Assignment 1: Learning and Memory PSY 306 (Winter 2024)

Name: Vansh

Roll Number: 2021363

1a)

Calculate the percent of total correct trials for each array set size for each participant and then the mean percent of total correct trials across all participants. Plot a simple bar diagram to represent the mean percent of total correct trials across all participants along with the standard error of the mean (as error bars) for each condition. [4 points]

Python Code:

```
import openpyxl
import numpy as np
import matplotlib.pyplot as plt

file_path_accuracy = 'all_participants_data_accuracy.xlsx'
file_path_change = 'all_participants_data_change.xlsx'
file_path_setSize = 'all_participants_data_setSize.xlsx'

data_accuracy = []
data_change = []
data_setSize = []

workbook = openpyxl.load_workbook(file_path_accuracy)

#Copying Data from all_participants_data_accuracy.xlsx and storing inside a list
of list data_accuracy

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))
```

```
    data_accuracy.append(data)

workbook.close()

workbook = openpyxl.load_workbook(file_path_change)

#Copying Data from all_participants_data_change.xlsx and storing inside a list of list data_change

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_change.append(data)

workbook.close()

workbook = openpyxl.load_workbook(file_path_setSize)

#Copying Data from all_participants_data_setSize.xlsx and storing inside a list of list data_setSize

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue
```

```
        data.append(list(row))

    data_setSize.append(data)

workbook.close()

total=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0]]

correct=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0]]

percentage=[]

dict ={4:0,6:0,8:0}

#calculating and storing in total(list of list) the total number of entries
corresponding to different setSize.

for a in range (17) :
    for b in range (48) :
        for c in range (4):
            dict[data_setSize[a][b][c]]+=1
    total[a][0]=dict[4]
    total[a][1]=dict[6]
    total[a][2]=dict[8]
    dict[4]=0
    dict[6]=0
    dict[8]=0

#calculating and storing in correct(list of list) the total number of entries with
accuracy 1 corresponding to different setSize.

for a in range (17) :
    for b in range (48) :
        for c in range (4):
            if data_accuracy[a][b][c]==1 :
                dict[data_setSize[a][b][c]]+=1
    correct[a][0]=dict[4]
    correct[a][1]=dict[6]
    correct[a][2]=dict[8]
    dict[4]=0
```

