Assignment 3: Neuroscience of Decision Making PSY 307 (Monsoon 2024)

Name: Anjaneya Sharma

Roll Number: 2021449

Instructions: Please write your own responses and do not copy or lift text/code from any source (including the 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_RollNo. before submitting. Please note that answers in this template only will be evaluated and hand-written or scanned answer sheets will not be evaluated.

[Strict deadline for submission: 23 November, Saturday, 11:00 AM]

Please paste the relevant code and graphs in this document

---

1. Fill out the google form: https://forms.gle/AFfSvHugnMg3JMCQ6

2. Two independent groups of participants (63 each) performed an Iowa gambling task. There are a total of 4 decks and 100 trials. Decks 1 and 2 yield immediate and steady rewards, but they are also characterised by unpredictable occasional losses that can result in negative long-term outcomes. Decks 3 and 4 offer relatively lower and steady immediate rewards, accompanied by even lower and less unpredictable occasional losses, leading to favourable long-term outcomes. The file 'choice.xlsx' contains the deck chosen by the participant from four available options - deck 1, 2, 3, 4, 'win.xlsx' contains the gain associated with the deck chosen and 'loss.xlsx' contains the loss associated with the deck chosen. Each excel file contains two sheets representing the two groups. Each sheet contains 63 rows and 100 columns, each row representing one participant's data and each column represents one trial.

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).

A) Calculate the proportion of switches made after a loss and a gain trial for each participant. Switch refers to a change in the choice of the deck in the subsequent trial. A loss trial is where the money received is less than the amount lost, and a gain trial is where the money received is greater than the amount lost. Create a larger plot with two subplots—one subplot representing each group. Plot bar diagrams representing the mean proportion of switched responses for the gain and loss trial and the standard error of the mean for each group. [6+4 marks]
Conduct appropriate statistical tests to compare the

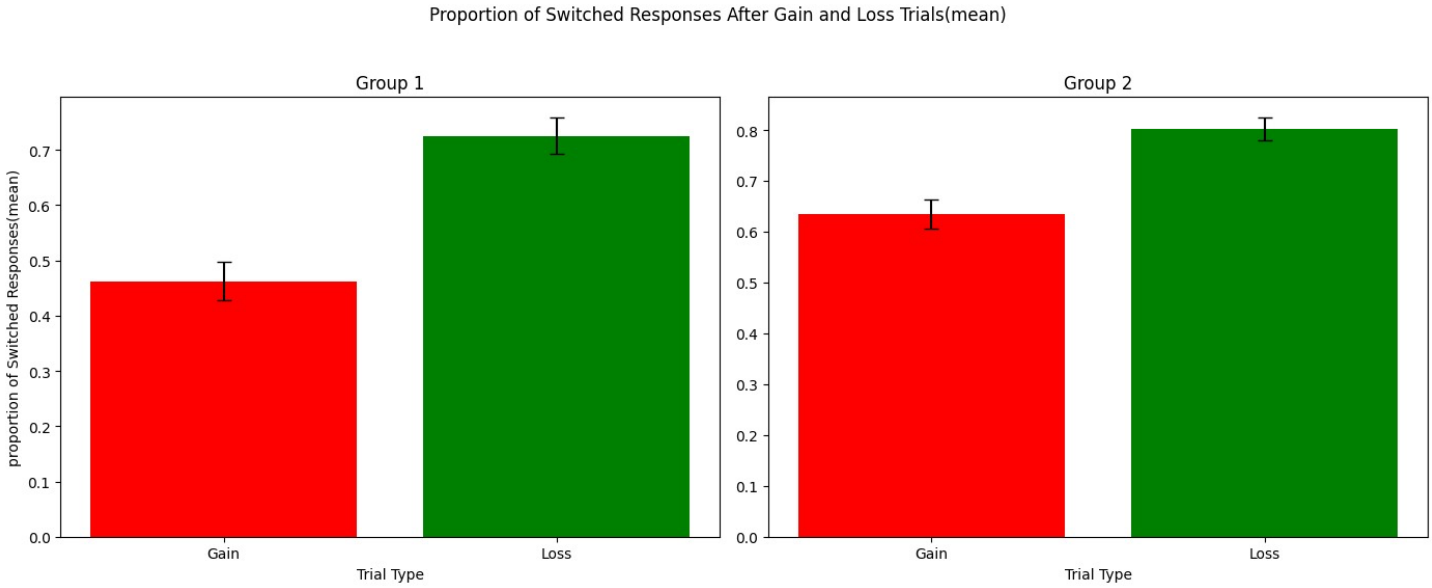
i) proportion of switched responses of gain/ loss trials between groups. Briefly explain the findings of the statistical analysis carried out.

ii) proportion of switched responses of gain trial and loss trials within each group. Briefly explain the findings of the statistical analysis carried out.

