Assignment 2: Learning and Memory PSY 306 (Winter 2024)

Name: Vansh

Roll Number: 2021363

Part B)

3 participants performed a task where they were presented with a series of choice sets, each consisting of two options: reward_today, which is the discounted value of delayed reward and reward_later, which is the amount of delayed reward. They were asked to play like they were playing for real money, and indicate their responses/preferences through key press.

Each sheet in the attached excel file **(LM_Assignment2-1.xlsx)** has data of 1 participant for 27 trials and there are a total of three sheets. Each sheet contains the following columns:

- reward_today (in Rs.)
- reward_later (in Rs.)
- delay (days) - represent the duration for receipt of reward_later
- Size: represents the category of the size of the reward_later where
  - 'S' represents small
  - 'M' represents medium
  - 'L' represents large
- key_press:
  - 'left' indicates preference for reward_today
  - 'right' indicates preference for reward_later

Assume the sensitivity to delay as 1.

Now, carry out the following…

1a)

i) Calculate the rate at which future rewards are devalued at every indifference point/trial. [Indifference point is the point in which participants do not discriminate between the two rewards.]

ii) Sort the entire data in ascending order with respect to the values calculated in (i) and then calculate the geometric mean of the two indifference points where the switch occurs. Switch here means where the response/preference of the participant changes either from 'reward_today' to 'reward_later' or 'reward_later' to 'reward_today'.

iii) To calculate cumulative rate, take the geometric mean of all the switch points.

Plot the cumulative rate calculated for each participant in a single bar graph. Report the cumulative rate on top of each bar. What can be concluded about self-control in reward driven learning from the cumulative rates calculated?

[2+3+2+3 points]

Python Code:

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

file_path='LM_Assignemnt2-1.xlsx'

data_1=[]
data_2=[]
data_3=[]

workbook = openpyxl.load_workbook(file_path)

i=0
#Accessing and storing the data in 'LM_Assignemnt2-1.xlsx' in 3 lists and
calculating the rate of devaluation of future rewards and storing inside the same
list.
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

        if (sheet_name=="P1") :
            data_1.append(list(row))
            data_1[i].append(((data_1[i][1]/data_1[i][0])-1)/data_1[i][2])
            i+=1

        if (sheet_name=="P2") :
            data_2.append(list(row))
            data_2[i].append(((data_2[i][1]/data_2[i][0])-1)/data_2[i][2])
            i+=1

        if (sheet_name=="P3") :
            data_3.append(list(row))
            data_3[i].append(((data_3[i][1]/data_3[i][0])-1)/data_3[i][2])
            i+=1
    i=0
workbook.close()

#sorting the data with respect to the rate of devaluation of future rewards.
sorted_data_1 = sorted (data_1, key = lambda x: x[5])
sorted_data_2 = sorted (data_2, key = lambda x: x[5])
sorted_data_3 = sorted (data_3, key = lambda x: x[5])

GM_1=[]
GM_2=[]
GM_3=[]

#Calculating the Geomtric mean of two indifference points where the switch occurs.
for i in range (1,len(sorted_data_1)) :
    if (sorted_data_1[i-1][4]!=sorted_data_1[i][4]) :
        GM_1.append((sorted_data_1[i-1][5]*sorted_data_1[i][5])**0.5)

for i in range (1,len(sorted_data_2)) :
    if (sorted_data_2[i-1][4]!=sorted_data_2[i][4]) :
        GM_2.append((sorted_data_2[i-1][5]*sorted_data_2[i][5])**0.5)

for i in range (1,len(sorted_data_3)) :
    if (sorted_data_3[i-1][4]!=sorted_data_3[i][4]) :
        GM_3.append((sorted_data_3[i-1][5]*sorted_data_3[i][5])**0.5)

CR_1=1
CR_2=1
CR_3=1

#Calculating the cumulative rate by taking the geometric mean of all the switch
points
for i in range (len(GM_1)) :
    CR_1=CR_1*GM_1[i]
CR_1=CR_1**(1/len(GM_1))

for i in range (len(GM_2)) :
    CR_2=CR_2*GM_2[i]
CR_2=CR_2**(1/len(GM_2))

for i in range (len(GM_3)) :
    CR_3=CR_3*GM_3[i]
CR_3=CR_3**(1/len(GM_3))

