Bioinformatics Assignment 3

Team 15---version by Harry Zarcadoolas

Preliminaries

In [1]:
```
!where python
```

```
C:\Users\harry\anaconda3\envs\cgs_assignment2\python.exe
C:\Users\harry\anaconda3\python.exe
C:\Program Files\Python312\python.exe
C:\Users\harry\AppData\Local\Microsoft\WindowsApps\python.exe
C:\msys64\ucrt64\bin\python.exe
```

In [2]:
```
# fixes warning with k-means efficiency process
import os
os.environ["OMP_NUM_THREADS"] = "1"
```

Imports

In [3]:
```
import pandas as pd
from sklearn.cluster import SpectralClustering
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import silhouette_score
from scipy.stats import chi2_contingency
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as colors
from sklearn.cluster import KMeans
import matplotlib.patches as mpatches
import numpy as np
```

In [4]:
```
# make sure plots show inline in jupyter
%matplotlib inline
```

Part 1

Load gene data

In [5]:
```
# load expression data
expression_data = pd.read_csv(r"C:\Users\harry\OneDrive - University of Florida\24-

# load expression metadata
metadata = pd.read_csv(r"C:\Users\harry\OneDrive - University of Florida\24-fall\CG
```

```
# check size and shape of expression matrix
num_genes, num_samples = expression_data.shape
print(f"The expression matrix has {num_genes} genes and {num_samples - 1} samples."

# ensure Ensembl IDs are strings and remove any version numbers (if present)
expression_data['Gene'] = expression_data['Gene'].astype(str).str.split('.').str[0]

# check for any missing conversions (actual conversion has been done in separate co
missing_gene_names = expression_data['Gene'].isnull().sum()
print(f"There are {missing_gene_names} genes with missing names.")
```

```
The expression matrix has 23870 genes and 94 samples.
There are 0 genes with missing names.
```

Part 2

a) Subset data to 5,000 most variable genes

In [6]:
```
# calculate variance for each gene (don't include gene name column)
gene_variances = expression_data.drop(columns=['Gene']).var(axis=1)

# 5,000 most variable genes
top_5000_genes = expression_data.iloc[gene_variances.nlargest(5000).index]

# confirm subset data shape
print(f"Top 5000 variable gene data shape: {top_5000_genes.shape}")
```

```
Top 5000 variable gene data shape: (5000, 95)
```

b-d) Clustering Algorithm --- v. Spectral Clustering

NOTE: Later for Part 2: Heatmap and Part 3: Statistics I also used k-means for more significant results