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

Answer:

Figure:



Captions for Figure:

Comparison of the mean proportion of switched responses after gain and loss trials for Groups 1 and 2. The results clearly indicate a signifiicant difference in switching behavior,with participants showing a much higher likelihood to switch after loss trials compared to gain trials in both groups. Group 2 exhibited slightly higher switching proportions overall, thereby reflecting a greater sensitivity to trial outcomes.

Statistical Tests used and Findings:

Statistical Tests Used:

Normality Check (Shapiro-Wilk Test):

- Group 1 Gain: p=0.0418 (not normal)
- Group 1 Loss: p=0.00008 (not normal)
- Group 2 Gain: p=0.0027 (not normal)
- Group 2 Loss: p=0.00003 (not normal)

Between-Group Comparisons (Mann-Whitney U Test):

- Gain Trials: U=1210.0, p=0.00016 (significant difference).

- Loss Trials: U=1741.0, p=0.2351 (no significant difference).

Within-Group Comparisons (Wilcoxon Test):

- Group 1 (Gain vs. Loss): W=62.0, p<0.00001 (significant difference).
- Group 2 (Gain vs. Loss): W=130.0, p.00001 (significant difference).

from all this information, we can conclude that participants were significantly more likely to switch their choices after experiencing a loss trial than after a gain trial, regardless of group. Group 2 exhibited slightly higher switching rates overall, which could suggest greater adaptability or responsiveness to trial outcomes. This emphasizes a loss aversion strategy where individuals tend to reconsider their choices more critically following losses than gains.

Code for the following is given below

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

# In[64]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import shapiro, mannwhitneyu, wilcoxon, ttest_ind, ttest_rel
from math import sqrt




# In[65]:


choice = pd.read_excel('choice.xlsx', sheet_name=None, header = None)
win = pd.read_excel('win.xlsx', sheet_name=None, header = None)
loss = pd.read_excel('loss.xlsx', sheet_name=None, header = None)


# In[66]:


len(choice['group2'])


# # Question 2 (A)

# In[66]:
```

```
# In[67]:


def calculate_switch_proportions(choice, win, loss):
    participants = choice.shape[0]
    trials = choice.shape[1]
    switch_after_gain = np.zeros(participants)
    switch_after_loss = np.zeros(participants)

    for i in range(participants):
        switches_gain = 0
        switches_loss = 0
        gain_trials = 0
        loss_trials = 0

        for t in range(trials - 1):
            if win.iloc[i, t] > abs(loss.iloc[i, t]):  # Gain trial (since loss values are <= 0)
                gain_trials += 1
                if choice.iloc[i, t] != choice.iloc[i, t + 1]:
                    switches_gain += 1
            else:  # Loss trial
                loss_trials += 1
                if choice.iloc[i, t] != choice.iloc[i, t + 1]:
                    switches_loss += 1

        switch_after_gain[i] = switches_gain / (gain_trials)
        switch_after_loss[i] = switches_loss / (loss_trials)

    return switch_after_gain, switch_after_loss


# In[68]:


switch_gain_group1, switch_loss_group1 = calculate_switch_proportions(choice['group1'], win['group1'],
loss['group1'])
switch_gain_group2, switch_loss_group2 = calculate_switch_proportions(choice['group2'], win['group2'],
loss['group2'])


# In[69]:
```

```
mean_gain_group1 = np.mean(switch_gain_group1)
mean_loss_group1 = np.mean(switch_loss_group1)
mean_gain_group2 = np.mean(switch_gain_group2)
mean_loss_group2 = np.mean(switch_loss_group2)


# In[70]:


sem_gain_group1 = stats.sem(switch_gain_group1)
sem_loss_group1 = stats.sem(switch_loss_group1)
sem_gain_group2 = stats.sem(switch_gain_group2)
sem_loss_group2 = stats.sem(switch_loss_group2)


