Assignment 2

Group 8---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
```

Imports

In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap
import warnings
from statsmodels.stats.multitest import multipletests
from scipy import stats
import gprofiler
from scipy.stats import ranksums
```

In [3]:
```
# make sure plots show inline in jupyter
%matplotlib inline
```

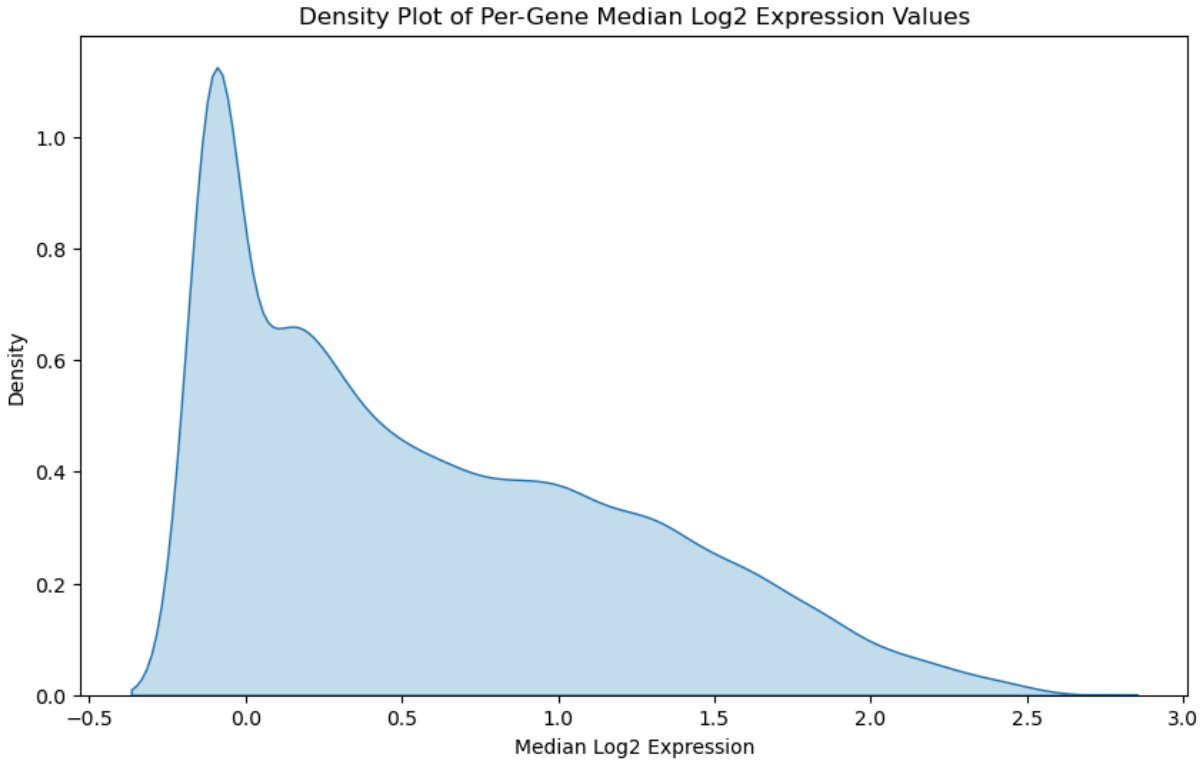
Part 1

Load gene data

In [4]:
```
# 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.")

# get number of unique genes
num_unique_genes = expression_data['Gene'].nunique()
print(f"The dataset has {num_unique_genes} unique genes.")

# log-scale the expression data values and calculate per-gene median expression
expression_values = expression_data.drop(columns=['Gene']) # remove gene name from
log_expression_values = np.log2(expression_values + 1)
median_expression = log_expression_values.median(axis=1)

# calculate expression range (max - min) for each gene across all samples
expression_range_per_gene = log_expression_values.max(axis=1) - log_expression_valu
# Calculate the median of these ranges
median_expression_range = expression_range_per_gene.median()
print(f"\nInsight to the average expression range across all genes:\nMedian express
mean_expression_range = expression_range_per_gene.mean()
print(f"Mean expression range: {mean_expression_range}\n") # repeat for mean


# Plot density
plt.figure(figsize=(10, 6))
sns.kdeplot(median_expression, fill=True)
plt.title('Density Plot of Per-Gene Median Log2 Expression Values')
plt.xlabel('Median Log2 Expression')
plt.ylabel('Density')
plt.show()
```

```
The expression matrix has 23870 genes and 94 samples.
There are 0 genes with missing names.
The dataset has 23798 unique genes.

Insight to the average expression range across all genes:
Median expression range: 0.743527807642619
Mean expression range: 0.836855693613459
```



Summary:

These findings show that my chicken dataset has 23,870 genes with a majority of them being unique. The median and mean expression range respectively was 0.74 and 0.84 (log2-scaled), so there's a relatively low level of epression across the board. Log2-scaled expression ranges for human genes are typically between 0.5 to 2. The density plot shows that most genes actually have a a low expression followed by a gradual tailing of gene density as median expression increases. So, only a relatively small portion of genes show high expression across different regions/tissues.

