import openai  
import csv  
import pandas as pd  
import json  
import re  
import matplotlib.pyplot as plt  
import seaborn as sns  
import numpy as np  
API\_KEY = ""   
openai.api\_key = API\_KEY

## Scenarios & Prompts   
  
##forest  
gain\_forest = """Scenario: A vast forest area is under severe threat from deforestation due to urban growth and logging activities, potentially resulting in the removal of 900 trees from a grove of ancient oak trees in the forest. Two alternative conservation strategies to address the deforestation threat to the grove have been proposed. Imagine you are on the conservation committee tasked with deciding on a proposal. Assume that the estimates are as follows:\n- Option A: If Proposal A is adopted, 300 of the trees will be protected.\n- Option B: If Proposal B is adopted, there is a 1/3 probability that all 900 of the trees will be protected, and 2/3 probability that no trees will be protected."""  
loss\_forest = """Scenario: A vast forest area is under severe threat from deforestation due to urban growth and logging activities, potentially resulting in the removal of 900 trees from a grove of ancient oak trees in the forest. Two alternative conservation strategies to address the deforestation threat to the grove have been proposed. Imagine you are on the conservation committee tasked with deciding on a proposal. Assume that the estimates are as follows:\n- Option A: If Proposal A is adopted, 600 of the trees will be cleared.\n- Option B: If Proposal B is adopted, there is a 1/3 probability that none of the trees will be cleared, and 2/3 probability that all 900 of the trees will be cleared."""  
  
##human  
gain\_human = """Scenario: A city is facing significant air quality issues due to high levels of pollution, which could lead to 900 premature deaths over the next year due to respiratory illness. Two alternative plans to improve the air quality and reduce pollution have been proposed. Imagine you are a city official on the committee tasked with deciding on a proposal. Assume that the estimates are as follows:\n- Option A: If Proposal A is adopted, 300 people will survive.\n- Option B: If Proposal B is adopted, there is a 1/3 probability that all 900 people will survive, and 2/3 probability that no one will survive."""  
loss\_human = """Scenario: A city is facing significant air quality issues due to high levels of pollution, which could lead to 900 premature deaths over the next year due to respiratory illness. Two alternative plans to improve the air quality and reduce pollution have been proposed. Imagine you are a city official on the committee tasked with deciding on a proposal. Assume that the estimates are as follows:\n- Option A: If Program A is adopted, 600 people will die.\n- Option B: If Program B is adopted, there is a 1/3 probability that no one will die, and 2/3 probability that all 900 people will die."""  
  
#animal

gain\_animal = """Scenario: An oil spill off the coast of a country is anticipated to cause significant environmental damage, and is projected to endanger the lives of 900 marine animals, including sea turtles, seals, and various types of seabirds. Two alternative clean-up efforts to address the oil spill have been proposed. Imagine you are on the clean-up committee tasked with deciding on a proposal. Assume that the estimates are as follows:\n- Option A: If Proposal A is adopted, 300 of the marine animals will survive.\n- Option B: If Proposal B is adopted, there is a 1/3 probability that all 900 of the marine animals will survive, and 2/3 probability that no marine animals will survive."""  
loss\_animal = """Scenario: An oil spill off the coast of a country is anticipated to cause significant environmental damage, and is projected to endanger the lives of 900 marine animals, including sea turtles, seals, and various types of seabirds. Two alternative clean-up efforts to address the oil spill have been proposed. Imagine you are on the clean-up committee tasked with deciding on a proposal. Assume that the estimates are as follows:\n- Option A: If Proposal A is adopted, 600 of the marine animals will die.\n- Option B: If Proposal B is adopted, there is a 1/3 probability that no marine animals will die, and 2/3 probability that all 900 of the marine animals will die."""  
  