# In[71]:


fig, axes = plt.subplots(1, 2, figsize=(14, 6), sharey=True)
# Group 1 plot

axes[0].bar(['Gain', 'Loss'], [mean_gain_group1, mean_loss_group1], yerr=[sem_gain_group1,
sem_loss_group1], capsize=5, color=['blue', 'red'])
axes[0].set_title('Group 1')
axes[0].set_xlabel('Trial Type')
axes[0].set_ylabel('Mean Proportion of Switched Responses')


# Group 2 plot
axes[1].bar(['Gain', 'Loss'], [mean_gain_group2, mean_loss_group2], yerr=[sem_gain_group2,
sem_loss_group2], capsize=5, color=['blue', 'red'])
axes[1].set_title('Group 2')
axes[1].set_xlabel('Trial Type')


plt.suptitle('Mean Proportion of Switched Responses After Gain and Loss Trials')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()


# ### Checking additional stats and see if there is a need for correction

# In[72]:


def diagnostic_check(data, group_name):
```

```
    print(f"Diagnostic check for {group_name}:")
    print(f"Mean: {np.mean(data):.3f}, Std Dev: {np.std(data):.3f}, Min: {np.min(data):.3f}, Max: {np.max(data):.3f}")
    print(f"Number of unique values: {len(np.unique(data))}")
    if len(np.unique(data)) < 5:
        print("Problem somewhere, variablity low")
    else:
        print("No problem detected")


# In[73]:


for group_data, group_name in zip([switch_gain_group1, switch_loss_group1, switch_gain_group2, switch_loss_group2],
                                  ["Group 1 Gain", "Group 1 Loss", "Group 2 Gain", "Group 2 Loss"]):
    diagnostic_check(group_data, group_name)


# In[73]:

# In[74]:



def perform_stat_tests(switch_gain_group1, switch_loss_group1, switch_gain_group2, switch_loss_group2):
    # Normality check using Shapiro-Wilk test
    p_gain_group1 = shapiro(switch_gain_group1).pvalue
    p_loss_group1 = shapiro(switch_loss_group1).pvalue
    p_gain_group2 = shapiro(switch_gain_group2).pvalue
    p_loss_group2 = shapiro(switch_loss_group2).pvalue

    print("Shapiro-Wilk test for normality:")
    print(f"Group 1 Gain trials: p-value = {p_gain_group1}")
    print(f"Group 1 Loss trials: p-value = {p_loss_group1}")
    print(f"Group 2 Gain trials: p-value = {p_gain_group2}")
    print(f"Group 2 Loss trials: p-value = {p_loss_group2}\n")

    def use_parametric_test(p_values):

        return all(p > 0.05 for p in p_values)