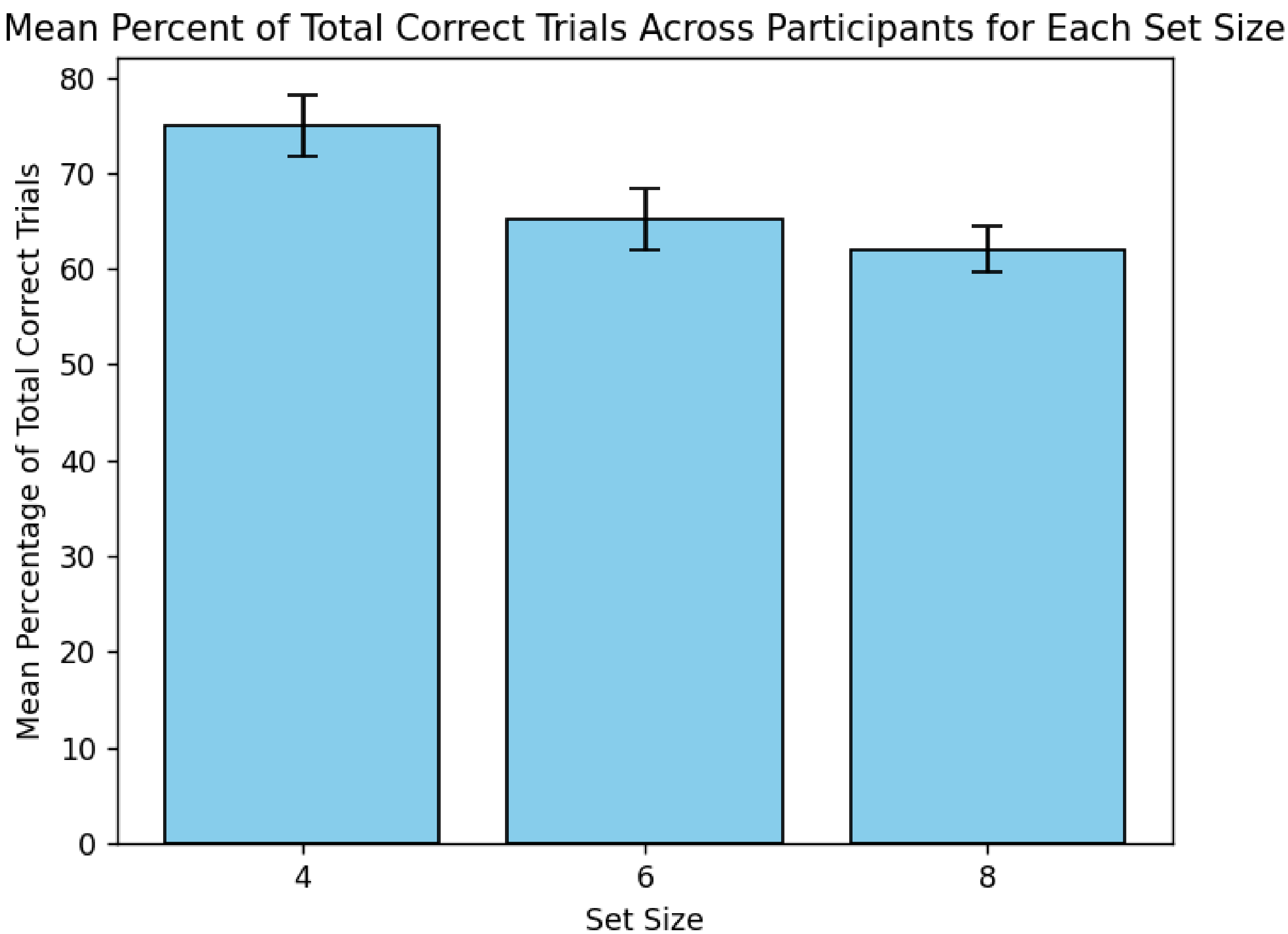
```
    dict[6]=0
    dict[8]=0

data_temp=[]

#calculating the percentage of entries with accuracy 1 for each individual
corresponding to different setSize.

for a in range (17) :
    for b in range (3) :
        percent=correct[a][b]*100/total[a][b]
        data_temp.append(percent)
    percentage.append(data_temp)
    data_temp=[]

mean_percent=[0,0,0]

#calculating the mean percentage for different setSize.

for a in range (17) :
    for b in range (3) :
        mean_percent[b]+=percentage[a][b]


for a in range (3) :
    mean_percent[a]=mean_percent[a]/17

arr_4=[]
arr_6=[]
arr_8=[]

#Storing the data in a structured way to use statistical in-built functions.

for a in range (17) :
    arr_4.append(percentage[a][0])
    arr_6.append(percentage[a][1])
    arr_8.append(percentage[a][2])

#Calculating standard error of the mean for differnt setSize

Error_4=np.std(arr_4)/np.sqrt(len(arr_4))
Error_6=np.std(arr_6)/np.sqrt(len(arr_6))
Error_8=np.std(arr_8)/np.sqrt(len(arr_8))
```

```
conditions = ['4','6','8']
x_pos = range(len(conditions))
errors=[Error_4, Error_6, Error_8]

#Printing on terminal

print ("SetSize :                          4                    6
8")
print ("Mean Percent              : ", end="")
print (mean_percent[0]," ",mean_percent[1]," ", mean_percent[2])
print ("Standard Error of the mean: ", end="")
print (Error_4," ",Error_6," ",Error_8)

#Plotting a simple bar diagram

plt.bar(x_pos, mean_percent, yerr=errors, capsize=5, color='skyblue',
edgecolor='black')

plt.xlabel('Set Size')
plt.ylabel('Mean Percentage of Total Correct Trials')
plt.title('Mean Percent of Total Correct Trials Across Participants for Each Set
Size')
plt.xticks(x_pos, conditions)

plt.show()
```

Output on Terminal:

SetSize :   4   6   8
Mean Percent   : 74.90808823529412   65.25735294117646   62.04044117647059
Standard Error of the mean: 3.2325528060660815   3.1932676563126416   2.4135011076612893

Bar Diagram:



1b)

To compare the mean percent correct trials (across participants) across three conditions, conduct an appropriate statistical test and report the results with the appropriate test statistics and p values. Based on a comparison of the accuracies in all conditions, what can be concluded about the relationship between response accuracy and visual working memory capacity from the experimental data? [3+2+1 points]
[Hint: Check for assumptions of appropriate statistical test stepwise to conduct a test followed by appropriate post-hoc test as discussed in the class to solve the above. Indicate the main steps in your code with clear comments.]