Part 2

Two Groups: Affected by Heat (Heat Stress - RED) vs Thermoneutral (Control - BLUE)

In [5]:
```
# function to simplify and assign groups based on keyword tags in the 'refinebio_ti
def assign_group(title):
    if 'Control' in title:
        return 'Thermoneutral'
    elif 'Heat Stress' in title:
        return 'Affected by Heat'
    else:
        return 'Unknown'

# creat a 'Group' col to add to the metadata using the simple grouping function
metadata['Group'] = metadata['refinebio_title'].apply(assign_group)
```

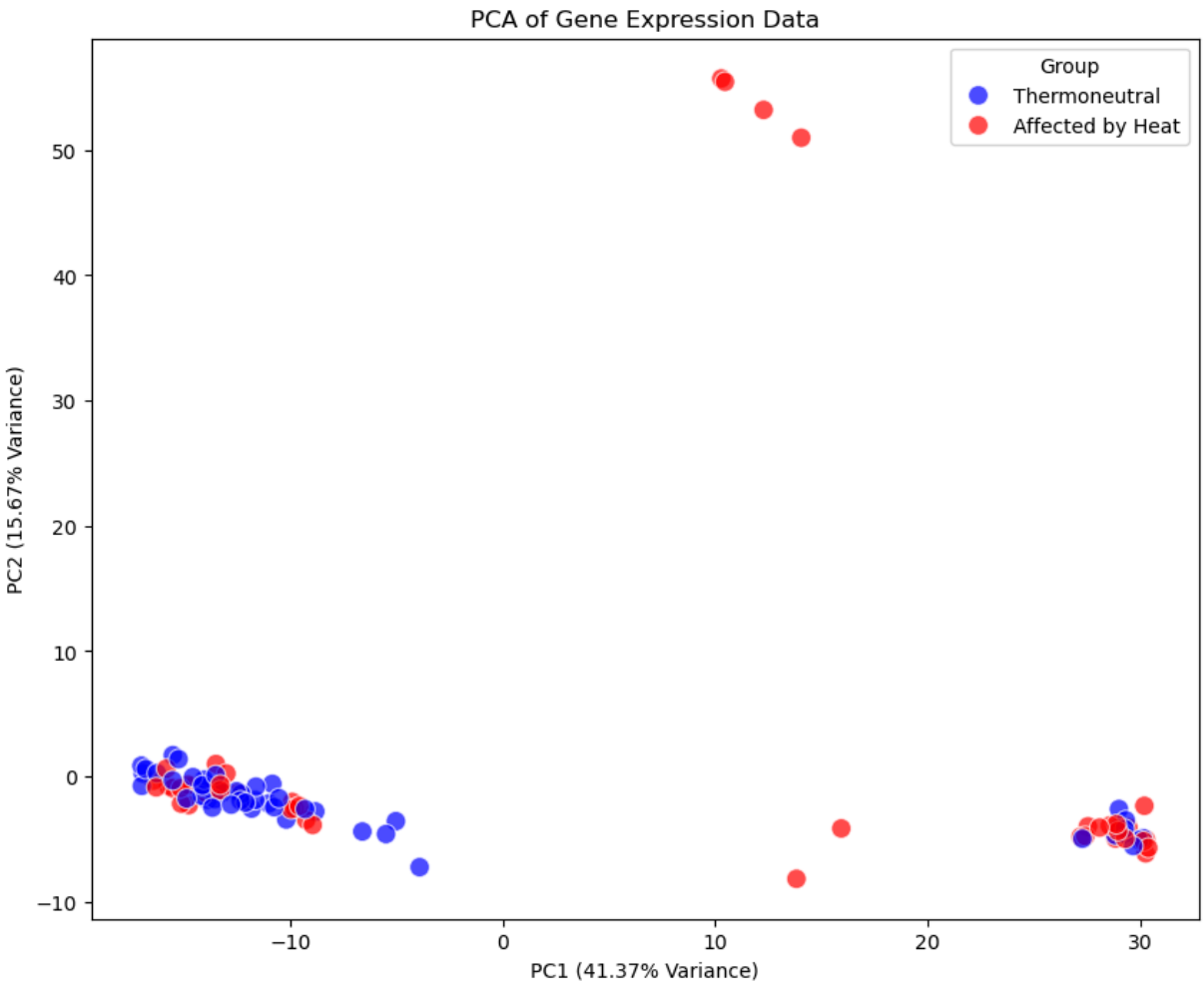
```
# check for missing samples between full expression data and metadata
expression_samples = expression_values.columns.tolist()
metadata_samples = metadata['refinebio_accession_code'].tolist()
missing_samples_in_metadata = set(expression_samples) - set(metadata_samples)
missing_samples_in_expression = set(metadata_samples) - set(expression_samples)
if missing_samples_in_metadata:
    print(f"Warning: The following samples are missing in metadata: {missing_sample
if missing_samples_in_expression:
    print(f"Warning: The following samples are missing in expression data: {missing
if not missing_samples_in_metadata and not missing_samples_in_expression:
    print("There are no data discrepencies between metadata and full expression dat
```

```
There are no data discrepencies between metadata and full expression data.
```

In [6]:
```
# transpose data so samples are now rows instaead of genes
expression_values = log_expression_values.T
```