    # Between groups comparison
    if use_parametric_test([p_gain_group1, p_gain_group2]):
        # Parametric test (t-test) for gain trials between groups
        t_gain, p_gain = ttest_ind(switch_gain_group1, switch_gain_group2)
```

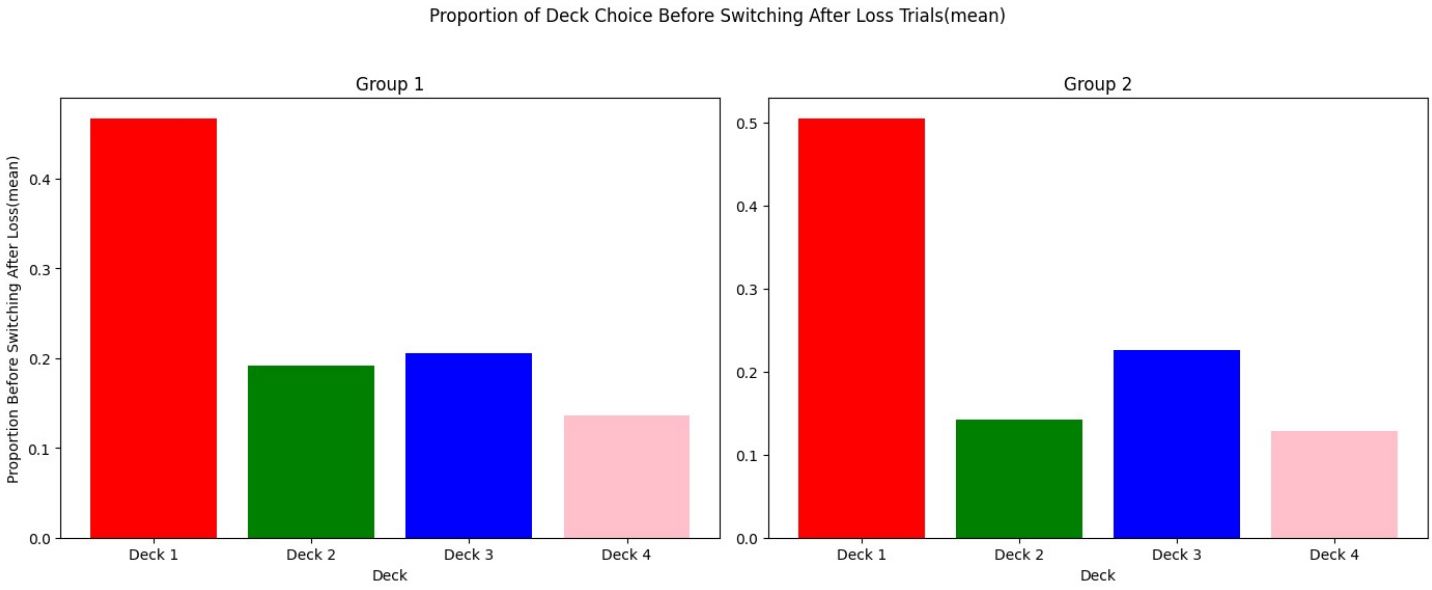
```
        print(f"Gain trials between groups (t-test): t-statistic = {t_gain}, p-value = {p_gain}")
    else:
        # else use non parametric – man whitney U
        u_gain, p_gain = mannwhitneyu(switch_gain_group1, switch_gain_group2, alternative='two-sided')
        print(f"Gain trials between groups (Mann-Whitney U): U-statistic = {u_gain}, p-value = {p_gain}")

    if use_parametric_test([p_loss_group1, p_loss_group2]):
        # Parametric test (t-test) for loss trials between groups
        t_loss, p_loss = ttest_ind(switch_loss_group1, switch_loss_group2)
        print(f"Loss trials between groups (t-test): t-statistic = {t_loss}, p-value = {p_loss}\n")
    else:
        u_loss, p_loss = mannwhitneyu(switch_loss_group1, switch_loss_group2, alternative='two-sided')
        print(f"Loss trials between groups (Mann-Whitney U): U-statistic = {u_loss}, p-value = {p_loss}\n")

    # Within group comparisons (Gain vs Loss)
    if use_parametric_test([p_gain_group1, p_loss_group1]):
        t_group1, p_group1 = ttest_rel(switch_gain_group1, switch_loss_group1)
        print(f"Group 1 (Gain vs Loss, t-test): t-statistic = {t_group1}, p-value = {p_group1}")
    else:
        w_group1, p_group1 = wilcoxon(switch_gain_group1, switch_loss_group1)
        print(f"Group 1 (Gain vs Loss, Wilcoxon): W-statistic = {w_group1}, p-value = {p_group1}")

    if use_parametric_test([p_gain_group2, p_loss_group2]):
        t_group2, p_group2 = ttest_rel(switch_gain_group2, switch_loss_group2)
        print(f"Group 2 (Gain vs Loss, t-test): t-statistic = {t_group2}, p-value = {p_group2}")
    else:
        w_group2, p_group2 = wilcoxon(switch_gain_group2, switch_loss_group2)
        print(f"Group 2 (Gain vs Loss, Wilcoxon): W-statistic = {w_group2}, p-value = {p_group2}")

perform_stat_tests(switch_gain_group1, switch_loss_group1, switch_gain_group2, switch_loss_group2)


# In[74]:


# In[75]:


# calculate proportions before and after loss trials
def calculate_deck_proportions(choice, loss,win):
    participants = choice.shape[0]
    trials = choice.shape[1]
    deck_choices_before_loss = {1: 0, 2: 0, 3: 0, 4: 0}
    deck_choices_after_loss = {1: 0, 2: 0, 3: 0, 4: 0}

    for i in range(participants):
        for t in range(1, trials):
```

```
            if win.iloc[i,t-1] + loss.iloc[i, t - 1] < 0 and choice.iloc[i,t] != choice.iloc[i,t-1]:  # Loss trial occurred
                deck_choices_after_loss[choice.iloc[i, t]] += 1
                deck_choices_before_loss[choice.iloc[i, t - 1]] += 1

