GWAS_encode_and_cluster

April 19, 2022

[1]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import AgglomerativeClustering
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 = '_cleaned.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]:
```
all_genes = set()
for df in dfs:
  genes = df['gene'].unique()
  [all_genes.add(gene) for gene in genes]

print(f"Found {len(all_genes)} total genes.")
```

```
Found 3259 total genes.
```

[5]:
```
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)}
```

[6]:
```
def encode_condition_df(df):
  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:
first_gene = all_genes_list[0]
df = pd.DataFrame([{'variant_and_allele': 'test', 'p_value': 0.01, 'trait':␣
 ↪'test', 'gene': first_gene}])
encoding = encode_condition_df(df)
assert encoding[0] == 1
assert sum(encoding) == 1
```

[7]:
```
# Encode them all!
encodings_vertical = pd.DataFrame({df['parent_trait'].unique()[0]:␣
 ↪encode_condition_df(df) for df in dfs})
encodings_vertical
```

[7]:

	ADHD	alzheimer disease	anxiety disorder	autism spectrum disorder	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	
4	0.0	1.0	0.0	0.0	
…	…	…	…	…	
3254	0.0	0.0	0.0	0.0	
3255	0.0	0.0	0.0	0.0	
3256	0.0	0.0	0.0	0.0	
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	1.0	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	0.0	0.0	
…	…	…	…	
3254	0.0	1.0	0.0	
3255	0.0	0.0	0.0	
3256	0.0	0.0	0.0	
3257	1.0	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	1.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	
…	…	…	…	
3254	0.0	0.0	0.0	
3255	0.0	1.0	0.0	
3256	0.0	1.0	0.0	
3257	0.0	0.0	0.0	
3258	0.0	1.0	0.0	

	unipolar depression
0	1.0
1	0.0

```
2                    0.0
3                    0.0
4                    0.0
…                    …
3254                 0.0
3255                 0.0
3256                 0.0
3257                 0.0
3258                 0.0

[3259 rows x 11 columns]
```

[8]:
```
# Actually needs to have vectors as rows not columns:
encodings = encodings_vertical.T
encodings
```

[8]:

	0	1	2	3	4	5	6	7	\
ADHD	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	
alzheimer disease	0.0	0.0	0.0	0.0	1.0	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	0.0	0.0	
bipolar disorder	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
drug dependence	0.0	1.0	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	1.0	0.0	0.0	1.0	0.0	1.0	
tourette syndrome	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
unipolar depression	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	8	9	…	3249	3250	3251	3252	3253	3254	\
ADHD	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	
alzheimer disease	0.0	0.0	…	0.0	0.0	0.0	1.0	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	…	1.0	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	1.0	
eating disorder	0.0	0.0	…	0.0	1.0	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	0.0	1.0	0.0	1.0	0.0	
tourette syndrome	0.0	0.0	…	0.0	0.0	0.0	1.0	0.0	0.0	
unipolar depression	1.0	1.0	…	0.0	0.0	0.0	0.0	0.0	0.0	

	3255	3256	3257	3258
ADHD	0.0	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	0.0	0.0	0.0

```
bipolar disorder           0.0   0.0   1.0   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              1.0   1.0   0.0   1.0
tourette syndrome          0.0   0.0   0.0   0.0
unipolar depression        0.0   0.0   0.0   0.0

[11 rows x 3259 columns]
```

