' ',
                          names = ['Block Number', 'Trial Number', 'Type of trial', 'Score', 'Match', 'Miss', 'False Alarm',
                                   'Reaction time', 'Memory', 'Current Letter', 'nback1', 'nback2'])
         df.head()
```

Out[2]:

	Block Number	Trial Number	Type of trial	Score	Match	Miss	False Alarm	Reaction time	Memory	Current Letter	nback1	nback2
0	1	1	0	1	0	0	0	3000	3	13	13	0
1	1	2	0	1	0	0	0	3000	3	7	7	13
2	1	3	0	1	0	0	0	3000	3	7	7	7
3	1	4	1	1	1	0	0	395	1	7	7	7
4	1	5	1	1	1	0	0	426	1	7	7	7

Extracting Reaction Times



```
In [3]:  reaction_time_matches = df.loc[df['Match'] == 1,'Reaction time']
         reaction_time_matches
```

```
Out[3]: 3      395
        4      426
        8     1332
        13     521
        15    1016
        29    1260
        30     753
        33    1024
        42     752
        43     357
        44     681
        46     510
        49     673
        52     926
        54     428
        55     544
        60     427
        65     653
        67     458
        72     454
        Name: Reaction time, dtype: int64
```

```
In [4]:  reaction_time_false_alarm = df.loc[df['False Alarm'] == 1,'Reaction time']
         reaction_time_false_alarm
```

```
Out[4]: 18     603
        22     510
        40    1028
        48     645
        68     652
        70     737
        71     526
        Name: Reaction time, dtype: int64
```

Plots

```
In [24]:  x_axis = ["Matches", "False Alarms"]
          y_axis = [reaction_time_matches.mean(), reaction_time_false_alarm.mean()]
          y_error = [np.std(reaction_time_matches), np.std(reaction_time_false_alarm)]

          print("Average Reaction time for matches: " + str(y_axis[0]) + "ms")
          print("Average Reaction time for false alarms: " + str(y_axis[1]) + "ms")
          print("Standard Deviation for matches: " + str(y_error[0]))
          print("Standard Deviation for false alarms: " + str(y_error[1]))
          print("\n")

          fig = plt.figure()
          plt.bar(x_axis, y_axis, yerr = y_error)
          plt.ylabel("Average Reaction Time (milliseconds)")
          plt.title("Average Reaction Time with Standard Deviation")
          caption = """Figure shows the average of the reaction times(blue bars)
                  and their standard deviations(black/error bars) for the match
              and false alarm trials across all trials of the experiment"""
          fig.text(0.5, -0.09, caption, ha = 'center')
          plt.show()
```

