Assignment 2: Neuroscience of Decision Making PSY 3/507 (Monsoon 2024)

Name: Anjaneya Sharma

Roll Number: 2021449

---

1. **Fill the following form https://forms.gle/68zPBahEw9JxakyN8**

2(A)

A researcher conducted a random dot motion discrimination task with 2 different motion coherence levels as different conditions. 30 participants performed 100 trials in each condition and the evidence accumulation was recorded. Column 1 of cell array = Condition 1 and each cell of Column 1 has one participant's data. Each cell has a 100 x 1000 matrix. Each row of the matrix = one trial for 1000 ms. The evidence accumulation starts from 300 and reaches the decision threshold at 600.The same convention applies to data from Column 2.

Now solve the following. Insert a figure (wherever required) and paste the MATLAB/Python/R code for the same. Any figure must provide all information necessary to interpret it including axes labels, captions/legends (simple figure titles as captions are not enough).

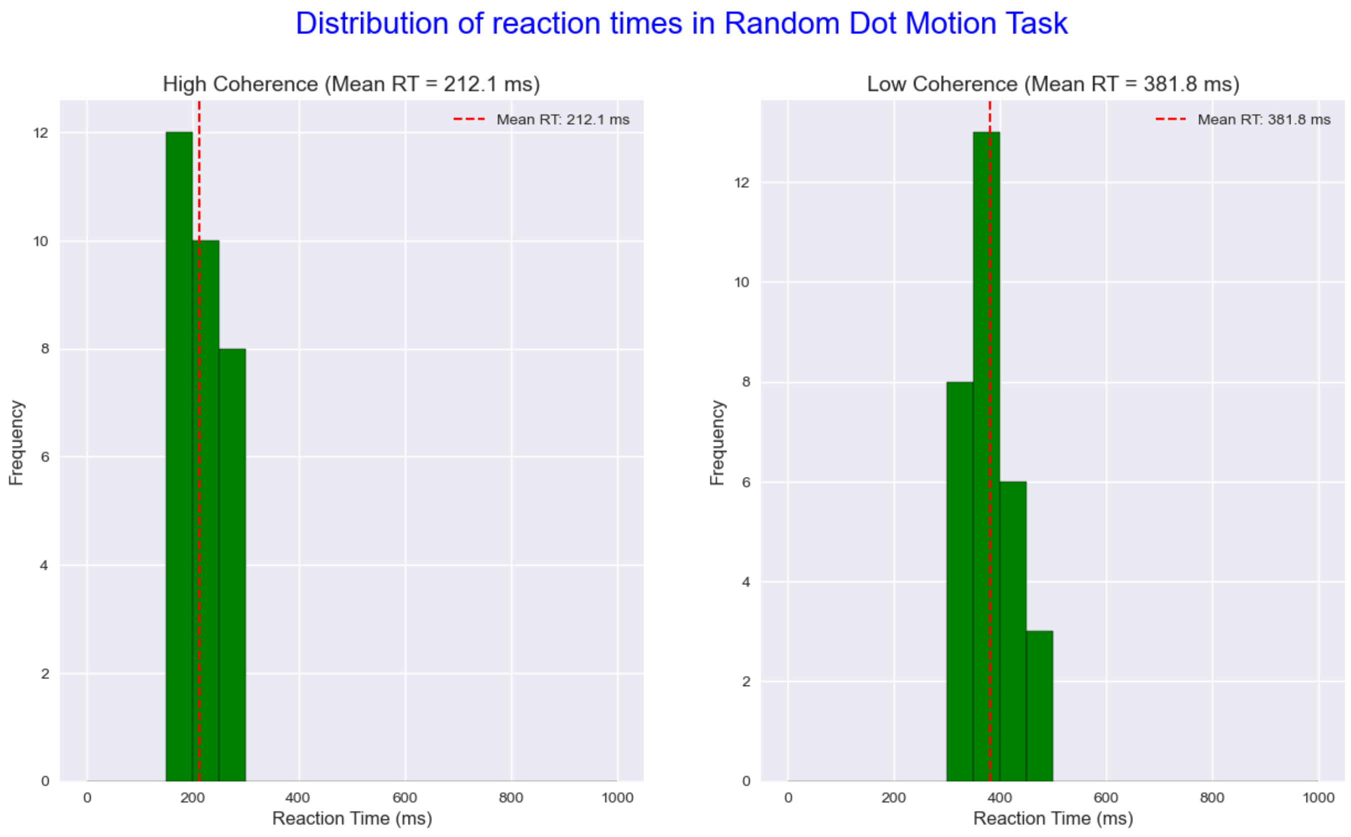
From the data, calculate reaction time (RT) for all 100 trials of each participant. Divide the time axis in 20 bins (each bin of 50 ms) and calculate the mean RT for all 30 participants for both conditions. Plot one histogram of the mean RT distribution for each condition separately (total = 2 histograms). Mark the mean of the distribution with a red line and report on the title. Conduct an appropriate statistical test to compare the mean RT (across participants). Report the results with test statistics and p values. [5 points + 1 point + 2 points]