  
##################################### Comboos for Experiment   
  
####### System Role #######   
  
###Combo 1: Baseline Condition   
 # participant   
 # capable of decisions  
 # basic instructions -- user messsage   
system\_message\_baseline1 = "You are a participant in a survey. You are capable of making decisions and choices. Provide your response as a JSON object."  
user\_message\_baseline1 = """You are participating in a survey-based study.\nInstructions: read the scenario below and then answer the questions.\n{}\n\nQuestions:\n1. Please choose an option:\n- Proposal A\n- Proposal B\n2. On a scale of 1 (Strong Preference for Proposal A) to 7 (Strong Preference for Proposal B), rate your preference for the option selected:\n- 1: Strong Preference for Proposal A\n- 2: Preference for Proposal A\n- 3: Slight Preference for Proposal A\n- 4: No Preference for Proposal A or B\n- 5: Slight Preference for Proposal B\n- 6: Preference for Proposal B\n- 7: Strong Preference for Proposal B\n3. Briefly describe the rationale for your preference rating:\n\nProvide your response as a JSON object. As an example: '{{"Q1\_Response": "Your Proposal Choice", "Q2\_Response": "Your Rating", "Q3\_Response": "Your Rationale."}}'.\n\nResponse:"""  
  
###Combo 2: Human Participant   
 # (+) human participant -- system messsage   
 # capable of decisions -- system messsage   
 # single JSON -- system message  
 # basic instructions -- user messsage   
 # Task Order Instructions via one JSON object -- user messsage   
system\_message\_hum = "You are a human participant in a survey. You are capable of making decisions and choices. Provide your response as a JSON object."  
user\_message = user\_message\_baseline1  
  
###Combo 3: Adds in Risk Preferences   
 # human participant -- system messsage   
 # capable of decisions -- system messsage   
 # (+) Consider human risk preferenes -- system messsage  
 # single JSON -- system message  
 # basic instructions -- user messsage   
 # Task Order Instructions via one JSON object -- user messsage   
system\_message\_risk = "You are a human participant in a survey. You are capable of making decisions and choices. You are responding to a survey with the same risk preferences as a human participant would. Provide your response as a JSON object."   
user\_message = user\_message\_baseline1  
  
  
###Combo 3B: Adds in Risk Preferences to user message  
 # human participant -- system messsage   
 # capable of decisions -- system messsage   
 # Consider human risk preferenes -- system messsage  
 # single JSON -- system message  
 # basic instructions -- user messsage   
 # Task Order Instructions via one JSON object -- user messsage   
 # (+) Consider human risk preferenes -- user messsage  
  
system\_message = system\_message\_risk  
user\_message\_risk = """You are participating in a survey-based study. Respond with the same risk preferences as a human participant would.\nInstructions: read the scenario below and then answer the questions.\n{}\n\nQuestions:\n1. Please choose an option:\n- Proposal A\n- Proposal B\n2. On a scale of 1 (Strong Preference for Proposal A) to 7 (Strong Preference for Proposal B), rate your preference for the option selected:\n- 1: Strong Preference for Proposal A\n- 2: Preference for Proposal A\n- 3: Slight Preference for Proposal A\n- 4: No Preference for Proposal A or B\n- 5: Slight Preference for Proposal B\n- 6: Preference for Proposal B\n- 7: Strong Preference for Proposal B\n3. Briefly describe the rationale for your preference rating:\n\nProvide your response as a JSON object. As an example: '{{"Q1\_Response": "Your Proposal Choice", "Q2\_Response": "Your Rating", "Q3\_Response": "Your Rationale."}}'.\n\nResponse:"""  
reference for Proposal B\n- 6: Preference for Proposal B\n- 7: Strong Preference for Proposal B\n\nProvide your response as a JSON object. As an example: '{{"Q1\_Response": "Your Proposal Choice", "Q2\_Response": "Your Rating"}}'.\n\nResponse:"""

## Define Functions & Model Call  
## Create functions to make prompt scenario dynamics   
def format\_scenario(scenario\_text):  
 return scenario\_text  
  
def format\_prompt(prompt\_text, scenario):  
 return prompt\_text.format(scenario)  
  
##define scenario and user\_message variables   
  
def generate\_response\_turbo(sys\_prompt, user\_prompt, temperature, top\_p):  
 messages=[{"role": "system", "content" : sys\_prompt },  
 {"role": "user", "content" : user\_prompt }]  
 response = openai.ChatCompletion.create(  
 #model="gpt-3.5-turbo",  
 model="gpt-3.5-turbo",  
  
 messages = messages,  
 temperature=temperature,  
 max\_tokens=200,  
 n=1,   
 top\_p=top\_p  
 )  
 prompt = sys\_prompt + user\_prompt  
 return [(prompt, response.choices[0].message['content'], temperature, top\_p) for choice in response.choices ]

