In [ ]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import ttest_rel
from statsmodels.stats.multitest import multipletests
```

In [ ]:
```
N_SUBJECTS = 100  #This is the number of participants

N_PARCELS = 360   #This is the number of regions

TR = 0.72  # Time resolution, in seconds

HEMIS = ["Right", "Left"]

RUNS   = ['LR','RL']  #A list to access Left to Right data or Right to Left
N_RUNS = 2 #or len(RUNS)

EXPERIMENTS = {
    'MOTOR'      : {'cond':['lf','rf','lh','rh','t','cue']},
    'WM'         : {'cond':['0bk_body','0bk_faces','0bk_places','0bk_tools','2bk_bo
    'EMOTION'    : {'cond':['fear','neut']},
    'GAMBLING'   : {'cond':['loss','win']},
    'LANGUAGE'   : {'cond':['math','story']},
    'RELATIONAL' : {'cond':['match','relation']},
    'SOCIAL'     : {'cond':['ment','rnd']}
}  # The values that we might need to access for our data path
```

In [ ]:
```
subjects_ids = np.loadtxt(os.path.join(r'C:\Users\Asus\Desktop\Neuroscience Project
```

In [ ]:
```
def load_single_timeseries(subject, experiment, run, remove_mean=True):
  """Load timeseries data for a single subject and single run.

  Args:
    subject (str):      subject ID to load
    experiment (str):   Name of experiment
    run (int):          (0 or 1)
    remove_mean (bool): If True, subtract the parcel-wise mean (typically the mean

  Returns
    ts (n_parcel x n_timepoint array): Array of BOLD data values

  """
  HCP_DIR=r'C:\Users\Asus\Desktop\Neuroscience Project\Neuromatch\hcp_task'
  bold_run  = RUNS[run]
  bold_path = f"{HCP_DIR}/subjects/{subject}/{experiment}/tfMRI_{experiment}_{bold_
  bold_file = "data.npy"
  ts = np.load(f"{bold_path}/{bold_file}")
  if remove_mean:
    ts -= ts.mean(axis=1, keepdims=True)
  return ts
```

In [ ]:
```
sub_data=load_single_timeseries('100307','GAMBLING',0,remove_mean=False)
```

In [ ]:
```
def add_consecutive_numbers(original_list):

  expanded_list = []
  for num in original_list:
    expanded_list.append(num)
    for i in range(1, 5):
      expanded_list.append(num + i)
  return expanded_list
#-----------------------------------------------------------------------

def loss_idx(id,HCP_DIR):
    loss_path=f"{HCP_DIR}/subjects//{id}/GAMBLING/tfMRI_GAMBLING_LR/EVs/loss_event.
    loss=np.loadtxt(loss_path)
    idx_list=[]
    for i in loss[:,0]:
        idx_list.append(round(i/0.72))
    return add_consecutive_numbers(idx_list)
#-----------------------------------------------------------------------

def win_idx(id,HCP_DIR):
    win_path=loss_path=f"{HCP_DIR}/subjects//{id}/GAMBLING/tfMRI_GAMBLING_LR/EVs/wi
    win=np.loadtxt(win_path)
    idx_list=[]
    for i in win[:,0]:
        idx_list.append(round(i/0.72))
    return add_consecutive_numbers(idx_list)
```

In [ ]:
```
regions_abbrevations_idx={'L_POS1': 210,
 'R_POS1': 30,
 'L_POS2': 194,
 'R_POS2': 14,
 'L_H': 299,
 'R_H': 119,
 'L_IPS1': 196,
 'R_IPS1': 16,
 'L_a32pr': 358,
 'R_a32pr': 178,
 'L_a24pr': 238,
 'R_a24pr': 58,
 'L_33pr': 237,
 'R_33pr': 57,
 'L_OFC': 272,
 'R_OFC': 92,
 'L_47s': 273,
 'R_47s': 93,
 'L_47m': 245,
 'R_47m': 65,
 'L_a10p': 268,
 'R_a10p': 88,
 'L_p10p': 349,
 'R_p10p': 169,
 'L_10d': 251,
```

```
 'R_10d': 71,
 'L_10pp': 269,
 'R_10pp': 89,
 'L_11l': 270,
 'R_11l': 90,
 'L_23d': 211,
 'R_23d': 31,
 'L_31a': 341,
 'R_31a': 161,
 'L_31pd': 340,
 'R_31pd': 160,
 'L_31pv': 214,
 'R_31pv': 34,
 'L_PCV': 206,
 'R_PCV': 26,
 'L_MI': 288,
 'R_MI': 108,
 'L_Ig': 347,
 'R_Ig': 167,
 'L_PI': 357,
 'R_PI': 177,
 'L_44': 253,
 'R_44': 73,
 'L_45': 254,
 'R_45': 74,
 'L_47l': 255,
 'R_47l': 75,
 'L_a47r': 256,
 'R_a47r': 76,
 'L_p47r': 350,
 'R_p47r': 170,
 'L_46': 263,
 'R_46': 83,
 'L_TGd': 310,
 'R_TGd': 130,
 'L_TGv': 351,
 'R_TGv': 171,
 'L_V2': 183,
 'R_V2': 3,
 'L_V3': 184,
 'R_V3': 4,
 'L_PGi': 329,
 'R_PGi': 149,
 'L_PGp': 322,
 'R_PGp': 142,
 'L_PGs': 330,
 'R_PGs': 150}
regions_idx=list(regions_abbrevations_idx.values())
```

In [ ]:
```
regions_idx.sort()
```

In [ ]:
```
mean_win_data=np.zeros((72,100))
mean_loss_data=np.zeros((72,100))
```

In [ ]:
```
HCP_DIR=r'C:\Users\Asus\Desktop\Neuroscience Project\Neuromatch\hcp_task'
```

In [ ]:
```
for idi,id in enumerate(subjects_ids):
    data=load_single_timeseries(id,'GAMBLING',0,remove_mean=False)
    win_idx_list=win_idx(id,HCP_DIR)
    loss_idx_list=-loss_idx(id,HCP_DIR)
    for roii,roi in enumerate(regions_idx):
        arr=data[roi,:]
        result_win=[arr[i] for i in win_idx_list]
        result_loss=[arr[i] for i in loss_idx_list]
        mean_win_data[roii,idi]=np.mean(np.array(result_win))
        mean_loss_data[roii,idi]=np.mean(np.array(result_loss))
```

In [ ]:
```
p_values = np.zeros(72)
```

In [ ]:
```
for region in range(72):
    t_stat, p_val = ttest_rel(mean_win_data[region], mean_loss_data[region])
    p_values[region] = p_val
```

In [ ]:
```
reject, pvals_corrected, _, _ = multipletests(p_values, method='fdr_bh')
```

In [ ]:
```
significant_regions_corrected = np.where(reject)[0]
```

In [ ]:
```
print(f"Significant brain regions after correction: {significant_regions_corrected}
print(f"Corrected P-values: {pvals_corrected}")
```