

# GWAS\_encode\_and\_cluster

April 24, 2022

```
[1]: from typing import Callable, Dict, List
import pandas as pd
import numpy as np
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram
import math
```

## 1 GWAS summary statistic clustering

Data will be encoded into a standardized representation and then clustered to derive potential condition associations.

### 1.1 Load & encode data

```
[2]: METADATA_FILE = 'gwas_trait_metadata.csv'
CLEANED_FILE_SUFFIX = '.csv'
UNKNOWN_GENE = "UNKNOWN"

metadata_df = pd.read_csv(METADATA_FILE)
all_traits = metadata_df['Trait'].tolist()
print(all_traits)
```

```
['attention deficit hyperactivity disorder', 'alzheimer disease', 'anxiety disorder', 'autism spectrum disorder', 'bipolar disorder', 'drug dependence', 'eating disorder', 'personality disorder', 'schizophrenia', 'tourette syndrome', 'unipolar depression']
```

```
[3]: def trait_to_cleaned_filename(trait):
    return trait.replace(" ", "_") + CLEANED_FILE_SUFFIX

def filter_unknown_genes(df):
```

```

    return df.loc[df['gene'] != UNKNOWN_GENE]

dfs = []
for trait in all_traits:
    df = pd.read_csv(trait_to_cleaned_filename(trait))
    df = filter_unknown_genes(df)
    if trait == 'attention deficit hyperactivity disorder':
        trait = 'ADHD'
    df['parent_trait'] = trait
    dfs.append(df)

```

### 1.1.1 Naive encoding

Just use 1-hot encoding of gene implication (i.e. number all genes implicated in the given conditions from 0...N-1. Then create an N-dimensional vector for each condition where element i is 1 if the condition is associated with that gene, 0 if not). The hypothesis is that similar conditions have implicated gene overlap.

```

[4]: def create_gene_to_index_dict(all_data_df: pd.DataFrame) -> Dict[str, int]:
    """Maps gene to index such that each gene is assigned a unique index."""
    all_genes = set()
    genes = all_data_df['gene'].unique()
    [all_genes.add(gene) for gene in genes]

    all_genes_list = list(all_genes)
    num_genes = len(all_genes_list)
    gene_to_index = {all_genes_list[i]: i for i in range(num_genes)}
    return gene_to_index

gene_to_index = create_gene_to_index_dict(pd.concat([df for df in dfs]))
print(f"Found {len(gene_to_index.keys())} total genes.")

```

Found 3259 total genes.

```

[5]: def encode_condition_df(df, gene_to_index):
    num_genes = len(gene_to_index.keys())
    vec = np.zeros(num_genes)
    condition_genes = df['gene'].unique()
    for gene in condition_genes:
        gene_index = gene_to_index[gene]
        vec[gene_index] = 1
    return vec

# Small test:

```

```

apoe_index = gene_to_index['APOE']
df = pd.DataFrame([{'variant_and_allele': 'test', 'p_value': 0.01, 'trait': 'test', 'gene': 'APOE'}])
encoding = encode_condition_df(df, gene_to_index)
assert encoding[apoe_index] == 1
assert sum(encoding) == 1

```

```

[6]: # Encode them all!
encodings_vertical = pd.DataFrame({df['parent_trait'].unique()[0]:
    encode_condition_df(df, gene_to_index) for df in dfs})
encodings_vertical

```

```

[6]:
      ADHD  alzheimer disease  anxiety disorder  autism spectrum disorder \
0      0.0                1.0                0.0                0.0
1      1.0                0.0                0.0                0.0
2      0.0                0.0                0.0                0.0
3      0.0                0.0                0.0                0.0
4      0.0                0.0                0.0                0.0
...
3254    0.0                0.0                0.0                0.0
3255    1.0                0.0                0.0                0.0
3256    0.0                0.0                0.0                1.0
3257    0.0                0.0                0.0                0.0
3258    0.0                0.0                0.0                0.0

      bipolar disorder  drug dependence  eating disorder \
0                0.0                0.0                0.0
1                0.0                0.0                0.0
2                0.0                0.0                0.0
3                0.0                0.0                0.0
4                0.0                0.0                0.0
...
3254                1.0                0.0                0.0
3255                0.0                0.0                0.0
3256                0.0                0.0                0.0
3257                0.0                0.0                0.0
3258                0.0                0.0                0.0

      personality disorder  schizophrenia  tourette syndrome \
0                0.0                0.0                0.0
1                0.0                1.0                0.0
2                0.0                0.0                0.0
3                0.0                0.0                0.0
4                0.0                1.0                0.0
...
3254                0.0                0.0                0.0
3255                0.0                0.0                0.0

```

3256	0.0	0.0	0.0
3257	0.0	0.0	0.0
3258	0.0	1.0	0.0

unipolar depression	
0	0.0
1	0.0
2	1.0
3	1.0
4	0.0
...	...
3254	0.0
3255	1.0
3256	0.0
3257	1.0
3258	1.0

[3259 rows x 11 columns]

```
[7]: # Actually needs to have vectors as rows not columns:
encodings = encodings_vertical.T
encodings
```

```
[7]:
```

	0	1	2	3	4	5	6	7	\
ADHD	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
alzheimer disease	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	
anxiety disorder	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
autism spectrum disorder	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
bipolar disorder	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
drug dependence	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
eating disorder	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
personality disorder	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
schizophrenia	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	
tourette syndrome	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
unipolar depression	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	

	8	9	...	3249	3250	3251	3252	3253	3254	\
ADHD	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
alzheimer disease	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
anxiety disorder	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
autism spectrum disorder	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	
bipolar disorder	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	1.0	
drug dependence	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
eating disorder	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
personality disorder	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
schizophrenia	0.0	0.0	...	0.0	1.0	1.0	0.0	0.0	0.0	
tourette syndrome	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

unipolar depression	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0
---------------------	-----	-----	-----	-----	-----	-----	-----	-----	-----

	3255	3256	3257	3258
ADHD	1.0	0.0	0.0	0.0
alzheimer disease	0.0	0.0	0.0	0.0
anxiety disorder	0.0	0.0	0.0	0.0
autism spectrum disorder	0.0	1.0	0.0	0.0
bipolar disorder	0.0	0.0	0.0	0.0
drug dependence	0.0	0.0	0.0	0.0
eating disorder	0.0	0.0	0.0	0.0
personality disorder	0.0	0.0	0.0	0.0
schizophrenia	0.0	0.0	0.0	1.0
tourette syndrome	0.0	0.0	0.0	0.0
unipolar depression	1.0	0.0	1.0	1.0

[11 rows x 3259 columns]

## 1.2 Naive Clustering (no special filtering)

```
[8]: # See docs here:
# https://scikit-learn.org/stable/modules/generated/sklearn.cluster.
# AgglomerativeClustering.html
model = AgglomerativeClustering(distance_threshold=0,
                                n_clusters=None,
                                linkage='ward')
clustering = model.fit(encodings)
```

```
[9]: # This code is from the scikit-learn examples!
# https://scikit-learn.org/stable/auto_examples/cluster/
# plot_agglomerative_dendrogram.
# html#sphx-glr-auto-examples-cluster-plot-agglomerative-dendrogram-py

def plot_dendrogram(model, conditions, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1 # leaf node
            else:
                current_count += counts[child_idx - n_samples]
```

```

        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)

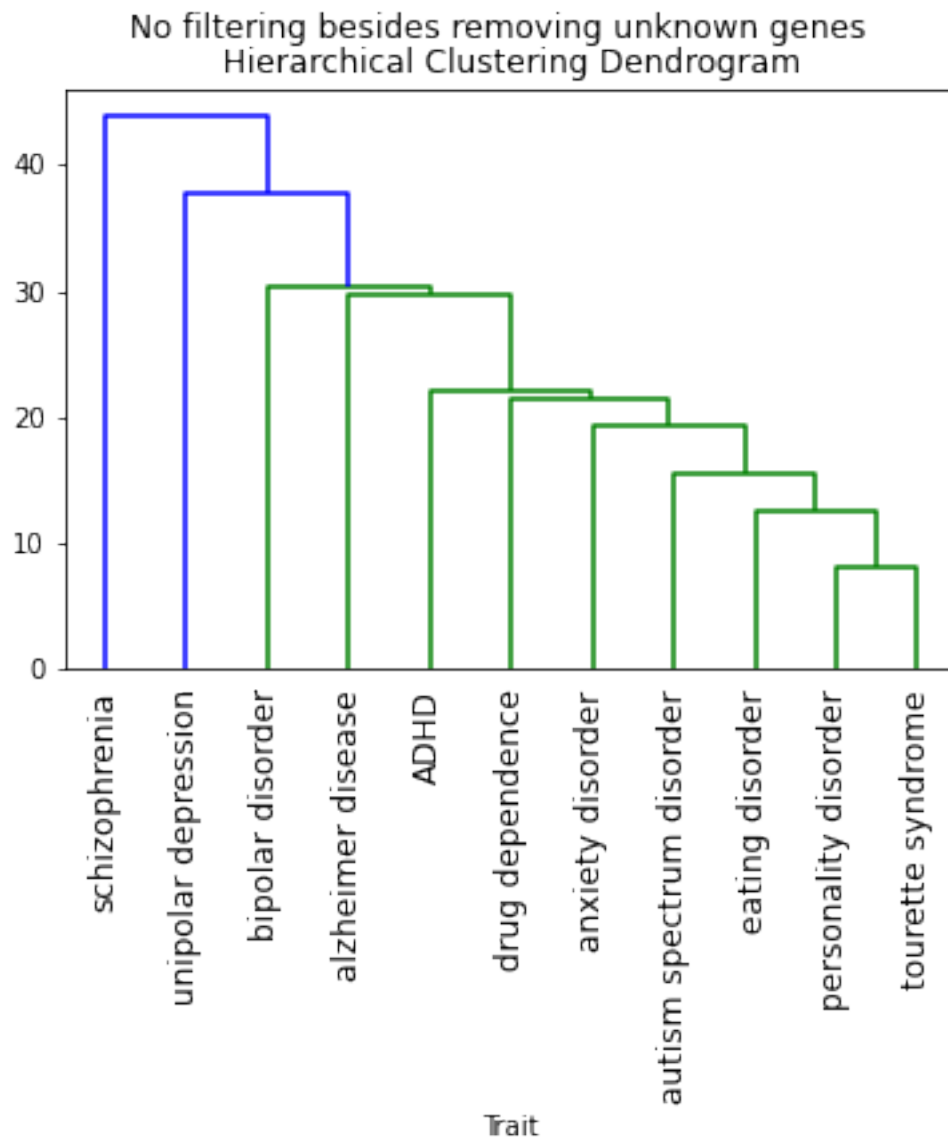
    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs, labels=conditions, leaf_rotation=90)

```

```

[10]: plt.suptitle('No filtering besides removing unknown genes')
plt.title("Hierarchical Clustering Dendrogram")
# plot all levels of the dendrogram
plot_dendrogram(model, encodings_vertical.columns, truncate_mode="level", p=11)
plt.xlabel("Trait")
plt.show()

```



### 1.2.1 Observations

It seems schizophrenia is the least similar to the others. This is a little surprising given that in my literature review I saw many mentions of Schizophrenia having overlap with other mental illnesses. The results may be skewed at this time because there is more data for schizophrenia.

The telescoping shape also seems peculiar (as opposed to distinct subgroups). The ordering seems to be (from left to right) traits with the most data points to traits with the least. Let's double-check:

```
[11]: trait_to_num_data_points = {}
      for df in dfs:
```

```

trait = df['parent_trait'].unique()[0]
num_rows = len(df)
trait_to_num_data_points[trait] = num_rows

{k: v for v, k in sorted(
    trait_to_num_data_points.items(), key=lambda item: item[1])}

```

```

[11]: {27: 'personality disorder',
      37: 'tourette syndrome',
      109: 'eating disorder',
      163: 'autism spectrum disorder',
      234: 'anxiety disorder',
      317: 'drug dependence',
      380: 'ADHD',
      724: 'alzheimer disease',
      748: 'bipolar disorder',
      1142: 'unipolar depression',
      1954: 'schizophrenia'}

```

Yes, the clustering hierarchy almost perfectly resembles how many data points there are per trait. This makes sense based on vector magnitudes.

### 1.3 Less naive clustering

Drop personality disorder and tourette syndrome as they likely don't have enough data to be meaningful compared to the other traits (also, from the post-cleaning analysis, it seems personality disorder variants have high p-values compared to all other conditions).

```

[12]: excluded_traits = ['personality disorder', 'tourette syndrome']
all_df = pd.concat(df for df in dfs if df['parent_trait'].unique()[0] not in
    ↪excluded_traits)
all_df['parent_trait'].unique()

```

```

[12]: array(['ADHD', 'alzheimer disease', 'anxiety disorder',
            'autism spectrum disorder', 'bipolar disorder', 'drug dependence',
            'eating disorder', 'schizophrenia', 'unipolar depression'],
          dtype=object)

```

```

[13]: all_df['tissues'] = all_df['tissues'].fillna('')

```

```

[14]: # Combine some of the code above into a reusable function
def gene_based_hierachical_clustering(all_df, title):
    traits = all_df['parent_trait']
    trait_dfs = [all_df.loc[all_df['parent_trait'] == trait] for trait in traits]
    encodings_vertical = pd.DataFrame({df['parent_trait'].unique()[0]:
    ↪encode_condition_df(df, gene_to_index) for df in trait_dfs})

```



```

encodings = encodings_vertical.T
model = AgglomerativeClustering(distance_threshold=0,
                                n_clusters=None,
                                linkage='ward')

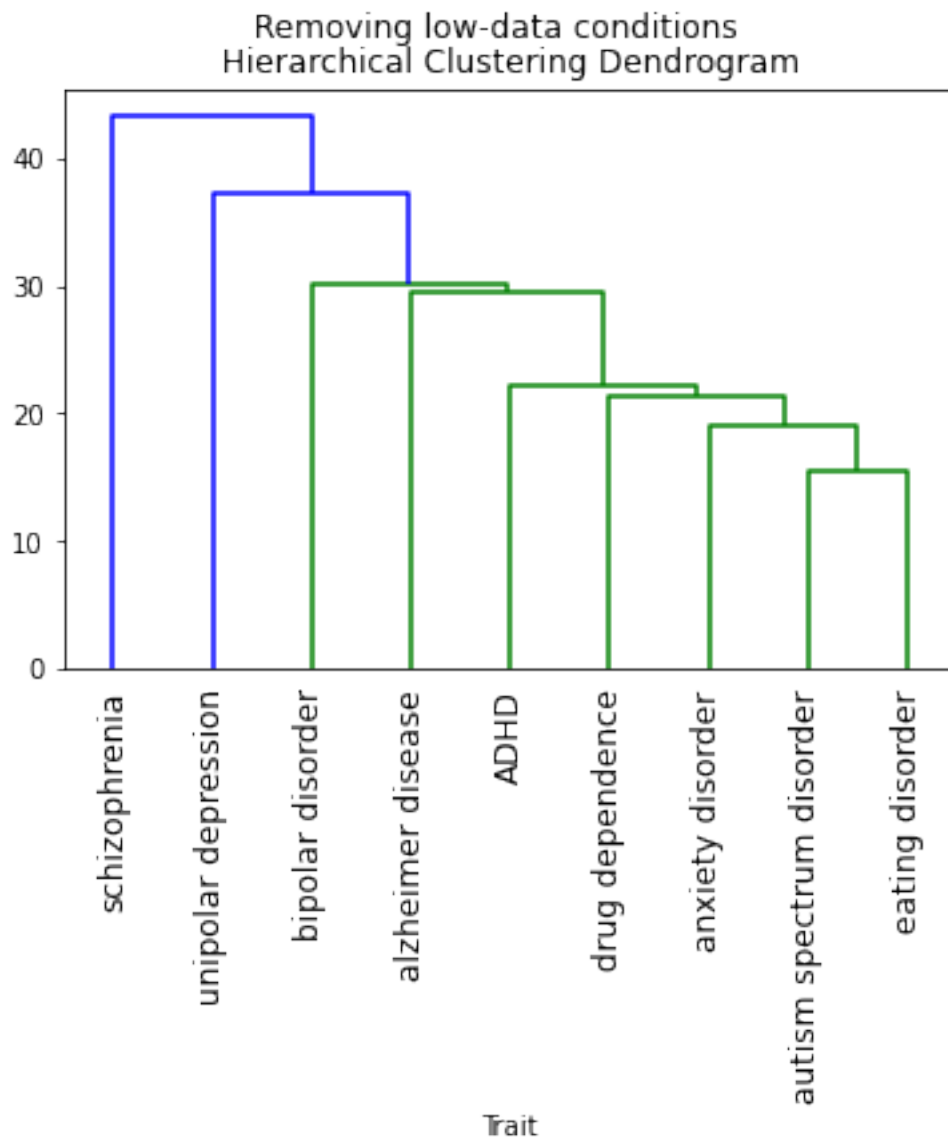
model.fit(encodings)

plt.suptitle(title)
plt.title("Hierarchical Clustering Dendrogram")
# plot all levels of the dendrogram
plot_dendrogram(model, encodings_vertical.columns, truncate_mode="level",
→p=11)
plt.xlabel('Trait')
plt.show()