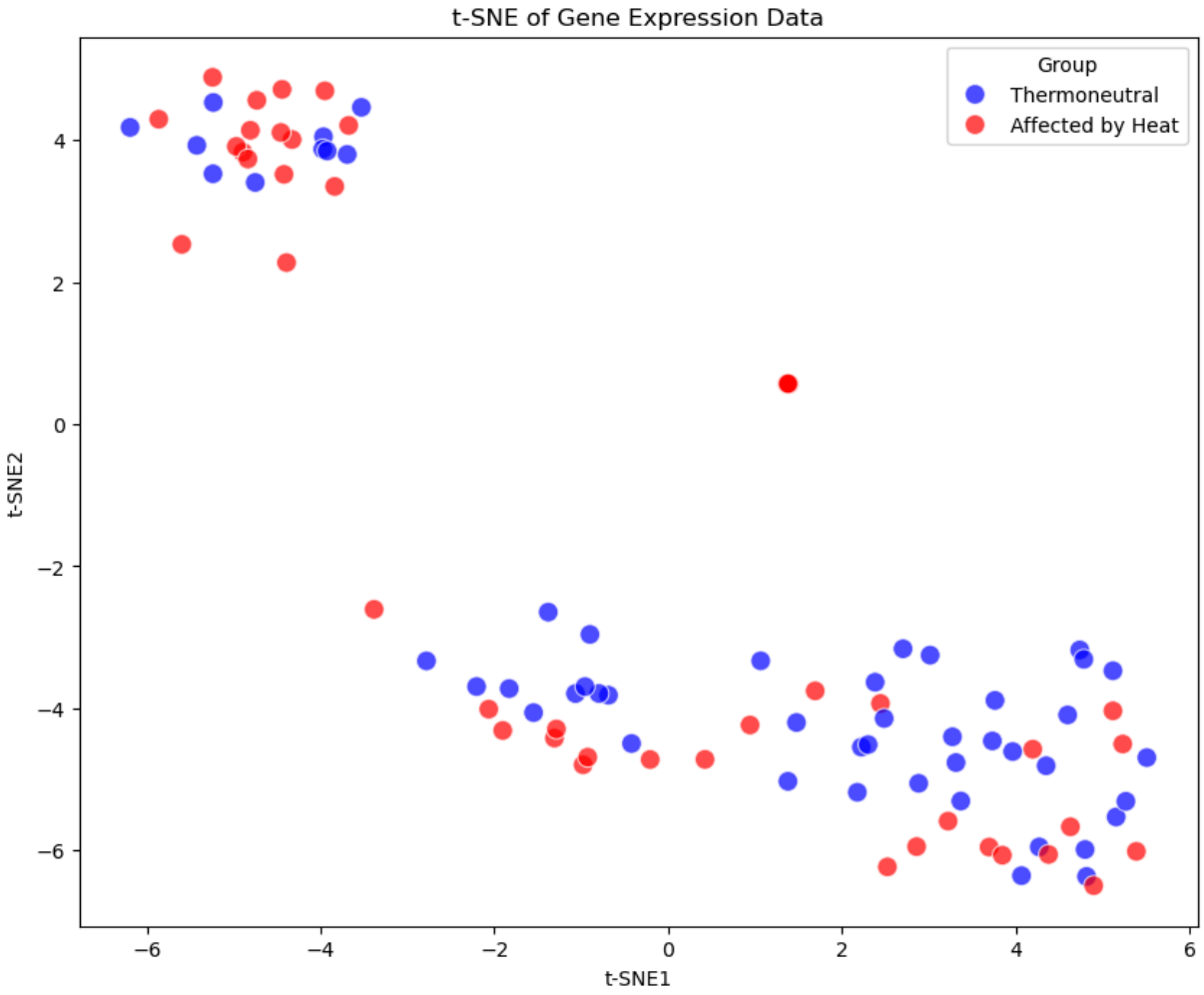
Principal Component Analysis (PCA)

In [7]:
```
# perform PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(expression_values)
pca_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'], index=expr

# merge PCA data with metadata to get groupings
pca_df = pca_df.merge(metadata[['refinebio_accession_code', 'Group']], left_index=T

# plot results
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='PC1', y='PC2',
    hue='Group',
    palette={'Thermoneutral': 'blue', 'Affected by Heat': 'red'},
    data=pca_df,
    s=100, alpha=0.7
)
plt.title('PCA of Gene Expression Data')
plt.xlabel(f'PC1 ({pca.explained_variance_ratio_[0]*100:.2f}% Variance)')
plt.ylabel(f'PC2 ({pca.explained_variance_ratio_[1]*100:.2f}% Variance)')
plt.legend(title='Group')
plt.show()
```



t-Distributed Stochastic Neighbor Embedding (t-SNE)

In [8]:
```
# perform t-SNE
tsne = TSNE(n_components=2, random_state=42)
tsne_components = tsne.fit_transform(expression_values)
tsne_df = pd.DataFrame(data=tsne_components, columns=['t-SNE1', 't-SNE2'], index=ex

# merge t-SNE data with metadata to get groupings
tsne_df = tsne_df.merge(metadata[['refinebio_accession_code', 'Group']], left_index

# plot results
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='t-SNE1', y='t-SNE2',
    hue='Group',
    palette={'Thermoneutral': 'blue', 'Affected by Heat': 'red'},
    data=tsne_df,
    s=100, alpha=0.7
)
plt.title('t-SNE of Gene Expression Data')
plt.xlabel('t-SNE1')
plt.ylabel('t-SNE2')
plt.legend(title='Group')
plt.show()
```

