import pandas as pd

import numpy as np

from collections import defaultdict

import ast

def load\_and\_process\_data(data, freq, row\_index):

"""Process response and sound timing data for a given frequency"""

# Handle special case for 6.5 and 7.5 Hz

freq\_str = str(freq).replace('.', '\_') if isinstance(freq, float) else str(freq)

resp\_started = data[f'resp\_{freq\_str}Hz.started'].dropna().iloc[0]

resp\_original = ast.literal\_eval(data.loc[row\_index, f'resp\_{freq\_str}Hz.rt'])

resp\_times = resp\_original - resp\_started

sound\_times\_raw = data.loc[row\_index, f'sound{freq\_str}Hz\_play\_times']

sound\_times = ast.literal\_eval(sound\_times\_raw)[:-1]

return resp\_times, sound\_times

def assign\_responses\_to\_sounds(responses, sound\_times):

"""Assign responses to their corresponding sounds"""

assigned\_responses = {sound\_time: [] for sound\_time in sound\_times}

current\_sound\_index = 0

for rt in responses:

while (current\_sound\_index < len(sound\_times) - 1) and (rt >= sound\_times[current\_sound\_index + 1]):

current\_sound\_index += 1

assigned\_responses[sound\_times[current\_sound\_index]].append(rt)

# Redistribute responses if needed

sound\_times\_list = list(assigned\_responses.keys())

for i in range(1, len(sound\_times\_list)):

prev\_sound = sound\_times\_list[i - 1]

current\_sound = sound\_times\_list[i]

if not assigned\_responses[prev\_sound] and len(assigned\_responses[current\_sound]) > 1:

assigned\_responses[prev\_sound].append(assigned\_responses[current\_sound].pop(0))

return assigned\_responses

def create\_response\_dataframe(assigned\_responses, sound\_times, freq):

"""Create a DataFrame with response analysis"""

freq\_str = str(freq).replace('.', '\_') if isinstance(freq, float) else str(freq)

table\_data = {

f"sound\_play\_times\_{freq\_str}Hz": [],

f"first\_response\_{freq\_str}Hz": [],

f"reaction\_time\_{freq\_str}Hz": [],

f"additional\_responses\_{freq\_str}Hz": [],

f"accuracy\_{freq\_str}Hz": []

}

accuracy\_sum = reaction\_time\_sum = reaction\_time\_count = 0

for sound\_time in sound\_times:

responses = assigned\_responses[sound\_time]

table\_data[f"sound\_play\_times\_{freq\_str}Hz"].append(round(sound\_time, 2))

if responses:

first\_response = round(responses[0], 2)

reaction\_time = round(first\_response - sound\_time, 2)

accuracy = 0.5 if len(responses) > 1 else 1

table\_data[f"first\_response\_{freq\_str}Hz"].append(first\_response)

table\_data[f"reaction\_time\_{freq\_str}Hz"].append(reaction\_time)

table\_data[f"accuracy\_{freq\_str}Hz"].append(accuracy)

table\_data[f"additional\_responses\_{freq\_str}Hz"].append([round(r, 2) for r in responses[1:]] if len(responses) > 1 else "None")

reaction\_time\_sum += reaction\_time

reaction\_time\_count += 1

accuracy\_sum += accuracy

else:

table\_data[f"first\_response\_{freq\_str}Hz"].append("None")

table\_data[f"reaction\_time\_{freq\_str}Hz"].append("N/A")

table\_data[f"accuracy\_{freq\_str}Hz"].append(0)

table\_data[f"additional\_responses\_{freq\_str}Hz"].append("None")

df = pd.DataFrame(table\_data)

total\_accuracy = round((accuracy\_sum / len(sound\_times)) \* 100, 2)

avg\_reaction\_time = round((reaction\_time\_sum / reaction\_time\_count), 2) if reaction\_time\_count > 0 else "N/A"

return df, total\_accuracy, avg\_reaction\_time

def analyze\_frequency(data, freq, row\_index):

"""Analyze responses for a single frequency"""

resp\_times, sound\_times = load\_and\_process\_data(data, freq, row\_index)

assigned\_responses = assign\_responses\_to\_sounds(resp\_times, sound\_times)

df, accuracy, reaction\_time = create\_response\_dataframe(assigned\_responses, sound\_times, freq)

freq\_str = str(freq).replace('.', '\_') if isinstance(freq, float) else str(freq)

print(f"\nHz{freq\_str}")

print(df)

print(f"\nTotal Accuracy\_{freq\_str}Hz: {accuracy}%")

print(f"Average Reaction Time\_{freq\_str}Hz: {reaction\_time}\n")

return df, accuracy, reaction\_time

# Main execution

file\_path = r'C:\Users\12035\Desktop\1. phd\24 motor task\data\0100.csv'

data = pd.read\_csv(file\_path)

# Dictionary mapping frequencies to their row indices

freq\_indices = {

1: 3, # 1Hz data

2: 7, # 2Hz data

3: 11, # 3Hz data

4: 15, # 4Hz data

5: 19, # 5Hz data

6: 23, # 6Hz data

6.5: 27, # 6.5Hz data

7: 31, # 7Hz data

7.5: 35, # 7.5Hz data

8: 39 # 8Hz data

}

# Analyze each frequency

for freq, index in freq\_indices.items():

analyze\_frequency(data, freq, index)