```
Average Reaction time for matches: 679.5ms
Average Reaction time for false alarms: 671.5714285714286ms
Standard Deviation for matches: 284.17943275332226
Standard Deviation for false alarms: 162.30910294237717
```

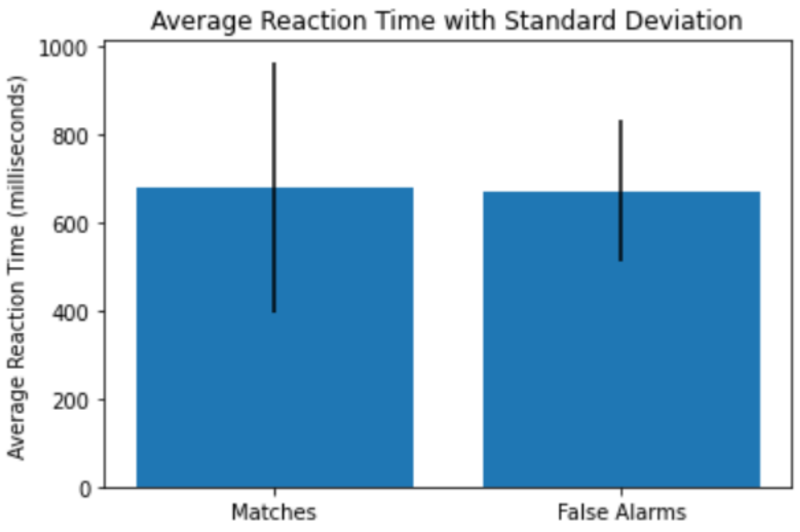


Figure shows the average of the reaction times(blue bars)
and their standard deviations(black/error bars) for the match
and false alarm trials across all trials of the experiment

C) Calculate the mean (M) reaction time of all trials. Split the original data into two parts such that all the trials of one part has reaction time less than M and the trials of the other part has reaction time equal to or greater than M. Next, calculate the total number of erroneous responses (i.e., incorrect responses, false alarms, misses, in match and non-match trials as relevant) and express that as a percentage of the total trial number for both parts of the split data. Report both percentages.

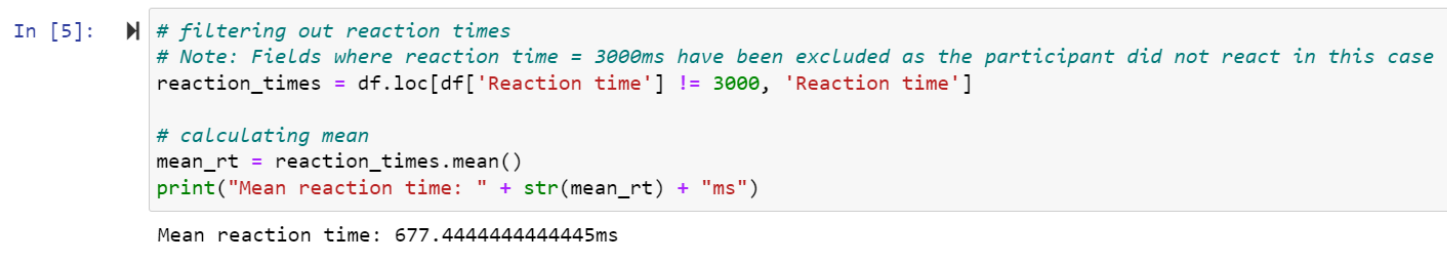
Based on a comparison of the above two percentages, what can be concluded about the relationship between response accuracy and reaction time in your experimental data? [9+1]

(Hint: report all steps; conclude based on your own data + analysis)

Code snippets and explanation

Step 1: Mean reaction time (M) is calculated as below.

Please note that where reaction time equals 3000 ms, we have excluded it from the calculation as, in this case, the person does not react.

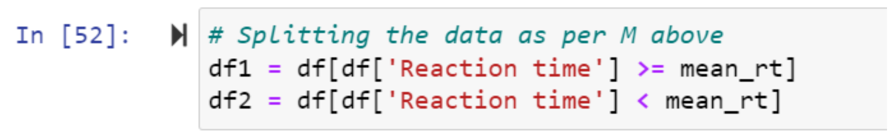


```
In [5]:  # filtering out reaction times
         # Note: Fields where reaction time = 3000ms have been excluded as the participant did not react in this case
         reaction_times = df.loc[df['Reaction time'] != 3000, 'Reaction time']

         # calculating mean
         mean_rt = reaction_times.mean()
         print("Mean reaction time: " + str(mean_rt) + "ms")

         Mean reaction time: 677.4444444444445ms
```

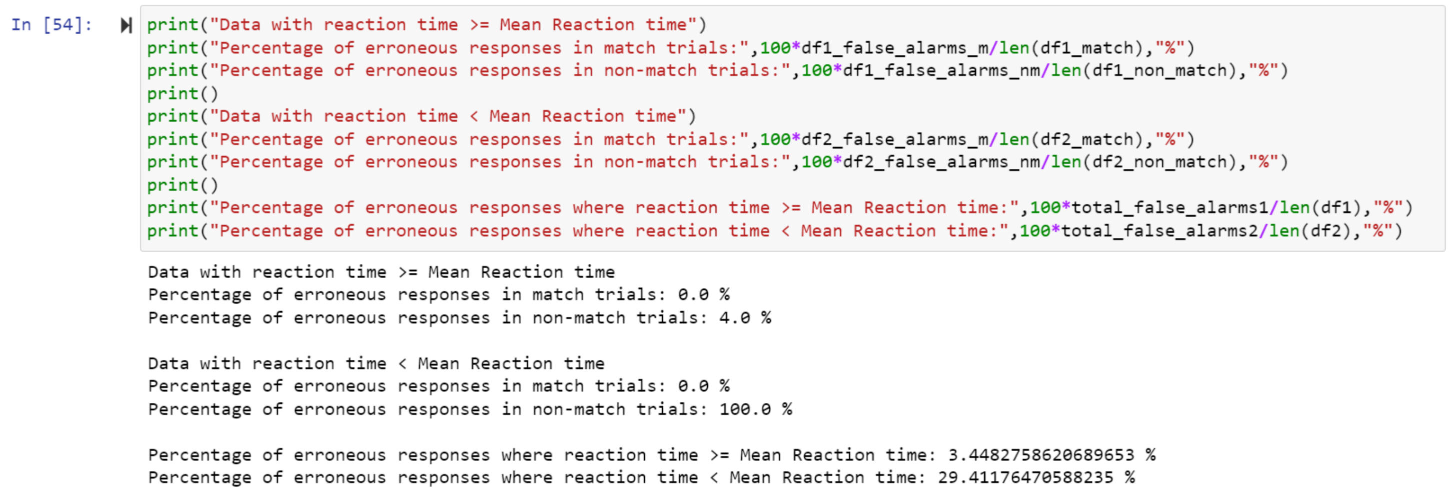
Step 2: Data is split into two parts as per M