```

Try again to see if there's any difference... One hypothesis is that only genes that appear in multiple traits are relevant to include in the encoding and clustering, because genes that are only implicated by one trait only show 'difference' from other traits... but on the other hand, if a trait has only one gene in common with others and hundreds of other unique implicated genes, it could be informative that the trait is actually quite different. Let's try both, starting with all genes.

```
[15]: gene_based_hierachical_clustering(all_df, 'Removing low-data conditions')
```



Still in order of data points. Probably need to do some sort of normalization based on data size. Let's first filter out genes that only appear in one trait though. If we're measuring similarity, unique genes don't really add information, or at least it's not clear how to account for that in an unbiased way.

```
[16]: all_genes = all_df['gene'].unique()
      # Stores genes that only appear for one trait
      genes_to_remove = []
      for gene in all_genes:
          num_traits = len(all_df.loc[all_df['gene'] == gene])
          if num_traits < 2:
              genes_to_remove.append(gene)
```

```
print(f'{len(genes_to_remove)} / {len(all_genes)} only appear in a single_  
↳trait')
```

2227 / 3213 only appear in a single trait

That's a good chunk of the genes, but let's proceed...

```
[17]: print(f'Initial df size: {len(all_df)}')
```

```
def is_common_gene(gene):  
    return gene not in genes_to_remove
```

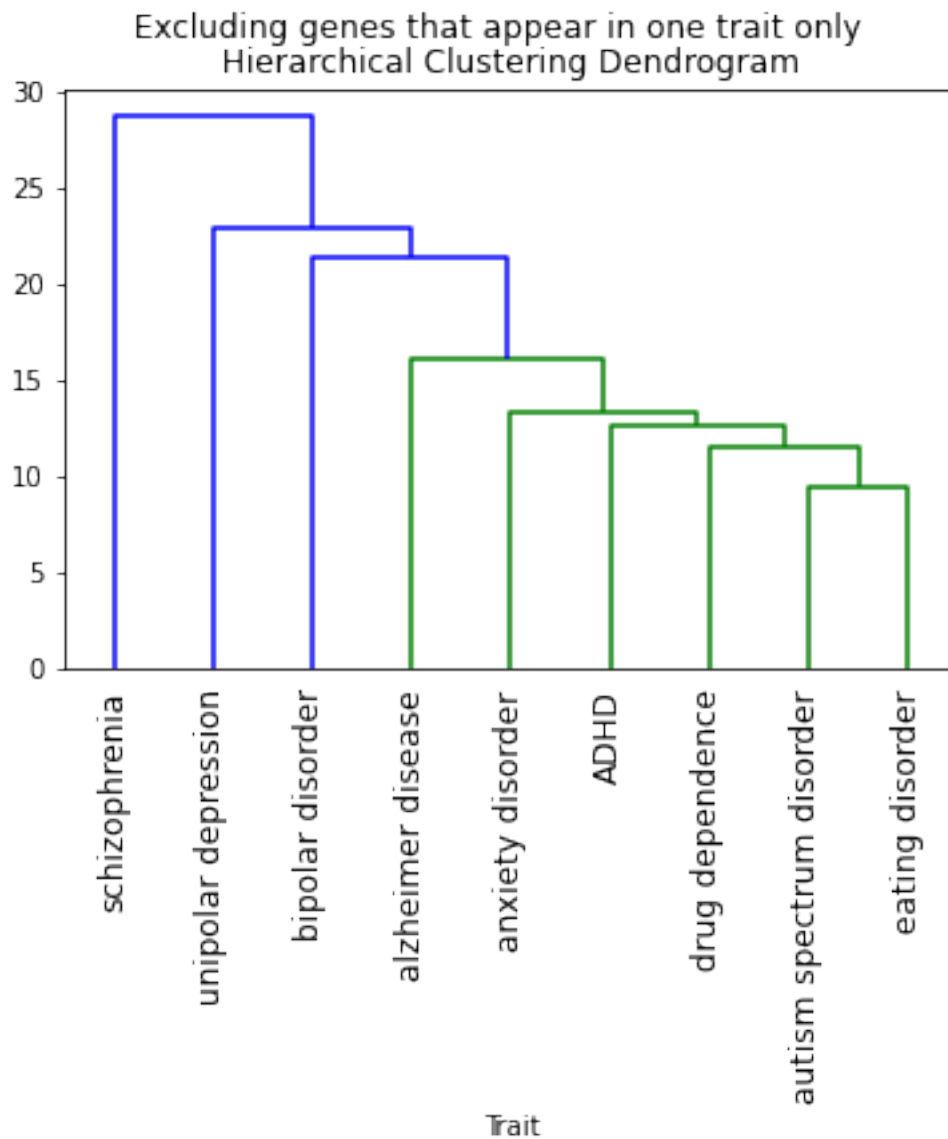
```
is_common_index = all_df['gene'].map(is_common_gene)  
all_df = all_df.loc[is_common_index]  
len(all_df)
```

Initial df size: 5771

[17]: 3544

Actually not that many data points were removed. Finally cluster this data:

```
[18]: gene_based_hierachical_clustering(all_df,  
                                          'Excluding genes that appear in one trait_  
↳only')
```



Well, a slightly different order of the telescoping effect, but still no distinct clusters.

#### 1.4 Account for data size

Vector magnitude (number of data points per condition) seems to be biasing clustering results. Try taking top X data points per trait where X is the number of data points for the trait with the smallest dataset size, and 'top' means lowest p-value variant associations.

```
[19]: trait_group_df = all_df.groupby('parent_trait').size()
      trait_group_df
```

```
[19]: parent_trait
      ADHD                209
      alzheimer disease    382
      anxiety disorder     115
      autism spectrum disorder  80
      bipolar disorder     511
      drug dependence      141
      eating disorder       41
      schizophrenia        1392
      unipolar depression   673
      dtype: int64
```

```
[20]: smallest_trait_size = trait_group_df.min()
      smallest_trait_size
```

```
[20]: 41
```

```
[21]: def filter_to_smallest_pvalues_same_sizes(in_df: pd.DataFrame) -> pd.DataFrame:
      """Creates DF with same smallest N p-value entries per trait.

      Specifically, finds smallest data count per-trait, and takes that many rows
      for each trait, where rows are the lowest p-values."""
      trait_group_df = in_df.groupby('parent_trait').size()
      smallest_trait_size = trait_group_df.min()
      new_df = pd.DataFrame()
      for trait in in_df['parent_trait'].unique():
          trait_df = in_df[in_df['parent_trait'] == trait]
          trait_df.sort_values(by='p_value',
                              inplace=True,
                              ascending=True)
          trait_df = trait_df.iloc[range(0, smallest_trait_size)]
          new_df = pd.concat([new_df, trait_df])
      return new_df

      new_df = filter_to_smallest_pvalues_same_sizes(all_df)
      trait_group_df = new_df.groupby('parent_trait').size()
      trait_group_df
```

/usr/local/lib/python3.7/dist-packages/pandas/util/\_decorators.py:311:

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

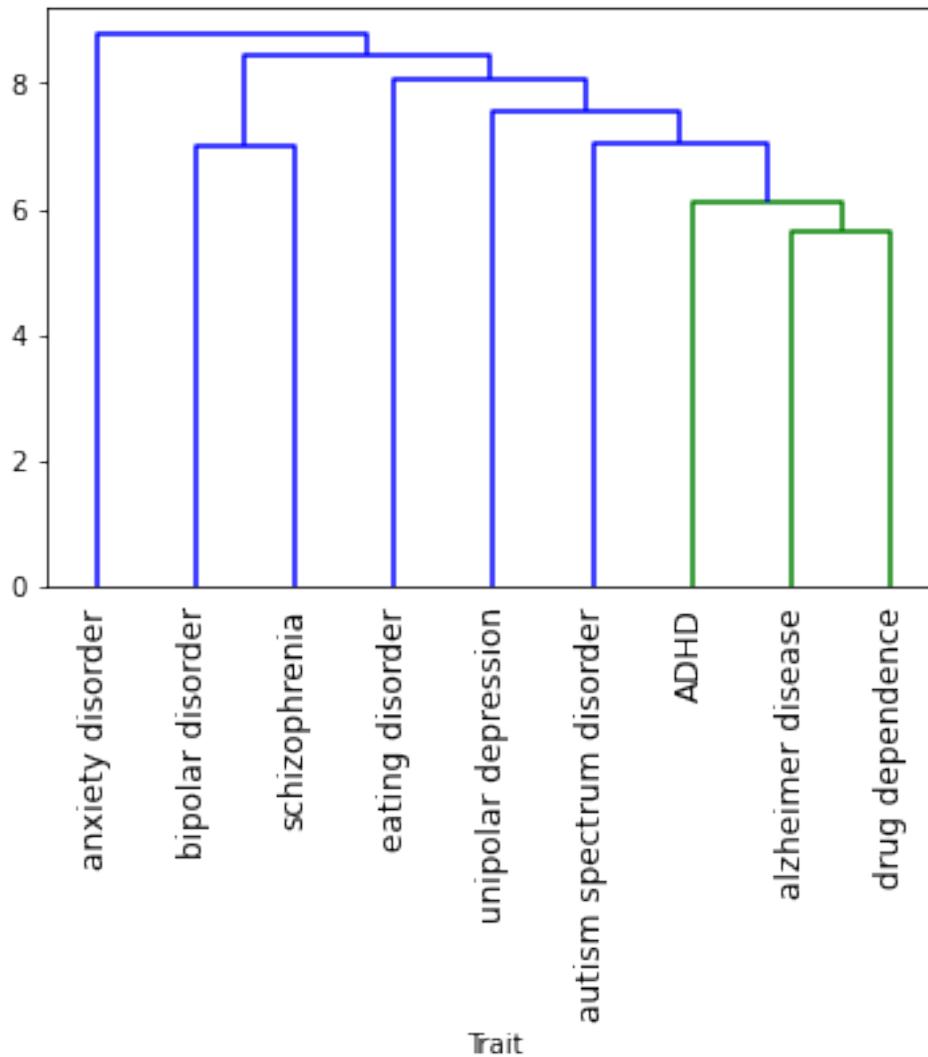
```
    return func(*args, **kwargs)
```

```
[21]: parent_trait
      ADHD 41
      alzheimer disease 41
      anxiety disorder 41
      autism spectrum disorder 41
      bipolar disorder 41
      drug dependence 41
      eating disorder 41
      schizophrenia 41
      unipolar depression 41
      dtype: int64
```

Now try clustering with most significant associations for common genes per trait.

```
[22]: gene_based_hierachical_clustering(
      new_df,
      'Most significant associations per trait; common genes only')
```

Most significant associations per trait; common genes only  
Hierarchical Clustering Dendrogram



Well, it's not sorted by magnitude anymore! Bipolar and schizophrenia are clustered together, which is also reported in the literature. I need to look into this a bit, but the association between Alzheimer's and drug dependence seems plausible too.

## 1.5 Experiment with different gene-based encodings

```
[23]: # Abstract the function from above slightly more to take in an encoding function
def hierachical_clustering(
    all_df: pd.DataFrame,
    title: str,
    encode: Callable[[pd.DataFrame], pd.DataFrame]) -> None:
```

```

encodings_vertical = encode(all_df)
encodings = encodings_vertical.T
model = AgglomerativeClustering(distance_threshold=0,
                                n_clusters=None,
                                linkage='ward')

model.fit(encodings)

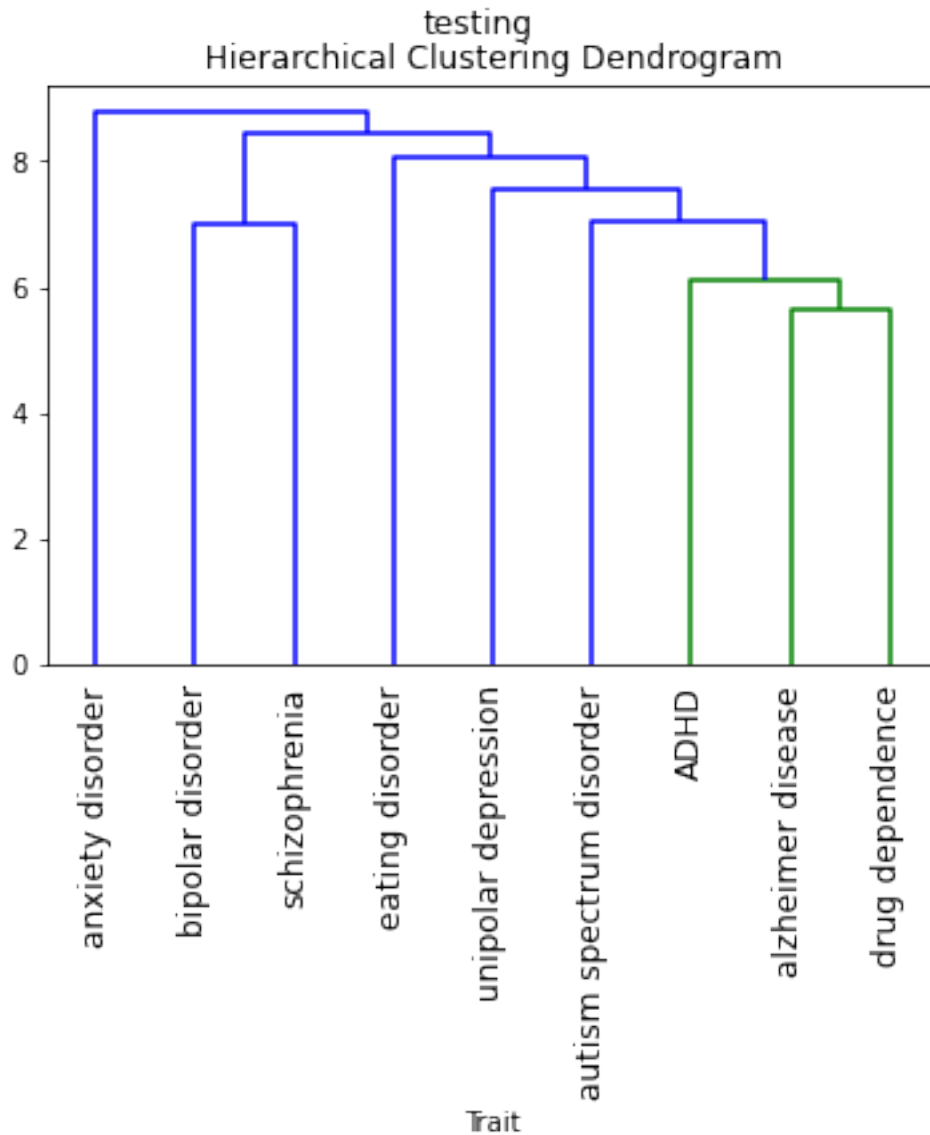
plt.suptitle(title)
plt.title("Hierarchical Clustering Dendrogram")
# plot all levels of the dendrogram
plot_dendrogram(model, encodings_vertical.columns, truncate_mode="level",
→p=11)
plt.xlabel('Trait')
plt.show()

# Copy code from above to test generalized clustering function.
def one_hot_gene_encoding(in_df: pd.DataFrame) -> pd.DataFrame:
    traits = in_df['parent_trait']
    trait_dfs = [in_df.loc[in_df['parent_trait'] == trait] for trait in traits]
    return pd.DataFrame({df['parent_trait'].unique()[0]: encode_condition_df(df,
→gene_to_index) for df in trait_dfs})

hierachical_clustering(new_df, 'testing', one_hot_gene_encoding)