Python Code:

```
import openpyxl
import numpy as np
```

```
import matplotlib.pyplot as plt
import scipy.stats as stats
import pandas as pd
import pingouin as pg
import statsmodels.api as sm

file_path_accuracy = 'all_participants_data_accuracy.xlsx'
file_path_change = 'all_participants_data_change.xlsx'
file_path_setSize = 'all_participants_data_setSize.xlsx'

data_accuracy = []
data_change = []
data_setSize = []

workbook = openpyxl.load_workbook(file_path_accuracy)

#Copying Data from all_participants_data_accuracy.xlsx and storing inside a list
of list data_accuracy

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_accuracy.append(data)

workbook.close()

workbook = openpyxl.load_workbook(file_path_change)

#Copying Data from all_participants_data_change.xlsx and storing inside a list of
list data_change

for sheet_name in workbook.sheetnames:
```

```
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_change.append(data)

workbook.close()

workbook = openpyxl.load_workbook(file_path_setSize)

#Copying Data from all_participants_data_setSize.xlsx and storing inside a list of
list data_setSize

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_setSize.append(data)

workbook.close()

total=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0]]
```

```
correct=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0]]


dict ={4:0,6:0,8:0}
percentage=[]

#calculating and storing in total(list of list) the total number of entries
corresponding to different setSize.

for a in range (17) :
    for b in range (48) :
        for c in range (4):
            dict[data_setSize[a][b][c]]+=1
    total[a][0]=dict[4]
    total[a][1]=dict[6]
    total[a][2]=dict[8]
    dict[4]=0
    dict[6]=0
    dict[8]=0

#calculating and storing in correct(list of list) the total number of entries with
accuracy 1 corresponding to different setSize.

for a in range (17) :
    for b in range (48) :
        for c in range (4):
            if data_accuracy[a][b][c]==1 :
                dict[data_setSize[a][b][c]]+=1
    correct[a][0]=dict[4]
    correct[a][1]=dict[6]
    correct[a][2]=dict[8]
    dict[4]=0
    dict[6]=0
    dict[8]=0

data_temp=[]

#calculating the percentage of entries with accuracy 1 for each individual
corresponding to different setSize.

for a in range (17) :
    for b in range (3) :
```

```
        percent=correct[a][b]*100/total[a][b]
        data_temp.append(percent)
    percentage.append(data_temp)
    data_temp=[]

data = {4:[],6:[],8:[]}

#Storing the data in a structured way to use statistical in-built functions.

for a in range (17) :
    data[4].append(percentage[a][0])
    data[6].append(percentage[a][1])
    data[8].append(percentage[a][2])

dframe = pd.DataFrame(data)

#doing the mauchly test(shericity assumption)

mauchly_result = pg.sphericity(dframe)

print ("Mauchly's Test for checking Sphericity assumption: \n\n",mauchly_result)
print()

#doint the shapiro-Wilk Test(Normality assumption) for different setSize

test_statistic_normality_4, p_value_normality_4 = stats.shapiro(data[4])

print("Shapiro-Wilk Test for assumption of Normality:\n")
print("Test Statistic for setSize=4: ", test_statistic_normality_4)
print("p-value for setSize=4:", p_value_normality_4)

test_statistic_normality_6, p_value_normality_6 = stats.shapiro(data[6])


print("Test Statistic for setSize=6: ", test_statistic_normality_6)
print("p-value for setSize=6: ", p_value_normality_6)

test_statistic_normality_8, p_value_normality_8 = stats.shapiro(data[8])


print("Test Statistic for setSize=8: ", test_statistic_normality_8)
print("p-value for setSize=8: ", p_value_normality_8)
```

```
#doint the Levene's test (Homogeneity of Variances assumption)

test_statistic_homogeneity, p_value_homogeneity = stats.levene(data[4], data[6],
data[8])

print ()
print("Levene's Test for checking the assumption homogeneity of Variances:\n")

print("Test Statistic:", test_statistic_homogeneity)
print("p-value:", p_value_homogeneity)

#doint Repeated Measures ANOVA test and Bonferroni post-hoc test

dframe_melt = pd.melt(dframe, var_name='Condition', value_name='Score')
x=[]
for i in range(3):
    for j in range(1,18):
        x.append(j)
dframe_melt['Participant'] = x
anova=pg.rm_anova(dv='Score', within='Condition', subject='Participant',
data=dframe_melt)
posthoc = sm.stats.multicomp.pairwise_tukeyhsd(dframe_melt['Score'],
dframe_melt['Condition'])

print ("\nRepeated Measures ANOVA test results: \n")
print(anova)

print("\nTukey's post-hoc Test:\n")

print (posthoc)
```