######################## Call Model for Each Scenario   
######## Humans ########  
  
## Human Loss   
##humans\_loss 37  
  
human\_loss\_turbo = []  
  
for i in range(37):  
 formatted\_scenario = format\_scenario(loss\_human) ##change for each run   
 user\_message = format\_prompt(user\_message\_baseline1,   
 formatted\_scenario) ##change for each run   
 human\_loss\_output = generate\_response\_turbo(system\_message\_baseline1,   
 user\_message, 1, 1) ##change for each run   
 prompt, response, temperature, top\_p = human\_loss\_output[0]   
 frame = 'loss\_human' ##change for each run   
 model = 'gpt-3.5-turbo'  
 test = ['combo 1A', 'system\_message\_baseline1', 'user\_message\_baseline1', 'system role']  
 num\_calls = 25 ## matches to qual exp.   
 human\_loss\_turbo.append({'prompt': prompt, 'response': response, 'temperature': temperature,'top\_p': top\_p, 'test': test, 'model': model, 'frame': frame, 'num\_calls': num\_calls})  
  
## Human Gain   
##humans\_gain 30  
  
human\_gain\_turbo = []  
  
for i in range(30):  
 formatted\_scenario = format\_scenario(gain\_human) ##change for each run   
 user\_message = format\_prompt(user\_message\_baseline1,   
 formatted\_scenario) ##change for each run   
 human\_gain\_output = generate\_response\_turbo(system\_message\_baseline1,   
 user\_message, 1, 1) ##change for each run   
 prompt, response, temperature, top\_p = human\_gain\_output[0]   
 frame = 'gain\_human' ##change for each run   
 model = 'gpt-3.5-turbo'  
 test = ['combo 1A', 'system\_message\_baseline1', 'user\_message\_baseline1', 'system role']  
 num\_calls = 20 ## matches to qual exp.   
   
 human\_gain\_turbo.append({'prompt': prompt, 'response': response, 'temperature': temperature,'top\_p': top\_p, 'test': test, 'model': model, 'frame': frame, 'num\_calls': num\_calls})  
  
######## Animals ########  
  
## Animal Loss   
##animals\_loss 37  
animal\_loss\_turbo = []  
for i in range(37):  
 formatted\_scenario = format\_scenario(loss\_animal)   
 user\_message = format\_prompt(user\_message\_baseline1, ##change for each run   
 formatted\_scenario)   
 animal\_loss\_output = generate\_response\_turbo(system\_message\_baseline1, ##change for each run   
 user\_message, 1, 1) ##change for each run   
 prompt, response, temperature, top\_p = animal\_loss\_output[0]   
 frame = 'loss\_animal'   
 model = 'gpt-3.5-turbo'  
 ##test = ['combo 2', 'system\_message\_hum', 'user\_message\_baseline1', 'human']   
  
 test = ['combo 1A', 'system\_message\_baseline1', 'user\_message\_baseline1', 'system role']  
 num\_calls = 10 ## matches to qual exp.   
 animal\_loss\_turbo.append({'prompt': prompt, 'response': response, 'temperature': temperature,'top\_p': top\_p, 'test': test, 'model': model, 'frame': frame, 'num\_calls': num\_calls})  
  
## Animal Gain   
##animals\_gain 34   
  
animal\_gain\_turbo = []  
for i in range(34):  
 formatted\_scenario = format\_scenario(gain\_animal)   
 user\_message = format\_prompt(user\_message\_baseline1, ##change for each run   
 formatted\_scenario)   
 animal\_gain\_output = generate\_response\_turbo(system\_message\_baseline1, ##change for each run   
 user\_message, 1, 1) ##change for each run   
 prompt, response, temperature, top\_p = animal\_gain\_output[0]   
 frame = 'gain\_animal'   
 model = 'gpt-3.5-turbo'  
 test = ['combo 1A', 'system\_message\_baseline1', 'user\_message\_baseline1', 'system role']  
 num\_calls = 34 ## matches to qual exp.   
 animal\_gain\_turbo.append({'prompt': prompt, 'response': response, 'temperature': temperature,'top\_p': top\_p, 'test': test, 'model': model, 'frame': frame, 'num\_calls': num\_calls})  
  