```





```
[24]: def implicated_gene_weighted_encoding(in_df: pd.DataFrame) -> pd.DataFrame:
    """Encodes trait vectors where elements are relative gene implications.

    Specifically, for each trait, creates a vector (V1, V2... Vn)
    where  $V_i = G_i / \text{Max}(G_i)$  where a trait has  $G_i$  variants implicated with  $i$ th gene
    and  $n$  is total genes in `in_df`.

    E.g. if trait has gene associations:
        gene1  gene2  gene3
        1      3      5

    Then encoding is:
```

```

(.2, .6, 1)

DataFrame is returned with traits as column headers and vectors as columns.
"""
gene_to_index = create_gene_to_index_dict(in_df)
traits = in_df['parent_trait']
trait_dfs = [in_df.loc[in_df['parent_trait'] == trait] for trait in traits]
trait_to_encoding = {}
for trait_df in trait_dfs:
    trait = trait_df['parent_trait'].unique()[0]
    gene_counts_encoding = weighted_gene_encoding(trait_df, gene_to_index)
    max_val = max(gene_counts_encoding)
    gene_counts_encoding = gene_counts_encoding / max_val
    trait_to_encoding[trait] = gene_counts_encoding
return pd.DataFrame(trait_to_encoding)

def weighted_gene_encoding(
    trait_df: pd.DataFrame, gene_to_index: Dict[str, int]) -> np.ndarray:
    num_genes = len(gene_to_index.keys())
    vec = np.zeros(num_genes)
    trait_genes = trait_df['gene']
    for gene in trait_genes:
        gene_index = gene_to_index[gene]
        vec[gene_index] += 1
    return vec

```

```

[25]: # Test encoding function:
implicated_gene_weighted_encoding(all_df)

```

```

[25]:
      ADHD  alzheimer disease  anxiety disorder  autism spectrum disorder \
0    0.000000      0.00000      0.0          0.00
1    0.000000      0.00000      0.0          0.00
2    0.266667      0.00000      0.0          0.00
3    0.000000      0.00000      0.0          0.00
4    0.000000      0.00000      0.0          0.00
..      ...      ...      ...      ...
981  0.000000      0.00000      0.0          0.25
982  0.000000      0.00000      0.0          0.00
983  0.000000      0.00000      0.0          0.00
984  0.000000      0.00000      0.0          0.00
985  0.000000      0.52381      0.0          0.00

      bipolar disorder  drug dependence  eating disorder  schizophrenia \
0          0.0          0.0          0.0      0.095238
1          0.1          0.0          0.0      0.047619
2          0.0          0.0          0.0      0.047619

```

```

3          0.4          0.0          0.0          0.000000
4          0.0          0.0          0.0          0.000000
..          ...          ...          ...          ...
981         0.1          0.0          0.0          0.000000
982         0.0          0.0          0.0          0.047619
983         0.1          0.0          0.0          0.047619
984         0.0          0.0          0.0          0.095238
985         0.0          0.0          0.0          0.000000

```

```

      unipolar depression
0          0.000000
1          0.000000
2          0.000000
3          0.000000
4          0.142857
..          ...
981         0.000000
982         0.071429
983         0.000000
984         0.000000
985         0.000000

```

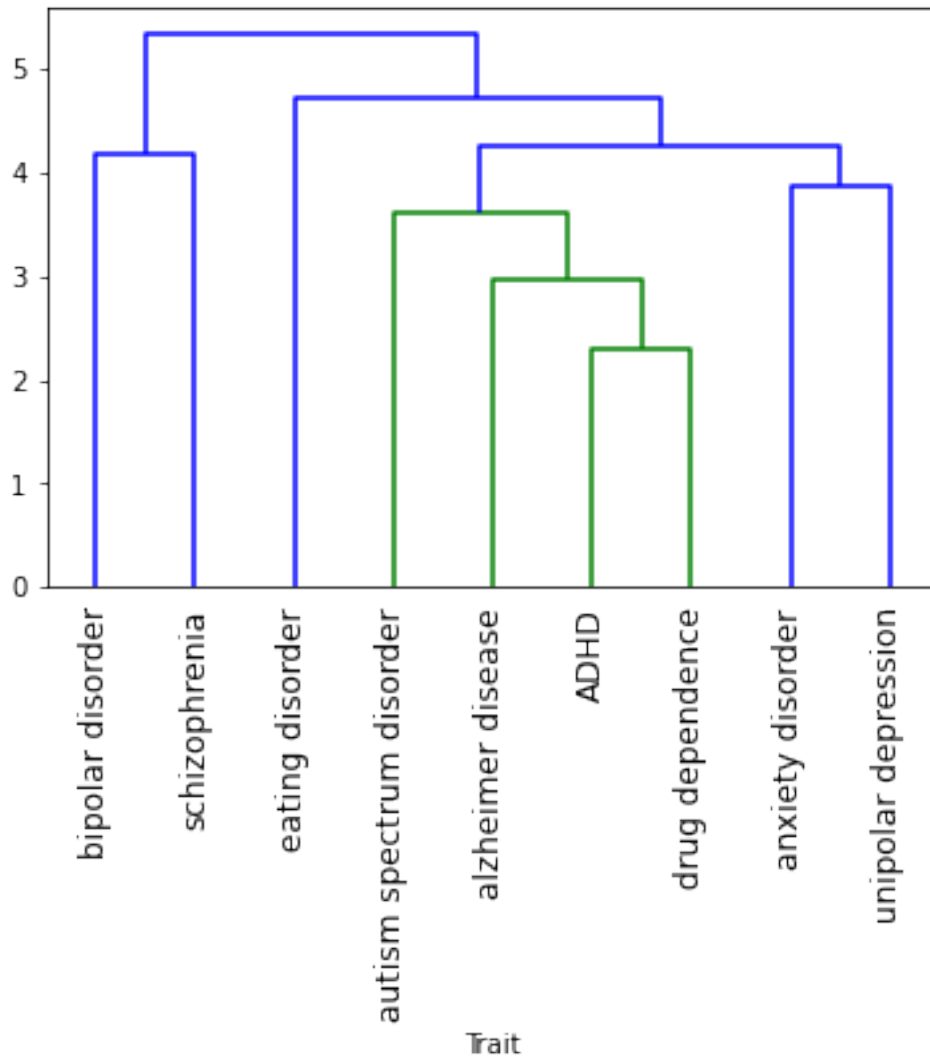
```
[986 rows x 9 columns]
```

```
[26]: implicated_gene_weighted_encoding(all_df)['schizophrenia'].unique()
```

```
[26]: array([0.0952381 , 0.04761905, 0.          , 0.23809524, 0.38095238,
            0.14285714, 0.42857143, 0.33333333, 0.28571429, 0.19047619,
            0.47619048, 0.71428571, 0.52380952, 1.          ])
```

```
[27]: hierachical_clustering(
      all_df,
      'Implicated-Gene-Weighted Hierarchical Clustering',
      implicated_gene_weighted_encoding)
```

Implicated-Gene-Weighted Hierarchical Clustering  
Hierarchical Clustering Dendrogram



Observations:

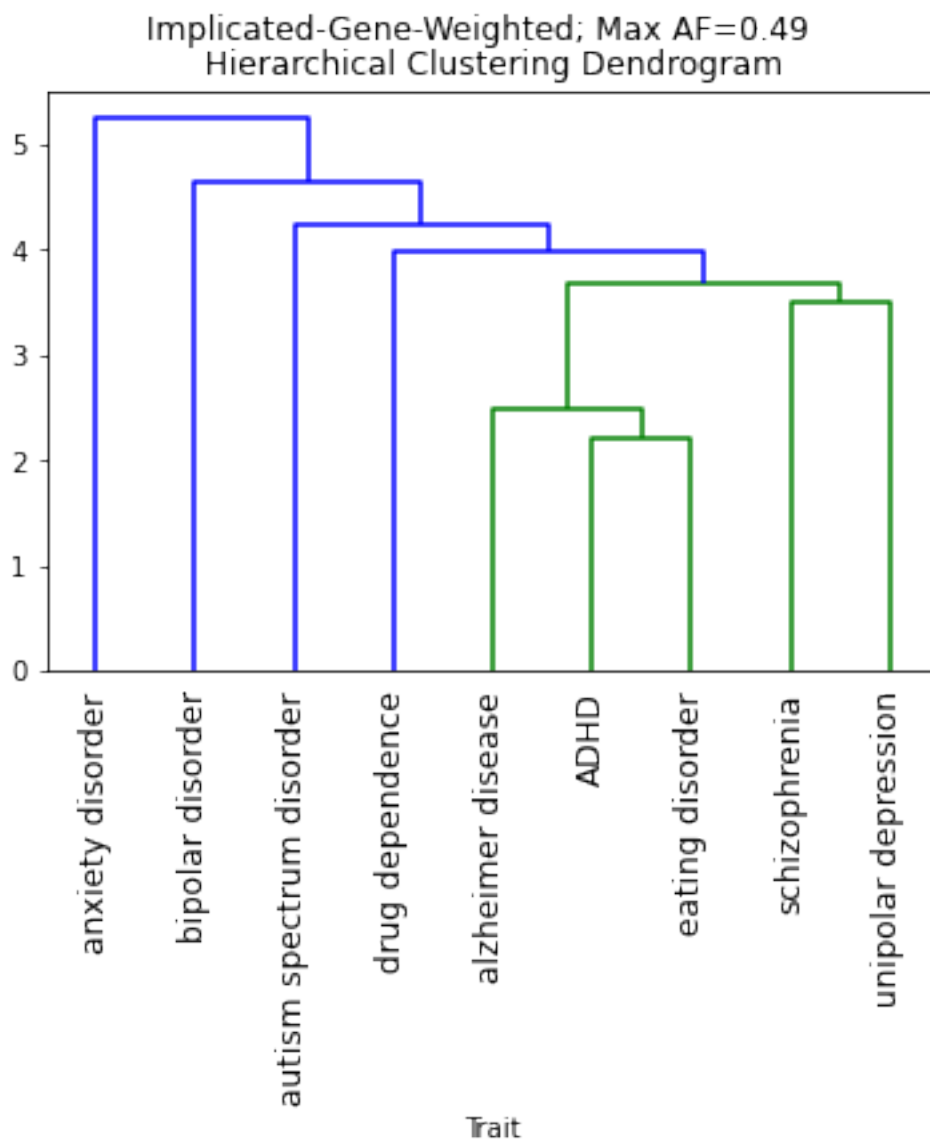
- Seems quite similar to the previous one, except anxiety disorder and unipolar depression are adjacent here.
- In the previous clustering, anxiety disorder was separated from all other traits, whereas here bipolar disorder and schizophrenia are clustered together and away from all others.
- I think this encoding is better in that it utilizes as much data as possible (not filtering to min p-values, just removing low-data conditions) but still accounts for bias in varying amounts of data for each condition.

### 1.5.1 Try same approach with AF filtering first.

Under the assumption these disorders affect a minority of the population, filter to variants with < 0.5 AF.

```
[28]: def filter_by_af(
        in_df: pd.DataFrame,
        max_af = None,
        min_af = None) -> pd.DataFrame:
    """Returns copy of `in_df` where `af` column is bounded by min/max AF."""
    out_df = in_df.copy()
    if max_af is not None:
        out_df = out_df[out_df['af'] <= max_af]
    if min_af is not None:
        out_df = out_df[out_df['af'] >= min_af]
    return out_df

# If we're utilizing AF, need to exclude those with unknown AF value,
# i.e. those with values of -1.
hierachical_clustering(
    filter_by_af(all_df, min_af=0.0, max_af=.49),
    'Implicated-Gene-Weighted; Max AF=0.49',
    implicated_gene_weighted_encoding)
```



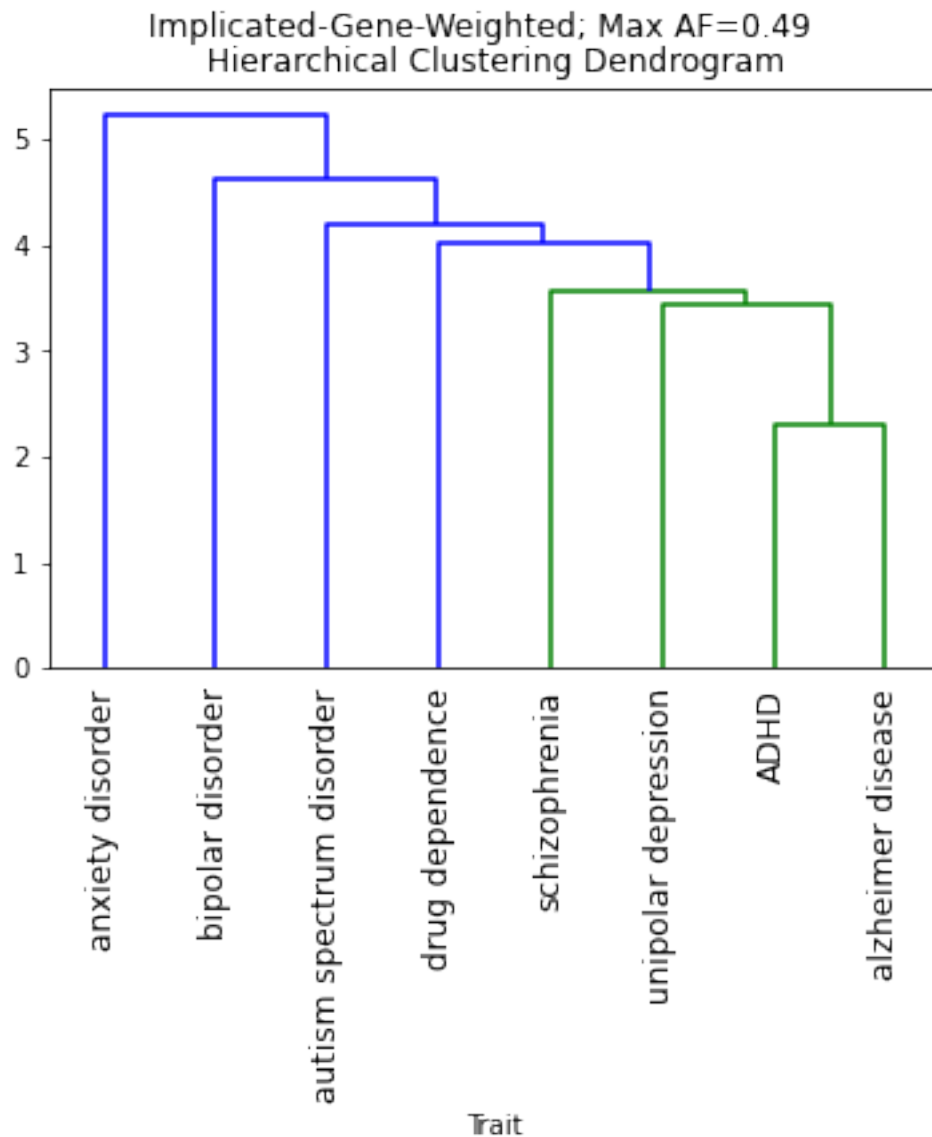
Similar to previous except eating disorder is now closer to ADHD and alzheimer, and schizophrenia and unipolar depression are less separated from the rest of the traits. Recalling from the post-cleaning analysis though, most of eating disorder variants had  $AF > 0.5$ , so this may be noise from low-data.

```
[29]: minority_df = filter_by_af(all_df, min_af=0.0, max_af=.49)
      len(minority_df[minority_df['parent_trait'] == 'eating disorder'])
```