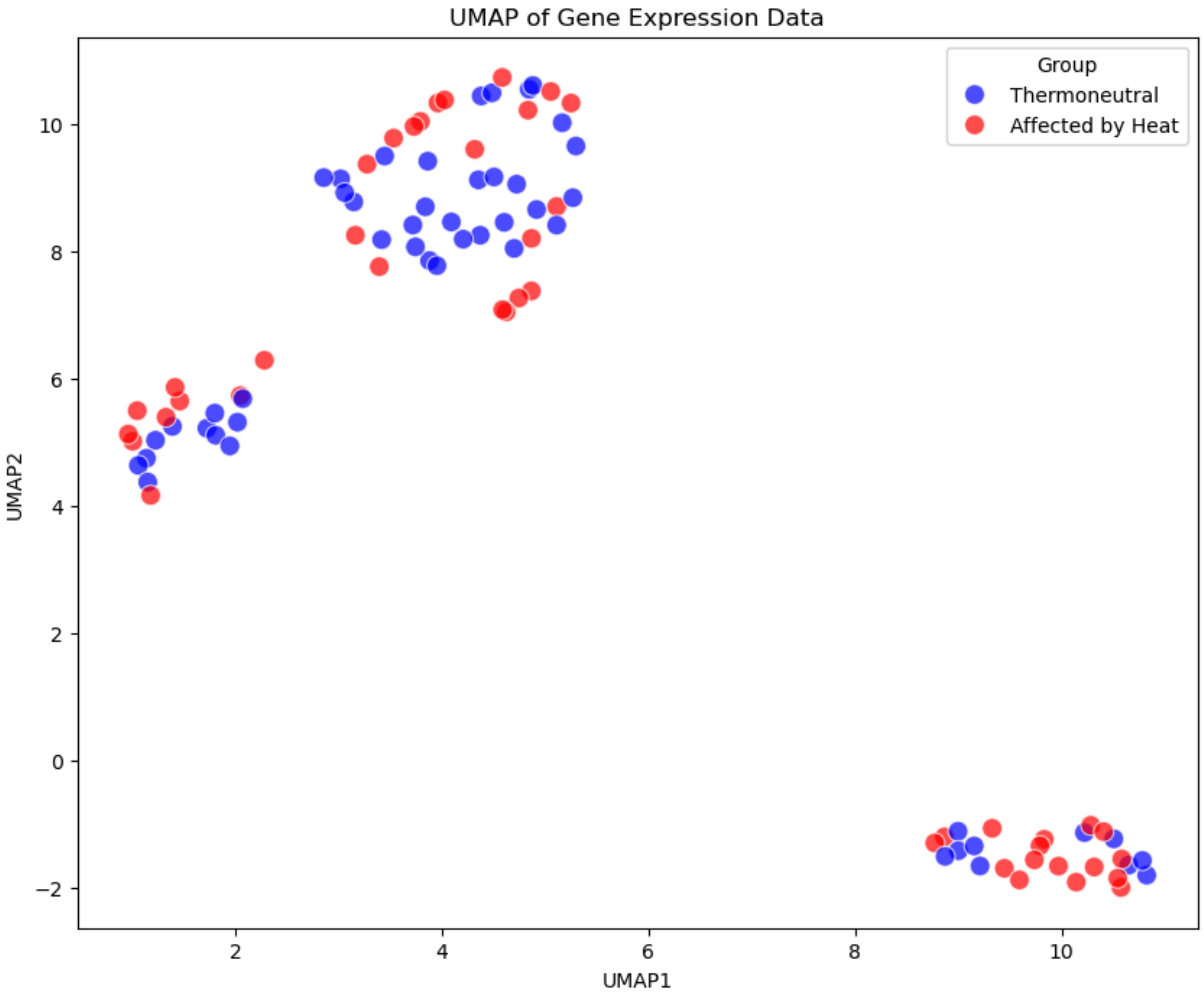


Uniform Manifold Approximation and Projection (UMAP)

In [9]:

```
# suppress the specific warning about parallelism processing (we need the two group
warnings.filterwarnings("ignore", message="n_jobs value .* overridden to 1 by setti

# perform UMAP
umap_reducer = umap.UMAP(n_components=2, random_state=42)
umap_components = umap_reducer.fit_transform(expression_values)
umap_df = pd.DataFrame(data=umap_components, columns=['UMAP1', 'UMAP2'], index=expr

# merge UMAP data with metadata to get groupings
umap_df = umap_df.merge(metadata[['refinebio_accession_code', 'Group']], left_index

# plot results
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x='UMAP1', y='UMAP2',
    hue='Group',
    palette={'Thermoneutral': 'blue', 'Affected by Heat': 'red'},
    data=umap_df,
    s=100, alpha=0.7
)
plt.title('UMAP of Gene Expression Data')
plt.xlabel('UMAP1')
```

```
plt.ylabel('UMAP2')
plt.legend(title='Group')
plt.show()
```



Summary:

Each of the three plots show the gene expression data with differentiation between the thermoneutral and heat stress affected. Each use different methods to reduce dimensionality and ultimately observe patterns between the two groups.