<u>Output on Terminal:</u>

Mauchly's Test for checking Sphericity assumption:

SpherResults(spher=True, W=0.8283093076955215, chi2=2.8255295150145585, dof=2, pval=0.24346921842062474)

Shapiro-Wilk Test for assumption of Normality:

Test Statistic for setSize=4: 0.9472741208625098 p-value for setSize=4: 0.41486039031850974

Test Statistic for setSize=6: 0.9503533879048294
p-value for setSize=6: 0.4621337557812606

Test Statistic for setSize=8: 0.9141280545478703
p-value for setSize=8: 0.11747480157154644

Levene's Test for checking the assumption homogeneity of Variances:

Test Statistic: 1.1158096232489343
p-value: 0.33599856447593074

Repeated Measures ANOVA test results:

	Source	ddof1	ddof2	F	p-unc	ng2	eps
0	Condition	2	32	15.228305	0.000023	0.16618	0.853468

Tukey's post-hoc Test:

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
4	6	-9.6507	0.0765	-20.1232	0.8218	False
4	8	-12.8676	0.0126	-23.3402	-2.3951	True
6	8	-3.2169	0.7393	-13.6894	7.2556	False

Result and Conclusion:

1. The p-value in Mauchly's Test is found to be 0.24346921842062474 which is larger than 0.0.5 which suggests that the assumption of Sphericity is met. The p-value in Shapiro Wilk's of all set sizes are found to be greater than 0.05 which shows that the data is approximately normally distributed which suggests that the assumption of Normality is met. The p-value in Levene's test (0.33599856447593074) is found to be greater than 0.05 which shows that there is no statistically significant difference in the variances between the groups of different set sizes which suggest that the assumption of homogeneity of variances is met. The F-statistic in Repeated Measures ANOVA test is large (15.228305). The p-value in Repeated measures ANOVA test is less than 0.05 which shows that there is a statistically significant difference between at least two of the conditions being compared. We are taking Tukey's test as post-hocs test. The p-value for pair(4,6) and (6,8) is greater than 0.05 so it depicts that there is no significant difference between accuracies of set sizes in these pairs and for the pair(4,8) the p value is less than 0.05 which depicts that there is significant difference between accuracies of set sizes in this pair.

2. It concludes that set size is inversely related to average response accuracy of individual participants.

2a)

Calculate the ‘d prime’ for each array size for all trials for each participant and average ‘d prime’ across participants. Create a bar diagram for each array size showing mean ‘d prime’ (across participants) and standard error of the mean as error bars. [5 points]