In [7]:
```
# drop 'Gene' column, leaving just data
top_5000_gene_data = top_5000_genes.drop(columns=['Gene'])

cluster_labels = {}
silhouette_scores = {}
p_values = []

# run spectral clustering and analysis with k values 2 to 8
for k in range(2, 9):
    print(f"\nRunning Spectral Clustering with k={k}...")

    #  spectral clustering with current k
    clustering = SpectralClustering(n_clusters=k, affinity='nearest_neighbors', ran
    labels = clustering.fit_predict(top_5000_gene_data.T)  # Transpose to cluster s

    # calc silhouette score
    silhouette_avg = silhouette_score(top_5000_gene_data.T, labels)

    # store labels and silhouette score
    cluster_labels[k] = labels
```

```
    silhouette_scores[k] = silhouette_avg

    # current sillhouette score
    print(f"Silhouette score for k={k}: {silhouette_avg}")

# chi-squared test results for multiple testing correction
chi2_results = []
p_values = []

# compare cluster memberships for k values using contingency tables
for k in range(3, 9):  # Start from k=3 to compare with k=2
    print(f"\nContingency table for k={k-1} vs k={k}:")

    # contingency for k-1 and k
    contingency_table = pd.crosstab(cluster_labels[k-1], cluster_labels[k])
    print(contingency_table)

    # chi-squared test on contingency table
    chi2, p, dof, _ = chi2_contingency(contingency_table)

    # store result for later correction
    chi2_results.append({
        "k-1": k-1,
        "k": k,
        "chi2": chi2,
        "p_value": p,
        "dof": dof
    })
    p_values.append(p)  # store p values for correction later

    print(f"Chi-squared test: Chi2 = {chi2}, p-value = {p}")

# adjust p-values using Bonferroni correction
adjusted_p_values = np.minimum(np.array(p_values) * len(p_values), 1.0)  # Bonferro

# add on adjusted p-values to the chi2_results
for i, result in enumerate(chi2_results):
    result['adjusted_p_value'] = adjusted_p_values[i]

# print final adjusted results
print("\nChi-squared test results with Bonferroni correction:")
for result in chi2_results:
    print(f"k={result['k-1']} vs k={result['k']}: Chi2 = {result['chi2']}, p-value
```

```
Running Spectral Clustering with k=2...
Silhouette score for k=2: 0.5446996723475832

Running Spectral Clustering with k=3...
Silhouette score for k=3: 0.32994683367643435

Running Spectral Clustering with k=4...
Silhouette score for k=4: 0.21334408693873366

Running Spectral Clustering with k=5...
Silhouette score for k=5: 0.19179243087285144

Running Spectral Clustering with k=6...
Silhouette score for k=6: 0.06909635410208506

Running Spectral Clustering with k=7...
Silhouette score for k=7: 0.16823079694456333

Running Spectral Clustering with k=8...
Silhouette score for k=8: 0.14469755750283164

Contingency table for k=2 vs k=3:
col_0   0   1   2
row_0
0       0   0  29
1      45  19   1
Chi-squared test: Chi2 = 89.46871794871795, p-value = 3.733487881184384e-20

Contingency table for k=3 vs k=4:
col_0   0   1   2   3
row_0
0      12   0   0  33
1       0   0  18   1
2       0  30   0   0
Chi-squared test: Chi2 = 181.17069143446852, p-value = 1.9140706753938645e-36

Contingency table for k=4 vs k=5:
col_0   0   1   2   3   4
row_0
0       0   0   0   0  12
1       0  29   0   1   0
2       0   0  18   0   0
3      12   0   1  21   0
Chi-squared test: Chi2 = 268.80349000844365, p-value = 1.61899020965234e-50

Contingency table for k=5 vs k=6:
col_0   0   1   2   3   4  5
row_0
0      12   0   0   0   0  0
1       0  19   0   0   1  9
2       0   0  19   0   0  0
3       0   0   0   0  22  0
4       0   0   0  12   0  0
Chi-squared test: Chi2 = 368.81259370314837, p-value = 5.850413701605445e-66

Contingency table for k=6 vs k=7:
```

```
col_0  0   1   2   3   4  5   6
row_0
0      0   0   0   0   0  0  12
1      1  18   0   0   0  0   0
2      0   0  19   0   0  0   0
3      0   0   0   0  12  0   0
4      3   0   0  20   0  0   0
5      0   0   0   0   0  9   0
Chi-squared test: Chi2 = 463.22425629290626, p-value = 4.030247091162422e-79

Contingency table for k=7 vs k=8:
col_0   0   1   2  3   4   5  6  7
row_0
0       0   0   0  0   0   0  4  0
1       0   0   0  0  18   0  0  0
2       0  15   0  0   0   0  0  4
3      15   0   0  0   0   0  0  5
4       0   0  12  0   0   0  0  0
5       0   0   0  9   0   0  0  0
6       1   0   0  0   0  10  0  1
Chi-squared test: Chi2 = 521.5763157894737, p-value = 5.191335330720188e-84

Chi-squared test results with Bonferroni correction:
k=2 vs k=3: Chi2 = 89.46871794871795, p-value = 3.733487881184384e-20, adjusted p-va
lue = 2.2400927287106307e-19
k=3 vs k=4: Chi2 = 181.17069143446852, p-value = 1.9140706753938645e-36, adjusted p-
value = 1.1484424052363187e-35
k=4 vs k=5: Chi2 = 268.80349000844365, p-value = 1.61899020965234e-50, adjusted p-va
lue = 9.71394125791404e-50
k=5 vs k=6: Chi2 = 368.81259370314837, p-value = 5.850413701605445e-66, adjusted p-v
alue = 3.5102482209632666e-65
k=6 vs k=7: Chi2 = 463.22425629290626, p-value = 4.030247091162422e-79, adjusted p-v
alue = 2.418148254697453e-78
k=7 vs k=8: Chi2 = 521.5763157894737, p-value = 5.191335330720188e-84, adjusted p-va
lue = 3.1148011984321128e-83
```