  
## Forest Gain  
##forest\_gain 35  
forest\_gain\_turbo = []  
for i in range(35):  
 formatted\_scenario = format\_scenario(gain\_forest)   
 user\_message = format\_prompt(user\_message\_baseline1, ##change for each run   
 formatted\_scenario)   
 forest\_gain\_output = generate\_response\_turbo(system\_message\_baseline1, ##change for each run   
 user\_message, 1, 1) ##change for each run   
 prompt, response, temperature, top\_p = forest\_gain\_output[0]   
 frame = 'gain\_forest'   
 model = 'gpt-3.5-turbo'  
 test = ['combo 1A', 'system\_message\_baseline1', 'user\_message\_baseline1', 'system role']  
 num\_calls = 0 ## matches to qual exp.   
 forest\_gain\_turbo.append({'prompt': prompt, 'response': response, 'temperature': temperature,'top\_p': top\_p, 'test': test, 'model': model, 'frame': frame, 'num\_calls': num\_calls})  
  
## Forest Loss  
##forest\_loss 28  
forest\_loss\_turbo = []  
for i in range(28):  
 formatted\_scenario = format\_scenario(loss\_forest)   
 user\_message = format\_prompt(user\_message\_baseline1, ##change for each run   
 formatted\_scenario)   
 forest\_loss\_output = generate\_response\_turbo(system\_message\_baseline1, ##change for each run   
 user\_message, 1, 1) ##change for each run   
 prompt, response, temperature, top\_p = forest\_loss\_output[0]   
 frame = 'loss\_forest'   
 model = 'gpt-3.5-turbo'  
 test = ['combo 1A', 'system\_message\_baseline1', 'user\_message\_baseline1', 'system role']  
 num\_calls = 5 ## matches to qual exp.   
 forest\_loss\_turbo.append({'prompt': prompt, 'response': response, 'temperature': temperature,'top\_p': top\_p, 'test': test, 'model': model, 'frame': frame, 'num\_calls': num\_calls})

#Append CSV File for multi call model version   
csv\_file\_path = "/Users/annaking/Documents/Github/riskychoiceframing\_AI/Experiment 2/tests/variable\_prompt\_test\_turbo\_run.csv"  
  
with open(csv\_file\_path, 'a', newline='') as file:  
 fieldnames = ['prompt', 'response', 'temperature', 'top\_p', 'test', 'model', 'frame', 'num\_calls']  
 writer = csv.DictWriter(file, fieldnames=fieldnames)  
 for response\_data in human\_loss\_turbo:  
 writer.writerow({  
 'prompt': response\_data['prompt'],  
 'response': response\_data['response'],  
 'temperature': response\_data['temperature'],  
 'top\_p': response\_data['top\_p'],  
 'test': response\_data['test'],   
 'model': response\_data['model'],   
 'frame': response\_data['frame'],  
 'num\_calls': response\_data['num\_calls']  
  
 })  
  
 for response\_data in human\_gain\_turbo:  
 writer.writerow({  
 'prompt': response\_data['prompt'],  
 'response': response\_data['response'],  
 'temperature': response\_data['temperature'],  
 'top\_p': response\_data['top\_p'],  
 'test': response\_data['test'],   
 'model': response\_data['model'],   
 'frame': response\_data['frame'],  
 'num\_calls': response\_data['num\_calls']  
 })  
  
 for response\_data in animal\_gain\_turbo:  
 writer.writerow({  
 'prompt': response\_data['prompt'],  
 'response': response\_data['response'],  
 'temperature': response\_data['temperature'],  
 'top\_p': response\_data['top\_p'],  
 'test': response\_data['test'],   
 'model': response\_data['model'],   
 'frame': response\_data['frame'],  
 'num\_calls': response\_data['num\_calls']  
 })  
 for response\_data in animal\_loss\_turbo:  
 writer.writerow({  
 'prompt': response\_data['prompt'],  
 'response': response\_data['response'],  
 'temperature': response\_data['temperature'],  
 'top\_p': response\_data['top\_p'],  
 'test': response\_data['test'],   
 'model': response\_data['model'],   
 'frame': response\_data['frame'],  
 'num\_calls': response\_data['num\_calls']  
 })  
  