The Principal Component Analysis (PCA) plot illustrates the difference between the two principal components (thermoneutral vs heat affected), taking a more 2-dimensional approach to the analysis. Together, they represent ~51% of the total variance, or 41.07% variance for heat affected and 15.56% for thermoneutral. The high variance of heat affected samples is significant because it suggests heat stress leads to larger differences in gene expression. However, while there are clear outliers in the heat-affected group from the plot, the overall separation between groups is not strongly apparent within the main clusters.

t-distributed Stochastic Neighbor Embedding (t-SNE) captures local relationships better than PCA. Here, the points are more distinguishable within each cluster. In these clusters, it can be seen there is some polarization of each groups--signifying that heat may cause expression to

be varied to a measurable extent. However, the overall structure of the data has loosesend makes the global organization of the data less out of scope.

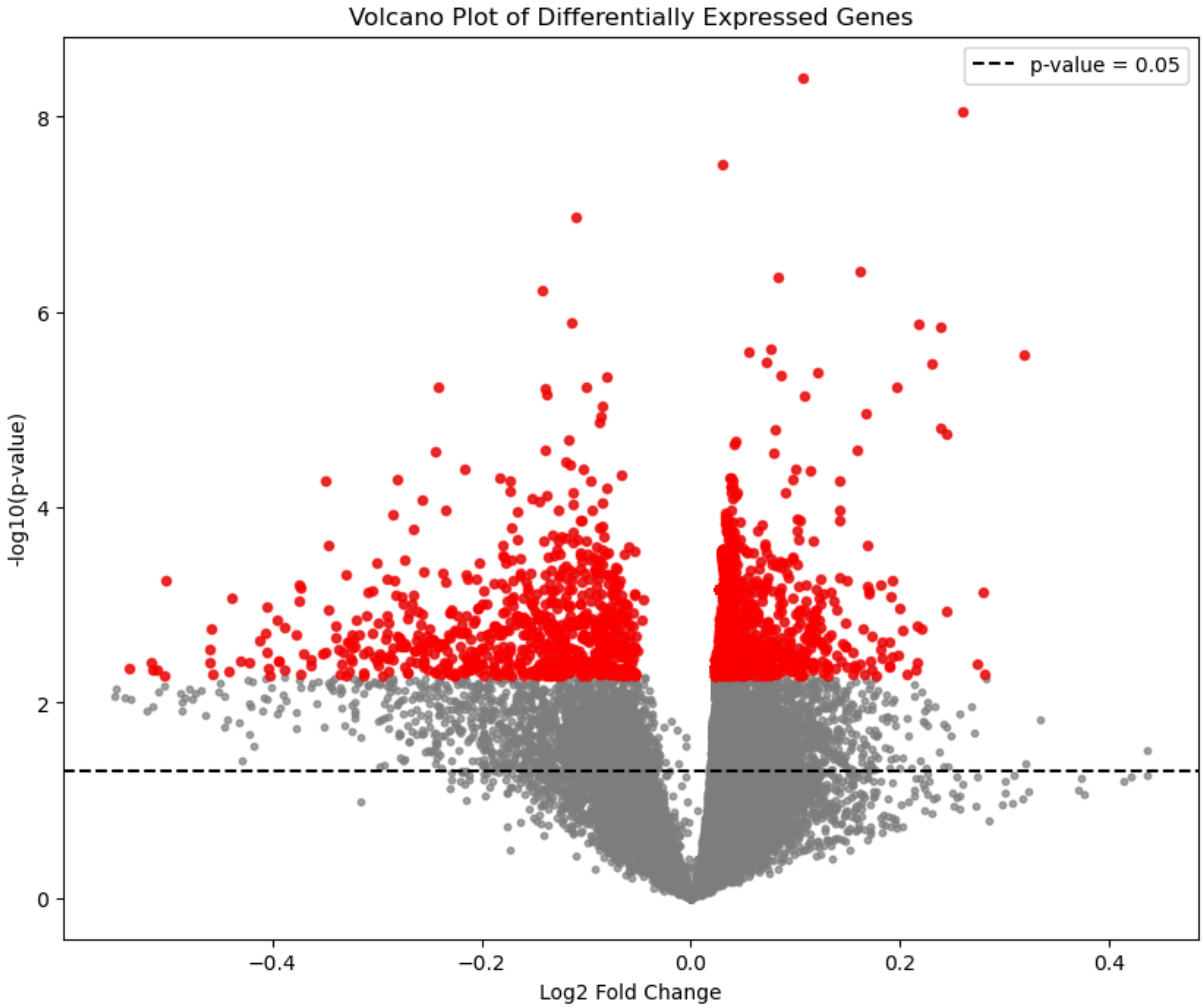
Emphasizing a balance bettween the two, Uniform Manifold Approximation and Projection (UMAP) takes into account local and global structuring of the data. It provides a more clear picture of how the different clusters relate to each other globally while still highlighting the polarization between the groups in each. This further pushes the idea of differing levels of gene expression resulting from heat stress.