I used spectral clustering, so k is predetermined when I pass it as an input. I noticed that the best sillhoutte and lowest chi-squared values (indicating better model fit), occurred when the value was the smallest, k=2, and generally became more poor as the k-value raised. The chi-squared values only continued to rise with increased k values, with a high level of significance as pointed out by the p-values.

e)

In [8]:
```python
# funct for top n most variable genes
def select_top_n_genes(data, n):
    # Calculate variance for each gene
    gene_variances = data.var(axis=1)
    # Get indices of the n most variable genes
    top_n_genes_idx = gene_variances.nlargest(n).index
    # Return the subset with the top n variable genes
    return data.loc[top_n_genes_idx]

# test gene sizes
```

```
gene_sizes = [10, 100, 1000, 10000]

# to store clustering results
clustering_results = {}

# loop through the gene sizes
for n_genes in gene_sizes:
    print(f"\nTesting with {n_genes} genes:")

    # top n most variable genes from the expression data
    subset_data = select_top_n_genes(expression_data.drop(columns=['Gene']), n_gene

    # transpose the subset--- samples are rows, genes are columns
    subset_data_transposed = subset_data.T

    # run spectral clustering with fixed k=2
    clustering = SpectralClustering(n_clusters=2, affinity='nearest_neighbors', ran
    labels = clustering.fit_predict(subset_data_transposed)

    # calc silhouette score
    silhouette_avg = silhouette_score(subset_data_transposed, labels)

    # store results
    clustering_results[n_genes] = labels
    print(f"Silhouette score for {n_genes} genes: {silhouette_avg}")

# chi-squared tests for different gene sizes
for size_1 in gene_sizes:
    for size_2 in gene_sizes:
        if size_1 >= size_2:
            continue
        print(f"\nChi-squared test between {size_1} genes and {size_2} genes:")
        labels_1 = clustering_results[size_1]
        labels_2 = clustering_results[size_2]

        # contingency table
        contingency_table = pd.crosstab(labels_1, labels_2)
        chi2, p, _, _ = chi2_contingency(contingency_table)
        print(f"Chi2 = {chi2}, p-value = {p}")
```

```
Testing with 10 genes:
Silhouette score for 10 genes: 0.8092100609849859

Testing with 100 genes:
Silhouette score for 100 genes: 0.7772257839043406

Testing with 1000 genes:
Silhouette score for 1000 genes: 0.6615722339072002

Testing with 10000 genes:
Silhouette score for 10000 genes: 0.5012880454409471

Chi-squared test between 10 genes and 100 genes:
Chi2 = 72.78770962845215, p-value = 1.4437313232793054e-17

Chi-squared test between 10 genes and 1000 genes:
Chi2 = 72.78770962845215, p-value = 1.4437313232793054e-17

Chi-squared test between 10 genes and 10000 genes:
Chi2 = 89.17898560755586, p-value = 3.6065978639909996e-21

Chi-squared test between 100 genes and 1000 genes:
Chi2 = 89.53011856455689, p-value = 3.020055773737905e-21

Chi-squared test between 100 genes and 10000 genes:
Chi2 = 72.78770962845215, p-value = 1.4437313232793054e-17

Chi-squared test between 1000 genes and 10000 genes:
Chi2 = 72.78770962845215, p-value = 1.4437313232793054e-17
C:\Users\harry\anaconda3\envs\cgs_assignment2\Lib\site-packages\sklearn\manifold\_spectral_embedding.py:329: UserWarning: Graph is not fully connected, spectral embedding may not work as expected.
  warnings.warn(
```

The number of genes, when referecing the chi-squared test, did not change significantly when using varying amounts of genes. This has me to believe that a few of the genes account for a lot of the significance. (I also used silhouette scores which seemed to indicate that the model became more loose fitting with a higher number of genes).