#Printing on the terminal
print("Cumulative Rate :")
print ()
print("Participant 1: ", end="")
print (CR_1)
print("Participant 2: ", end="")
print (CR_2)
print("Participant 3: ", end="")
print (CR_3)

#Plotting the graph
participants = ['Participant 1', 'Participant 2', 'Participant 3']
cumulative_rates = [CR_1, CR_2, CR_3]
plt.figure(figsize=(8, 6))
bars = plt.bar(participants, cumulative_rates, color='skyblue')
plt.xlabel('Participant')
plt.ylabel('Cumulative Rate')
plt.title('Cumulative Rate of Devaluation for Each Participant')
#Reporting the cumulative rate on top of each bar.
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval, f'{yval:.4f}', ha='center',
va='bottom')


plt.show()
```

Output on terminal:

```
Cumulative Rate :

Participant 1: 0.006135775914207119
Participant 2: 0.034887712616626515
Participant 3: 0.058534263621015714
```

Bar Graph:

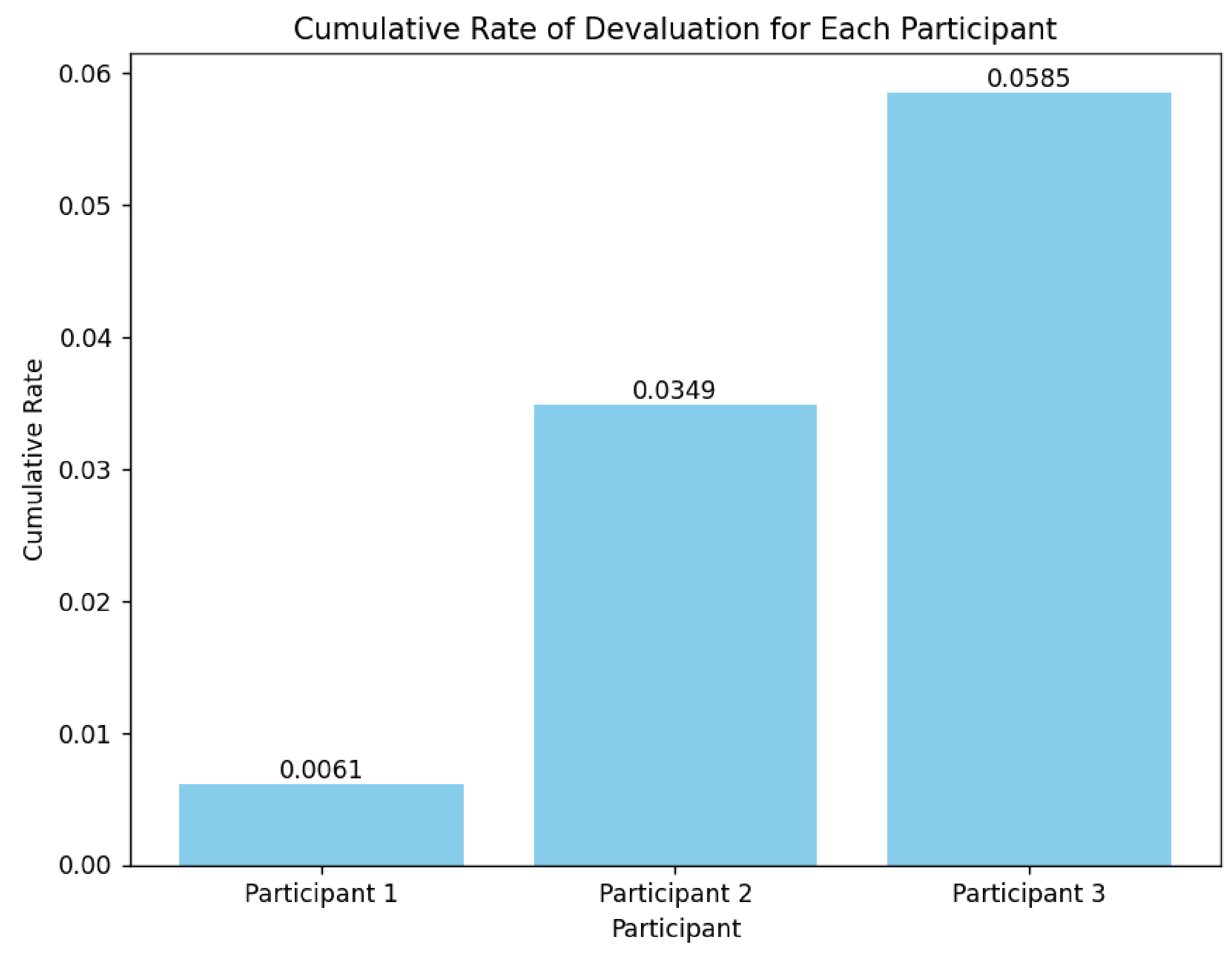


Figure Caption: Bar graph showing the cumulative rate of preference for delayed reward across three participants. Cumulative rate represents the proportion of trials in which the delayed reward was preferred over immediate reward. Higher cumulative rates indicate greater self-control in favor of delayed rewards."

conclusion about self-control in reward driven learning from the cumulative rates calculated:

From the provided cumulative rates for each participant:
Participant 1 has the lowest cumulative rate (0.0061), indicating a higher tendency to prefer immediate rewards over delayed rewards. This suggests lower self-control in reward-driven learning for Participant 1.
Participant 2 has a higher cumulative rate (0.0349) compared to Participant 1, but still relatively low, indicating some level of self-control but still a preference for immediate rewards.
Participant 3 has the highest cumulative rate (0.0585), suggesting a greater ability to delay gratification and a higher level of self-control in reward-driven learning compared to the other participants.
Overall, these cumulative rates suggest that Participant 3 demonstrates the highest level of self-control in reward-driven learning, followed by Participant 2, and then Participant 1.