Part 3

In [10]:
```
# extract the groups for thermoneutral and heat-affected samples
thermoneutral_samples = metadata[metadata['Group'] == 'Thermoneutral']['refinebio_a
heat_stress_samples = metadata[metadata['Group'] == 'Affected by Heat']['refinebio_

# get the expression data for each group
thermoneutral_data = log_expression_values[thermoneutral_samples]
heat_stress_data = log_expression_values[heat_stress_samples]

# perform differential expression analysis
results = []
for gene in log_expression_values.index:
    thermoneutral_values = thermoneutral_data.loc[gene]
    heat_stress_values = heat_stress_data.loc[gene]

    # perform t-test between the two groups
    t_stat, p_value = stats.ttest_ind(thermoneutral_values, heat_stress_values)

    # calculate log2 fold change
    mean_thermoneutral = np.mean(thermoneutral_values)
    mean_heat_stress = np.mean(heat_stress_values)
    log2_fold_change = mean_heat_stress - mean_thermoneutral

    # Store the results---gene, log2FC, and p-value
    results.append([gene, log2_fold_change, p_value])

# convert results to its own DataFrame
results_df = pd.DataFrame(results, columns=['Gene', 'Log2FoldChange', 'p_value'])

# adjust p-values using Benjamini-Hochberg FDR correction
results_df['adjusted_p_value'] = multipletests(results_df['p_value'], method='fdr_b

# filter for significant results (adjusted p-value < 0.05)
significant_genes = results_df[results_df['adjusted_p_value'] < 0.05]

# sort results by log2 fold change (absolute value) to find the most up-regulated g
significant_genes = significant_genes.sort_values(by='Log2FoldChange', ascending=Fa
```

Volcano plot

In [11]:
```
# create a -log10(p_value) column for volcano plot generation
results_df['-log10(p_value)'] = -np.log10(results_df['p_value'])


plt.figure(figsize=(10, 8))

# non-significant genes are gray
plt.scatter(results_df['Log2FoldChange'], results_df['-log10(p_value)'], color='gra

# significant genes highlighted red
significant = results_df['adjusted_p_value'] < 0.05
plt.scatter(results_df.loc[significant, 'Log2FoldChange'],
            results_df.loc[significant, '-log10(p_value)'],
            color='red', alpha=0.7, s=20)


plt.title('Volcano Plot of Differentially Expressed Genes')
plt.xlabel('Log2 Fold Change')
plt.ylabel('-log10(p-value)')

# add a significance threshold line
plt.axhline(y=-np.log10(0.05), color='black', linestyle='--', label='p-value = 0.05
plt.legend()

plt.show()
```



In [12]:
```
# show top results
print(f"Top significant genes:\n{significant_genes.head(50)}")

# outut the differential expression results to a CSV
significant_genes.to_csv('differential_expression_results.csv', index=False)
```

Top significant genes:

	Gene	Log2FoldChange	p_value	adjusted_p_value
3199	3199	0.318154	2.715913e-06	0.004987
14761	14761	0.280992	5.142592e-03	0.049101
7138	7138	0.278726	7.375657e-04	0.014671
3556	3556	0.273936	3.984796e-03	0.043334
20166	20166	0.259784	8.811341e-09	0.000105
10398	10398	0.244270	1.779855e-05	0.013705
12035	12035	0.243929	1.176882e-03	0.020853
14863	14863	0.239278	1.423557e-06	0.003398
429	429	0.238108	1.523923e-05	0.012543
8268	8268	0.231012	3.337037e-06	0.005310
2816	2816	0.220401	1.764780e-03	0.027038
4660	4660	0.218286	1.333546e-06	0.003398
1738	1738	0.216856	3.839610e-03	0.042549
2732	2732	0.216515	1.650005e-03	0.025776
4316	4316	0.214890	4.658422e-03	0.046668
7901	7901	0.206587	5.133722e-03	0.049078
19616	19616	0.203029	1.812718e-03	0.027508
13786	13786	0.199224	1.077914e-03	0.019478
5614	5614	0.197680	3.288980e-03	0.039431
13015	13015	0.196696	5.951336e-06	0.006518
2291	2291	0.192548	3.322186e-03	0.039571
15118	15118	0.192162	5.639210e-04	0.014099
4880	4880	0.190582	8.239417e-04	0.015977
9851	9851	0.189673	4.173626e-03	0.044534
11915	11915	0.189317	4.231305e-03	0.044909
4190	4190	0.189177	2.418579e-03	0.033003
3880	3880	0.183247	3.459367e-03	0.040417
7829	7829	0.183018	2.362376e-03	0.032512
1095	1095	0.181584	2.816471e-03	0.036125
7570	7570	0.180747	6.307096e-04	0.014099
2121	2121	0.176682	5.237473e-03	0.049552
21197	21197	0.173593	2.069923e-03	0.029977
2888	2888	0.172939	4.667000e-03	0.046668
17737	17737	0.170929	3.697425e-03	0.041710
12571	12571	0.170501	7.175236e-04	0.014356
13149	13149	0.170091	7.611678e-04	0.015003
1153	1153	0.169236	6.477091e-04	0.014099
13736	13736	0.168209	2.409547e-04	0.014099
14924	14924	0.167900	1.090412e-05	0.010011
12039	12039	0.166641	5.062840e-03	0.048802
10752	10752	0.165072	1.774120e-03	0.027146
5083	5083	0.162223	3.887547e-07	0.001722
12807	12807	0.161665	4.005063e-03	0.043475
2798	2798	0.159021	2.582538e-05	0.014099
11799	11799	0.157952	4.513033e-03	0.045899
13412	13412	0.155505	4.117366e-03	0.044291
12210	12210	0.154756	5.093894e-03	0.048950
14349	14349	0.154304	3.060888e-03	0.037603
11564	11564	0.149383	5.671465e-04	0.014099
22945	22945	0.148534	4.289313e-03	0.045223