Interpret the findings with respect to motion coherence. [2 points]

(Hint: If the data in each of the two groups follow a normal distribution, use a parametric statistical test for testing the difference of two independent group means. Otherwise, use a suitable non-parametric counterpart of the parametric test.)

Note to the reader/checker : The standard deviation calculated for the histograms is approximately 15% of the mean, which ssuggests that the mean RT values are not perfect precise but still they provide reasonable comparison between the two groups.

Reaction Times Distribution:



<u>Caption and conclusion</u> based on the Curve and the Results and statistics given below:

The histograms display mean reaction times (RT) for 30 participants in a random dot motion discrimination task under two conditions: High Coherence and Low Coherence.

Each histogram shows the distribution of mean RTs in 50 ms bins, with the red dashed line marking the average RT for each condition.

The mean RT in the High Coherence condition is 212.1 ms, while in the Low Coherence condition, it is 381.8 ms. These findings indicate that higher coherence levels lead to faster decision-making.

<u>Results and Statistics:</u>

```
Results and metrics :
Mann-Whitney U test test was used
Test statistic: 0.0
p-value: 3.012*e-11 (Rounded of to nearest 3 digits)

Condition 1: Mean RT = 212.1 ms (When the Coherence is High)
Condition 2: Mean RT = 381.8 ms (Coherence is low)
```

The Mann-Whitney U test reveals a statistically significant difference in reaction times between the two conditions, with a test statistic of 0.0 and a p-value of 3.012×10^{-11}, which is way below the 0.05 standard threshold.

This, combined with the observation that the standard deviation is approximately 15% of the mean, suggests that while the mean RTs provide quite significant difference between conditions, individual variations exist. Higher coherence here improves perceptual clarity which furthr facilitates quicker responses.

Answer:
```
#!/usr/bin/env python
# coding: utf-8

# In[82]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
from scipy.io import loadmat
import seaborn as sns
from scipy.integrate import trapz


# ***Part 2 A***

# In[ ]:


# ************************************************************************************************


# In[131]:


data = scipy.io.loadmat('Assignment2_2A_NDM_2024.mat')['NDM_Assignment2']
n_participants, n_conditions = data.shape ######## check

#setting up a dictionary for reaction times in the beginning
rtimeDict = {0: [], 1: []}


# In[132]:
```

```
# ************************************************************************************************


# In[133]:


for condition in range(2):
    for participant in range(n_participants):
        participant_data = data[participant, condition]
        trial_rts = []

        for trial in participant_data:
            threshold_indices = np.where(trial >= 600)[0]
            if len(threshold_indices) > 300:
                rt = threshold_indices[0]  #first time threshold crssoing
            else:
                rt = np.nan  # If threshold is never reached, which doesn't happen here
                #added this just in case
            trial_rts.append(rt)

        rtimeDict[condition].append(trial_rts)

mean_rt_cond1 = np.nanmean(rtimeDict[0], axis=1)
mean_rt_cond2 = np.nanmean(rtimeDict[1], axis=1)


# In[134]:


#i have checked the normality of data and with the help of p values(using shapiro wilk test),
determining which test to perform
_, p_val_1 = stats.shapiro(mean_rt_cond1)
_, p_val_2 = stats.shapiro(mean_rt_cond2)


if p_val_1 > 0.05 and p_val_2 > 0.05:
    statistic, p_value = stats.ttest_ind(mean_rt_cond1, mean_rt_cond2, nan_policy='omit')
    test_name = "Independent t-test"
else:
    statistic, p_value = stats.mannwhitneyu(mean_rt_cond1, mean_rt_cond2)
    test_name = "Mann-Whitney U test"




# In[135]:


# *************************************************************************
# In[136]:


# Print statistical results
print("\nResults and metrics :")
print(f"{test_name} test was used")
print(f"Test statistic: {statistic}")
print(f"p-value: {p_value}")
print(f"\nCondition 1 (High Coherence): Mean RT = {np.nanmean(mean_rt_cond1):.1f} ms")
print(f"\nCondition 2 (Low Coherence): Mean RT = {np.nanmean(mean_rt_cond2):.1f} ms")


# In[137]:


# Plotting
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 8), facecolor='white')
bins = np.linspace(0, 1000, 21)
condition_names = ['High Coherence', 'Low Coherence']

# Basic plot for each condition
for condition, (ax, name) in enumerate(zip([ax1, ax2], condition_names)):
    participant_means = np.nanmean(rtimeDict[condition], axis=1)
    mean_rt = np.nanmean(participant_means)

    ax.hist(participant_means, bins=bins, color='green', edgecolor='black')
    ax.axvline(mean_rt, color='red', linestyle='--', linewidth=1.5, label=f'Mean RT: {mean_rt:.1f}
ms')

    ax.set_xlabel('Reaction Time (ms)', fontsize=12)
    ax.set_ylabel('Frequency', fontsize=12)
    ax.set_title(f'{name} (Mean RT = {mean_rt:.1f} ms)', fontsize=14)
    ax.legend(fontsize=10)

plt.suptitle('Distribution of reaction times in Random Dot Motion Task', fontsize=20, color='blue')


plt.show()


# In[138]:


# ******************************************************************************************


# In[139]:
```

```
mean_rt_cond1

# In[130]:

mean_rt_cond2
```