Sankey plot

In [9]:
```python
# for rendering with Juptyer
pio.renderers.default = 'plotly_mimetype'

clustering_results = {
    10: clustering_results[10],
    100: clustering_results[100],
    1000: clustering_results[1000],
    10000: clustering_results[10000]
}

# Define the gene sizes
gene_sizes = [10, 100, 1000, 10000]
```

```python
# list of nodes based on the cluster labels (for each gene size)
nodes = []
for n_genes in gene_sizes:
    unique_labels = set(clustering_results[n_genes])
    for label in unique_labels:
        nodes.append(f'{n_genes} genes - Cluster {label}')

# mapping from node labels to their index in the Sankey diagram
node_indices = {node: i for i, node in enumerate(nodes)}

# source, target, and values arrays
sources = []
targets = []
values = []

# go through gene sizes to create links between successive clustering results
for i in range(len(gene_sizes) - 1):
    size_1 = gene_sizes[i]
    size_2 = gene_sizes[i + 1]

    labels_1 = clustering_results[size_1]
    labels_2 = clustering_results[size_2]

    # for each sample, create connections between clusters of consecutive gene size
    for sample_idx in range(len(labels_1)):
        source_label = f'{size_1} genes - Cluster {labels_1[sample_idx]}'
        target_label = f'{size_2} genes - Cluster {labels_2[sample_idx]}'

        # indices for source and target nodes
        source_idx = node_indices[source_label]
        target_idx = node_indices[target_label]

        # add on source, target, and the value of the flow (each sample is a unit f
        sources.append(source_idx)
        targets.append(target_idx)
        values.append(1)

# color for each cluster
cluster_colors = colors.qualitative.Set1

# list of colors for each gene size's cluster labels
link_colors = []
for i in range(len(sources)):
    source_label = sources[i]
    link_colors.append(cluster_colors[source_label % len(cluster_colors)])

# sankey diagram with colors
fig = go.Figure(go.Sankey(
    node=dict(
        pad=15,
        thickness=20,
        line=dict(color="black", width=0.5),
        label=nodes,
        color="blue"
    ),
    link=dict(
```

```
        source=sources,
        target=targets,
        value=values,
        color=link_colors
    )
))

# show diagram
fig.update_layout(title_text="Alluvial Diagram of Cluster Membership Across Differe
fig.show()
```

The Sankey diagram was appended to the end of the pdf because of rendering. This diagram showed how cluster memberships changed as we increased the number of genes being used for clustering. I used four different gene sizes: 10, 100, 1,000, and 10,000. For each size, I used spectral clustering to group the samples. Then, I used colored ribbons to show the flow of how the genes were being clustered as the gene size increased. From the diagram, it could be shown that there is cluster stability since most samples remain in the same cluster. But, there are still a few transitions which indicate that a larger dataset provides more granular insight. When the size changed from 10 to 100 genes, some samples were changed to cluster 1 from cluster 0. However, when the size changes from 1000 to 10,000, there is a near opposite effect, so it may just be noise and further shows cluster stability.