Summary

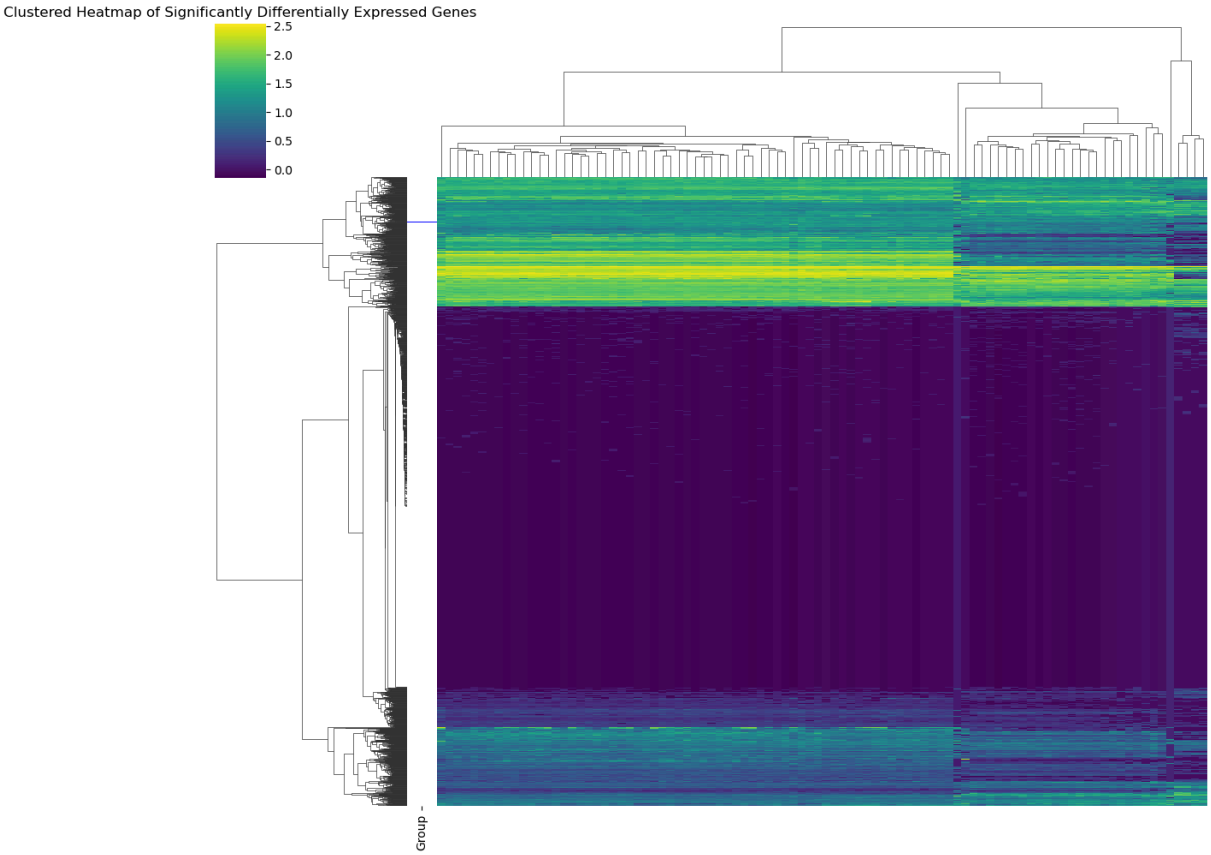
From the differential analysis and visualization with the volcano plot, I see that there are genes with a relatively high statistical significance showing upregulation response to heat stress. Most of the significant genes cluster near a log2 fold change---shown on the x-axis of the volcano plot---around zero. While this means that the differential expression is not very extreme across all significant genes (and genes in general), there are still genes with larger changes, suggesting they may be greatly influenced by heat. To show which genes are considered significant, I used an adjusted p-value of 0.05 and colored them in red, after testing correction with Benjamini-Hochberg. The statistically insignificant genes are colored gray and this means their expression changes could be due to random variation rather than heat stress. Going back to the fact that there are significant genes with upregulation responses, this demonstrates that changes in gene expression are triggered as part of the chicken's biological response to thermal stress and might potentially include things like stress response pathways, metabolic changes, and heat shock proteins.