[29]: 10

Yeah... may be best to exclude eating disorder as well since these were near-outliers according to the post-cleaning analysis.

```
[30]: minority_df = minority_df[minority_df['parent_trait'] != 'eating disorder']
      hierachical_clustering(
        minority_df,
        'Implicated-Gene-Weighted; Max AF=0.49',
        implicated_gene_weighted_encoding)
```



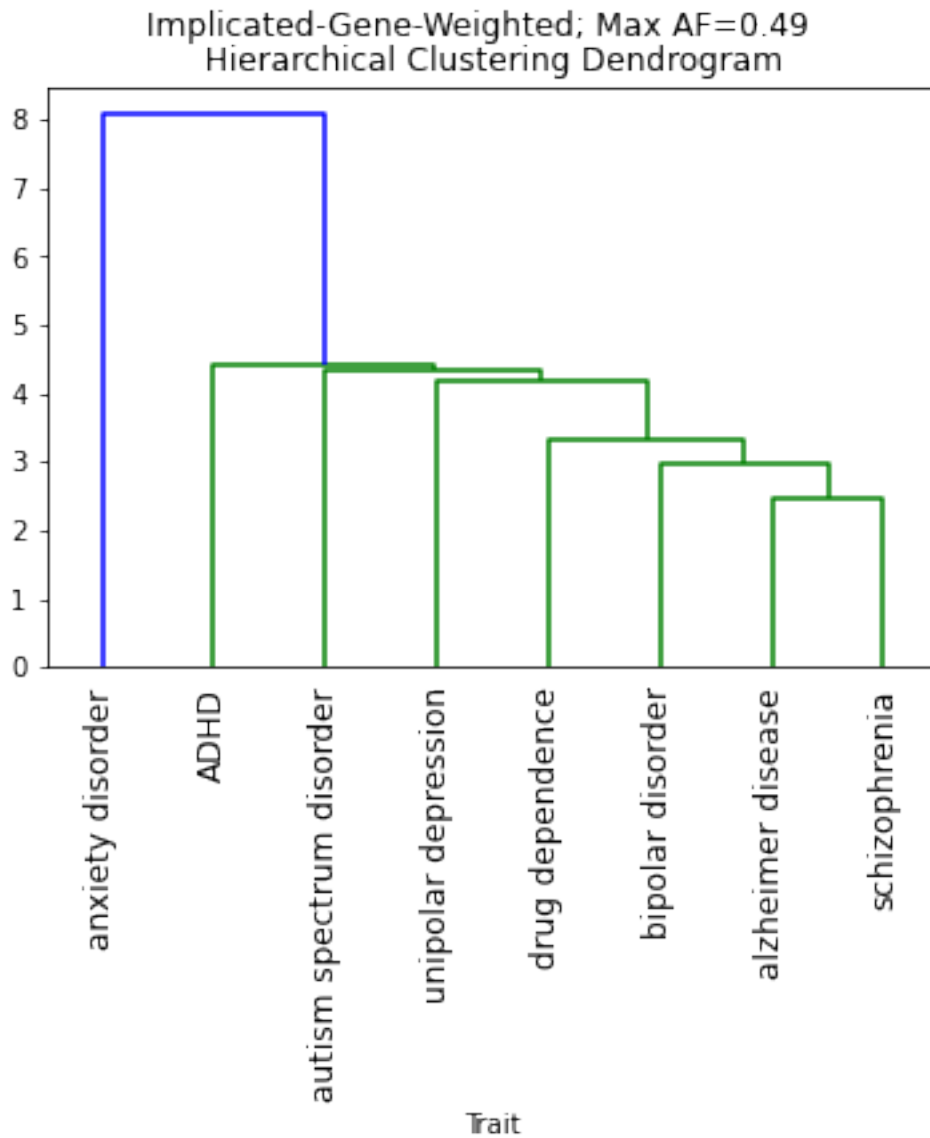
Getting the telescoping effect again... try normalizing data points per trait again.

```
[31]: hierachical_clustering(
      filter_to_smallest_pvalues_same_sizes(minority_df),
      'Implicated-Gene-Weighted; Max AF=0.49',
      implicated_gene_weighted_encoding)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/util/_decorators.py:311:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
return func(*args, **kwargs)
```

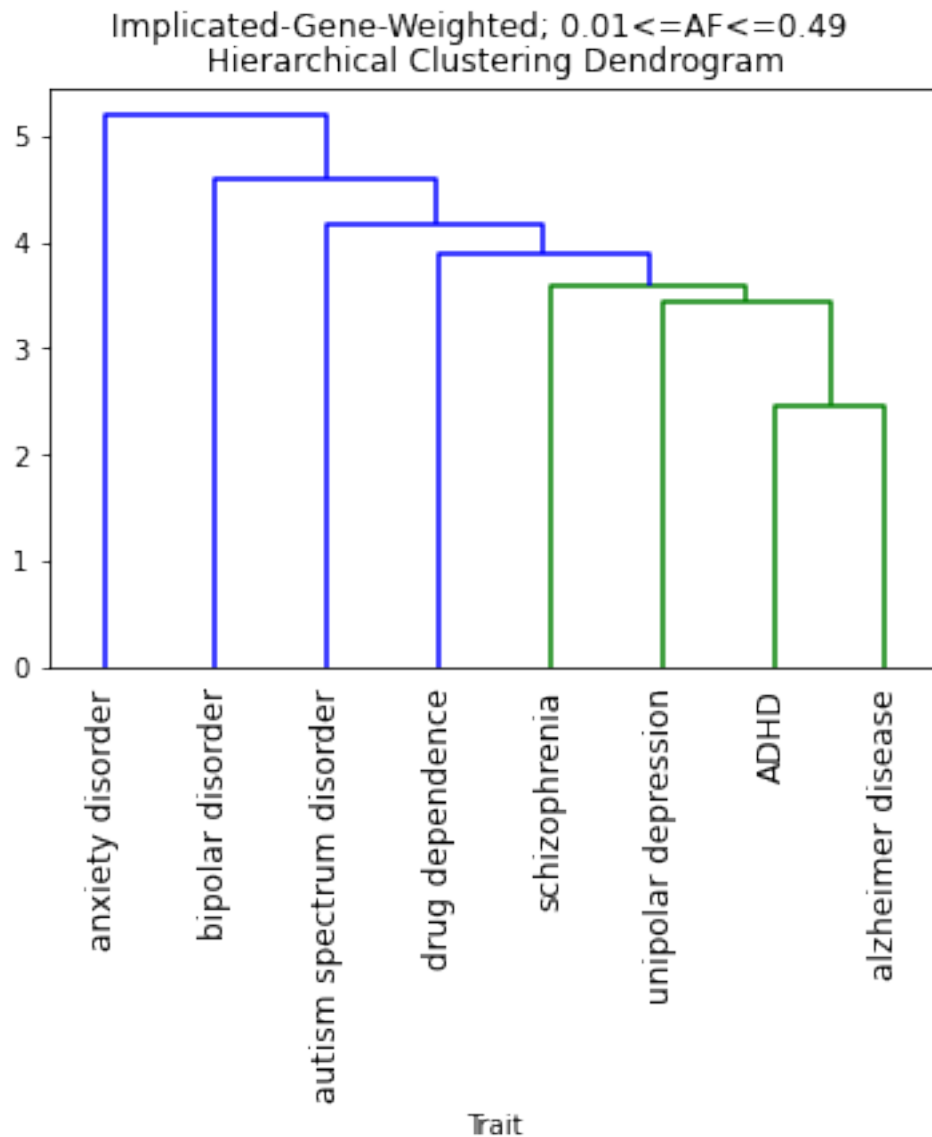


Appears to still be telescoped, but different ordering, and anxiety disorder seems especially unrelated now.

According to a [paper](#) I read, excluding SNPs with  $AF < 0.01$  may be viable too. My assumption is these may be 'too rare' to play a role in common(ish) traits.



```
[32]: min_and_max_filtered_df = filter_by_af(minority_df, min_af=0.01)
      hierarchical_clustering(
          min_and_max_filtered_df,
          'Implicated-Gene-Weighted; 0.01<=AF<=0.49',
          implicated_gene_weighted_encoding)
```



Interestingly, this really changes the structure. Double-check how much data is being filtered at each step.

```
[33]: pre_af_filtering_size = len(all_df)
      max_af_filtering_size = len(minority_df) # this also removed eating disorder
      min_and_max_filtering_size = len(min_and_max_filtered_df)
```

```
print(f'{pre_af_filtering_size} variants before AF filtering.')
print(f'{max_af_filtering_size} variants with max AF filtering.')
print(f'{min_and_max_filtering_size} variants with min and max AF filtering.')
```

3544 variants before AF filtering.  
 1278 variants with max AF filtering.  
 1237 variants with min and max AF filtering.

About a third of original variants remaining after min & max AF filtering (and excluding those with unknown AF values). Check for condition skew

```
[34]: min_and_max_filtered_df.groupby('parent_trait').size().sort_values()
```

```
[34]: parent_trait
autism spectrum disorder    36
drug dependence             51
anxiety disorder           58
ADHD                       83
alzheimer disease          115
bipolar disorder           178
unipolar depression        296
schizophrenia              420
dtype: int64
```

Still at least 30 rows per trait.

```
[35]: # Just double-checking the logic.
min_and_max_filtered_df['af'].describe()
```

```
[35]: count    1237.000000
mean         0.257523
std          0.140542
min          0.010266
25%          0.133620
50%          0.275067
75%          0.378422
max          0.489979
Name: af, dtype: float64
```

What if we focus on rarest variants only?

```
[36]: rare_df = filter_by_af(all_df, min_af=0.0, max_af=.01)
len(rare_df)
```

```
[36]: 41
```

Probably not enough data points for meaningful clustering. Let's just look at the conditions with significant rare variants.

```
[37]: rare_df.groupby('parent_trait').size().sort_values()
```

```
[37]: parent_trait
anxiety disorder      1
autism spectrum disorder  1
bipolar disorder      3
drug dependence       4
schizophrenia         4
unipolar depression   6
alzheimer disease    22
dtype: int64
```

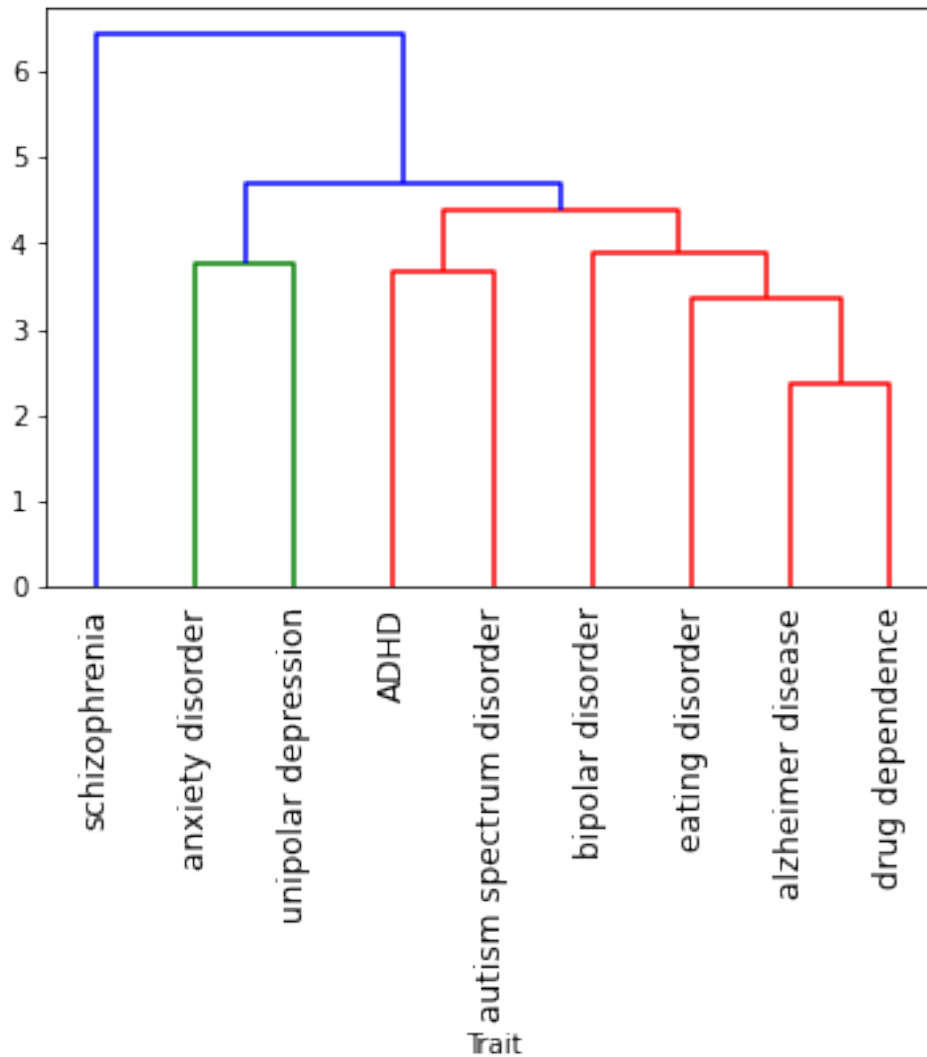
Surprisingly, alzheimer disease actually has more significant rare ( $AF \leq 0.01$ ) variants than schizophrenia, even though schizophrenia has more known significant variants (by more than 2x according to post-cleaning analysis).

Let's try one more filtering approach for gene-weighted clustering. Only consider those variants which have known significant tissue associations.

```
[38]: def filter_to_tissue_associations(in_df: pd.DataFrame) -> pd.DataFrame:
      """Filters to rows where 'tissues' != ''. """
      out_df = in_df.copy()
      out_df = out_df[out_df['tissues'].map(lambda tissues: len(tissues) > 0)]
      return out_df

data_with_tissues_df = filter_to_tissue_associations(all_df)
hierachical_clustering(
    data_with_tissues_df,
    'Implicated-Gene-Weighted; Variants associated with tissues',
    implicated_gene_weighted_encoding)
```

Implicated-Gene-Weighted; Variants associated with tissues  
Hierarchical Clustering Dendrogram



I see some similarities in other clusterings but also some differences.

Similarities: I've noticed alzheimer and drug dependence are often together, as are anxiety and depression.

Differences: I think this is the first time I've seen autism and ADHD closest together.