 for response\_data in forest\_gain\_turbo:  
 writer.writerow({  
 'prompt': response\_data['prompt'],  
 'response': response\_data['response'],  
 'temperature': response\_data['temperature'],  
 'top\_p': response\_data['top\_p'],  
 'test': response\_data['test'],   
 'model': response\_data['model'],   
 'frame': response\_data['frame'],  
 'num\_calls': response\_data['num\_calls']  
 })  
 for response\_data in forest\_loss\_turbo:  
 writer.writerow({  
 'prompt': response\_data['prompt'],  
 'response': response\_data['response'],  
 'temperature': response\_data['temperature'],  
 'top\_p': response\_data['top\_p'],  
 'test': response\_data['test'],   
 'model': response\_data['model'],   
 'frame': response\_data['frame'],  
 'num\_calls': response\_data['num\_calls']  
 })

### Analyze Results

##Load Data   
gpt3\_run = "/Users/annaking/Documents/Github/riskychoiceframing\_AI/Experiment 2/tests/variable\_prompt\_test\_turbo\_run.csv"  
df\_gpt3 = pd.read\_csv(gpt3\_run)

## Cleaning  
  
##scenario & frame

df\_gpt3[['frame', 'scenario']] = df\_gpt3['frame'].str.split('\_', n=1, expand=True)  
pattern2 = r'Scenario:(.\*?)Questions'  
df\_gpt3['scenario\_text'] = df\_gpt3['prompt'].str.extract(pattern2, flags=re.DOTALL)

## Clean json   
# import json   
def clean\_json(x):  
 x = x.strip("'") # remove leading/trailing single quotes  
 x = x.strip("\n") # remove leading/trailing newline characters  
 x = x.encode('utf-8', 'ignore').decode('utf-8') # ignore non utf-8 characters  
 x = re.sub(r'"\s\*"', '","', x) # replace spaces between quotes with commas  
 #replace incorrect JSON keys with correct keys  
 for key in ["Q1\_Response", "Q2\_Response", "Q3\_Response"]:  
 x = re.sub(f'(?<=[{{,])\s\*{key}\s\*(?=:)', f' "{key}"', x)  
 # make sure quotes   
 x = re.sub(r':\s\*([0-9]+)\s\*(?=[,}])', r': "\1"', x)  
  
 # cleaning steps  
 x = x.replace('Q1\_response','Q1\_Response')  
 x = x.replace('Q2\_response','Q2\_Response')  
 x = x.replace('Q3\_response','Q3\_Response')  
 x = x.replace('it"s', 'it\'s')  
 x = x.replace('B"s', "B's")  
 x = x.replace("}, {", ",")  
 x = x.replace("},{", ",")  
 x = x.replace(".}", '."}')  
 x = x.replace('"}" }', '"}')  
  
 for proposal in ["Proposal A", "Proposal B"]:  
 x = re.sub(f'(?<=:)\s\*{proposal}(?=\s\*[^"]\w\*,)', f' "{proposal}"', x)  
 x = re.sub(r'}\s\*{', ', ', x)  
 if not x.startswith('{'):  
 x = '{' + x  
 if not x.endswith('}'):  
 x = x + '}'  
 try:  
 x = x.replace('\n', ' ')  
 x = json.dumps(json.loads(x))  
 except json.JSONDecodeError:  
 return x  
 return x  
  
df\_gpt3['response'] = df\_gpt3['response'].apply(clean\_json)  
responses = df\_gpt3.apply(lambda x: pd.Series(json.loads(x['response'])), axis=1, result\_type='expand')  
df\_gpt3 = pd.concat([df\_gpt3, responses], axis = 1)  
df\_gpt3 = df\_gpt3.reset\_index()  
df\_gpt3['Q1\_Response'] = df\_gpt3['Q1\_Response'].apply(lambda x: 'Proposal B' if x == 'Option B' else x)  
df\_gpt3['Q1\_Response'] = df\_gpt3['Q1\_Response'].apply(lambda x: 'Proposal A' if x == 'Option A' else x)

## Save File for Analysis  
df\_gpt3.to\_csv('/Users/annaking/Documents/Github/riskychoiceframing\_AI/Experiment 1/analysis/df\_gpt3\_testing\_072623.csv')