Python Code:

```
import openpyxl
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

file_path_accuracy = 'all_participants_data_accuracy.xlsx'
file_path_change = 'all_participants_data_change.xlsx'
file_path_setSize = 'all_participants_data_setSize.xlsx'

data_accuracy = []
data_change = []
data_setSize = []

workbook = openpyxl.load_workbook(file_path_accuracy)

#Copying Data from all_participants_data_accuracy.xlsx and storing inside a list
of list data_accuracy

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_accuracy.append(data)
```

```
workbook.close()

workbook = openpyxl.load_workbook(file_path_change)

#Copying Data from all_participants_data_change.xlsx and storing inside a list of list data_change

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_change.append(data)

workbook.close()

workbook = openpyxl.load_workbook(file_path_setSize)

#Copying Data from all_participants_data_setSize.xlsx and storing inside a list of list data_setSize

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue
```

```
        data.append(list(row))

    data_setSize.append(data)

workbook.close()


hits=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0]]
falsealarm=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0]]
total_change_0=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0]]
total_change_1=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0]]

Dict={4:0,6:0,8:0}

#calculating and storing the total number of 0 change in total_change_0(list of
list) the total number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4):
            if data_change[a][b][c]==0 :
                if data_setSize[a][b][c]==4 :
                    total_change_0[a][0]+=1
                elif data_setSize[a][b][c]==6 :
                    total_change_0[a][1]+=1
                elif data_setSize[a][b][c]==8 :
                    total_change_0[a][2]+=1

#calculating and storing the total number of 1 change in total_change_1(list of
list) the total number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4) :
            if data_change[a][b][c]==1 :
                if data_setSize[a][b][c]==4 :
```

```
                    total_change_1[a][0]+=1
                elif data_setSize[a][b][c]==6 :
                    total_change_1[a][1]+=1
                elif data_setSize[a][b][c]==8 :
                    total_change_1[a][2]+=1

#false alarm = where change is 0 and accuracy is 0
#calculating and storing total number of falsealarm in falsealarm(list of list)
the total number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4) :
            if data_change[a][b][c]==0 and data_accuracy[a][b][c]==0 :
                Dict[data_setSize[a][b][c]]+=1
    falsealarm[a][0]=Dict[4]
    falsealarm[a][1]=Dict[6]
    falsealarm[a][2]=Dict[8]
    Dict[4]=0
    Dict[6]=0
    Dict[8]=0

#hit = where change is 1 and accuracy is 1
#calculating and storing total number of hits in hits(list of list) the total
number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4) :
            if data_change[a][b][c]==1 and data_accuracy[a][b][c]==1 :
                Dict[data_setSize[a][b][c]]+=1
    hits[a][0]=Dict[4]
    hits[a][1]=Dict[6]
    hits[a][2]=Dict[8]
    Dict[4]=0
    Dict[6]=0
    Dict[8]=0

#calculating and storing the z value of false alarm in itself
#calculating and storing the z value of hits in itself

for i in range (17) :
    for j in range (3) :
```

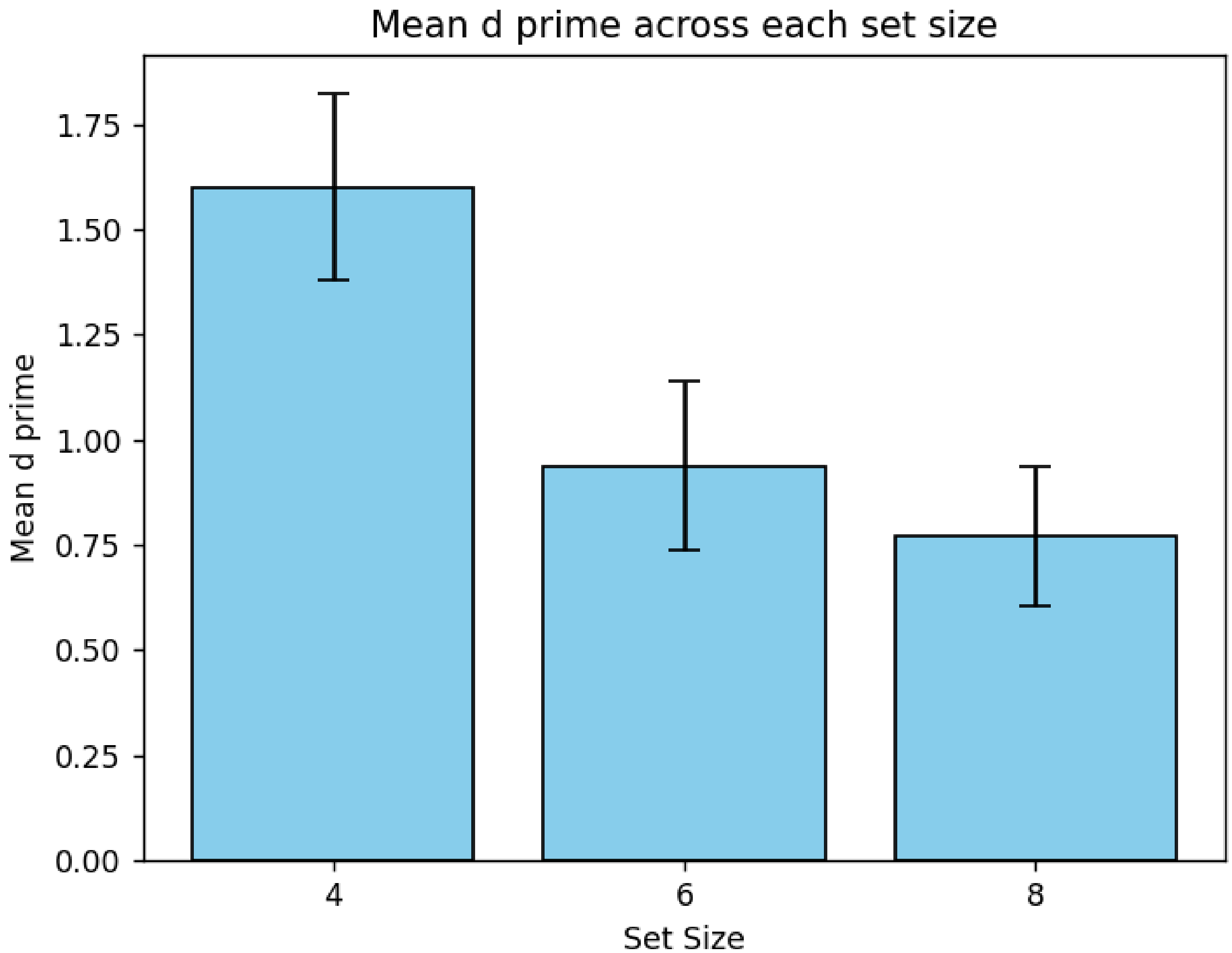
```
        hits[i][j]=hits[i][j]/total_change_1[i][j]
        falsealarm[i][j]=falsealarm[i][j]/total_change_0[i][j]

#calculating and storing the d prime

for i in range (17) :
    for j in range (3) :
        hits[i][j]=stats.norm.ppf(hits[i][j])-stats.norm.ppf(falsealarm[i][j])

d_prime_4=0
d_prime_6=0
d_prime_8=0

for i in range (17) :
    d_prime_4+=hits[i][0]
    d_prime_6+=hits[i][1]
    d_prime_8+=hits[i][2]

d_prime_4=d_prime_4/17
d_prime_6=d_prime_6/17
d_prime_8=d_prime_8/17

average_d_prime=(d_prime_8+d_prime_6+d_prime_4)/3

d_prime=[d_prime_4,d_prime_6,d_prime_8]

arr_4=[]
arr_6=[]
arr_8=[]

#Storing the data in a structured way to use statistical in-built functions.

for i in range (17) :
    arr_4.append(hits[i][0])
    arr_6.append(hits[i][1])
    arr_8.append(hits[i][2])

#calculating and storing the standard error of mean

Error_4=np.std(arr_4)/np.sqrt(len(arr_4))
Error_6=np.std(arr_6)/np.sqrt(len(arr_6))
Error_8=np.std(arr_8)/np.sqrt(len(arr_8))
```

```
conditions = ['4','6','8']

x_pos = range(len(conditions))
errors= [Error_4, Error_6, Error_8]

#Printing on terminal

print ("SetSize :                          4                    6
8")
print ("Mean d prime              : ", end="")
print (d_prime[0]," ",d_prime[1]," ", d_prime[2])
print ("Standard Error of the mean: ", end="")
print (Error_4," ",Error_6," ",Error_8)

#Plotting a simple bar diagram

plt.bar(x_pos, d_prime, yerr=errors, capsize=5, color='skyblue',
edgecolor='black')

plt.xlabel('Set Size')
plt.ylabel('Mean d prime')
plt.title('Mean d prime across each set size')
plt.xticks(x_pos, conditions)

plt.show()
```

Output on terminal:

SetSize : 4 6 8
Mean d prime : 1.6007323915730254 0.9375675732377838 0.7711763832559827
Standard Error of the mean: 0.22217459936936373 0.20097877150277169 0.16703690842575525

Bar Diagram:



2b)

To compare the mean ‘d prime’ (across participants) across three conditions (array size), conduct an appropriate statistical test and report the results with test statistics and p values. Interpret the results of the test statistics. [2+2+1 points]
[Hint: Check for assumptions of appropriate statistical test stepwise to conduct a test followed by appropriate post-hoc test as discussed in the class to solve the above. Indicate the main steps in your code with clear comments.]