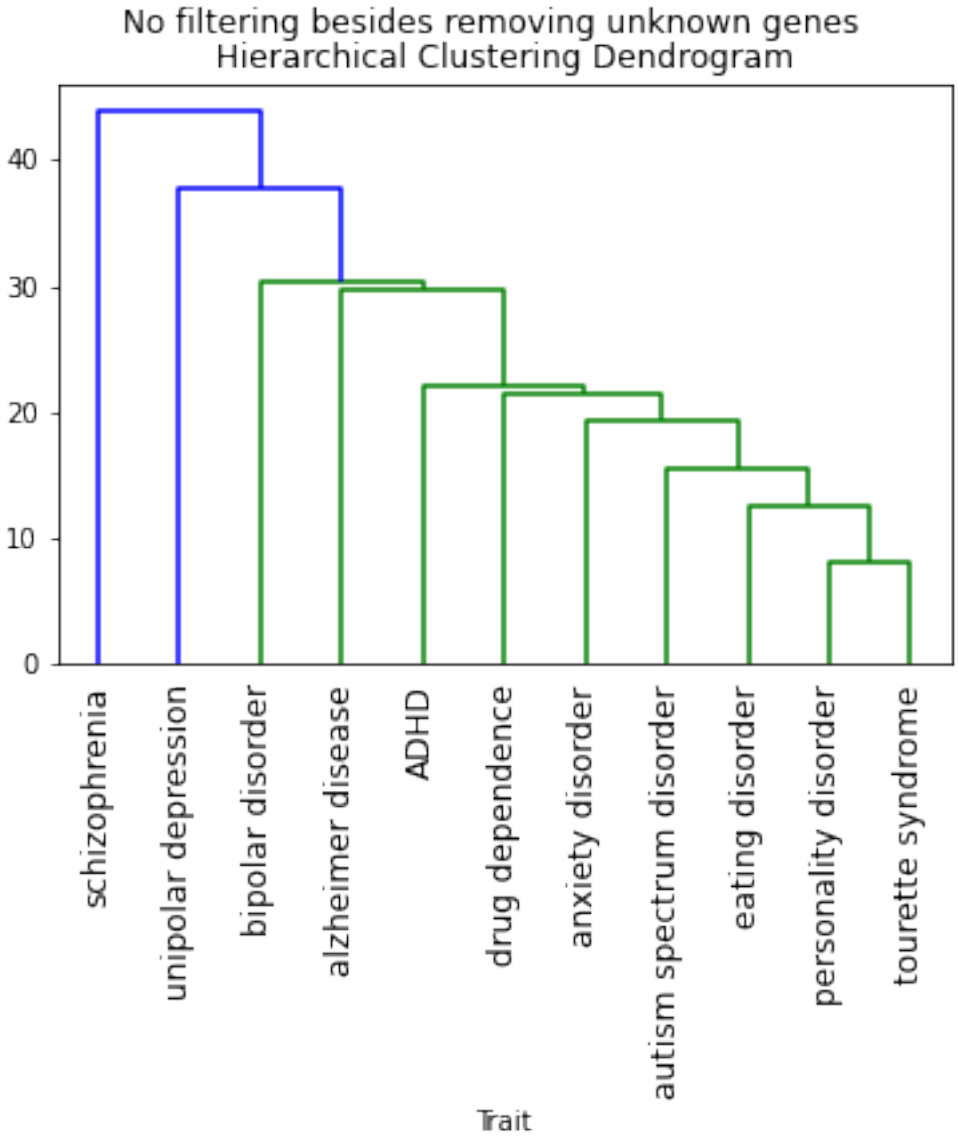
1.2 Naive Clustering (no special filtering)

[9]:
```
# 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)
```

[10]:
```
# 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)
```

[11]:
```
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:

[12]:
```
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])}
```

[12]:
```
{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).