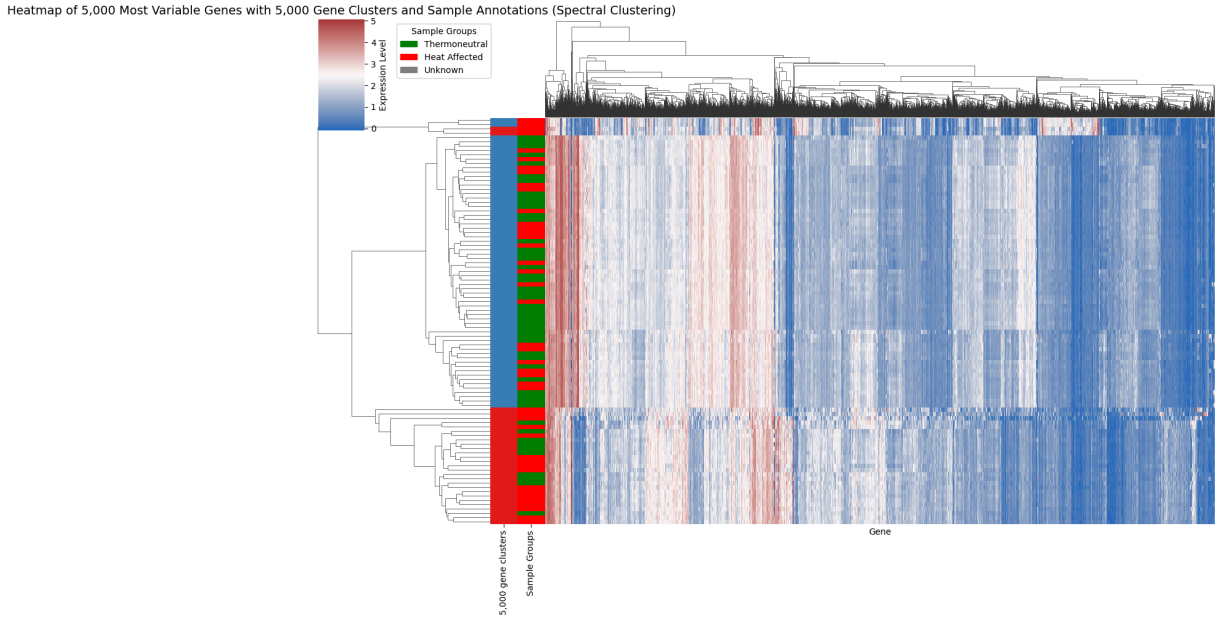
f) Heatmaps and Dendrograms

In [10]:

```
# reload data in case of error
expression_data = pd.read_csv(r"C:\Users\harry\OneDrive - University of Florida\24-
metadata = pd.read_csv(r"C:\Users\harry\OneDrive - University of Florida\24-fall\CG

gene_variances = expression_data.drop(columns=['Gene']).var(axis=1)
top_5000_genes_idx = gene_variances.nlargest(5000).index
top_5000_genes = expression_data.iloc[top_5000_genes_idx].set_index('Gene')

expression_data_transposed = top_5000_genes.T

# map samples to groups (Thermoneutral vs Heat Affected)
metadata['condition'] = metadata['refinebio_title'].apply(lambda x: 'Heat Affected'
sample_to_condition = dict(zip(metadata['refinebio_accession_code'], metadata['cond

# ensure all sample IDs in expression_data_transposed are found in metadata
missing_samples = set(expression_data_transposed.index) - set(sample_to_condition.k
if missing_samples:
    print(f"Warning: Missing sample IDs in metadata: {missing_samples}")

# sample groups should be in the same order as the expression data
sample_groups = [sample_to_condition.get(sample, 'Unknown') for sample in expressio

# perform clustering (Spectral Clustering with k=2)
spectral_clustering = SpectralClustering(n_clusters=2, affinity='nearest_neighbors'
cluster_labels = spectral_clustering.fit_predict(expression_data_transposed)

# annotations
```

```
annotations = pd.DataFrame({
    "5,000 gene clusters": cluster_labels,
    "Sample Groups": sample_groups
}, index=expression_data_transposed.index)

# map annotations to colors using sns.color_palette()
cluster_palette = sns.color_palette("Set1", 2)  # Two clusters, so two colors
group_palette = {'Thermoneutral': 'green', 'Heat Affected': 'red', 'Unknown': 'gray

# color mappings
cluster_lut = dict(zip(sorted(annotations["5,000 gene clusters"].unique()), cluster
group_lut = group_palette  # Use pre-defined group_palette directly

# map annotations to color codes
row_colors = pd.DataFrame({
    "5,000 gene clusters": annotations["5,000 gene clusters"].map(cluster_lut),
    "Sample Groups": annotations["Sample Groups"].map(group_lut)
}, index=annotations.index)

# heatmap
g = sns.clustermap(
    expression_data_transposed,  # The expression data
    method="average",            # Clustering method
    metric="euclidean",          # Distance metric for clustering
    cmap="vlag",                 # Color map for heatmap
    row_cluster=True,            # Create row dendrogram (for genes)
    col_cluster=True,            # Create column dendrogram (for samples)
    figsize=(15, 10),            # Figure size
    row_colors=row_colors,       # Add row annotations as a sidebar
    cbar_kws={"label": "Expression Level"},  # Color bar for the heatmap
    xticklabels=False,           # Hide x tick labels (sample names)
    yticklabels=False,           # Hide y tick labels (gene names)
)

# add legend for sample groups
legend_patches = [
    mpatches.Patch(color='green', label='Thermoneutral'),
    mpatches.Patch(color='red', label='Heat Affected'),
    mpatches.Patch(color='gray', label='Unknown')
]
plt.legend(handles=legend_patches, title="Sample Groups", bbox_to_anchor=(1.6, 1),

# labels and title
plt.title("Heatmap of 5,000 Most Variable Genes with 5,000 Gene Clusters and Sample

# display
plt.show()
```

```
C:\Users\harry\anaconda3\envs\cgs_assignment2\Lib\site-packages\seaborn\matrix.py:560: UserWarning:

Clustering large matrix with scipy. Installing `fastcluster` may give better performance.

C:\Users\harry\anaconda3\envs\cgs_assignment2\Lib\site-packages\seaborn\matrix.py:560: UserWarning:

Clustering large matrix with scipy. Installing `fastcluster` may give better performance.
```