## 2 Tissue-association based clustering

Re-using some code from post-cleaning analysis to get started with tissue associated variants. Total numbers may be different as low-data traits have already been filtered.

```

[39]: TISSUE_DELIM = "&"

tissue_metadata_file = 'tissue_metadata.txt'
all_tissues = []
with open(tissue_metadata_file) as tissue_file:
    for tissue in tissue_file.readlines():
        all_tissues.append(tissue.replace("\n", ""))

print(f'Tissues: {all_tissues}')

# In previous cleaning, if a variant mapped to multiple genes, I created
# separate rows for it. For tissue association I just blindly match the variant
# location and don't care about the gene though... so drop duplicate locations
# as they may misrepresent total counts in subsequent tissue analysis here.
tissues_df = all_df.copy()
tissues_df.drop_duplicates(subset=['location', 'parent_trait'], inplace=True)
total_variants = len(tissues_df)
tissues_df['has_tissues'] = tissues_df['tissues'].map(lambda tissues:
    ↪len(tissues) > 0)
tissues_df = tissues_df[tissues_df['has_tissues'] == True]
num_variants_with_tissues = len(tissues_df)
print(f'{num_variants_with_tissues} variants significantly associated with
    ↪tissues of {total_variants} total.')

# Aggregate number of per-tissue associations per trait.
# This is a dict from trait to Dict[tissue, tissue association count]
trait_to_tissue_to_count = {}
all_traits = tissues_df['parent_trait'].unique()
for trait in all_traits:
    trait_to_tissue_to_count[trait] = {tissue: 0 for tissue in all_tissues}

for _, row in tissues_df.iterrows():
    trait = row['parent_trait']
    variant_tissues = row['tissues'].split(TISSUE_DELIM)
    for tissue in variant_tissues:
        trait_to_tissue_to_count[trait][tissue] += 1

rows = []
index = []
for trait, tissue_dict in trait_to_tissue_to_count.items():
    index.append(trait)
    rows.append(tissue_dict)

trait_to_tissue_count_df = pd.DataFrame(rows, index=index)
trait_to_tissue_count_df.head()

```

Tissues: ['Adipose\_Subcutaneous', 'Adipose\_Visceral\_Omentum', 'Adrenal\_Gland', 'Artery\_Aorta', 'Artery\_Coronary', 'Artery\_Tibial', 'Brain\_Amygdala', 'Brain\_Anterior\_cingulate\_cortex\_BA24', 'Brain\_Caudate\_basal\_ganglia', 'Brain\_Cerebellar\_Hemisphere', 'Brain\_Cerebellum', 'Brain\_Cortex', 'Brain\_Frontal\_Cortex\_BA9', 'Brain\_Hippocampus', 'Brain\_Hypothalamus', 'Brain\_Nucleus\_accumbens\_basal\_ganglia', 'Brain\_Putamen\_basal\_ganglia', 'Brain\_Spinal\_cord\_cervical\_c-1', 'Brain\_Substantia\_nigra', 'Breast\_Mammary\_Tissue', 'Cells\_Cultured\_fibroblasts', 'Cells\_EBV-transformed\_lymphocytes', 'Colon\_Sigmoid', 'Colon\_Transverse', 'Esophagus\_Gastroesophageal\_Junction', 'Esophagus\_Mucosa', 'Esophagus\_Muscularis', 'Heart\_Atrial\_Appendage', 'Heart\_Left\_Ventricle', 'Kidney\_Cortex', 'Liver', 'Lung', 'Minor\_Salivary\_Gland', 'Muscle\_Skeletal', 'Nerve\_Tibial', 'Ovary', 'Pancreas', 'Pituitary', 'Prostate', 'Skin\_Not\_Sun\_Exposed\_Suprapubic', 'Skin\_Sun\_Exposed\_Lower\_leg', 'Small\_Intestine\_Terminal\_Ileum', 'Spleen', 'Stomach', 'Testis', 'Thyroid', 'Uterus', 'Vagina', 'Whole\_Blood']

948 variants significantly associated with tissues of 2615 total.

[39]:	Adipose_Subcutaneous	Adipose_Visceral_Omentum	\
ADHD	15	11	
alzheimer disease	25	13	
anxiety disorder	7	9	
autism spectrum disorder	7	8	
bipolar disorder	62	51	
	Adrenal_Gland	Artery_Aorta	Artery_Coronary \
ADHD	5	11	4
alzheimer disease	9	18	5
anxiety disorder	6	7	4
autism spectrum disorder	4	6	2
bipolar disorder	43	62	28
	Artery_Tibial	Brain_Amygdala	\
ADHD	15	2	
alzheimer disease	21	3	
anxiety disorder	13	3	
autism spectrum disorder	8	2	
bipolar disorder	72	8	
	Brain_Anterior_cingulate_cortex_BA24	\	
ADHD		2	
alzheimer disease		3	
anxiety disorder		6	
autism spectrum disorder		5	
bipolar disorder		24	
	Brain_Caudate_basal_ganglia	\	

ADHD	6				
alzheimer disease	12				
anxiety disorder	7				
autism spectrum disorder	4				
bipolar disorder	41				
		Brain_Cerebellar_Hemisphere	...	\	
ADHD	6	...			
alzheimer disease	14	...			
anxiety disorder	10	...			
autism spectrum disorder	9	...			
bipolar disorder	37	...			
		Skin_Not_Sun_Exposed_Suprapubic	\		
ADHD	15				
alzheimer disease	18				
anxiety disorder	12				
autism spectrum disorder	8				
bipolar disorder	60				
		Skin_Sun_Exposed_Lower_leg	\		
ADHD	15				
alzheimer disease	21				
anxiety disorder	12				
autism spectrum disorder	8				
bipolar disorder	74				
		Small_Intestine_Terminal_Ileum	Spleen	Stomach	\
ADHD	2	5	6		
alzheimer disease	6	15	16		
anxiety disorder	4	9	9		
autism spectrum disorder	3	5	3		
bipolar disorder	29	43	42		
	Testis	Thyroid	Uterus	Vagina	Whole_Blood
ADHD	12	21	2	2	10
alzheimer disease	18	21	2	2	25
anxiety disorder	14	14	2	2	15
autism spectrum disorder	9	9	0	2	8
bipolar disorder	68	82	7	12	59

[5 rows x 49 columns]

```
[40]: # Clustering approach for tissues will be very similar to the weighted gene
# encoding, except now each element in a vector here represents
# tissue associations instead of gene associations.
```

```

def implicated_tissue_weighted_encoding(in_df: pd.DataFrame) -> pd.DataFrame:
    """Encodes trait vectors where elements are relative tissue implications.

    Specifically, for each trait, creates a vector (V1, V2... Vn)
    where  $V_i = T_i / \text{Max}(T_i)$  where a trait has  $T_i$  variants implicated with  $i$ th tissue
    and  $n$  is total tissues in `in_df`.

    E.g. if trait has tissue associations:
        tissue1  tissue2  tissue3
        1        3        5

    Then encoding is:
        (.2,    .6,    1)

    DataFrame is returned with traits as column headers and vectors as columns.
    """
    tissue_to_index = create_tissue_to_index_dict()
    traits = in_df.index.tolist()
    trait_to_encoding = {}
    for trait in traits:
        trait_row = in_df.loc[trait]
        tissue_counts_encoding = trait_row.to_numpy()
        max_val = max(tissue_counts_encoding)
        tissue_counts_encoding = tissue_counts_encoding / max_val
        trait_to_encoding[trait] = tissue_counts_encoding
    return pd.DataFrame(trait_to_encoding)

def create_tissue_to_index_dict() -> Dict[str, int]:
    """Maps tissue to index such that each tissue is assigned a unique index."""
    num_tissues = len(all_tissues)
    tissue_to_index = {all_tissues[i]: i for i in range(num_tissues)}
    return tissue_to_index

# Test
implicated_tissue_weighted_encoding(trait_to_tissue_count_df)

```

```

[40]:
      ADHD  alzheimer disease  anxiety disorder  autism spectrum disorder \
0  0.714286                1.00             0.466667                0.777778
1  0.523810                0.52             0.600000                0.888889
2  0.238095                0.36             0.400000                0.444444
3  0.523810                0.72             0.466667                0.666667
4  0.190476                0.20             0.266667                0.222222
5  0.714286                0.84             0.866667                0.888889
6  0.095238                0.12             0.200000                0.222222
7  0.095238                0.12             0.400000                0.555556

```



8	0.285714	0.48	0.466667	0.444444
9	0.285714	0.56	0.666667	1.000000
10	0.380952	0.80	0.733333	0.888889
11	0.333333	0.60	0.400000	0.666667
12	0.095238	0.24	0.333333	0.555556
13	0.142857	0.44	0.333333	0.222222
14	0.095238	0.20	0.266667	0.333333
15	0.190476	0.52	0.533333	0.555556
16	0.142857	0.56	0.333333	0.555556
17	0.000000	0.16	0.466667	0.111111
18	0.000000	0.16	0.133333	0.111111
19	0.476190	0.32	0.533333	0.444444
20	0.571429	0.72	0.933333	1.000000
21	0.047619	0.24	0.266667	0.111111
22	0.380952	0.80	0.266667	0.666667
23	0.380952	0.56	0.533333	0.555556
24	0.523810	0.76	0.400000	0.666667
25	0.666667	0.88	0.600000	0.666667
26	0.571429	0.96	0.533333	0.888889
27	0.571429	0.68	0.533333	0.888889
28	0.380952	0.64	0.666667	0.444444
29	0.000000	0.16	0.066667	0.000000
30	0.142857	0.36	0.266667	0.222222
31	0.380952	0.84	0.866667	0.666667
32	0.047619	0.04	0.200000	0.111111
33	0.476190	0.84	0.533333	0.777778
34	0.857143	0.80	0.533333	0.888889
35	0.047619	0.24	0.200000	0.111111
36	0.380952	0.64	0.600000	0.555556
37	0.380952	0.44	0.400000	0.777778
38	0.238095	0.32	0.333333	0.333333
39	0.714286	0.72	0.800000	0.888889
40	0.714286	0.84	0.800000	0.888889
41	0.095238	0.24	0.266667	0.333333
42	0.238095	0.60	0.600000	0.555556
43	0.285714	0.64	0.600000	0.333333
44	0.571429	0.72	0.933333	1.000000
45	1.000000	0.84	0.933333	1.000000
46	0.095238	0.08	0.133333	0.000000
47	0.095238	0.08	0.133333	0.222222
48	0.476190	1.00	1.000000	0.888889

	bipolar disorder	drug dependence	eating disorder	schizophrenia \
0	0.756098	0.600000	0.333333	0.867704
1	0.621951	0.285714	0.000000	0.723735
2	0.524390	0.114286	0.000000	0.501946
3	0.756098	0.485714	0.333333	0.762646

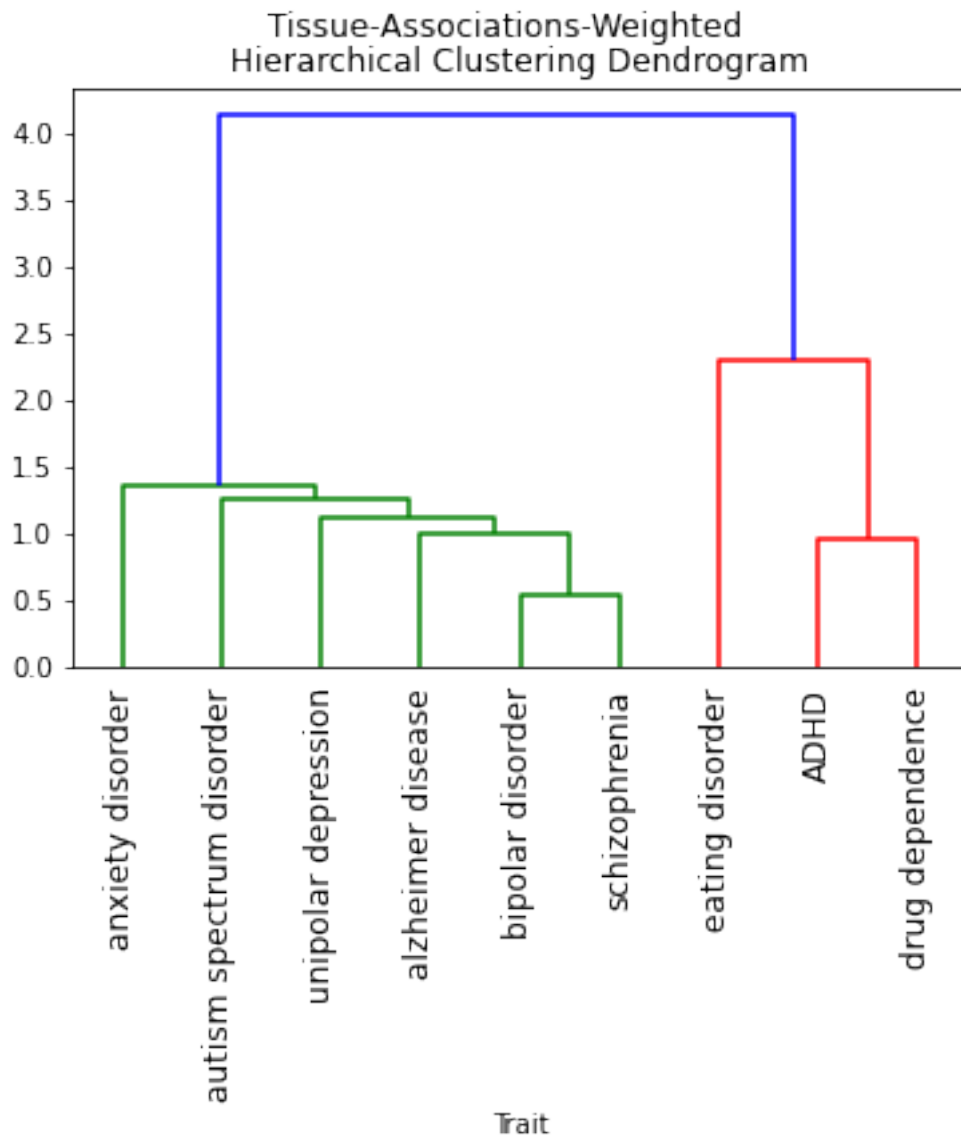
4	0.341463	0.228571	0.000000	0.373541
5	0.878049	0.771429	0.333333	0.871595
6	0.097561	0.057143	0.000000	0.241245
7	0.292683	0.028571	0.000000	0.322957
8	0.500000	0.200000	0.333333	0.517510
9	0.451220	0.342857	0.000000	0.494163
10	0.634146	0.400000	0.333333	0.622568
11	0.512195	0.028571	0.000000	0.501946
12	0.341463	0.028571	0.000000	0.455253
13	0.243902	0.057143	0.000000	0.365759
14	0.292683	0.028571	0.000000	0.338521
15	0.341463	0.057143	0.000000	0.459144
16	0.487805	0.200000	0.000000	0.431907
17	0.195122	0.057143	0.000000	0.315175
18	0.060976	0.028571	0.000000	0.143969
19	0.573171	0.457143	0.000000	0.614786
20	0.865854	0.457143	0.666667	0.898833
21	0.231707	0.085714	0.000000	0.338521
22	0.573171	0.285714	0.333333	0.599222
23	0.524390	0.342857	0.000000	0.595331
24	0.512195	0.371429	0.000000	0.626459
25	0.731707	0.514286	0.666667	0.832685
26	0.792683	0.428571	0.333333	0.797665
27	0.646341	0.628571	0.000000	0.696498
28	0.560976	0.628571	0.000000	0.642023
29	0.024390	0.028571	0.000000	0.081712
30	0.219512	0.285714	0.000000	0.354086
31	0.719512	0.742857	0.000000	0.747082
32	0.219512	0.028571	0.000000	0.214008
33	0.719512	0.742857	0.333333	0.793774
34	0.865854	0.600000	0.000000	0.922179
35	0.219512	0.028571	0.000000	0.315175
36	0.548780	0.257143	0.000000	0.591440
37	0.512195	0.085714	0.000000	0.560311
38	0.304878	0.085714	0.000000	0.459144
39	0.731707	0.542857	0.333333	0.859922
40	0.902439	0.571429	1.000000	0.894942
41	0.353659	0.228571	0.000000	0.377432
42	0.524390	0.314286	0.333333	0.575875
43	0.512195	0.371429	0.000000	0.568093
44	0.829268	0.514286	0.666667	0.902724
45	1.000000	1.000000	0.333333	1.000000
46	0.085366	0.085714	0.000000	0.217899
47	0.146341	0.028571	0.000000	0.264591
48	0.719512	0.314286	0.333333	0.785992

unipolar depression

0	0.961165
1	0.825243
2	0.660194
3	0.708738
4	0.456311
5	0.951456
6	0.203883
7	0.359223
8	0.718447
9	0.679612
10	0.786408
11	0.689320
12	0.504854
13	0.417476
14	0.262136
15	0.631068
16	0.475728
17	0.368932
18	0.116505
19	0.747573
20	0.941748
21	0.378641
22	0.572816
23	0.747573
24	0.669903
25	0.990291
26	0.932039
27	0.660194
28	0.766990
29	0.029126
30	0.475728
31	0.815534
32	0.291262
33	0.980583
34	1.000000
35	0.504854
36	0.699029
37	0.582524
38	0.485437
39	0.873786
40	0.961165
41	0.456311
42	0.601942
43	0.650485
44	0.970874
45	0.961165
46	0.262136

```
47         0.349515
48         0.893204
```

```
[41]: hierarchical_clustering(
        trait_to_tissue_count_df,
        'Tissue-Associations-Weighted',
        implicated_tissue_weighted_encoding)
```



Actually quite different from other plots. The ADHD, eating disorder, drug dependence sub-cluster seems to be very distinct from the other group. Schizophrenia and bipolar disorder are still closely related.