Part 4

Heatmap

In [25]:
```
# filter significant genes (wtih adjusted p-value < 0.05)
significant_genes = results_df[results_df['adjusted_p_value'] < 0.05]['Gene'].tolis

# get expression data for only significant genes
significant_expression_data = log_expression_values.loc[significant_genes]

# display number of significantly differentially expressed genes
print(f"Number of significantly differentially expressed genes: {len(significant_ge

# create color map for groups (thermoneutral = blue, affected by Heat = red)
group_colors = metadata['Group'].replace({'Thermoneutral': 'blue', 'Affected by Hea

# surpress warning about clustering speed; I want a high resolution graph
warnings.filterwarnings("ignore", message="Clustering large matrix with scipy. Inst

# create clustered heatmap
sns.clustermap(
    significant_expression_data,
    cmap='viridis',
    row_colors=group_colors,
    xticklabels=False,
    yticklabels=False,
    figsize=(12, 10)
)

# display heatmap
plt.title('Clustered Heatmap of Significantly Differentially Expressed Genes')
plt.show()
```

```
Number of significantly differentially expressed genes: 2559
<Figure size 1000x600 with 0 Axes>
```



Summary

This enrichment analysis helped identify the processes and pathways most relevant to the high heat response in affected chickens. From the heatmap, the genes that are upregulated (more green) in the heat-stressed chickens may be involved in stress response pathways, while downregulated genes (more purple) could be linked to regular functions that are suppressed under stress conditions. Looking at the heatmap, there are not really any downregulated genes. Also, the hierarchical clustering on both rows--genes---and columns---samples---shows clear groupings and implies biological differences between the two groups (thermoneutral and heat affected).

Part 5

In [23]:
```
results_harry = []

# loop through each gene and perform Wilcoxon rank-sum test
for gene in log_expression_values.index:
    thermoneutral_values = log_expression_values.loc[gene, thermoneutral_samples]
    heat_stress_values = log_expression_values.loc[gene, heat_stress_samples]

    # perform test
    stat, p_value = ranksums(thermoneutral_values, heat_stress_values)

    # store result
    results_harry.append([gene, stat, p_value])
```

```
# make results into a proper DataFrame table
results_harry_df = pd.DataFrame(results_harry, columns=['Gene', 'Stat', 'p_value'])

# incorporate Benjamini-Hochberg correction for multiple testing
results_harry_df['adjusted_p_value'] = multipletests(results_harry_df['p_value'], m

# filter for significant results (adjusted p-value < 0.05)
significant_genes = results_harry_df[results_harry_df['adjusted_p_value'] < 0.05]

# display the top significant genes
print(significant_genes.sort_values(by='adjusted_p_value').head(50))

# output results to a CSV
results_harry_df.to_csv('wilcoxon_rank_sum_test_results.csv', index=False)
```

	Gene	Stat	p_value	adjusted_p_value
15142	15142	5.637550	1.724864e-08	0.000412
20166	20166	-5.387497	7.144547e-08	0.000781
335	335	-5.277626	1.308683e-07	0.000781
13015	13015	-5.315513	1.063575e-07	0.000781
3924	3924	-5.038939	4.681189e-07	0.001862
3109	3109	-5.050305	4.411045e-07	0.001862
4660	4660	-4.978320	6.413840e-07	0.002187
14863	14863	-4.883604	1.041644e-06	0.002488
3199	3199	-4.872238	1.103414e-06	0.002488
17843	17843	4.864660	1.146536e-06	0.002488
5083	5083	-4.898758	9.644421e-07	0.002488
4336	4336	-4.800253	1.584656e-06	0.002496
429	429	-4.788887	1.677092e-06	0.002496
4408	4408	4.754789	1.986542e-06	0.002496
1015	1015	-4.830562	1.361481e-06	0.002496
3136	3136	-4.754789	1.986542e-06	0.002496
11938	11938	-4.766155	1.877752e-06	0.002496
14924	14924	-4.781309	1.741572e-06	0.002496
5845	5845	-4.769943	1.842778e-06	0.002496
10252	10252	4.743423	2.101373e-06	0.002508
14093	14093	-4.720690	2.350454e-06	0.002672
12621	12621	-4.697958	2.627749e-06	0.002851
8268	8268	-4.675226	2.936295e-06	0.003047
6879	6879	-4.663860	3.103320e-06	0.003087
12417	12417	4.607030	4.084607e-06	0.003900
9047	9047	-4.535045	5.759111e-06	0.005287
6593	6593	4.519891	6.187158e-06	0.005470
12526	12526	-4.489581	7.136333e-06	0.005678
13515	13515	4.493370	7.010485e-06	0.005678
1097	1097	4.500947	6.765129e-06	0.005678
3651	3651	4.428962	9.468752e-06	0.007291
6171	6171	4.330457	1.488003e-05	0.011100
1662	1662	4.292570	1.766168e-05	0.012775
21326	21326	-4.273627	1.923189e-05	0.013502
4807	4807	4.266049	1.989648e-05	0.013569
9097	9097	-4.243317	2.202397e-05	0.014603
18410	18410	4.235740	2.278003e-05	0.014696
14259	14259	-4.186487	2.833050e-05	0.017191
20295	20295	4.190276	2.786156e-05	0.017191
13583	13583	-4.182698	2.880693e-05	0.017191
16672	16672	-3.387076	7.064169e-04	0.017972
16563	16563	-3.224163	1.263413e-03	0.017972
16565	16565	-3.428752	6.063635e-04	0.017972
16727	16727	-3.224163	1.263413e-03	0.017972
16718	16718	-3.224163	1.263413e-03	0.017972
16577	16577	-3.239318	1.198159e-03	0.017972
16729	16729	-3.224163	1.263413e-03	0.017972
16612	16612	-3.224163	1.263413e-03	0.017972
16701	16701	-3.610608	3.054796e-04	0.017972
16626	16626	-3.224163	1.263413e-03	0.017972

***Tried my best but unable to finish parts 6-9**

In [ ]: