GWAS_encode_and_cluster

April 18, 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

[14]:
```
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']
```

[56]:
```
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.

[57]:
```
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.
```

[58]:
```
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)}
```

[59]:
```
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
```

[60]:
```
# Encode them all!
encodings_vertical = pd.DataFrame({df['parent_trait'].unique()[0]:␣
 ↪encode_condition_df(df) for df in dfs})
encodings_vertical
```

[60]:

	ADHD	alzheimer disease	anxiety disorder	autism spectrum disorder	bipolar disorder	drug dependence	eating disorder	personality disorder	schizophrenia	tourette syndrome	unipolar depression
0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
…	…	…	…	…	…	…	…	…	…	…	…
3254	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3255	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3256	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3257	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
3258	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
2                    0.0
3                    0.0
4                    0.0
…                    …
3254                 0.0
3255                 1.0
3256                 0.0
3257                 0.0
3258                 1.0

[3259 rows x 11 columns]
```

[61]:
```
# Actually needs to have vectors as rows not columns:
encodings = encodings_vertical.T
encodings
```

[61]:

	0	1	2	3	4	5	6	7	\
ADHD	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
alzheimer disease	0.0	0.0	0.0	1.0	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	0.0	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	1.0	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	0.0	1.0	1.0	
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	

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

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

[11 rows x 3259 columns]
```

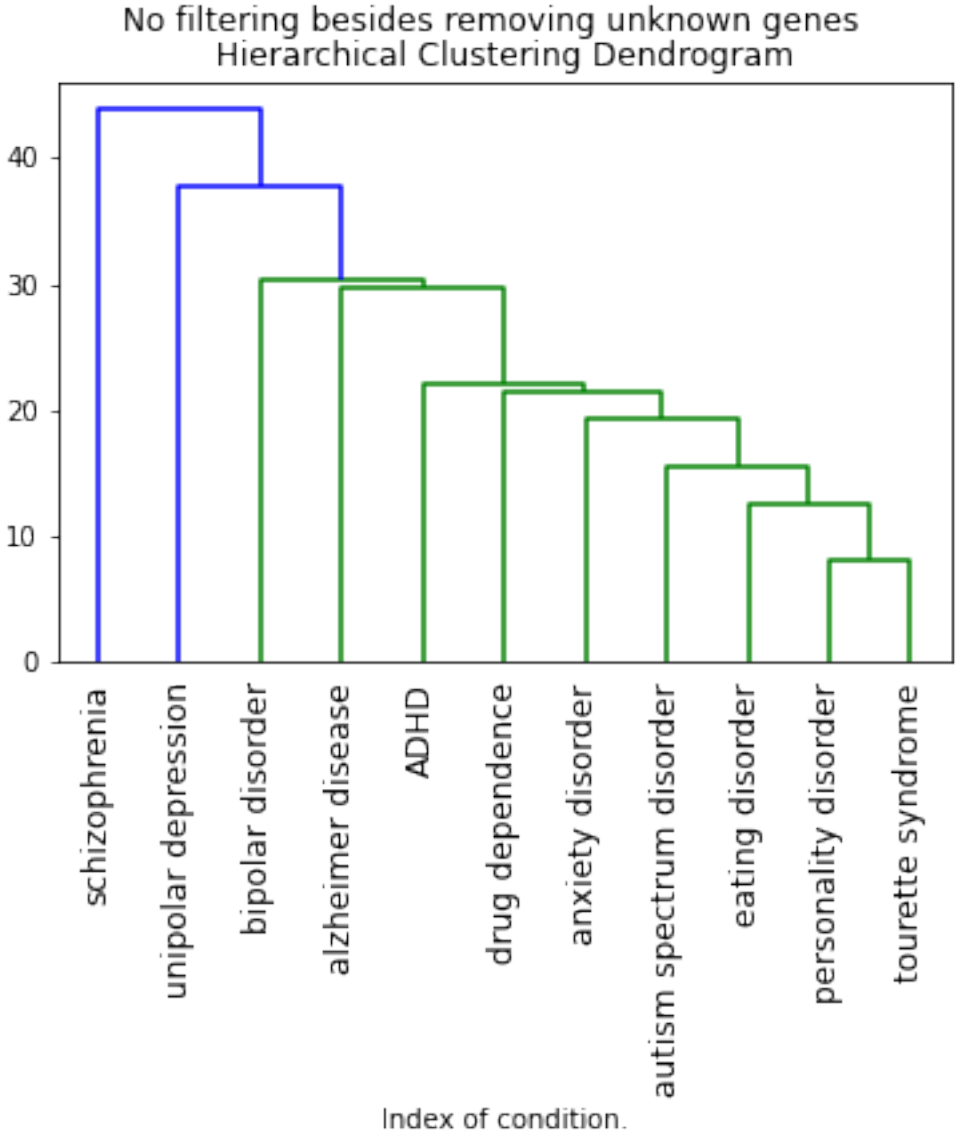
1.2 Cluster

[62]:
```python
# 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)
```

[63]:
```python
# 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, **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=encodings_vertical.columns,␣
↪leaf_rotation=90)
```

[64]:
```
plt.suptitle('No filtering besides removing unknown genes')
plt.title("Hierarchical Clustering Dendrogram")
# plot all levels of the dendrogram
plot_dendrogram(model, truncate_mode="level", p=11)
plt.xlabel("Index of condition.")
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).