Python Code:

```
import openpyxl
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import pandas as pd
```

```python
import pingouin as pg

file_path_accuracy = 'all_participants_data_accuracy.xlsx'
file_path_change = 'all_participants_data_change.xlsx'
file_path_setSize = 'all_participants_data_setSize.xlsx'

data_accuracy = []
data_change = []
data_setSize = []

workbook = openpyxl.load_workbook(file_path_accuracy)

#Copying Data from all_participants_data_accuracy.xlsx and storing inside a list
of list data_accuracy

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_accuracy.append(data)

workbook.close()

workbook = openpyxl.load_workbook(file_path_change)
#Copying Data from all_participants_data_change.xlsx and storing inside a list of
list data_change

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True
```

```
    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_change.append(data)

workbook.close()

workbook = openpyxl.load_workbook(file_path_setSize)
#Copying Data from all_participants_data_setSize.xlsx and storing inside a list of
list data_setSize

for sheet_name in workbook.sheetnames:
    sheet = workbook[sheet_name]

    data = []

    skip_first_row = True

    for row in sheet.iter_rows(values_only=True):
        if skip_first_row:
            skip_first_row = False
            continue

        data.append(list(row))

    data_setSize.append(data)

workbook.close()

hits=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0]]
falsealarm=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0]]
total_change_0=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0]]
```

```
total_change_1=[[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0], [0,0,0],
[0,0,0]]

Dict={4:0,6:0,8:0}
#calculating and storing the total number of 0 change in total_change_0(list of
list) the total number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4):
            if data_change[a][b][c]==0 :
                if data_setSize[a][b][c]==4 :
                    total_change_0[a][0]+=1
                elif data_setSize[a][b][c]==6 :
                    total_change_0[a][1]+=1
                elif data_setSize[a][b][c]==8 :
                    total_change_0[a][2]+=1

#calculating and storing the total number of 1 change in total_change_1(list of
list) the total number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4) :
            if data_change[a][b][c]==1 :
                if data_setSize[a][b][c]==4 :
                    total_change_1[a][0]+=1
                elif data_setSize[a][b][c]==6 :
                    total_change_1[a][1]+=1
                elif data_setSize[a][b][c]==8 :
                    total_change_1[a][2]+=1
#false alarm = where change is 0 and accuracy is 0
#calculating and storing total number of falsealarm in falsealarm(list of list)
the total number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4) :
            if data_change[a][b][c]==0 and data_accuracy[a][b][c]==0 :
                Dict[data_setSize[a][b][c]]+=1
    falsealarm[a][0]=Dict[4]
    falsealarm[a][1]=Dict[6]
```

```
    falsealarm[a][2]=Dict[8]
    Dict[4]=0
    Dict[6]=0
    Dict[8]=0

#hit = where change is 1 and accuracy is 1
#calculating and storing total number of hits in hits(list of list) the total
number of entries corresponding to different setSize

for a in range (17) :
    for b in range (48) :
        for c in range (4) :
            if data_change[a][b][c]==1 and data_accuracy[a][b][c]==1 :
                Dict[data_setSize[a][b][c]]+=1
    hits[a][0]=Dict[4]
    hits[a][1]=Dict[6]
    hits[a][2]=Dict[8]
    Dict[4]=0
    Dict[6]=0
    Dict[8]=0

#calculating and storing the z value of false alarm in itself
#calculating and storing the z value of hits in itself

for i in range (17) :
    for j in range (3) :
        hits[i][j]=hits[i][j]/total_change_1[i][j]
        falsealarm[i][j]=falsealarm[i][j]/total_change_0[i][j]

#calculating and storing the d prime

for i in range (17) :
    for j in range (3) :
        hits[i][j]=stats.norm.ppf(hits[i][j])-stats.norm.ppf(falsealarm[i][j])

data={4:[],6:[],8:[]}

#Storing the data in a structured way to use statistical in-built functions.


for i in range (17) :
    data[4].append(hits[i][0])
    data[6].append(hits[i][1])
```

```
    data[8].append(hits[i][2])

dframe = pd.DataFrame(data)

#doing the mauchly test(shericity assumption)


mauchly_result = pg.sphericity(dframe)

print ("Mauchly's Test for checking Sphericity assumption: \n\n",mauchly_result)
print()

#doint the shapiro-Wilk Test(Normality assumption) for different setSize

test_statistic_normality_4, p_value_normality_4 = stats.shapiro(data[4])

print("Shapiro-Wilk Test for assumption of Normality:\n")
print("Test Statistic for setSize=4: ", test_statistic_normality_4)
print("p-value for setSize=4:", p_value_normality_4)

test_statistic_normality_6, p_value_normality_6 = stats.shapiro(data[6])


print("Test Statistic for setSize=6: ", test_statistic_normality_6)
print("p-value for setSize=6: ", p_value_normality_6)

test_statistic_normality_8, p_value_normality_8 = stats.shapiro(data[8])


print("Test Statistic for setSize=8: ", test_statistic_normality_8)
print("p-value for setSize=8: ", p_value_normality_8)

test_statistic_homogeneity, p_value_homogeneity = stats.levene(data[4], data[6],
data[8])

#doint the Levene's test (Homogeneity of Variances assumption)

print ()
print("Levene's Test for checking the assumption homogeneity of Variances:\n")

print("Test Statistic:", test_statistic_homogeneity)
print("p-value:", p_value_homogeneity)
```

```
#doint Friedman's Test and Bonferroni post-hoc test

dframe_melt = pd.melt(dframe, var_name='Condition', value_name='Score')
x=[]
for i in range(3):
    for j in range(1,18):
        x.append(j)
dframe_melt['Participant'] = x
friedman_result = pg.friedman(data=dframe)
posthoc=pg.pairwise_tests(data=dframe_melt, dv='Score', within='Condition',
subject='Participant', padjust='bonferroni')

#Printing result on terminal

print ("\nFriedman Test Result: \n")
print(friedman_result)

print("\nBonferroni post-hoc Test:\n")

print (posthoc)
```