```
In [52]:  # Splitting the data as per M above
          df1 = df[df['Reaction time'] >= mean_rt]
          df2 = df[df['Reaction time'] < mean_rt]
```

Step 3: The number of erroneous responses is calculated in match trials, and non-match trials in the above two data splits. Furthermore, they are expressed as %s both separately and combined.

Also, in the dataset (present at the end), we can clearly see that there are no misses, only false alarms so only their count is considered in the calculation of erroneous responses.



```
In [54]:  print("Data with reaction time >= Mean Reaction time")
          print("Percentage of erroneous responses in match trials:",100*df1_false_alarms_m/len(df1_match),"%")
          print("Percentage of erroneous responses in non-match trials:",100*df1_false_alarms_nm/len(df1_non_match),"%")
          print()
          print("Data with reaction time < Mean Reaction time")
          print("Percentage of erroneous responses in match trials:",100*df2_false_alarms_m/len(df2_match),"%")
          print("Percentage of erroneous responses in non-match trials:",100*df2_false_alarms_nm/len(df2_non_match),"%")
          print()
          print("Percentage of erroneous responses where reaction time >= Mean Reaction time:",100*total_false_alarms1/len(df1),"%")
          print("Percentage of erroneous responses where reaction time < Mean Reaction time:",100*total_false_alarms2/len(df2),"%")

          Data with reaction time >= Mean Reaction time
          Percentage of erroneous responses in match trials: 0.0 %
          Percentage of erroneous responses in non-match trials: 4.0 %

          Data with reaction time < Mean Reaction time
          Percentage of erroneous responses in match trials: 0.0 %
          Percentage of erroneous responses in non-match trials: 100.0 %

          Percentage of erroneous responses where reaction time >= Mean Reaction time: 3.4482758620689653 %
          Percentage of erroneous responses where reaction time < Mean Reaction time: 29.41176470588235 %
```

Comparing the two percentages, we observe that the percentage of erroneous responses where the reaction time is less than the mean reaction time is greater than the percentage of erroneous responses where the reaction time is more than the mean reaction time.

Now we calculate the percentage of correct responses (Accuracy)

Percentage of correct responses where reaction time >= Mean reaction time = 100 - 3.44 (shown above)

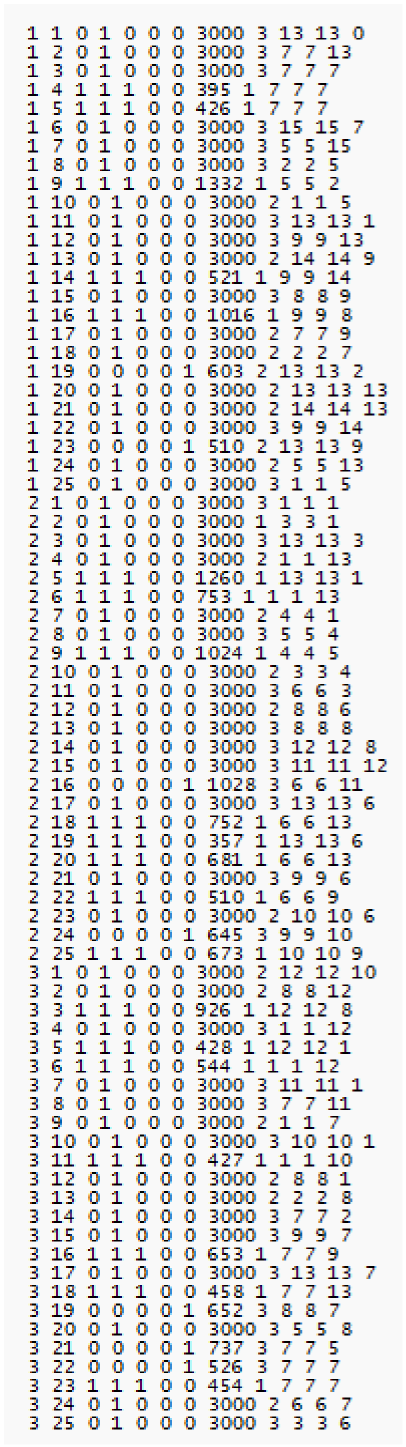
= 96.56 %

Percentage of correct responses where reaction time < Mean reaction time = 100 - 29.41 (shown above)

= 70.59 %

From this we conclude that greater the reaction time, greater the percentage of correct responses or accuracy. So, reaction time and accuracy are directly proportional.

<u>Dataset</u>