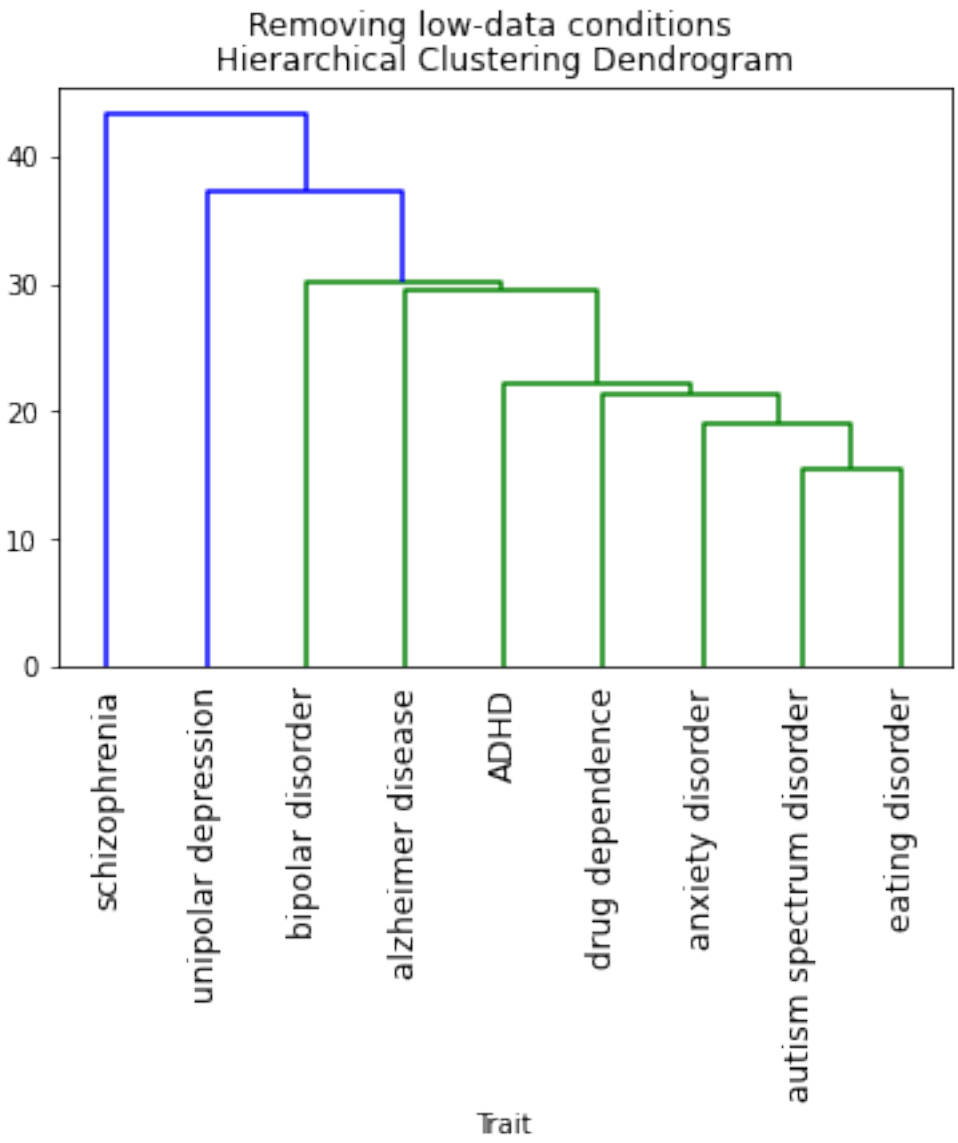
[13]:
```
excluded_traits = ['personality disorder', 'tourette syndrome']
all_df = pd.concat(df for df in dfs if df['parent_trait'].unique()[0] not in␣
 ↪excluded_traits)
all_df['parent_trait'].unique()
```

[13]:
```
array(['ADHD', 'alzheimer disease', 'anxiety disorder',
       'autism spectrum disorder', 'bipolar disorder', 'drug dependence',
       'eating disorder', 'schizophrenia', 'unipolar depression'],
      dtype=object)
```

[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) 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...