                # print("participant: ",i+1)
                # print("trial :", t)
                # print("choice after loss : ", choice.iloc[i,t])
                # print("choice before loss : ", choice.iloc[i,t-1])


    total_before = sum(deck_choices_before_loss.values())
    total_after = sum(deck_choices_after_loss.values())

    proportions_before = {deck: count / total_before for deck, count in deck_choices_before_loss.items()}
    proportions_after = {deck: count / total_after for deck, count in deck_choices_after_loss.items()}

    return proportions_before, proportions_after



proportions_before_group1, proportions_after_group1 = calculate_deck_proportions(choice['group1'], loss['group1'], win['group2'])
proportions_before_group2, proportions_after_group2 = calculate_deck_proportions(choice['group2'], loss['group2'], win['group2'])


# In[76]:


def print_deck_analysis(proportions_before, proportions_after, group_name):
    print(f"{group_name} deck rankings (before loss):")
    for rank, (deck, proportion) in enumerate(sorted(proportions_before.items(), key=lambda x: x[1], reverse=True), 1):
        print(f"Rank {rank}: Deck {deck} (proportion: {round(proportion,3)})")
    print()
    print(f"{group_name} deck rankings (after loss):")
    for rank, (deck, proportion) in enumerate(sorted(proportions_after.items(), key=lambda x: x[1], reverse=True), 1):
        print(f"Rank {rank}: Deck {deck} (proportion: {round(proportion,3)})")
    print()

print("Analysis of deck choices BEFORE and AFTER loss trials:")
print_deck_analysis(proportions_before_group1, proportions_after_group1, "Group 1")
print_deck_analysis(proportions_before_group2, proportions_after_group2, "Group 2")
```

B) For each group, determine the deck chosen by each participant immediately before switching decks after encountering a loss trial. Subsequently, calculate the proportion of each deck chosen relative to the total number of loss trials for each participant. Create a larger plot with two subplots—one subplot representing each group. Plot the mean proportion as a bar diagram and the standard error of the mean for each of the four deck choices during loss trials. Rank the decks in decreasing order based on their mean proportions for each group. [4+ 1 marks]

Answer:

Figure:



Captions for Figure:

Deck choice preferences before and after loss trials for Groups 1 and 2 reveal a clear post loss shift in behavior. Before loss trials, Deck 1 was the most preferred choice for both groups. However,after experiencing losses, participants in both groups demonstrated a significant preference for Deck 2, thereby showing a strategy adjustment to potentially reduce future losses.

Statistical Tests used and Findings:

(mention of statistical tests has been made in the beginning of the doc only)

Statistical Observations:

Group 1 Preferences:

- Before Loss: Deck 1 (52.4%) was most preferred.
- After Loss: Deck 2 (37.5%) became the dominant choice.

Group 2 Preferences:

- Before Loss: Deck 1 (50.4%) was most preferred.
- After Loss: Deck 2 (40.3%) became the dominant choice.

Observations from these conclusions is :
Participants adjusted their strategies after loss trials, shifting their preference to Deck 2, which suggests a response aimed at exploring safer options. This change implies a loss driven adaptation mechanism where individuals alter their decision making to optimize outcomes after negative experiences.

****

Ques 2B

In[77]:

```
# Function to determine the deck chosen before switching after a loss trial
def calculate_deck_before_switch(choice, loss, win):
    participants = choice.shape[0]
    trials = choice.shape[1]
    deck_before_switch = {1: 0, 2: 0, 3: 0, 4: 0}
    total_switches = 0

    for i in range(participants):
        for t in range(trials - 1):
            if win.iloc[i,t] + loss.iloc[i, t] < 0 and choice.iloc[i, t] != choice.iloc[i, t + 1]:  # Loss trial and switch occurred
                deck_before_switch[choice.iloc[i, t]] += 1
                total_switches += 1

    proportions_before_switch = {deck: count / total_switches for deck, count in deck_before_switch.items()}
    return proportions_before_switch