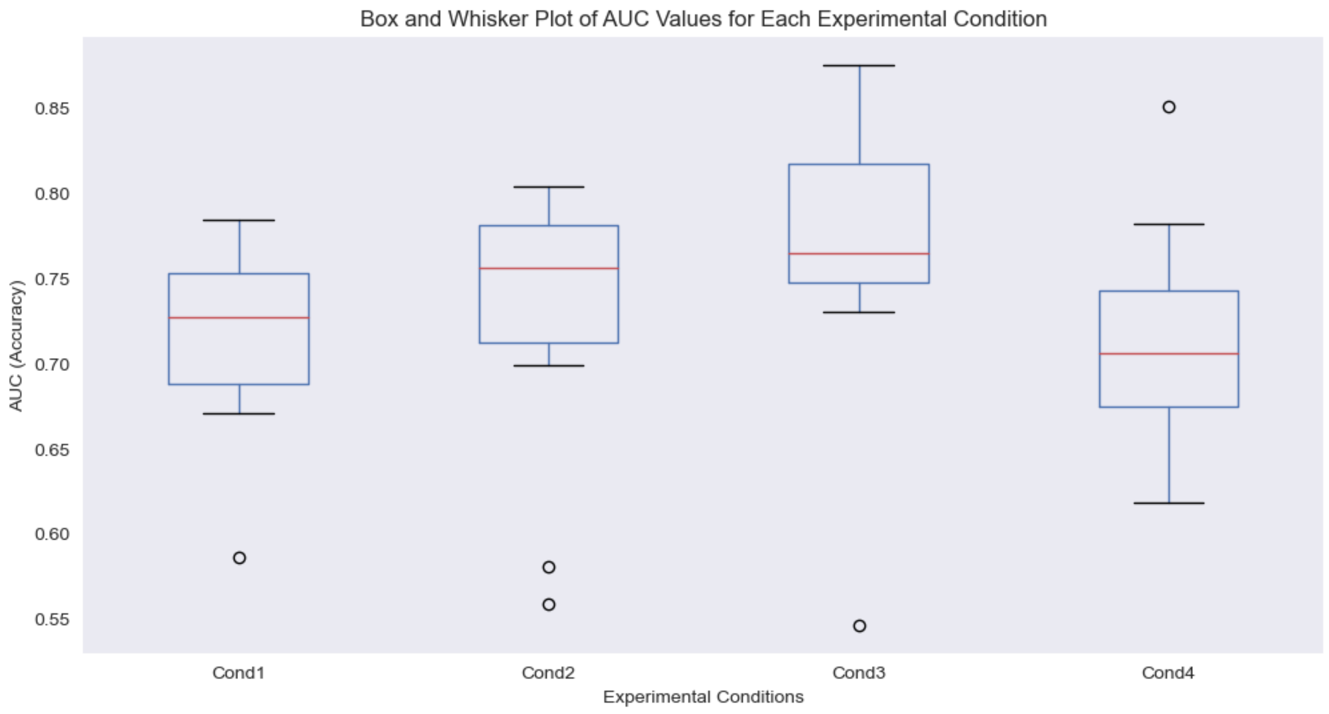
2(B)