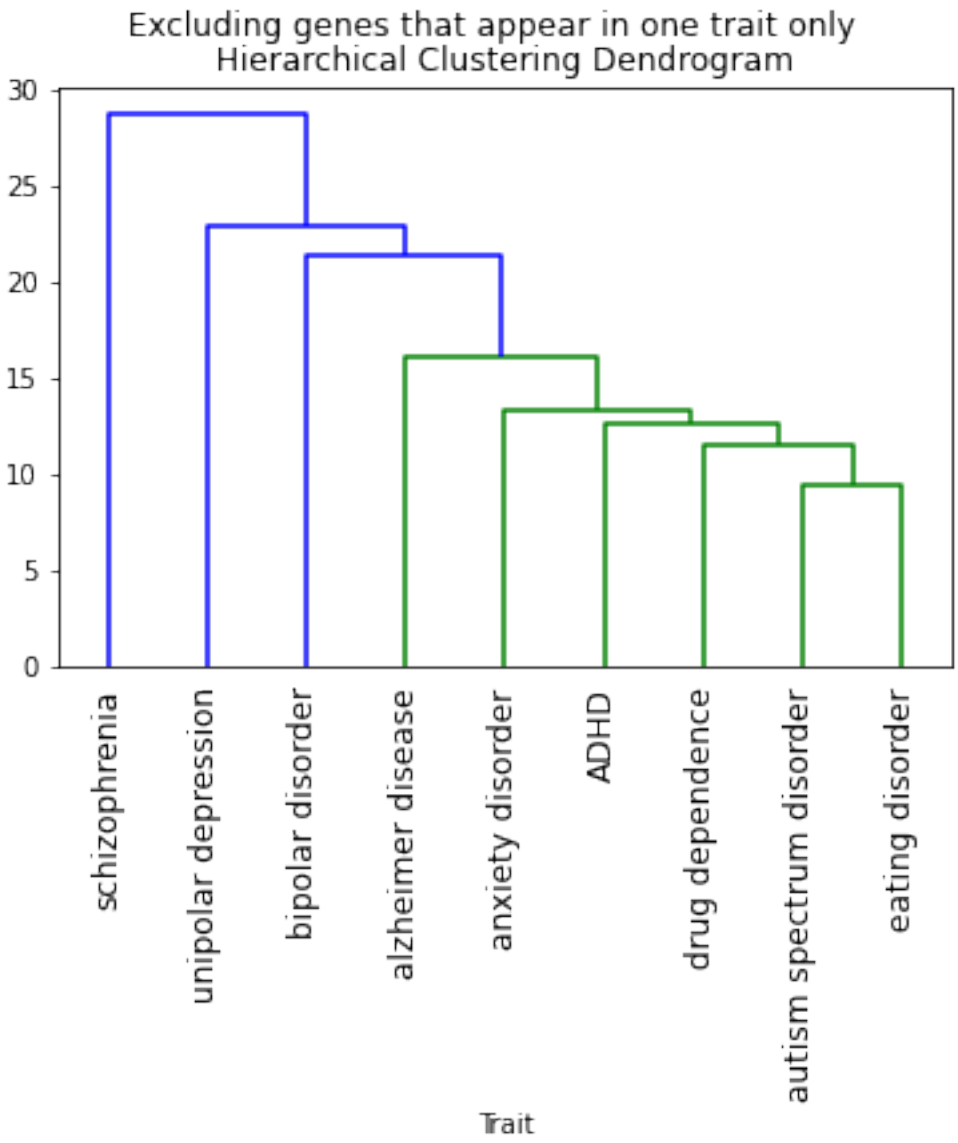
[17]:
```
print(f'Initial df size: {len(all_df)}')

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:

[19]:
```
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.

