In [270…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

import pickle as pkl
```

In [271…
```
with open('dense_rank_distance_matrix_experts_BSU1_BSU2_df.pkl', 'rb')
    dense_rank_distance_matrix_experts_BSU1_BSU2_df = pkl.load(f)

with open('kmeans_rank_distance_matrix_experts_BSU1_BSU2_df.pkl', 'rb'
    kmeans_rank_distance_matrix_experts_BSU1_BSU2_df = pkl.load(f)
```

In [272…
```
with open('dense_rank_linkage_matrix_experts_BSU1_BSU2.pkl', 'rb') as
    dense_rank_linkage_matrix_experts_BSU1_BSU2 = pkl.load(f)

with open('kmeans_rank_linkage_matrix_experts_BSU1_BSU2.pkl', 'rb') as
    kmeans_rank_linkage_matrix_experts_BSU1_BSU2 = pkl.load(f)
```

Cluster Assignments

Clusters for 30 trials

In [273…
```
# Perform hierarchical clustering on columns (trials/audio samples) fo
linkage_matrix_dense_audio = linkage(dense_rank_distance_matrix_expert
linkage_matrix_kmeans_audio = linkage(kmeans_rank_distance_matrix_expe

# Extract 5 clusters from both hierarchical trees
num_clusters_trials = 5
dense_audio_clusters = fcluster(linkage_matrix_dense_audio, num_cluste
kmeans_audio_clusters = fcluster(linkage_matrix_kmeans_audio, num_clus

# Create DataFrames mapping audio samples to their clusters
dense_audio_cluster_df = pd.DataFrame({
    'Audio Sample': dense_rank_distance_matrix_experts_BSU1_BSU2_df.co
    'Dense Cluster': dense_audio_clusters
})

kmeans_audio_cluster_df = pd.DataFrame({
    'Audio Sample': kmeans_rank_distance_matrix_experts_BSU1_BSU2_df.c
    'KMeans Cluster': kmeans_audio_clusters
})
```

```
# Merge to align the clusters from both methods
merged_audio_clusters = dense_audio_cluster_df.merge(kmeans_audio_clus

# Build a contingency table comparing the two clustering results
contingency_table_trials = pd.crosstab(merged_audio_clusters['Dense Cl

# Display the contingency table
contingency_table_trials
```

Out [273…

KMeans Cluster	1	2	3	4	5
Dense Cluster					
1	1	0	2	0	2
2	1	0	3	3	2
3	0	1	0	1	1
4	2	1	2	2	0
5	0	1	2	1	2

In [274…

```
# Initialize contingency table for trials/audio samples with lists
contingency_table_trials_specified = pd.DataFrame(
    [[[] for _ in range(5)] for _ in range(5)],  # Adjust for 5 cluste
    columns=[1, 2, 3, 4, 5],
    index=[1, 2, 3, 4, 5]
)
contingency_table_trials_specified.index.name = 'Dense Cluster'
contingency_table_trials_specified.columns.name = 'KMeans Cluster'

# Define a function to retrieve audio sample lists
def get_audio_samples(dense_label, kmeans_label):
    return merged_audio_clusters[
        (merged_audio_clusters['Dense Cluster'] == dense_label) &
        (merged_audio_clusters['KMeans Cluster'] == kmeans_label)
    ]['Audio Sample'].tolist()

# Populate the contingency table with actual audio sample lists
for dense in [1, 2, 3, 4, 5]:
    for kmeans in [1, 2, 3, 4, 5]:
        contingency_table_trials_specified.at[dense, kmeans] = get_aud
```

In [275…

```
contingency_table_trials_specified
```

Out [275…

KMeans Cluster	1	2	
Dense Cluster			
1	[PE_CreatureFromTheBlackjackTable]	[]	TI
2	[PE_AmateurOnPurpose]	[]	LP_Creat
3	[]	[PE_27_castanets]	
4	[SH_AmateurOnPurpose, SH_CreatureFromTheBlackj...	[DE_SitaSings_remix2_LD6]	[T
5	[]	[DE_SitaSings_remix1_LD0]	[DE_Cosr

In [276…
```
# export the contingency table to a csv file
contingency_table_trials_specified.to_csv('contingency_table_trials_sp
```

Clusters for 42 Subjects

In [277…
```
# Perform hierarchical clustering on rows (subjects) for both distance
linkage_matrix_dense_subjects = linkage(dense_rank_distance_matrix_exp
linkage_matrix_kmeans_subjects = linkage(kmeans_rank_distance_matrix_e

# Extract 3 clusters from both hierarchical trees
num_clusters_subjects = 3
dense_subject_clusters = fcluster(linkage_matrix_dense_subjects, num_c
kmeans_subject_clusters = fcluster(linkage_matrix_kmeans_subjects, num

# Create DataFrames mapping subjects to their clusters
dense_subject_cluster_df = pd.DataFrame({
    'Subject': dense_rank_distance_matrix_experts_BSU1_BSU2_df.index,
    'Dense Cluster': dense_subject_clusters
})

kmeans_subject_cluster_df = pd.DataFrame({
    'Subject': kmeans_rank_distance_matrix_experts_BSU1_BSU2_df.index,
    'KMeans Cluster': kmeans_subject_clusters
})

# Merge to align clusters from both methods
merged_subject_clusters = dense_subject_cluster_df.merge(kmeans_subjec

# Build a contingency table for subjects
contingency_table_subjects = pd.crosstab(merged_subject_clusters['Dens
# Display the contingency table
contingency_table_subjects
```

Out[277…

KMeans Cluster	1	2	3
Dense Cluster			
1	1	2	14
2	9	0	5
3	10	0	1

In [ ]:

```
contingency_table_subjects_specified = pd.DataFrame(
    [[[] for _ in range(3)] for _ in range(3)],
    columns=[1, 2, 3],
    index=[1, 2, 3]
)
contingency_table_subjects_specified.index.name = 'Dense Cluster'
contingency_table_subjects_specified.columns.name = 'KMeans Cluster'

def get_subjects(dense_label, kmeans_label):
    return merged_subject_clusters[
        (merged_subject_clusters['Dense Cluster'] == dense_label) &
        (merged_subject_clusters['KMeans Cluster'] == kmeans_label)
    ]['Subject'].tolist()
1
# Populate the contingency table with actual subject lists
for dense in [1, 2, 3]:
    for kmeans in [1, 2, 3]:
        contingency_table_subjects_specified.at[dense, kmeans] = get_s
```

In [279…

```
contingency_table_subjects_specified
```

Out[279…

KMeans Cluster	1	2	3
Dense Cluster			
1	[42]	[16, 23]	[1, 4, 5, 7, 8, 9, 12, 17, 22, 26, 37, 39, 40,...
2	[13, 15, 18, 19, 20, 21, 25, 27, 36]	[]	[3, 6, 11, 14, 24]
3	[10, 28, 29, 30, 31, 32, 33, 34, 35, 38]	[]	[2]

In [280…

```
# export the contingency table to a csv file
contingency_table_subjects_specified.to_csv('contingency_table_subject
```