Output on terminal:

Mauchly's Test for checking Sphericity assumption:

SpherResults(spher=True, W=0.8760147458122959, chi2=1.9855853260254264, dof=2, pval=0.37054045014988723)

Shapiro-Wilk Test for assumption of Normality:

Test Statistic for setSize=4: 0.9784430756828919
p-value for setSize=4: 0.9416202683875113

Test Statistic for setSize=6: 0.9596358738237221
p-value for setSize=6: 0.6246459526608572

Test Statistic for setSize=8: 0.8812880584176418
p-value for setSize=8: 0.0333645522595167

Levene's Test for checking the assumption homogeneity of Variances:

Test Statistic: 0.9577512343807209
p-value: 0.3909659748473192

Friedman Test result:

	Source	W	ddof1	Q	p-unc
Friedman	Within	0.387543	2	13.176471	0.001376

Bonferroni post-hoc Test:

	Contrast	A	B	Paired	Parametric	T	dof	alternative	p-unc	p-corr	p-adjust	BF10	hedges
0	Condition	4	6	True	True	5.316530	16.0	two-sided	0.000070	0.000209	bonferroni	391.393	0.719182
1	Condition	4	8	True	True	4.879293	16.0	two-sided	0.000167	0.000501	bonferroni	180.268	0.969626
2	Condition	6	8	True	True	1.005881	16.0	two-sided	0.329444	0.988331	bonferroni	0.386	0.206863

Result and Conclusion:

1. The p-value in Mauchly's Test is found to be 0.37054045014988723 which is larger than 0.0.5 which suggests that the assumption of Sphericity is met. The p-value in Shapiro Wilk's of set sizes 4 and 6 are found to be greater than 0.05 which shows that the data is approximately normally distributed and p-value for set size 8 is found to be less than 0.05 which shows that that data is not normally distributed in setsize 8. The p-value in Levene's test is found to be greater than 0.05 which shows that there is no statistically significant difference in the variances between the groups of different set sizes which suggest that the assumption of homogeneity of variances is met. The p-value in Friedman's test (0.001376) is found to be smaller than 0.05 which shows that there is a statistically significant difference between at least two of the conditions being compared. In the post-hocs test, we are using the Bonferroni test. The corrected p-value of pair 4 & 8 and 4 & 6 in Bonferroni test are found to be less than 0.05 as there is a significant difference between average d-primes of 4 & 6 and 4 & 8. The corrected p-value of pair 6&8 is larger than 0.05 as there is no significant difference in the average d-primes.