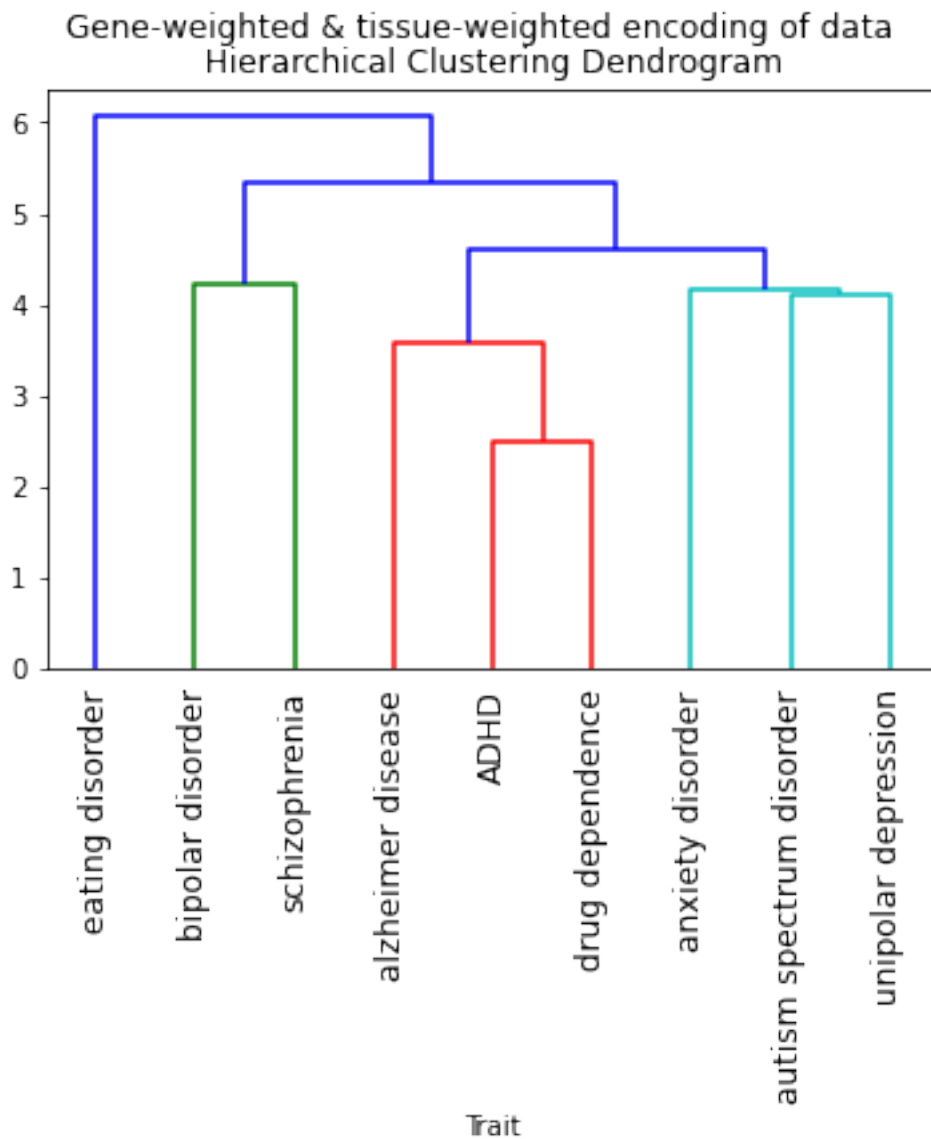
Although normalizing the encoded values to the ranges 0-1 somewhat accounts for varying data

sizes, it may not be sufficient since more data means more potential to have tissue associations, which could lead to denser vectors for these traits which may not actually have any clinical significance.

## 2.1 Combinations of encodings & filtering?

```
[42]: # Each encoding is a DF with traits as columns, so can just concatenate them...  
tissue_encoding = implicated_tissue_weighted_encoding(trait_to_tissue_count_df)  
gene_weighted_encoding = implicated_gene_weighted_encoding(all_df)  
both_encoding = pd.concat([tissue_encoding, gene_weighted_encoding])
```

```
[43]: hierachical_clustering(  
    both_encoding,  
    'Gene-weighted & tissue-weighted encoding of data',  
    lambda df: df)
```



### 3 Different clustering algorithm: k-means

With k-means, the desired number of clusters must be pre-specified.

```
[44]: input = implicated_gene_weighted_encoding(all_df).T
      input
```

```
[44]:
```

	0	1	2	3	4	5	\
ADHD	0.000000	0.000000	0.266667	0.0	0.000000	0.0	
alzheimer disease	0.000000	0.000000	0.000000	0.0	0.000000	0.0	

anxiety disorder	0.000000	0.000000	0.000000	0.0	0.000000	0.0
autism spectrum disorder	0.000000	0.000000	0.000000	0.0	0.000000	0.0
bipolar disorder	0.000000	0.100000	0.000000	0.4	0.000000	0.2
drug dependence	0.000000	0.000000	0.000000	0.0	0.000000	0.0
eating disorder	0.000000	0.000000	0.000000	0.0	0.000000	0.0
schizophrenia	0.095238	0.047619	0.047619	0.0	0.000000	0.0
unipolar depression	0.000000	0.000000	0.000000	0.0	0.142857	0.0

	6	7	8	9	...	\
ADHD	0.000000	0.000000	0.000000	0.000000	...	
alzheimer disease	0.000000	0.000000	0.000000	0.000000	...	
anxiety disorder	0.000000	0.000000	0.000000	0.000000	...	
autism spectrum disorder	0.000000	0.000000	0.000000	0.500000	...	
bipolar disorder	0.100000	0.000000	0.000000	0.000000	...	
drug dependence	0.000000	0.000000	0.000000	0.000000	...	
eating disorder	0.000000	0.000000	0.000000	0.000000	...	
schizophrenia	0.238095	0.047619	0.380952	0.142857	...	
unipolar depression	0.000000	0.142857	0.000000	0.000000	...	

	976	977	978	979	980	981	\
ADHD	0.000000	0.000000	0.000000	0.000000	0.00	0.00	
alzheimer disease	0.142857	0.857143	0.238095	0.000000	0.00	0.00	
anxiety disorder	0.000000	0.000000	0.000000	0.250000	0.00	0.00	
autism spectrum disorder	0.000000	0.000000	0.000000	0.000000	0.75	0.25	
bipolar disorder	0.000000	0.000000	0.000000	0.000000	0.40	0.10	
drug dependence	0.000000	0.000000	0.000000	0.000000	0.00	0.00	
eating disorder	0.000000	0.000000	0.000000	0.000000	0.50	0.00	
schizophrenia	0.000000	0.000000	0.000000	0.000000	0.00	0.00	
unipolar depression	0.000000	0.000000	0.000000	0.214286	0.00	0.00	

	982	983	984	985
ADHD	0.000000	0.000000	0.000000	0.000000
alzheimer disease	0.000000	0.000000	0.000000	0.52381
anxiety disorder	0.000000	0.000000	0.000000	0.000000
autism spectrum disorder	0.000000	0.000000	0.000000	0.000000
bipolar disorder	0.000000	0.100000	0.000000	0.000000
drug dependence	0.000000	0.000000	0.000000	0.000000
eating disorder	0.000000	0.000000	0.000000	0.000000
schizophrenia	0.047619	0.047619	0.095238	0.000000
unipolar depression	0.071429	0.000000	0.000000	0.000000

[9 rows x 986 columns]

```
[45]: # 4 seems like a reasonable start as it divides the traits into 3 clusters.
# (9 total traits since personality & tourette are dropped).
num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=0).fit(input)
```

```
[46]: kmeans.labels_
```

```
[46]: array([1, 1, 1, 1, 0, 1, 2, 0, 1], dtype=int32)
```

```
[47]: kmeans.cluster_centers_
```

```
[47]: array([[0.04761905, 0.07380952, 0.02380952, ..., 0.07380952, 0.04761905,
           0.          ],
          [0.          , 0.          , 0.04444444, ..., 0.          , 0.          ,
           0.08730159],
          [0.          , 0.          , 0.          , ..., 0.          , 0.          ,
           0.          ]])
```

```
[48]: trait_order = input.index.tolist()
      trait_order
```

```
[48]: ['ADHD',
      'alzheimer disease',
      'anxiety disorder',
      'autism spectrum disorder',
      'bipolar disorder',
      'drug dependence',
      'eating disorder',
      'schizophrenia',
      'unipolar depression']
```

```
[49]: for idx in range(len(trait_order)):
      print(f'{trait_order[idx]}: cluster {kmeans.labels_[idx] + 1}')
```

```
ADHD: cluster 2
alzheimer disease: cluster 2
anxiety disorder: cluster 2
autism spectrum disorder: cluster 2
bipolar disorder: cluster 1
drug dependence: cluster 2
eating disorder: cluster 3
schizophrenia: cluster 1
unipolar depression: cluster 2
```

```
[50]: NUM_COMPONENTS = 3
      PCA_DF_COLUMNS = [f'PC{i}' for i in range(1, NUM_COMPONENTS + 1)]
      COLORS = ['gold', 'red', 'forestgreen', 'royalblue', 'magenta', 'pink',
                'darkorange', 'turquoise', 'gray', 'rosybrown', 'navy']

      def abbreviate_trait(trait: str) -> str:
          """Hacky, but gets the job done for abbreviating legend labels."""
```



```

if trait == 'ADHD':
    return trait
if trait == 'anxiety disorder':
    return 'AnxD'
if trait == 'alzheimer disease':
    return 'AlzD'
if " " not in trait:
    return trait.upper()[0]
words = trait.split(" ")
return "".join([word.upper()[0] for word in words])

def get_pca_and_components(matrix_data):
    """Returns tuple of the PCA object and DF of components."""
    scaled_matrix_data = StandardScaler().fit_transform(matrix_data)

    pca = PCA(n_components=NUM_COMPONENTS)

    components = pca.fit_transform(scaled_matrix_data)
    components_df = pd.DataFrame(data=components, columns=PCA_DF_COLUMNS)

    return (pca, components_df)

def get_legend_labels(trait: List[str], cluster_labels, n_clusters: int) -> Dict[int, str]:
    cluster_to_name_list = {i + 1: [] for i in range(n_clusters)}
    for idx in range(len(trait)):
        trait = trait[idx]
        cluster = cluster_labels[idx] + 1
        cluster_to_name_list[cluster].append(abbreviate_trait(trait))

    cluster_to_name = {k: ",".join(v) for k, v in cluster_to_name_list.items()}
    return cluster_to_name

def kmeans_top_two_pcs_scatter_plot(components_df: pd.DataFrame,
                                     traits: List[str],
                                     cluster_labels: np.ndarray,
                                     n_clusters: int,
                                     extra_title=None) -> None:
    """Plots top 2 PCs in scatter plot, using X for cluster centers.
    Points belonging to different clusters are colored accordingly."""
    cluster_to_name = get_legend_labels(traits, cluster_labels, n_clusters)

    fig, ax = plt.subplots(figsize=(12, 8))
    title = f'K-Means clustering; {n_clusters} clusters; data reduced to 2D'

```

```

if extra_title is not None:
    title = f'{title}; {extra_title}'
ax.set(title=title)
num_rows = len(components_df)
cluster_to_color = {i: COLORS[i] for i in range(n_clusters)}
labeled = set()
for idx in range(num_rows):
    if idx >= len(trait_order):
        ax.scatter(components_df.iloc[idx]['PC1'],
                    components_df.iloc[idx]['PC2'],
                    c='black',
                    marker='x')
    else:
        label = cluster_to_name[cluster_labels[idx] + 1]
        if label in labeled:
            ax.scatter(components_df.iloc[idx]['PC1'],
                        components_df.iloc[idx]['PC2'],
                        c=cluster_to_color[cluster_labels[idx]],
                        marker='o')
        else:
            labeled.add(label)
            ax.scatter(components_df.iloc[idx]['PC1'],
                        components_df.iloc[idx]['PC2'],
                        c=cluster_to_color[cluster_labels[idx]],
                        marker='o',
                        label=label)

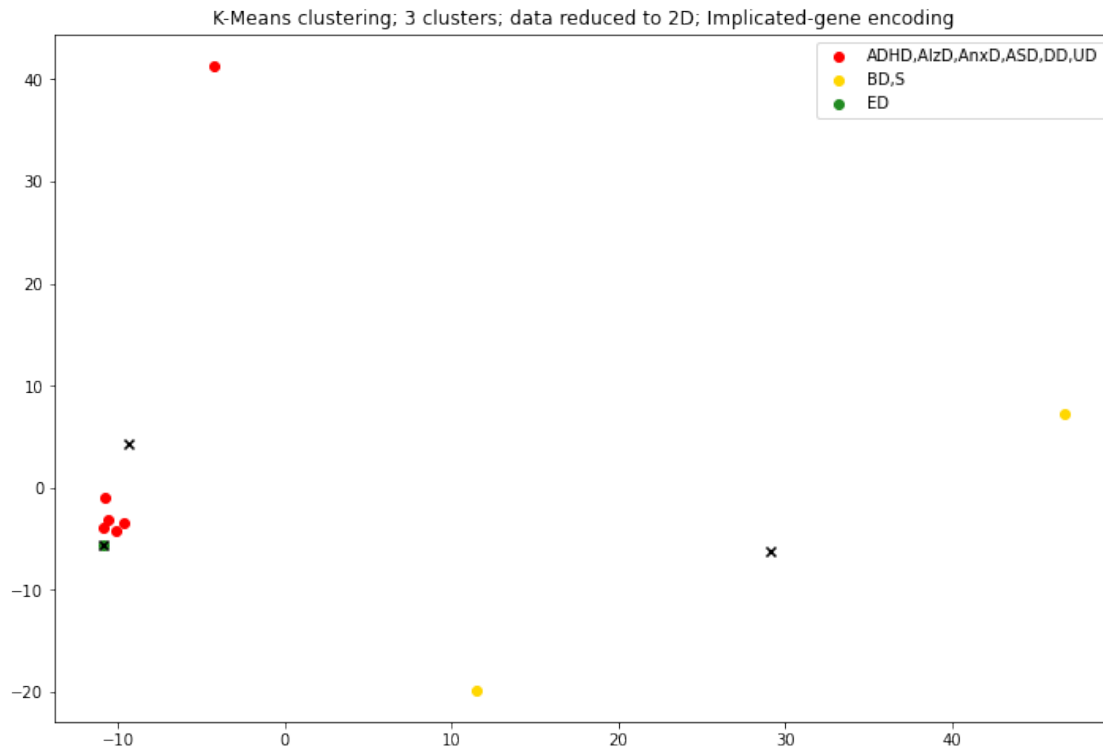
plt.legend(loc='upper right')
plt.show()
plt.clf()

# Concat encoded input data and kmeans cluster centers (each have same x-dim).
input_and_centers_data_matrix = np.concatenate(
    (input.values, kmeans.cluster_centers_),
    axis=0)

# PCA to reduce 986 dimensions to 2 for visualizing.
pca, component_df = get_pca_and_components(
    input_and_centers_data_matrix)

# Plot, coloring the centers distinctly.
kmeans_top_two_pcs_scatter_plot(component_df,
                                trait_order,
                                kmeans.labels_,
                                num_clusters,
                                extra_title='Implicated-gene encoding')

```



<Figure size 432x288 with 0 Axes>

Even using a different algorithm, there are similar outcomes: \* Schizophrenia and bipolar disorder are still together \* Eating disorder is by itself \* The rest are together.