```
1 1 0 1 0 0 0 3000 3 13 13 0
1 2 0 1 0 0 0 3000 3 7 7 13
1 3 0 1 0 0 0 3000 3 7 7 7
1 4 1 1 1 0 0 395 1 7 7 7
1 5 1 1 1 0 0 426 1 7 7 7
1 6 0 1 0 0 0 3000 3 15 15 7
1 7 0 1 0 0 0 3000 3 5 5 15
1 8 0 1 0 0 0 3000 3 2 2 5
1 9 1 1 1 0 0 1332 1 5 5 2
1 10 0 1 0 0 0 3000 2 1 1 5
1 11 0 1 0 0 0 3000 3 13 13 1
1 12 0 1 0 0 0 3000 3 9 9 13
1 13 0 1 0 0 0 3000 2 14 14 9
1 14 1 1 1 0 0 521 1 9 9 14
1 15 0 1 0 0 0 3000 3 8 8 9
1 16 1 1 1 0 0 1016 1 9 9 8
1 17 0 1 0 0 0 3000 2 7 7 9
1 18 0 1 0 0 0 3000 2 2 2 7
1 19 0 0 0 0 1 603 2 13 13 2
1 20 0 1 0 0 0 3000 2 13 13 13
1 21 0 1 0 0 0 3000 2 14 14 13
1 22 0 1 0 0 0 3000 3 9 9 14
1 23 0 0 0 0 1 510 2 13 13 9
1 24 0 1 0 0 0 3000 2 5 5 13
1 25 0 1 0 0 0 3000 3 1 1 5
2 1 0 1 0 0 0 3000 3 1 1 1
2 2 0 1 0 0 0 3000 1 3 3 1
2 3 0 1 0 0 0 3000 3 13 13 3
2 4 0 1 0 0 0 3000 2 1 1 13
2 5 1 1 1 0 0 1260 1 13 13 1
2 6 1 1 1 0 0 753 1 1 1 13
2 7 0 1 0 0 0 3000 2 4 4 1
2 8 0 1 0 0 0 3000 3 5 5 4
2 9 1 1 1 0 0 1024 1 4 4 5
2 10 0 1 0 0 0 3000 2 3 3 4
2 11 0 1 0 0 0 3000 3 6 6 3
2 12 0 1 0 0 0 3000 2 8 8 6
2 13 0 1 0 0 0 3000 3 8 8 8
2 14 0 1 0 0 0 3000 3 12 12 8
2 15 0 1 0 0 0 3000 3 11 11 12
2 16 0 0 0 0 1 1028 3 6 6 11
2 17 0 1 0 0 0 3000 3 13 13 6
2 18 1 1 1 0 0 752 1 6 6 13
2 19 1 1 1 0 0 357 1 13 13 6
2 20 1 1 1 0 0 681 1 6 6 13
2 21 0 1 0 0 0 3000 3 9 9 6
2 22 1 1 1 0 0 510 1 6 6 9
2 23 0 1 0 0 0 3000 2 10 10 6
2 24 0 0 0 0 1 645 3 9 9 10
2 25 1 1 1 0 0 673 1 10 10 9
3 1 0 1 0 0 0 3000 2 12 12 10
3 2 0 1 0 0 0 3000 2 8 8 12
3 3 1 1 1 0 0 926 1 12 12 8
3 4 0 1 0 0 0 3000 3 1 1 12
3 5 1 1 1 0 0 428 1 12 12 1
3 6 1 1 1 0 0 544 1 1 1 12
3 7 0 1 0 0 0 3000 3 11 11 1
3 8 0 1 0 0 0 3000 3 7 7 11
3 9 0 1 0 0 0 3000 2 1 1 7
3 10 0 1 0 0 0 3000 3 10 10 1
3 11 1 1 1 0 0 427 1 1 1 10
3 12 0 1 0 0 0 3000 2 8 8 1
3 13 0 1 0 0 0 3000 2 2 2 8
3 14 0 1 0 0 0 3000 3 7 7 2
3 15 0 1 0 0 0 3000 3 9 9 7
3 16 1 1 1 0 0 653 1 7 7 9
3 17 0 1 0 0 0 3000 3 13 13 7
3 18 1 1 1 0 0 458 1 7 7 13
3 19 0 0 0 0 1 652 3 8 8 7
3 20 0 1 0 0 0 3000 3 5 5 8
3 21 0 0 0 0 1 737 3 7 7 5
3 22 0 0 0 0 1 526 3 7 7 7
3 23 1 1 1 0 0 454 1 7 7 7
3 24 0 1 0 0 0 3000 2 6 6 7
3 25 0 1 0 0 0 3000 3 3 3 6
```

Complete Code (both parts)