# Calculate deck choice proportions before switching after loss trials for both groups
proportions_before_switch_group1 = calculate_deck_before_switch(choice['group1'], loss['group1'], win['group1'])
proportions_before_switch_group2 = calculate_deck_before_switch(choice['group2'], loss['group2'], win['group2'])
```

In[78]:

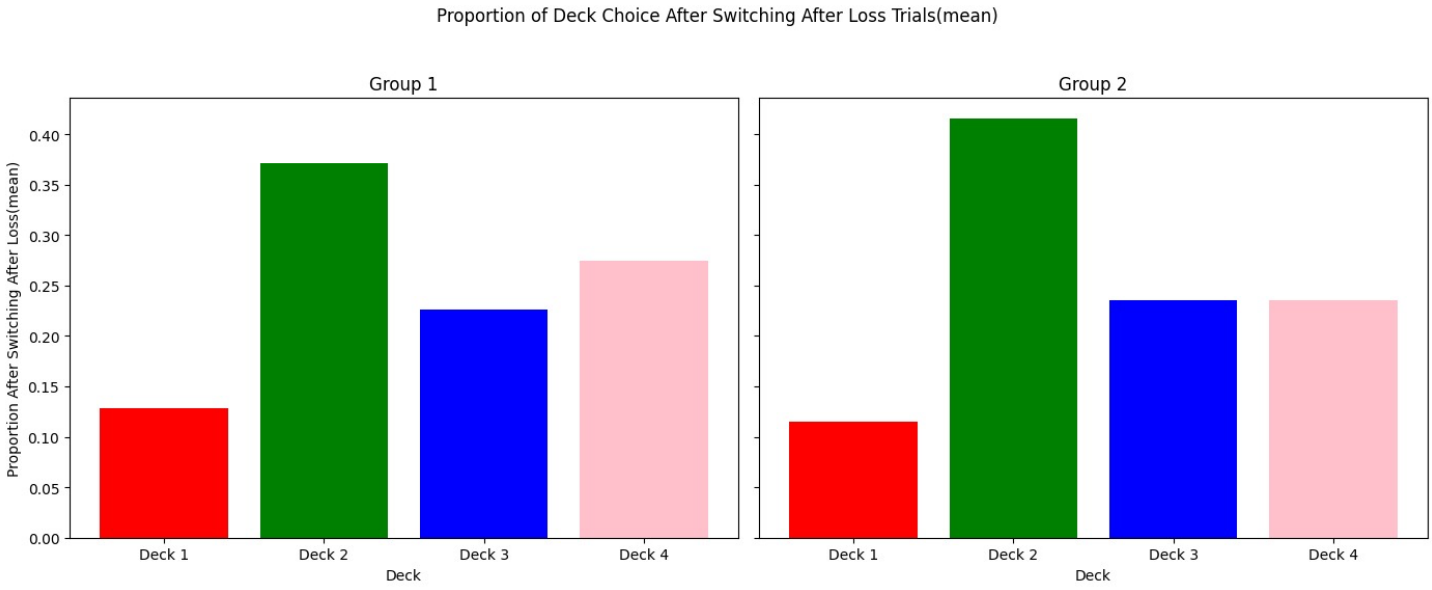
```
fig, axes = plt.subplots(1, 2, figsize=(14, 6), sharey=True)
```

```
# Group 1 plot
axes[0].bar(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], list(proportions_before_switch_group1.values()), capsize=5, color=['blue', 'green', 'orange', 'purple'])
axes[0].set_title('Group 1')
axes[0].set_xlabel('Deck')
axes[0].set_ylabel('Mean Proportion Before Switching After Loss')

# Group 2 plot
axes[1].bar(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], list(proportions_before_switch_group2.values()), capsize=5, color=['blue', 'green', 'orange', 'purple'])
axes[1].set_title('Group 2')
axes[1].set_xlabel('Deck')

plt.suptitle('Mean Proportion of Deck Choice Before Switching After Loss Trials')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

ranking_group1 = sorted(zip(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], proportions_before_switch_group1.values()), key=lambda x: x[1], reverse=True)
ranking_group2 = sorted(zip(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], proportions_before_switch_group2.values()), key=lambda x: x[1], reverse=True)