1b)

Sort each sheet (participant’s data) based on the column ‘size’. Repeat the three steps mentioned above to calculate the cumulative rate. [If there is only one switch point, consider it as cumulative rate.]

Create one bar graph and plot the mean cumulative rate (across participants) calculated along y axis and the size of the reward along x-axis. Also, calculate the standard error of the mean (across participants) and add as error bars over the mean.
What is the role of the size of the reward on self-control in this reward driven learning?

[4+1 points]

Python Code:

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

file_path='LM_Assignemnt2-1.xlsx'

data_1=[]
data_2=[]
data_3=[]

workbook = openpyxl.load_workbook(file_path)

i=0
```

```
#Accessing and storing the data in 'LM_Assignemnt2-1.xlsx' in 3 lists and
calculating the rate of devaluation of future rewards and storing inside the same
list.
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

        if (sheet_name=="P1") :
            data_1.append(list(row))
            data_1[i].append(((data_1[i][1]/data_1[i][0])-1)/data_1[i][2])
            i+=1

        if (sheet_name=="P2") :
            data_2.append(list(row))
            data_2[i].append(((data_2[i][1]/data_2[i][0])-1)/data_2[i][2])
            i+=1

        if (sheet_name=="P3") :
            data_3.append(list(row))
            data_3[i].append(((data_3[i][1]/data_3[i][0])-1)/data_3[i][2])
            i+=1
    i=0
workbook.close()

#Sorting the data of each participants w.r.t. the size
sorted_data_1 = sorted (data_1, key = lambda x: x[3], reverse=True)
sorted_data_2 = sorted (data_2, key = lambda x: x[3], reverse=True)
sorted_data_3 = sorted (data_3, key = lambda x: x[3], reverse=True)

#Storing the data of each size of each participant in different lists
data_1_s=sorted_data_1[:9]
data_1_m=sorted_data_1[9:18]
data_1_l=sorted_data_1[18:]

data_2_s=sorted_data_2[:9]
```

```
data_2_m=sorted_data_2[9:18]
data_2_l=sorted_data_2[18:]

data_3_s=sorted_data_3[:9]
data_3_m=sorted_data_3[9:18]
data_3_l=sorted_data_3[18:]