[26]:
```
trait_group_df = all_df.groupby('parent_trait').size()
trait_group_df
```

[26]:
```
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
```

[27]:
```
smallest_trait_size = trait_group_df.min()
smallest_trait_size
```

[27]:
```
41
```

[30]:
```
new_df = pd.DataFrame()
for trait in all_df['parent_trait'].unique():
  trait_df = all_df[all_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])

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

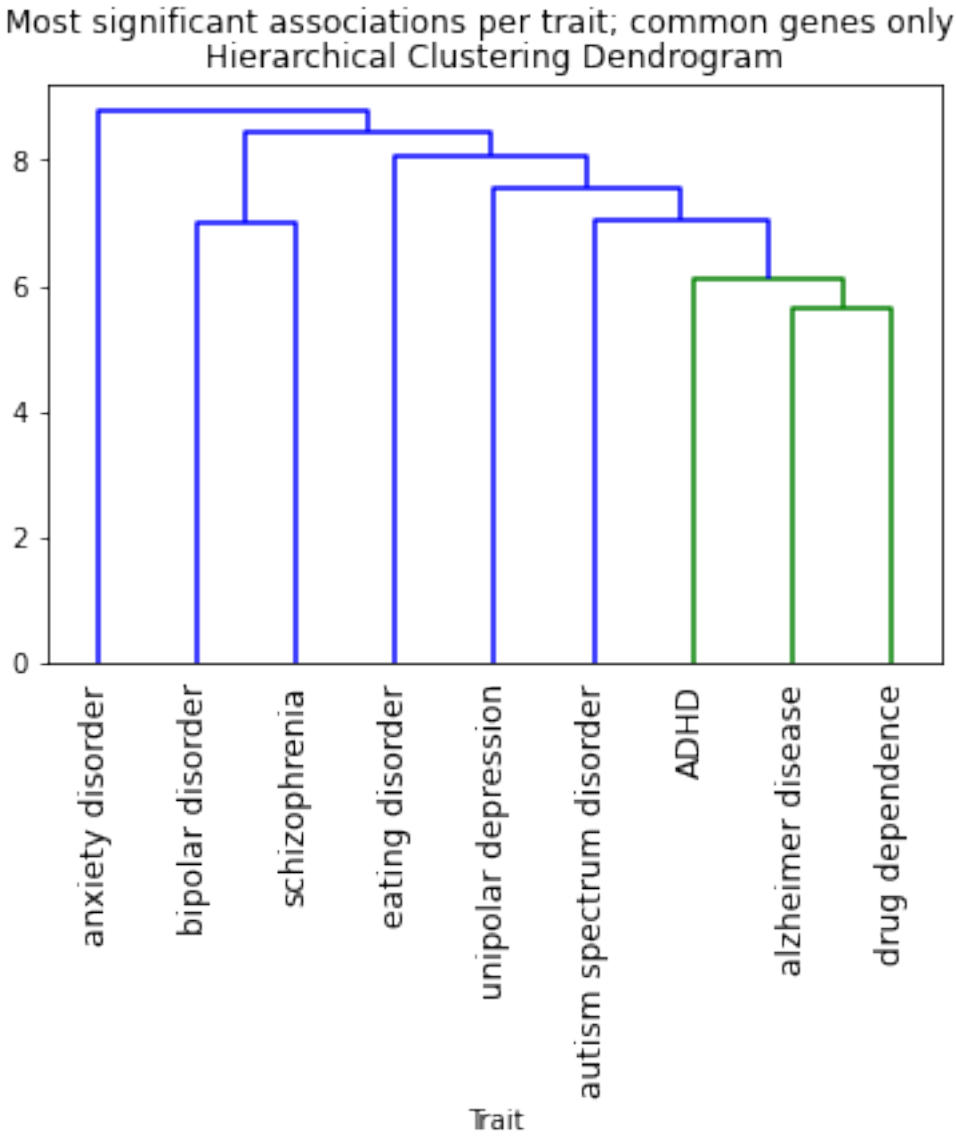
```
/usr/local/lib/python3.7/dist-packages/pandas/util/_decorators.py:311:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return func(*args, **kwargs)
```

[30]:
```
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.

[32]:
```
gene_based_hierachical_clustering(new_df,
                                  'Most significant associations per trait;␣
↪common genes only')
```



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.