# In[79]:


print("Ranking of decks for Group 1 (based on mean proportions before switching after loss):")
for rank, (deck, proportion) in enumerate(ranking_group1, start=1):
    print(f"Rank {rank}: {deck} (Proportion: {proportion:.3f})")

print("\nRanking of decks for Group 2 (based on mean proportions before switching after loss):")
for rank, (deck, proportion) in enumerate(ranking_group2, start=1):
    print(f"Rank {rank}: {deck} (Proportion: {proportion:.3f})")
```

C) For each group, determine the deck switched to by each participant immediately after encountering a loss trial. Subsequently, calculate the proportion of each deck chosen relative to the total number of loss trials for each participant. Create a larger plot with two subplots—one subplot representing each group. Plot the mean proportion as a bar diagram and the standard error of the mean for each of the four deck choices during loss trials. Rank the decks in decreasing order based on their mean proportions for each group. [4+1 marks]

Answer:

Figure:



Captions for Figure:

These trials highlights Deck 1 as the most frequently chosen deck for both groups, even before participants decided to switch to another deck. This behavior indicates that participants initially persisted with their preferred deck, potentially reflecting a hesitancy to abandon their primary strategy until switching became necessary.

Statistical Tests used and Findings:

Statistical Observations:

Group 1:

- Deck 1 (46.6%) was most frequently chosen before switching.
- Deck 3 (20.6%) followed as the next most frequent choice.

Group 2:

- Deck 1 (50.4%) dominated before switching.
- Deck 3 (22.5%) ranked second.

We can conclude from this that despite losses, participants demonstrated an inclination to stick with Deck 1 before deciding to switch. This behavior reflects an initial confidence or reluctance to abandon familiar strategies. The eventual switching behavior suggests that participants reassessed their choices based on continued negative feedback, emphasizing the role of trial-and-error learning in dynamic decision-making processes.

****

Ques 2C

```
# In[80]:


# Function to determine the deck chosen after switching after a loss trial
def calculate_deck_after_switch(choice, loss, win):
    participants = choice.shape[0]
    trials = choice.shape[1]
    deck_after_switch = {1: 0, 2: 0, 3: 0, 4: 0}
    total_switches = 0

    for i in range(participants):
        for t in range(trials - 1):
            if win.iloc[i,t] + loss.iloc[i, t] < 0 and choice.iloc[i, t] != choice.iloc[i, t + 1]:  # Loss trial and switch
occurred
                deck_after_switch[choice.iloc[i, t + 1]] += 1
                total_switches += 1

    proportions_after_switch = {deck: count / total_switches for deck, count in deck_after_switch.items()}
    return proportions_after_switch

# Calculate deck choice proportions after switching after loss trials for both groups
proportions_after_switch_group1 = calculate_deck_after_switch(choice['group1'], loss['group1'],
win['group1'])
proportions_after_switch_group2 = calculate_deck_after_switch(choice['group2'], loss['group2'],
win['group1'])


# In[81]:


# Plotting the bar plots for deck choice after switching after loss trials
fig, axes = plt.subplots(1, 2, figsize=(14, 6), sharey=True)

# Group 1 plot
axes[0].bar(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], list(proportions_after_switch_group1.values()),
capsize=5, color=['blue', 'green', 'orange', 'purple'])
axes[0].set_title('Group 1')
axes[0].set_xlabel('Deck')
axes[0].set_ylabel('Mean Proportion After Switching After Loss')

# Group 2 plot
axes[1].bar(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], list(proportions_after_switch_group2.values()),
capsize=5, color=['blue', 'green', 'orange', 'purple'])
axes[1].set_title('Group 2')
axes[1].set_xlabel('Deck')

plt.suptitle('Mean Proportion of Deck Choice After Switching After Loss Trials')
```

```
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()



# In[82]:


ranking_after_switch_group1 = sorted(zip(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], proportions_after_switch_group1.values()), key=lambda x: x[1], reverse=True)
ranking_after_switch_group2 = sorted(zip(['Deck 1', 'Deck 2', 'Deck 3', 'Deck 4'], proportions_after_switch_group2.values()), key=lambda x: x[1], reverse=True)

print("Ranking of decks for Group 1 (based on mean proportions after switching after loss):")
for rank, (deck, proportion) in enumerate(ranking_after_switch_group1, start=1):
    print(f"Rank {rank}: {deck} (Proportion: {proportion:.3f})")

print("\nRanking of decks for Group 2 (based on mean proportions after switching after loss):")
for rank, (deck, proportion) in enumerate(ranking_after_switch_group2, start=1):
    print(f"Rank {rank}: {deck} (Proportion: {proportion:.3f})")


# In[82]:
```