Please use the attached data file (Assignment2-2B-NDM-2024.xlsx). Each page of the Excel file contains data of one behavioural experimental condition (Cond) for 10 participants (P1 to P10). The participants were shown noisy images of different conditions and instructed to recognize those images as targets or lures. There are a total of four experimental conditions (Cond 1 – Cond 4). On each sheet and under each participant, the data shows hit rate (HT) and false alarm rate (FA) for each criterion value of inner confidence in their decision (each row). Solve the following and insert a figure (wherever required) and paste the MATLAB/Python/R code for the same. Any figure must provide all information necessary to interpret it including axes labels, captions/legends (simple figure titles as captions are not enough).

Calculate the accuracies for recognizing noisy images for all participants in each of the four conditions from the above graph. Use the accuracies thus calculated to graph four box and whisker plots (for four experimental conditions). Describe all details in the figure caption. Run a 'Friedman's test' to statistically test for difference in accuracies across the four conditions and report the Friedman's Chi square test statistic and p value for the same. Briefly explain the result of your statistical analysis. [6 + 2 + 2 points]

Answer:

Box and Whisker Plots:



<u>Caption and conclusion</u> based on the Curve and the Results and statistics given below:

The figure above displays the distribution of AUC (accuracy) values across four experimental conditions, reflecting the ability of the participants to recognize noisy images.

Each boxplot summarizes the spread, median, and variability of AUC values for each condition, with differences shown in central tendency of the Area under curve for different conditions.

Results and Statistics:

```
Friedman's Chi-square statistic: 5.400000000000006
p-value: 0.14474357941485547
```

Despite observed differences in median AUC across conditions, the Friedman's test (Chi-square = 5.4, p value = 0.1447) reveals the no statistically significant effect of experimental conditions on recognition accuracy.

Thereby suggesting that the conditions do not substantially impact participants' performance based on AUC metrics.

```
data_xl = pd.read_excel('Assignment2-2B-NDM-2024.xlsx', index_col = False)

file_path = 'Assignment2-2B-NDM-2024.xlsx'
xls = pd.ExcelFile(file_path)

auc_dict = {}


# In[87]:


for cnt, sheet_name in enumerate(xls.sheet_names, start=1):
    #skipping the first row containing HA and FT
    df = pd.read_excel(file_path, sheet_name=sheet_name, skiprows=1)

    condition_aucs = []

    # i have seperated the HT and FA columns from the dataframe to make it simpler for me to
    # perform the test

    if cnt == 1: # i have also used cnt to correct the first sheet column offset problem

        ht_cols = df.iloc[:, 1::3]
        fa_cols = df.iloc[:, 2::3]
    else:

        ht_cols = df.iloc[:, ::3]
        fa_cols = df.iloc[:, 1::3]

    # converting columns from object data typee to float64
    ht_cols = ht_cols.apply(pd.to_numeric, errors='coerce')
    fa_cols = fa_cols.apply(pd.to_numeric, errors='coerce')


    fa_cols.columns = ht_cols.columns    #in order to ensure no problems occur while performing
operations on them


    for participant in range(ht_cols.shape[1]):
        ht = ht_cols.iloc[:, participant].values
        fa = fa_cols.iloc[:, participant].values
        sorted_indices = np.argsort(fa)
        fa_sorted = fa[sorted_indices]
        ht_sorted = ht[sorted_indices]

        auc_value = trapz(ht_sorted, fa_sorted)
        condition_aucs.append(auc_value)

    auc_dict[sheet_name] = condition_aucs

auc_df = pd.DataFrame(auc_dict, columns=['Cond1', 'Cond2', 'Cond3', 'Cond4'])




# In[88]:


print(auc_df)
for i in range(4):
    print(f"Mean AUC for Cond{i+1}: {auc_df.iloc[:, i].mean()}")


# In[89]:


plt.figure(figsize=(12, 6))
auc_df.boxplot()
plt.title('Box and Whisker Plot of AUC Values for Each Experimental Condition')
plt.xlabel('Experimental Conditions')
plt.ylabel('AUC (Accuracy)')
plt.grid(False)
plt.show()

friedman_stat, p_value = stats.friedmanchisquare(
    auc_df['Cond1'], auc_df['Cond2'], auc_df['Cond3'], auc_df['Cond4']
)

print(f"Friedman's Chi-square statistic: {friedman_stat}")
print(f"p-value: {p_value}")


# In[ ]:
```

```
# In[ ]:
```

```
# In[ ]:
```