In this heatmap, I clustered the 5,000 gene data with the most notable variance again using spectral clustering with k=2. I grouped the samples on expression patterns. I made the rows represent genes and columns represent samples and provided annotations for the clustering profiles compared to the actual data markers that show Heat vs. Thermoneutral sample grouping. I added a legend to help show which groups go with thermoneutral and heat affected groups. Altogether, I used the combination of annotations to highlight if there could be overlap and/or trends that show good correlation in clustering and gene expression level. Based on the diagram, it looks like there is possible concistency that go well with the red cluster form spectral clustering to go with heat affected, but the results get less significant when taking into account the blue cluster and thermoneutral samples.

Part 3: Statistics

Note: I also utilized kmeans clustering as it had significant results when compared to the original sample groupings. So, I added another heatmap with k=2 kmeans clustering as well additional statistics for that section

In [11]:
```
# chi-squared test of independence
# contingency table comparing cluster membership with sample groups
```

```
contingency_table = pd.crosstab(annotations["5,000 gene clusters"], annotations["Sa

# actual chi-squared test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# results
print(f"Chi-squared test statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")

# show contingency table and expected counts
print("\nContingency Table:")
print(contingency_table)
print("\nExpected Counts under the null hypothesis:")
print(pd.DataFrame(expected, columns=contingency_table.columns, index=contingency_t

# adjust for multiple hypothesis testing using Bonferroni correction (adjusted p-va
alpha = 0.05  # Significance level
adjusted_p_value = p_value * len(contingency_table.columns)
adjusted_p_value = min(adjusted_p_value, 1.0)  # make sure the p-value doesn't exce

print(f"\nAdjusted P-value (Bonferroni correction): {adjusted_p_value}")

# basic interpretation
if adjusted_p_value < alpha:
    print("There is a significant association between cluster membership and sample
else:
    print("There is no significant association between cluster membership and sampl
```

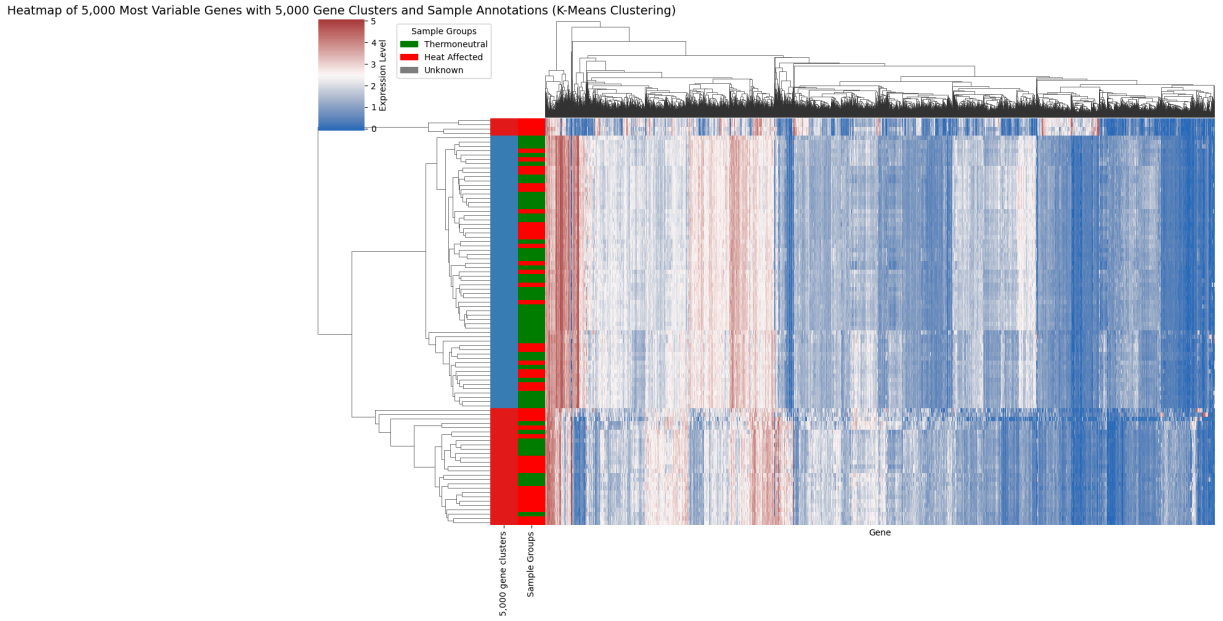
```
Chi-squared test statistic: 4.859099590065107
P-value: 0.027500684875791458
Degrees of freedom: 1

Contingency Table:
Sample Groups        Heat Affected  Thermoneutral
5,000 gene clusters
0                               19             10
1                               25             40

Expected Counts under the null hypothesis:
Sample Groups        Heat Affected  Thermoneutral
5,000 gene clusters
0                       13.574468      15.425532
1                       30.425532      34.574468

Adjusted P-value (Bonferroni correction): 0.055001369751582915
There is no significant association between cluster membership and sample groups.
```