```
#!/usr/bin/env python

# coding: utf-8


# ### Part B


# Importing libraries


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt



# Data to Dataframe


# Importing data and converting to dataframe for analysis

# Column names have been collected from documentation

df = pd.read_csv('Nback2data.txt',

                 sep = ' ',

                 names = ['Block Number', 'Trial Number', 'Type of trial', 'Score',
'Match', 'Miss', 'False Alarm',

                           'Reaction time', 'Memory', 'Current Letter', 'nback1',
'nback2'])

df.head()



# Extracting Reaction Times


reaction_time_matches = df.loc[df['Match'] == 1,'Reaction time']

reaction_time_matches
```

```
reaction_time_false_alarm = df.loc[df['False Alarm'] == 1,'Reaction time']
reaction_time_false_alarm


# Plots

x_axis = ["Matches", "False Alarms"]
y_axis = [reaction_time_matches.mean(), reaction_time_false_alarm.mean()]
y_error = [np.std(reaction_time_matches), np.std(reaction_time_false_alarm)]

print("Average Reaction time for matches: " + str(y_axis[0]) + "ms")
print("Average Reaction time for false alarms: " + str(y_axis[1]) + "ms")
print("Standard Deviation for matches: " + str(y_error[0]))
print("Standard Deviation for false alarms: " + str(y_error[1]))
print("\n")

fig = plt.figure()
plt.bar(x_axis, y_axis, yerr = y_error)
plt.ylabel("Average Reaction Time (milliseconds)")
plt.title("Average Reaction Time with Standard Deviation")
caption = """Figure shows the average of the reaction times(blue bars)
        and their standard deviations(black/error bars) for the match
    and false alarm trials across all trials of the experiment"""
fig.text(0.5, -0.09, caption, ha = 'center')
plt.show()
```

```
# ### Part C


# filtering out reaction times

# Note: Fields where reaction time = 3000ms have been excluded as the participant did
not react in this case

reaction_times = df.loc[df['Reaction time'] != 3000, 'Reaction time']


# calculating mean

mean_rt = reaction_times.mean()

print("Mean reaction time: " + str(mean_rt) + "ms")



# Splitting the data as per M above

df1 = df[df['Reaction time'] >= mean_rt]

df2 = df[df['Reaction time'] < mean_rt]



# split into match and non-match

df1_match = df1.loc[df1['Match'] == 1]

df1_non_match = df1.loc[df1['Match'] == 0]


df2_match = df2.loc[df2['Match'] == 1]

df2_non_match = df2.loc[df2['Match'] == 0]


# calculating total number of erroneous responses (false alarm columns)

# number of false alarms in df1

df1_false_alarms_m = df1_match['False Alarm'].sum()

df1_false_alarms_nm = df1_non_match['False Alarm'].sum()

total_false_alarms1 = df1_false_alarms_m + df1_false_alarms_nm
```

```
# number of false alarms in df2

df2_false_alarms_m = df2_match['False Alarm'].sum()

df2_false_alarms_nm = df2_non_match['False Alarm'].sum()

total_false_alarms2 = df2_false_alarms_m + df2_false_alarms_nm





print("Data with reaction time >= Mean Reaction time")

print("Percentage of erroneous responses in match
trials:",100*df1_false_alarms_m/len(df1_match),"%")

print("Percentage of erroneous responses in non-match
trials:",100*df1_false_alarms_nm/len(df1_non_match),"%")

print()

print("Data with reaction time < Mean Reaction time")

print("Percentage of erroneous responses in match
trials:",100*df2_false_alarms_m/len(df2_match),"%")

print("Percentage of erroneous responses in non-match
trials:",100*df2_false_alarms_nm/len(df2_non_match),"%")

print()

print("Percentage of erroneous responses where reaction time >= Mean Reaction
time:",100*total_false_alarms1/len(df1),"%")

print("Percentage of erroneous responses where reaction time < Mean Reaction
time:",100*total_false_alarms2/len(df2),"%")
```