Let's try upping num clusters to see if there are differences in 'the rest' / cluster 2

```
[51]: # First make a re-usable function
def kmeans_cluster_and_pca_plot(in_df: pd.DataFrame, n_clusters: int,
    extra_title=None) -> None:
    kmeans = KMeans(n_clusters=n_clusters, random_state=0).fit(in_df)
    # Concat encoded input data and kmeans cluster centers (each have same x-dim).
    input_and_centers_data_matrix = np.concatenate(
        (in_df.values, kmeans.cluster_centers_),
        axis=0)

    # PCA to reduce 986 dimensions to 2 for visualizing.
    pca, component_df = get_pca_and_components(
        input_and_centers_data_matrix)

    # Plot, coloring the centers distinctly.
    kmeans_top_two_pcs_scatter_plot(component_df,
        in_df.index.tolist(),
```

```

kmeans.labels_,
n_clusters,
extra_title=extra_title)

```

```

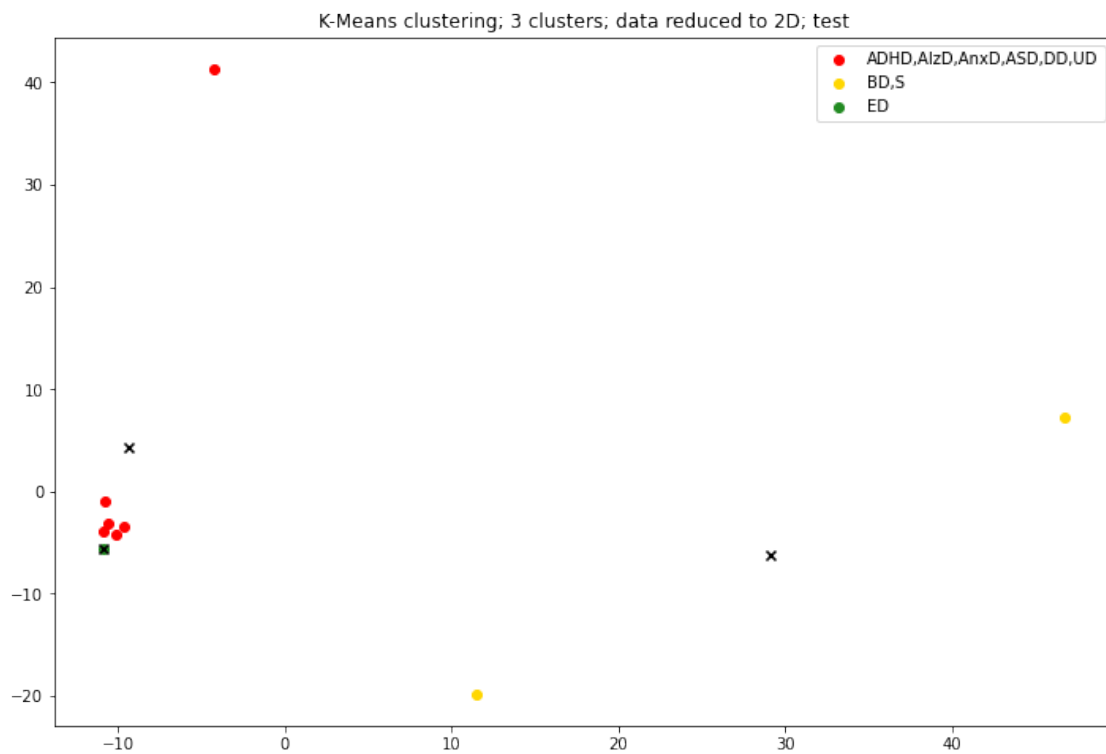
# Test, should equal above plot

```

```

kmeans_cluster_and_pca_plot(input, num_clusters, extra_title='test')

```

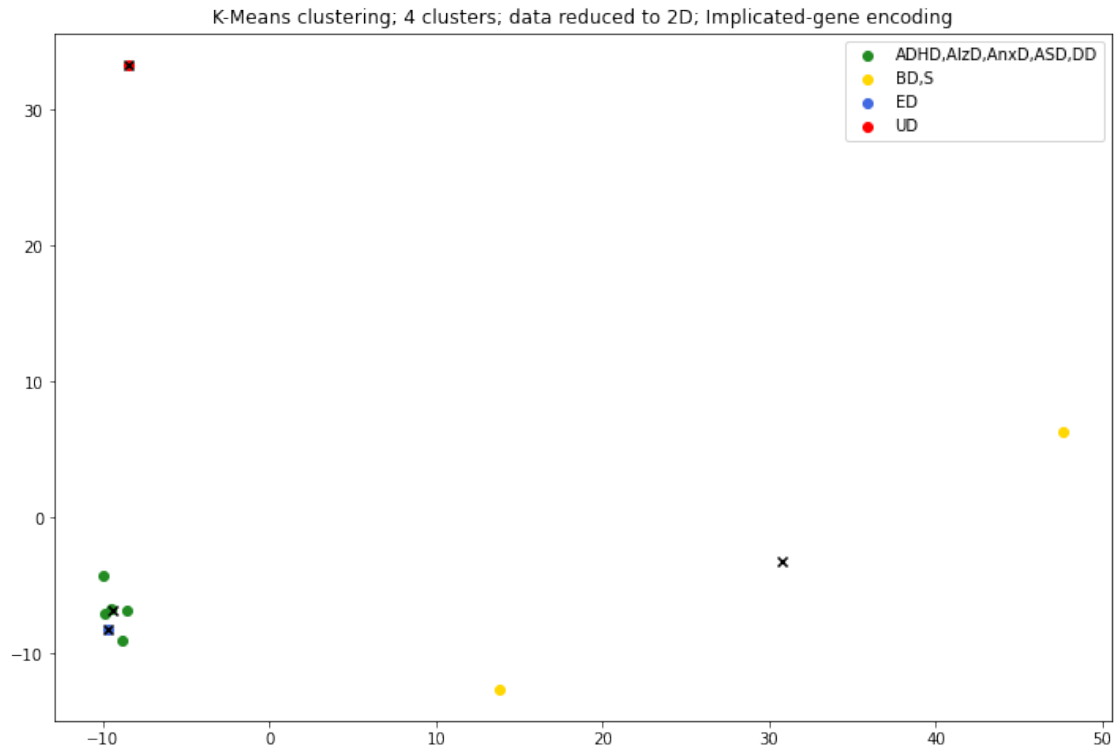


<Figure size 432x288 with 0 Axes>

```

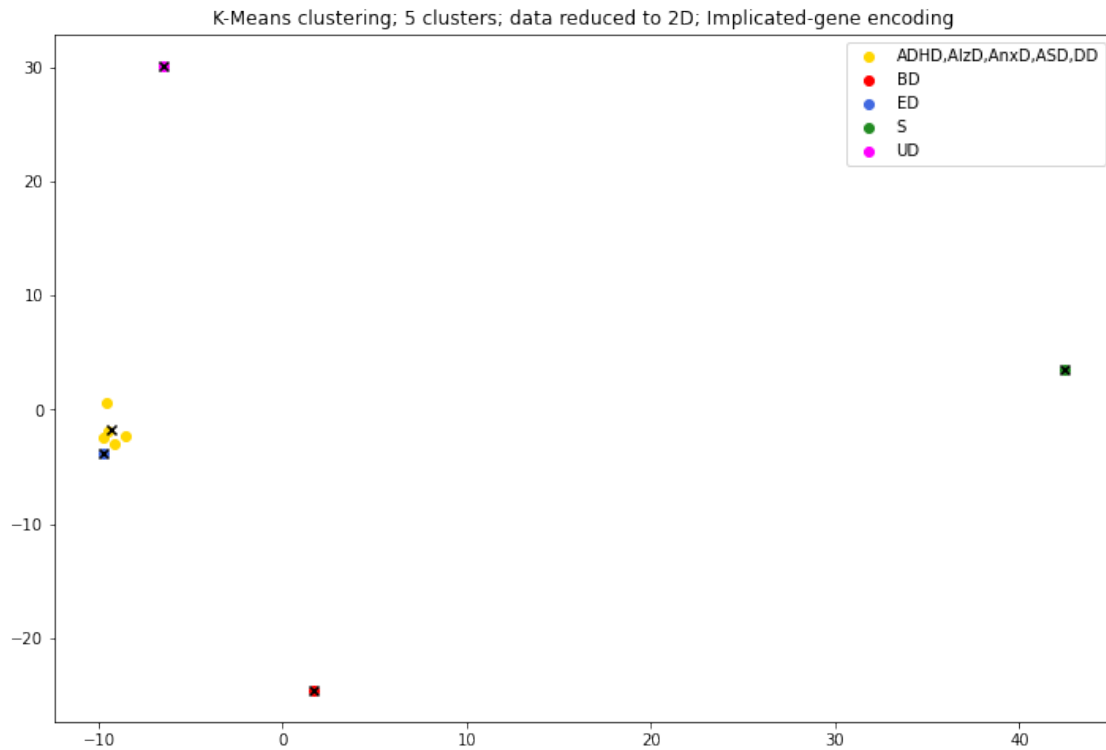
[52]: # Up num_clusters
num_clusters = 4
kmeans_cluster_and_pca_plot(input, num_clusters, extra_title='Implicated-gene_
↳encoding')

```



<Figure size 432x288 with 0 Axes>

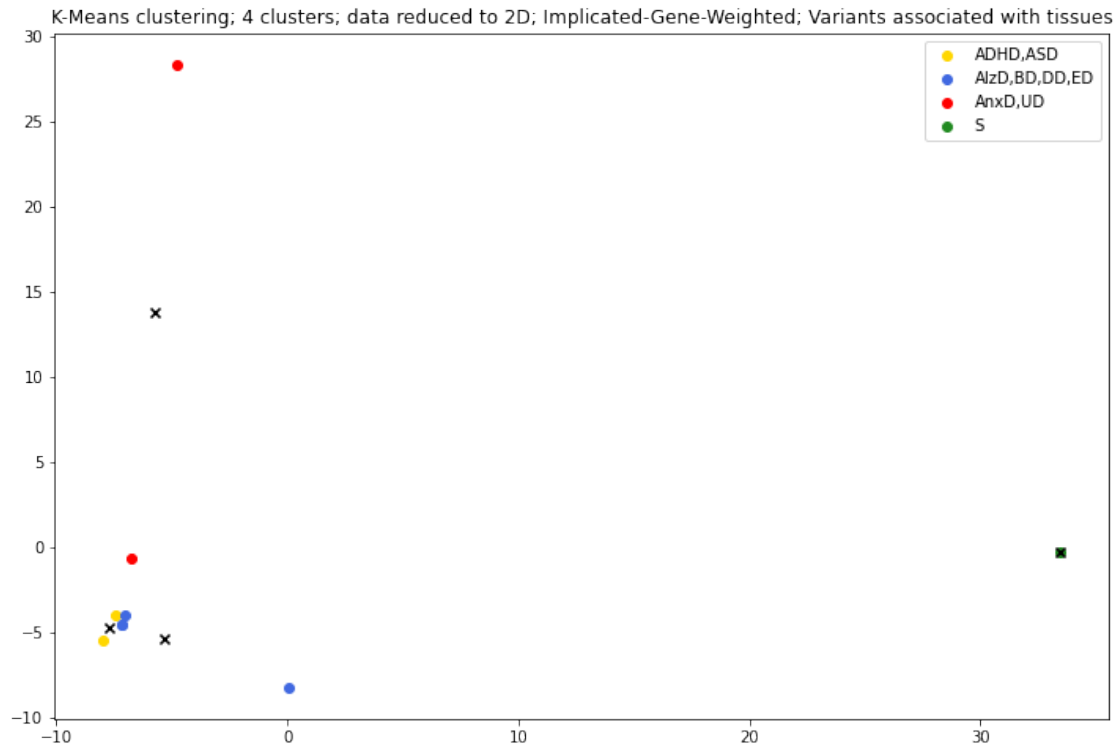
```
[53]: # One more time
num_clusters = 5
kmeans_cluster_and_pca_plot(input, num_clusters, extra_title='Implicated-gene_
↳encoding')
```



<Figure size 432x288 with 0 Axes>

It seems to still be skewed by traits with more data. Try some filtering...

```
[60]: input = implicated_gene_weighted_encoding(data_with_tissues_df).T
num_clusters = 4
kmeans_cluster_and_pca_plot(
    input,
    num_clusters,
    extra_title='Implicated-Gene-Weighted; Variants associated with tissues')
```



<Figure size 432x288 with 0 Axes>

Not especially skewed by trait data size, but schizophrenia is still very distinct. Blue and yellow clusters are quite close.

```
[61]: input = ↳ implicated_gene_weighted_encoding(filter_to_smallest_pvalues_same_sizes(all_df)).  

↳ T  

num_clusters = 4  

kmeans_cluster_and_pca_plot(  

    input,  

    num_clusters,  

    extra_title='Implicated-Gene-Weighted; Same # data points per trait (min_  

↳ p-value variants)')
```

/usr/local/lib/python3.7/dist-packages/pandas/util/\_decorators.py:311:

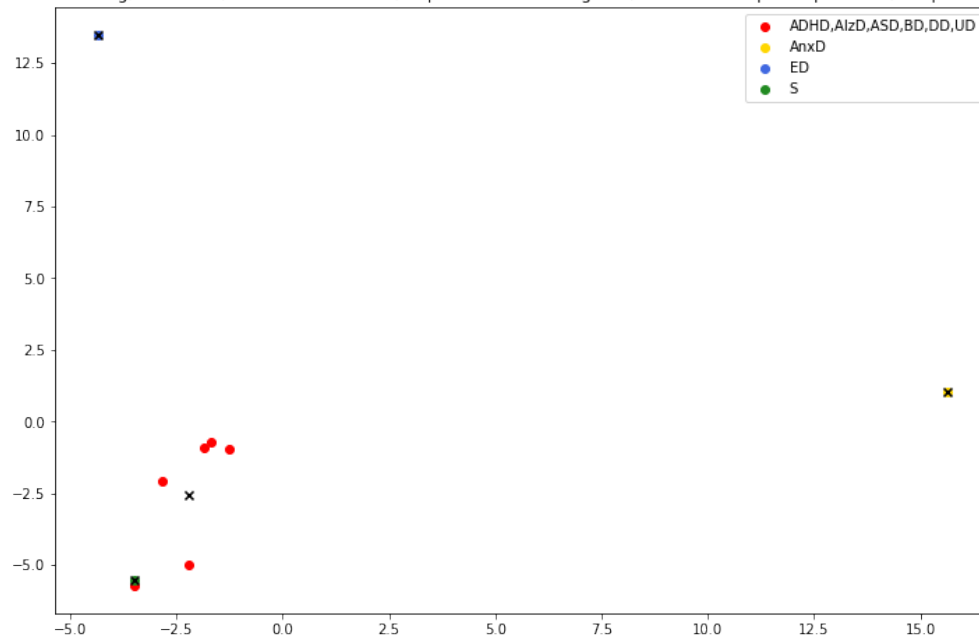
SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
return func(*args, **kwargs)
```

K-Means clustering; 4 clusters; data reduced to 2D; Implicated-Gene-Weighted; Same # data points per trait (min p-value variants)



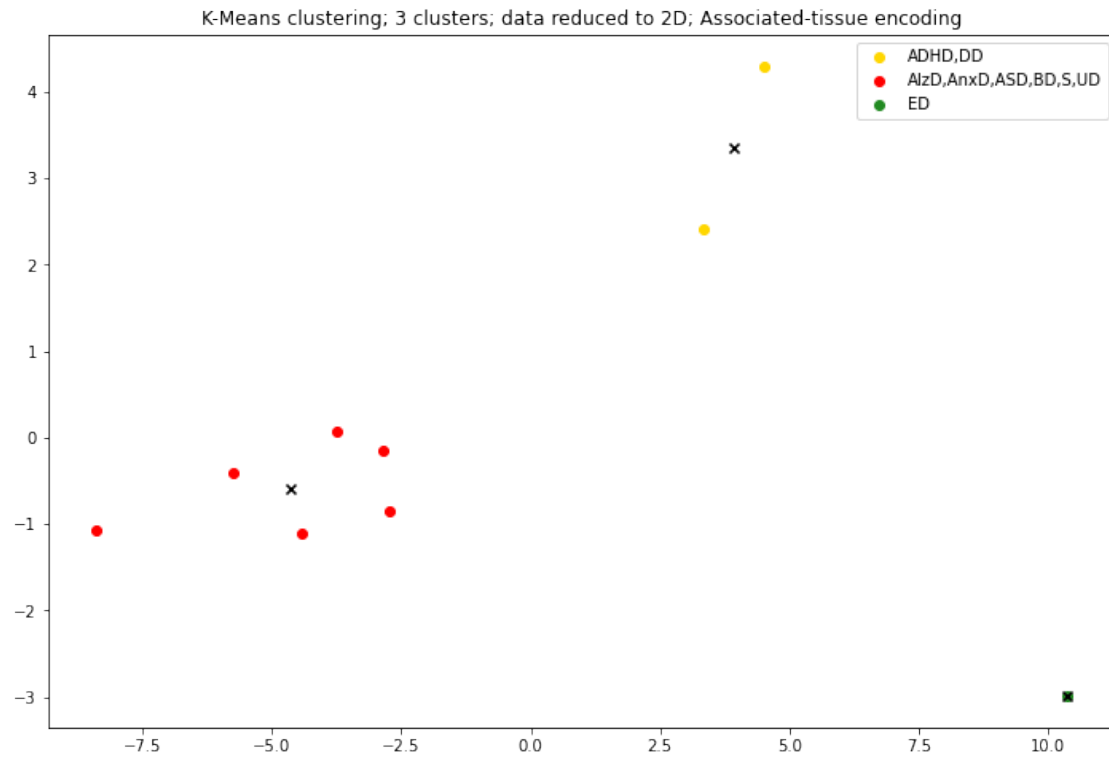
<Figure size 432x288 with 0 Axes>

Even with same # data points per trait, it seems eating disorder is still quite distinct, and for some reason anxiety disorder too.

### 3.1 Try k-means with tissue-encodings

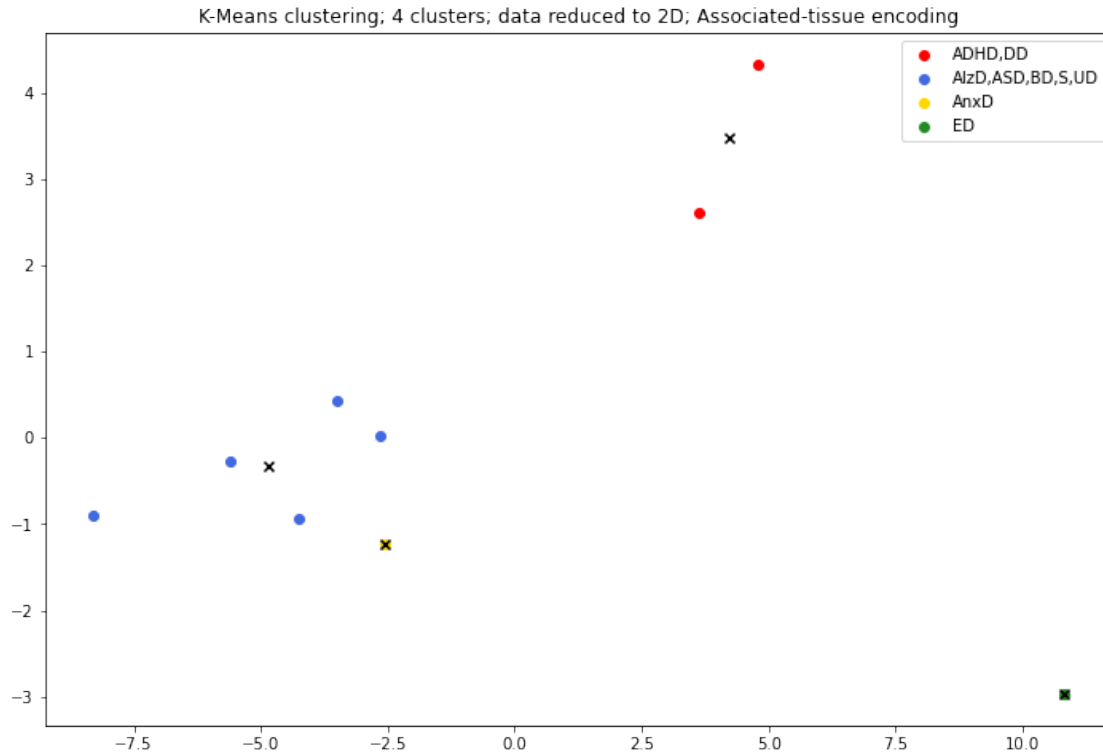
```
[54]: tissue_encoding = implicated_tissue_weighted_encoding(trait_to_tissue_count_df)
tissue_input = tissue_encoding.T
num_clusters = 3
kmeans_cluster_and_pca_plot(
    tissue_input, num_clusters, extra_title='Associated-tissue encoding')
```





<Figure size 432x288 with 0 Axes>

```
[55]: num_clusters = 4
      kmeans_cluster_and_pca_plot(
          tissue_input, num_clusters, extra_title='Associated-tissue encoding')
```



<Figure size 432x288 with 0 Axes>

## 4 PCA on gene & tissue-encoded matrix

As one last experiment, simply run PCA on the concatenated gene-encoding and tissue-encoding matrix and plot top 2 to see if there are any insights from reducing dimensions.

[56]: both\_encoding

```
[56]:
```

	ADHD	alzheimer disease	anxiety disorder	autism spectrum disorder	\
0	0.714286	1.00000	0.466667		0.777778
1	0.523810	0.52000	0.600000		0.888889
2	0.238095	0.36000	0.400000		0.444444
3	0.523810	0.72000	0.466667		0.666667
4	0.190476	0.20000	0.266667		0.222222
..	...	...	...		...
981	0.000000	0.00000	0.000000		0.250000
982	0.000000	0.00000	0.000000		0.000000
983	0.000000	0.00000	0.000000		0.000000
984	0.000000	0.00000	0.000000		0.000000
985	0.000000	0.52381	0.000000		0.000000

	bipolar disorder	drug dependence	eating disorder	schizophrenia \
0	0.756098	0.600000	0.333333	0.867704
1	0.621951	0.285714	0.000000	0.723735
2	0.524390	0.114286	0.000000	0.501946
3	0.756098	0.485714	0.333333	0.762646
4	0.341463	0.228571	0.000000	0.373541
..	...	...	...	...
981	0.100000	0.000000	0.000000	0.000000
982	0.000000	0.000000	0.000000	0.047619
983	0.100000	0.000000	0.000000	0.047619
984	0.000000	0.000000	0.000000	0.095238
985	0.000000	0.000000	0.000000	0.000000

	unipolar depression
0	0.961165
1	0.825243
2	0.660194
3	0.708738
4	0.456311
..	...
981	0.000000
982	0.071429
983	0.000000
984	0.000000
985	0.000000

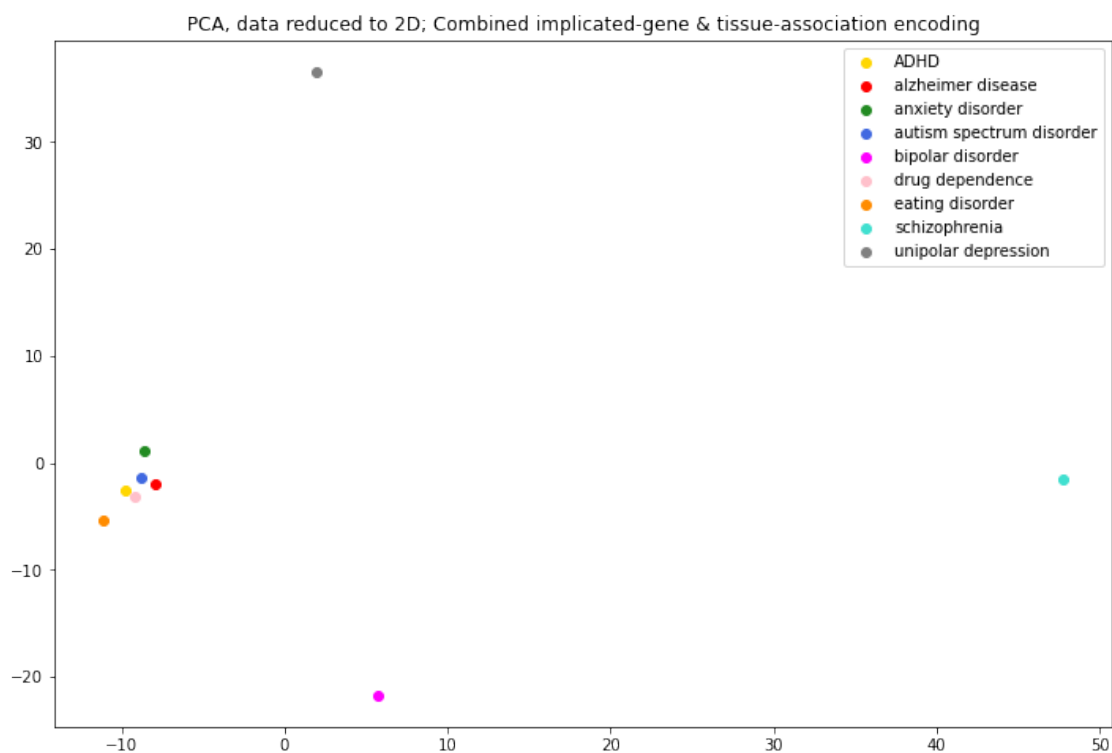
[1035 rows x 9 columns]

```
[57]: def top_two_pcs_scatter_plot(components_df: pd.DataFrame,
                                traits: List[str],
                                extra_title=None) -> None:
    """Plots top 2 PCs in scatter plot.
    Colors each point differently as each point is a distinct trait."""

    fig, ax = plt.subplots(figsize=(12, 8))
    title = f'PCA, data reduced to 2D'
    if extra_title is not None:
        title = f'{title}; {extra_title}'
    ax.set(title=title)
    num_rows = len(components_df)
    for idx in range(num_rows):
        ax.scatter(components_df.iloc[idx]['PC1'],
                    components_df.iloc[idx]['PC2'],
                    c=COLORS[idx],
                    marker='o',
                    label=traits[idx])
```

```
plt.legend(loc='upper right')
plt.show()
plt.clf()

input_df = both_encoding.T
pca, component_df = get_pca_and_components(input_df)
top_two_pcs_scatter_plot(
    component_df,
    input_df.index.tolist(),
    extra_title='Combined implicated-gene & tissue-association encoding')
```



<Figure size 432x288 with 0 Axes>

Seems to be again skewed by data size per trait.

```
[63]: input_df = implicated_gene_weighted_encoding(
    filter_to_smallest_pvalues_same_sizes(data_with_tissues_df)).T
pca, component_df = get_pca_and_components(input_df)
top_two_pcs_scatter_plot(
    component_df,
    input_df.index.tolist(),
```

```
extra_title='Implicated-gene encoding; only using data with tissue_
↳associations; filtered to lowest X p-value variants per trait')
```

/usr/local/lib/python3.7/dist-packages/pandas/util/\_decorators.py:311:

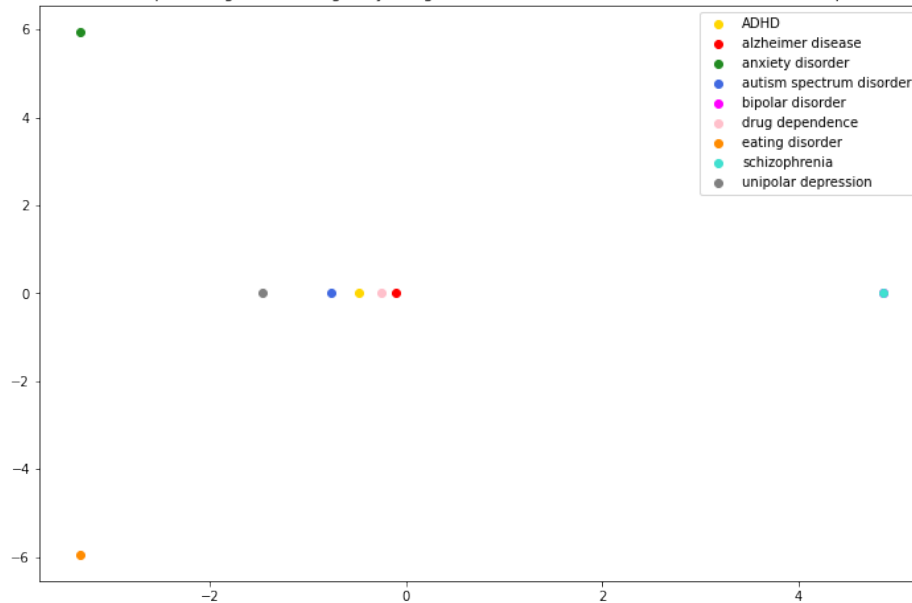
SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
return func(*args, **kwargs)
```

PCA, data reduced to 2D; Implicated-gene encoding; only using data with tissue associations; filtered to lowest X p-value variants per trait



<Figure size 432x288 with 0 Axes>

Interesting that all but two conditions (AnxD, ED) fall on a straight line (no Y component).