In [12]:
```
# reload data in case of error
expression_data = pd.read_csv(r"C:\Users\harry\OneDrive - University of Florida\24-
metadata = pd.read_csv(r"C:\Users\harry\OneDrive - University of Florida\24-fall\CG

gene_variances = expression_data.drop(columns=['Gene']).var(axis=1)
top_5000_genes_idx = gene_variances.nlargest(5000).index
top_5000_genes = expression_data.iloc[top_5000_genes_idx].set_index('Gene')
```

```
expression_data_transposed = top_5000_genes.T

# map samples to groups (Thermoneutral vs Heat Affected)
metadata['condition'] = metadata['refinebio_title'].apply(lambda x: 'Heat Affected'
sample_to_condition = dict(zip(metadata['refinebio_accession_code'], metadata['cond

# ensure all sample IDs in expression_data_transposed are found in metadata
missing_samples = set(expression_data_transposed.index) - set(sample_to_condition.k
if missing_samples:
    print(f"Warning: Missing sample IDs in metadata: {missing_samples}")

# sample groups should be in the same order as the expression data
sample_groups = [sample_to_condition.get(sample, 'Unknown') for sample in expressio

# perform clustering (kmeans with k=2)
kmeans = KMeans(n_clusters=2, random_state=42).fit(expression_data_transposed)
cluster_labels = kmeans.labels_

# annotations
annotations = pd.DataFrame({
    "5,000 gene clusters": cluster_labels,
    "Sample Groups": sample_groups
}, index=expression_data_transposed.index)

# map annotations to colors using sns.color_palette()
cluster_palette = sns.color_palette("Set1", 2)  # Two clusters, so two colors
group_palette = {'Thermoneutral': 'green', 'Heat Affected': 'red', 'Unknown': 'gray

# color mappings
cluster_lut = dict(zip(sorted(annotations["5,000 gene clusters"].unique()), cluster
group_lut = group_palette  # Use pre-defined group_palette directly

# map annotations to color codes
row_colors = pd.DataFrame({
    "5,000 gene clusters": annotations["5,000 gene clusters"].map(cluster_lut),
    "Sample Groups": annotations["Sample Groups"].map(group_lut)
}, index=annotations.index)

# heatmap
g = sns.clustermap(
    expression_data_transposed,  # The expression data
    method="average",            # Clustering method
    metric="euclidean",          # Distance metric for clustering
    cmap="vlag",                 # Color map for heatmap
    row_cluster=True,            # Create row dendrogram (for genes)
    col_cluster=True,            # Create column dendrogram (for samples)
    figsize=(15, 10),            # Figure size
    row_colors=row_colors,       # Add row annotations as a sidebar
    cbar_kws={"label": "Expression Level"},  # Color bar for the heatmap
    xticklabels=False,           # Hide x tick labels (sample names)
    yticklabels=False,           # Hide y tick labels (gene names)
)

# add legend for sample groups
legend_patches = [
    mpatches.Patch(color='green', label='Thermoneutral'),
```

```
    mpatches.Patch(color='red', label='Heat Affected'),
    mpatches.Patch(color='gray', label='Unknown')
]
plt.legend(handles=legend_patches, title="Sample Groups", bbox_to_anchor=(1.6, 1),

# labels and title
plt.title("Heatmap of 5,000 Most Variable Genes with 5,000 Gene Clusters and Sample

# display
plt.show()
```

```
C:\Users\harry\anaconda3\envs\cgs_assignment2\Lib\site-packages\seaborn\matrix.py:56
0: UserWarning:

Clustering large matrix with scipy. Installing `fastcluster` may give better perform
ance.

C:\Users\harry\anaconda3\envs\cgs_assignment2\Lib\site-packages\seaborn\matrix.py:56
0: UserWarning:

Clustering large matrix with scipy. Installing `fastcluster` may give better perform
ance.
```