#Sorting w.r.t. rate of devaluation of future rewards.
data_1_s=sorted (data_1_s, key= lambda x: x[5])
data_1_m=sorted (data_1_m, key= lambda x: x[5])
data_1_l=sorted (data_1_l, key= lambda x: x[5])

data_2_s=sorted (data_2_s, key= lambda x: x[5])
data_2_m=sorted (data_2_m, key= lambda x: x[5])
data_2_l=sorted (data_2_l, key= lambda x: x[5])

data_3_s=sorted (data_3_s, key= lambda x: x[5])
data_3_m=sorted (data_3_m, key= lambda x: x[5])
data_3_l=sorted (data_3_l, key= lambda x: x[5])

GM_1_s=[]
GM_1_m=[]
GM_1_l=[]

GM_2_s=[]
GM_2_m=[]
GM_2_l=[]

GM_3_s=[]
GM_3_m=[]
GM_3_l=[]

#Calculating the geometric mean of the two indifference points where the switch
occurs.
for i in range (1, len(data_1_s)) :
    if data_1_s[i-1][4]!=data_1_s[i][4] :
        GM_1_s.append((data_1_s[i-1][5]*data_1_s[i][5])**0.5)

for i in range (1, len(data_1_m)) :
    if data_1_m[i-1][4]!=data_1_m[i][4] :
        GM_1_m.append((data_1_m[i-1][5]*data_1_m[i][5])**0.5)

for i in range (1, len(data_1_l)) :
    if data_1_l[i-1][4]!=data_1_l[i][4] :
        GM_1_l.append((data_1_l[i-1][5]*data_1_l[i][5])**0.5)

for i in range (1, len(data_2_s)) :
    if data_2_s[i-1][4]!=data_2_s[i][4] :
        GM_2_s.append((data_2_s[i-1][5]*data_2_s[i][5])**0.5)

for i in range (1, len(data_2_m)) :
    if data_2_m[i-1][4]!=data_2_m[i][4] :
        GM_2_m.append((data_2_m[i-1][5]*data_2_m[i][5])**0.5)

for i in range (1, len(data_2_l)) :
    if data_2_l[i-1][4]!=data_2_l[i][4] :
        GM_2_l.append((data_2_l[i-1][5]*data_2_l[i][5])**0.5)

for i in range (1, len(data_3_s)) :
    if data_3_s[i-1][4]!=data_3_s[i][4] :
        GM_3_s.append((data_3_s[i-1][5]*data_3_s[i][5])**0.5)

for i in range (1, len(data_3_m)) :
    if data_3_m[i-1][4]!=data_3_m[i][4] :
        GM_3_m.append((data_3_m[i-1][5]*data_3_m[i][5])**0.5)

for i in range (1, len(data_3_l)) :
    if data_3_l[i-1][4]!=data_3_l[i][4] :
        GM_3_l.append((data_3_l[i-1][5]*data_3_l[i][5])**0.5)

CR_1_S=1
CR_1_M=1
CR_1_L=1
CR_2_S=1
CR_2_M=1
CR_2_L=1
CR_3_S=1
CR_3_M=1
CR_3_L=1

#Calculating the cumulative rate by taking the geometric mean of all the switch
points.
for i in range (len(GM_1_s)) :
    CR_1_S=CR_1_S*GM_1_s[i]
CR_1_S=CR_1_S**(1/len(GM_1_s))
```

```
for i in range (len(GM_1_m)) :
    CR_1_M=CR_1_M*GM_1_m[i]
CR_1_M=CR_1_M**(1/len(GM_1_m))

for i in range (len(GM_1_l)) :
    CR_1_L=CR_1_L*GM_1_l[i]
CR_1_L=CR_1_L**(1/len(GM_1_l))

for i in range (len(GM_2_s)) :
    CR_2_S=CR_2_S*GM_2_s[i]
CR_2_S=CR_2_S**(1/len(GM_2_s))

for i in range (len(GM_2_m)) :
    CR_2_M=CR_2_M*GM_2_m[i]
CR_2_M=CR_2_M**(1/len(GM_2_m))

for i in range (len(GM_2_l)) :
    CR_2_L=CR_2_L*GM_2_l[i]
CR_2_L=CR_2_L**(1/len(GM_2_l))

for i in range (len(GM_3_s)) :
    CR_3_S=CR_3_S*GM_3_s[i]
CR_3_S=CR_3_S**(1/len(GM_3_s))

for i in range (len(GM_3_m)) :
    CR_3_M=CR_3_M*GM_3_m[i]
CR_3_M=CR_3_M**(1/len(GM_3_m))

for i in range (len(GM_3_l)) :
    CR_3_L=CR_3_L*GM_3_l[i]
CR_3_L=CR_3_L**(1/len(GM_3_l))

CR_s=[CR_1_S, CR_2_S, CR_3_S]
CR_m=[CR_1_M, CR_2_M, CR_3_M]
CR_l=[CR_1_L, CR_2_L, CR_3_L]

#Calculating the standard error of the mean across participants
Error_s=np.std(CR_s)/np.sqrt(len(CR_s))
Error_m=np.std(CR_m)/np.sqrt(len(CR_m))
Error_l=np.std(CR_l)/np.sqrt(len(CR_l))

#Calculating mean cumulative rate
Mean_s=(CR_1_S + CR_2_S + CR_3_S)/3
```

```
Mean_m=(CR_1_M+ CR_2_M+ CR_3_M)/3
Mean_l=(CR_1_L+ CR_2_L+ CR_3_L)/3

# Size labels
Size = ['Small', 'Medium', 'Large']

Mean=[Mean_s, Mean_m, Mean_l]

Errors=[Error_s, Error_m, Error_l]

#Printing on terminal
print ("Cumulative rate of each participant with size=small: ", end="")
print (CR_s)
print ("Cumulative rate of each participant with size=medium: ", end="")
print (CR_m)
print ("Cumulative rate of each participant with size=large: ", end="")
print (CR_l)
print ()
print ("Mean Cumulative rate for all participants across different sizes: ",
end="")
print (Mean)
print ("Standard Error of mean: ", end="")
print (Errors)

#Plotting
plt.figure(figsize=(8, 6))
plt.bar(Size, Mean, yerr=Errors, capsize=5, color='skyblue', alpha=0.7)
plt.xlabel('Size of Reward')
plt.ylabel('Mean Cumulative Rate')
plt.title('Mean Cumulative Rate Across Participants for Each Reward Size')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```

Output on terminal:

```
Cumulative rate of each participant with size=small: [0.0098789024227099,
0.06499698718864959, 0.15936381457791915]
Cumulative rate of each participant with size=medium: [0.009563873279839782,
0.02532513782091461, 0.02532513782091461]
Cumulative rate of each participant with size=large: [0.0038221028470741786,
0.009856283612727638, 0.009856283612727638]
```

Mean Cumulative rate for all participants across different sizes: [0.07807990139642622, 0.02007138297388967, 0.007844890024176486]
Standard Error of mean: [0.035636456750040214, 0.004289672869628334, 0.0016422959879519661]

Bar Graph:

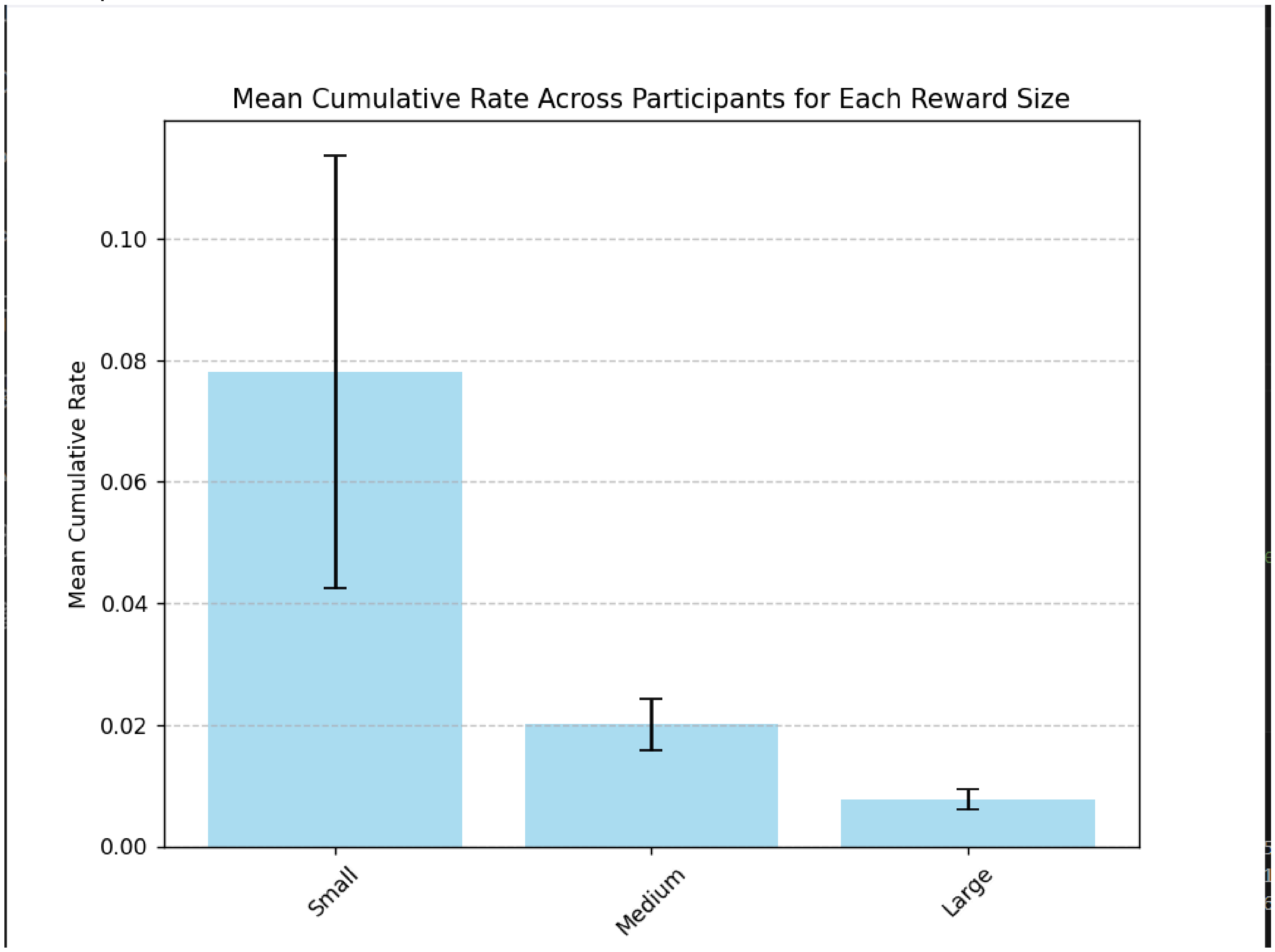


Figure Caption : Bar graph showing the mean cumulative rate of preference for delayed reward across reward sizes. Error bars represent the standard error of the mean (SEM). The mean cumulative rate is calculated across three participants."

The role of the size of the reward on self-control in this reward driven learning:

Effectiveness of Different Reward Sizes: Comparing the mean cumulative rates across different reward sizes can reveal which size was most effective in promoting self-control. A higher mean cumulative rate for a particular reward size suggests that it was more successful in promoting self-control compared to others.

Small Reward Size: This size appears to be the most effective in promoting self-control, as it has the highest mean cumulative rate (0.0781). A higher mean cumulative rate suggests that participants were more inclined to choose delayed rewards over immediate rewards when presented with a small reward size.

Medium Reward Size: The mean cumulative rate for the medium reward size is lower (0.0201) compared to the small reward size. This indicates that participants showed less self-control when presented with medium-sized rewards compared to small-sized rewards.

Large Reward Size: The large reward size has the lowest mean cumulative rate (0.0078) among all sizes. This suggests that participants exhibited the least self-control when presented with large rewards, as they were more likely to choose immediate rewards over delayed rewards.

2)

An experimenter carries out three pilot experiments of 30 trials each in human subjects to study the relationship between time (# trials) and Associative learning between the exposure to sets of environmental stimuli (Conditioned and Unconditioned Stimuli). She collects and averages the data across an equal number of subjects for each pilot experiment. This data is entered in the **(LM_Assignment2-2.xlsx)**.
Each row = 1 pilot experiment. Each column is the value/magnitude of the CR (arbitrary units).
Now carry out the following...

a) Computationally estimate the Rates and Asymptotes of Learning for the three pilot experiments. Create three subplots for three experiments as part of one larger plot to graph the individual data points (as open circle markers; black color) and overlay of the learning curve (blue color) on each subplot. Indicate the Learning rate and Learning asymptote on top of each subplot (as title).
Also, report any one metric of "goodness of fit" for each of the three learning curves to the underlying experimental data and briefly explain the quality of your curve fit to the experimental data based on the metric.

Hint: - Use unconstrained nonlinear optimization to find the optimal parameters of the negatively accelerated learning curve which best describes the relationship within the data, quantitatively.
For a measure of goodness of curve fit to the experimental data, explore and report any one of these metrics - sum of squared errors OR R square OR adjusted R square.

b) Based on your analysis of the data what can you conclude about the intensities of the Unconditioned Stimuli in the three pilot experiments and why?

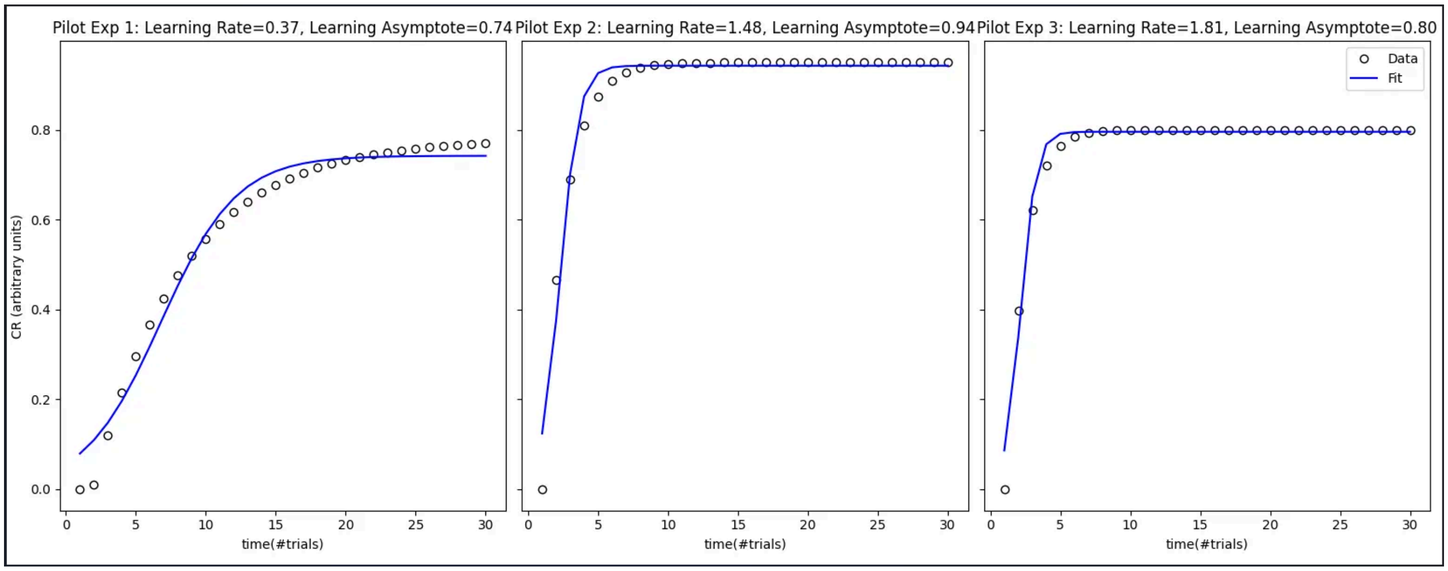
[4+1 points]

Python Code:

```
from scipy.optimize import curve_fit
import matplotlib.pyplot as plot
import pandas as panda
import numpy as nump

temp1=0
```

```python
#Accessing/reading the excel file './LM_Assignment2-2.xlsx'
df = panda.read_excel('./LM_Assignment2-2.xlsx', header=None)
#Seperating the data and considering the first column is just text/label so
skipping that.
data_1 = df.iloc[0, 1:].reset_index(drop=True)
data_2 = df.iloc[1, 1:].reset_index(drop=True)
temp1=temp1+1
data_3 = df.iloc[2, 1:].reset_index(drop=True)
temp1=temp1-1

#function_logistic as learning curve model
def function_logistic(a, asym, rate, mid):
    return asym / (1 + nump.exp(-rate * (a - mid)))

temp1=temp1+2

trials = nump.arange(1, 31)
#Fitting logistic curve to the data of every pilot exp.
data_1_params, _ = curve_fit(function_logistic, trials, data_1)
temp1=temp1*2
data_2_params, _ = curve_fit(function_logistic, trials, data_2)
data_3_params, _ = curve_fit(function_logistic, trials, data_3)
temp1=temp1/2
#Fitted values generation for plot
data_1_fitted_values = function_logistic(trials, *data_1_params)
data_2_fitted_values = function_logistic(trials, *data_2_params)
temp1=temp1*2
data_3_fitted_values = function_logistic(trials, *data_3_params)
#Plotting
temp1=temp1-1
temp, plotting = plot.subplots(1, 3, figsize=(18, 6), sharey=True)

def r_squared(y_true, y_pred):
    ss_res = nump.sum((y_true - y_pred) ** 2)
    ss_tot = nump.sum((y_true - nump.mean(y_true)) ** 2)
    return 1 - (ss_res / ss_tot)
r_squared_exp_1 = r_squared(data_1, data_1_fitted_values)
r_squared_exp_2 = r_squared(data_2, data_2_fitted_values)
r_squared_exp_3 = r_squared(data_3, data_3_fitted_values)
print ("Experiment 1: R-squared value = ", end="")
print (r_squared_exp_1)
print ("Experiment 2: R-squared value = ", end="")
print (r_squared_exp_2)
print ("Experiment 3: R-squared value = ", end="")
print (r_squared_exp_3)


plotting[0].plot(trials, data_1, 'o', mfc='none', mec='black', label='Data')
temp1=temp1+1
plotting[0].plot(trials, data_1_fitted_values, '-', color='blue', label='Fit')
plotting[0].set_title(f'Pilot Exp 1: Learning Rate={data_1_params[1]:.2f},
Learning Asymptote={data_1_params[0]:.2f}')
temp1=temp1-1
plotting[0].set_xlabel('time(#trials)')
plotting[0].set_ylabel('CR (arbitrary units)')
temp2=temp1+1

plotting[1].plot(trials, data_2, 'o', mfc='none', mec='black', label='Data')
plotting[1].plot(trials, data_2_fitted_values, '-', color='blue', label='Fit')
temp1=temp1+3
plotting[1].set_title(f'Pilot Exp 2: Learning Rate={data_2_params[1]:.2f},
Learning Asymptote={data_2_params[0]:.2f}')
plotting[1].set_xlabel('time(#trials)')
temp1=temp2-5


plotting[2].plot(trials, data_3, 'o', mfc='none', mec='black', label='Data')
plotting[2].plot(trials, data_3_fitted_values, '-', color='blue', label='Fit')
temp2=temp1-3
plotting[2].set_title(f'Pilot Exp 3: Learning Rate={data_3_params[1]:.2f},
Learning Asymptote={data_3_params[0]:.2f}')
plotting[2].set_xlabel('time(#trials)')
temp1=temp2+3

plot.legend()
plot.tight_layout()
temp2=temp1-4
plot.show()
```

Plot:



Output on terminal:

```
Experiment 1: R-squared value = 0.9792540885072844
Experiment 2: R-squared value = 0.9704850173922641
Experiment 3: R-squared value = 0.9797677686498184
```

Quality of your curve fit to the experimental data based on the metric:

Experiment 1: R-squared value = 0.9793

This indicates that approximately 97.93% of the variability in the data is explained by the logistic growth model fitted to Experiment 1.
A value close to 1 suggests that the model provides an excellent fit to the experimental data, capturing almost all of the observed variability.
Experiment 2: R-squared value = 0.9705

For Experiment 2, the R-squared value is approximately 0.9705, indicating that around 97.05% of the variability in the data is explained by the fitted model.
This value also suggests a very good fit, with the model explaining a substantial portion of the observed variability in the experimental data.
Experiment 3: R-squared value = 0.9798

Similarly, Experiment 3 has an R-squared value of approximately 0.9798, indicating that roughly 97.98% of the variability in the data is explained by the model.
Like the other experiments, this high R-squared value suggests an excellent fit of the logistic growth model to the experimental data.
Overall, based on the high R-squared values obtained for all three experiments (all above 0.97), we can conclude that the logistic growth model provides an excellent fit to the experimental data.