In this heatmap, I clustered the 5,000 gene data with the most notable variance again using k-means clustering with k=2. I grouped the samples on expression patterns. I made the rows represent genes and columns represent samples and provided annotations for the clustering profiles compared to the actual data markers that show Heat vs. Thermoneutral sample grouping. I added a legend to help show which groups go with thermoneutral and heat affected groups. Altogether, I used the combination of annotations to highlight if there could be overlap and/or trends that show good correlation in clustering and gene expression level. Based on the diagram, there seems to be more concistency to show that the red cluster from spectral clustering aligns with heat affected samples. Unlike the last heatmap, there also seems to be stronger constitency as well for the blue cluster matching to the thermoneutral sample grouping, with still some exceptions.

In [13]:
```
# chi-squared test of independence
# contingency table comparing cluster membership with sample groups
contingency_table = pd.crosstab(annotations["5,000 gene clusters"], annotations["Sa

# actual chi-squared test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# results
print(f"Chi-squared test statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")

# show contingency table and expected counts
print("\nContingency Table:")
print(contingency_table)
print("\nExpected Counts under the null hypothesis:")
print(pd.DataFrame(expected, columns=contingency_table.columns, index=contingency_t

# adjust for multiple hypothesis testing using Bonferroni correction (adjusted p-va
alpha = 0.05  # Significance level
adjusted_p_value = p_value * len(contingency_table.columns)
adjusted_p_value = min(adjusted_p_value, 1.0)  # make sure the p-value doesn't exce

print(f"\nAdjusted P-value (Bonferroni correction): {adjusted_p_value}")

# basic interpretation
if adjusted_p_value < alpha:
    print("There is a significant association between cluster membership and sample
else:
    print("There is no significant association between cluster membership and sampl
```

```
Chi-squared test statistic: 6.934572918121306
P-value: 0.008454524833439107
Degrees of freedom: 1

Contingency Table:
Sample Groups        Heat Affected  Thermoneutral
5,000 gene clusters
0                               21             10
1                               23             40

Expected Counts under the null hypothesis:
Sample Groups        Heat Affected  Thermoneutral
5,000 gene clusters
0                       14.510638      16.489362
1                       29.489362      33.510638

Adjusted P-value (Bonferroni correction): 0.016909049666878215
There is a significant association between cluster membership and sample groups.
```

For results with the spectral clustering analysis, at first glance there seemed to be a decent correlation matching the chosen groups in Assignment 1. However, upon closer inspection the adjusted p-value showed the within a 95% confidence, you can't say that the chi-squared value was truly significant. This is backed by the heatmap which shows

inconcistent correlations with the thermoneutral group attempting to correspond with the blue cluster This led me to wanting to see if using a basic k-means algorithm can help provide slightly more significant results. This proved to be true. The chi-squared value was still low enough to show strong model fit and the adjusted p-value fell well within the 95% confidence interval. Still, when looking at the contigency table, the number of samples in the cluster (red) that is supposed to correspond with heat affected is only a bit higher than the blue cluster. This is probably because k-means is a relatively basic model. Altogether, the clustering with k-means showed that are significant results when comparing to the assignment 1 groupings and that spectral clustering falls a little short.

Alluvial Diagram of Cluster Membership Across Different Gene Sizes