Simple usage example

To run the entire analysis pipeline, load base.py and then run :
snpProportion, snpProportionNoInterpolation, sampleMeta, embedding, db_communities, output = runPipeline(<path to parameter file>)

Detailed usage example

To select appropriate parameter values, it is useful to run the code in several stages, optimizing one parameter value before moving on to the next stage.

Step 1: Load and prepare data
First, load the parameter file:
minSample, minloci, umapSeed, epsilon, cutHeight, admixedCutoff, filePrefix, inputCountsFile, inputMetaFile = loadParameters(parameterFile)

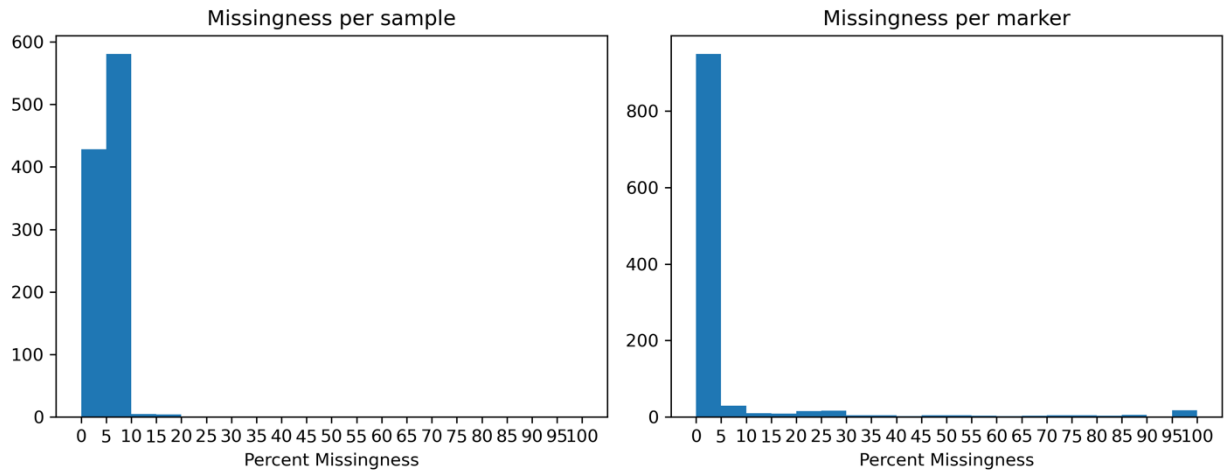
And then load the counts and metadata files:
snpProportion, snpProportionNoInterpolation, sampleMeta = filterData(inputCountsFile, inputMetaFile, minloci, minSample)

This step will convert the counts data into allele frequencies, and then filter out samples. Any references labeled “REMOVE” in the reference column will be removed. Next, the minloci parameter will remove markers that are missing data from greater than X proportion of samples, and then the minSample parameter will remove samples that are missing data from greater than Y proportion of the remaining markers. Any remaining missing values will be interpolated by taking the average of the row

Next, calculate the UMAP embedding of the data:
embedding = embedData(snpProportion, umapSeed)

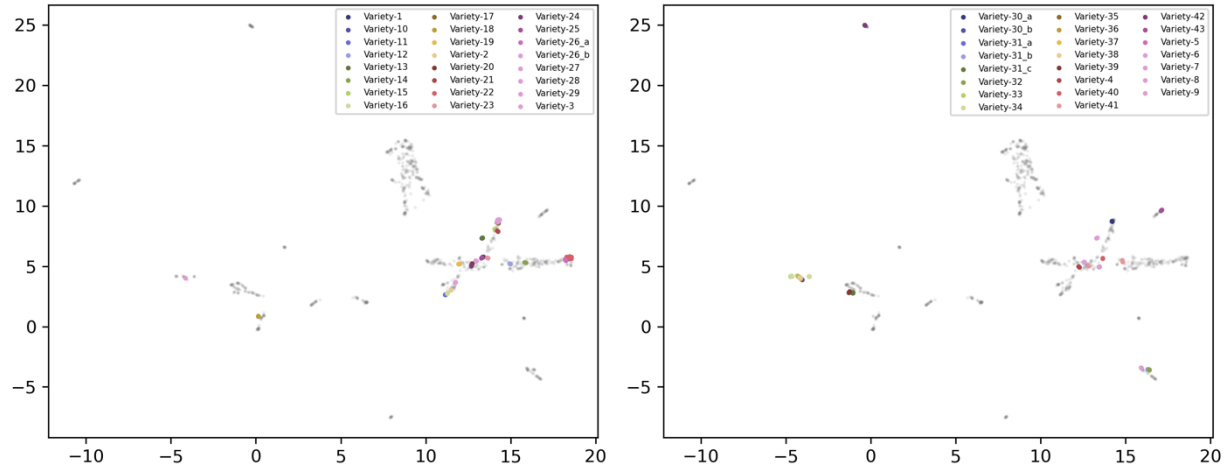
The default values for minloci and minSample are both 0.2, which should be appropriate for most datasets. To confirm this, you can run histogramMissingness which will generate a histograms of sample and marker missingness before any of the low quality samples have been removed, which can be helpful to select values for minSample and minloci.

plot.histogramMissingness(snpProportionNoInterpolation)



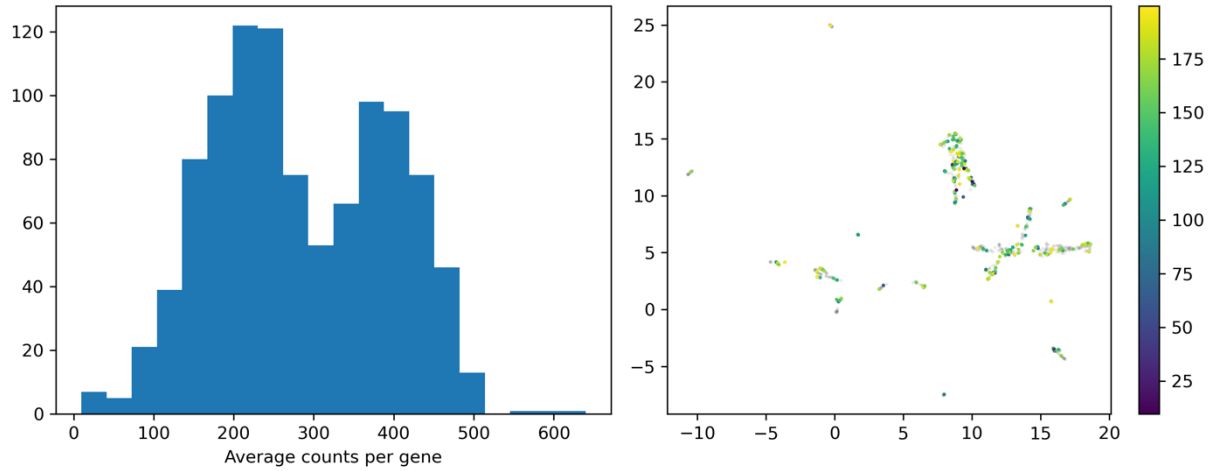
umapReferenceSeparate will generate a scatter plot with the UMAP embedding of the data where each reference variety is a different color (this function is not ideal when there are more than 40 reference varieties).

plot.umapReferenceSeparate(snpProportion, embedding, sampleMeta)



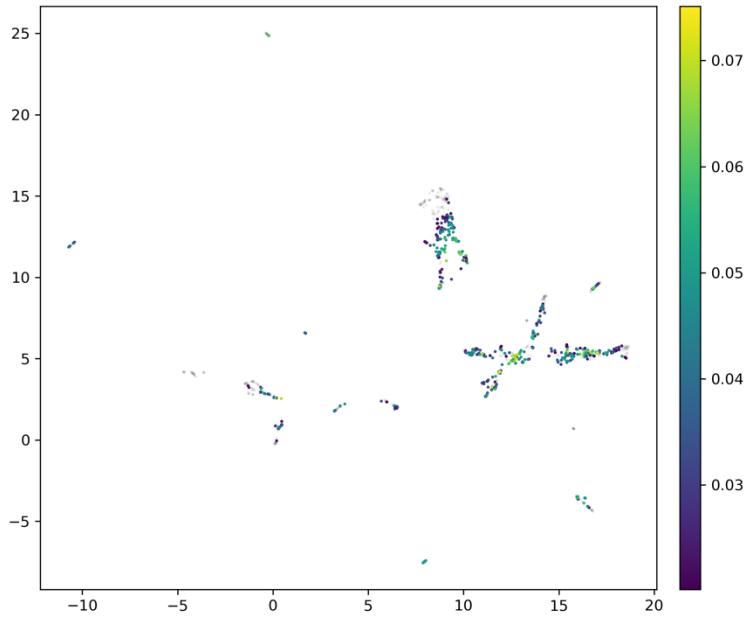
averageCounts will generate a figure with two subplots—a histogram of average counts per gene and scatterplot of the UMAP embedding where the samples are colored according to the with average number of counts per marker. This can be helpful to get an overall sense of sample quality. If all of the low count samples are clustering together then the filtering step should probably be more stringent and use larger values for minloci and/or minSample.

plot.histogramAverageCounts (inputCountsFile,snpProportion, embedding)



umapDivergence will generate a scatter plot of the UMAP embedding where samples are colored by divergence value. This is helpful to identify if there are clusters that only contain admixed samples.

plot.umapDivergence(snpProportion, embedding)



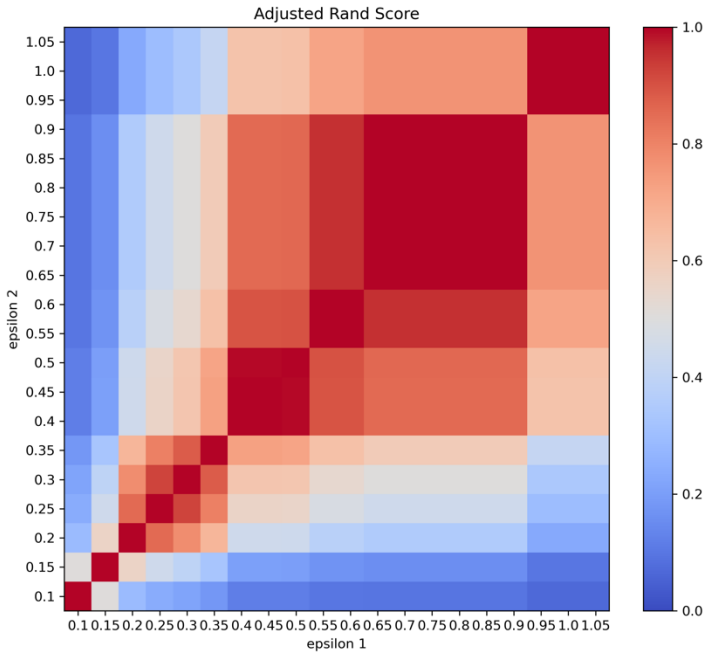
Step 2: DBSACN clustering

DBSCAN is a clustering method that identifies dense, well separated groupings of samples. The main parameter for DBSCAN clustering is epsilon, which controls how dense samples must be to get labeled as a single cluster.

evaluateEpsilon will return a matrix of adjusted rand score values for a range of epsilon values, which is helpful for deciding which particular values to compare. Adjusted Rand scores range

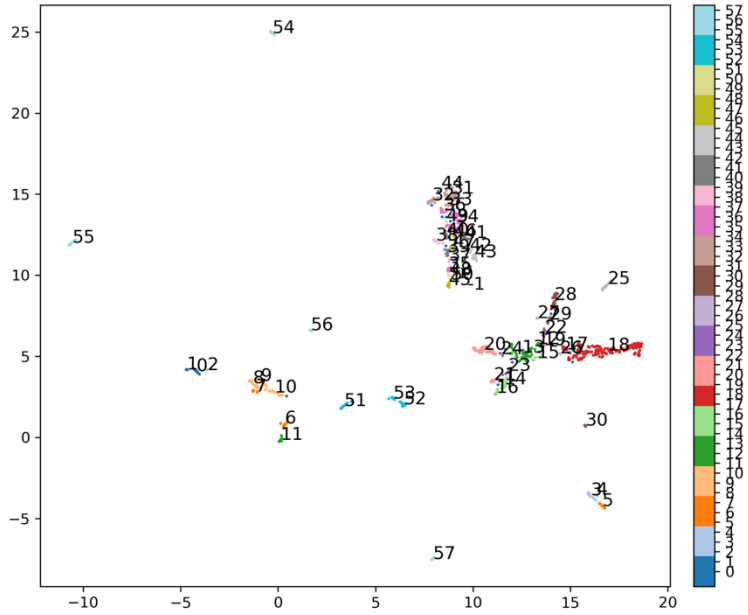
from 0 to 1 where 1 indicates samples have the exact same cluster assignments. For example, with this dataset epsilon values between 0.65 and 0.9 result in the same DBSCAN output, and a very different output from epsilon == 0.2.

evaluateEpsilon(embedding, filePrefix)

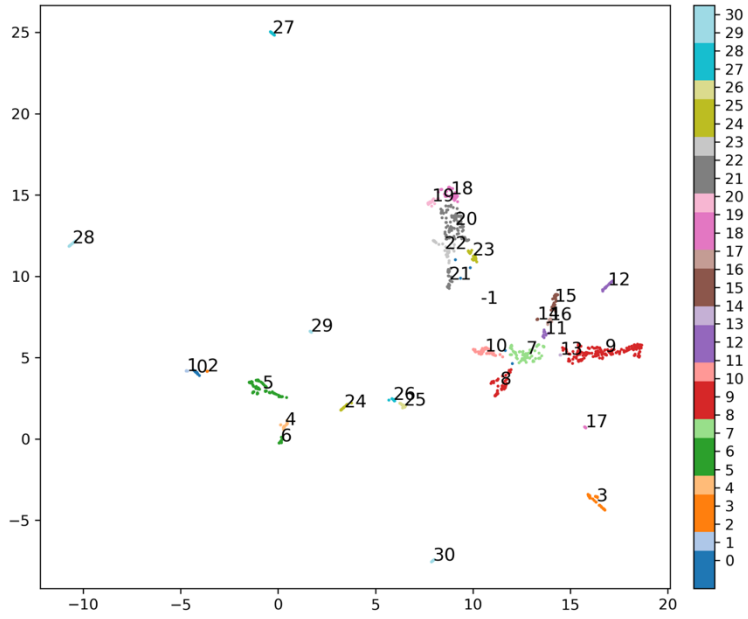


To perform DBSCAN clustering for a single epsilon value run:
db_communities = clusteringDBSCAN(snpProportion, sampleMeta, embedding, epsilon, filePrefix, admixedCutoff)

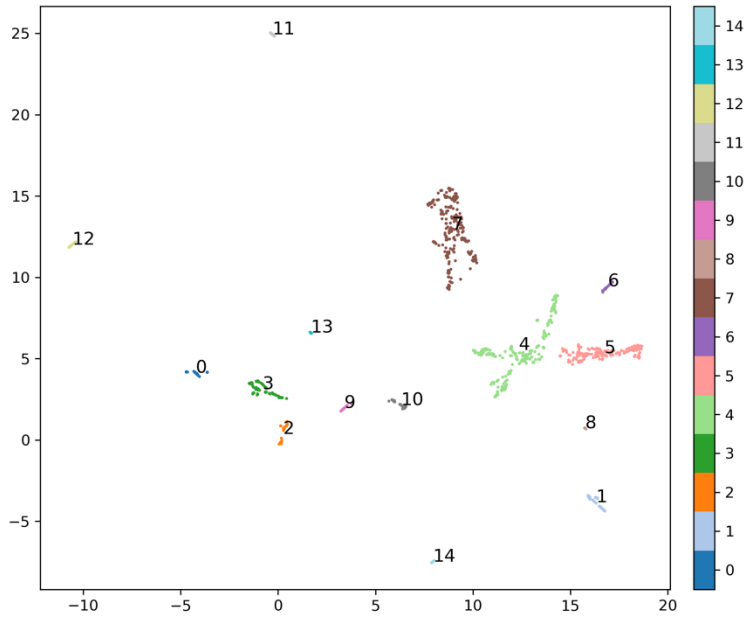
Epsilon 0.2



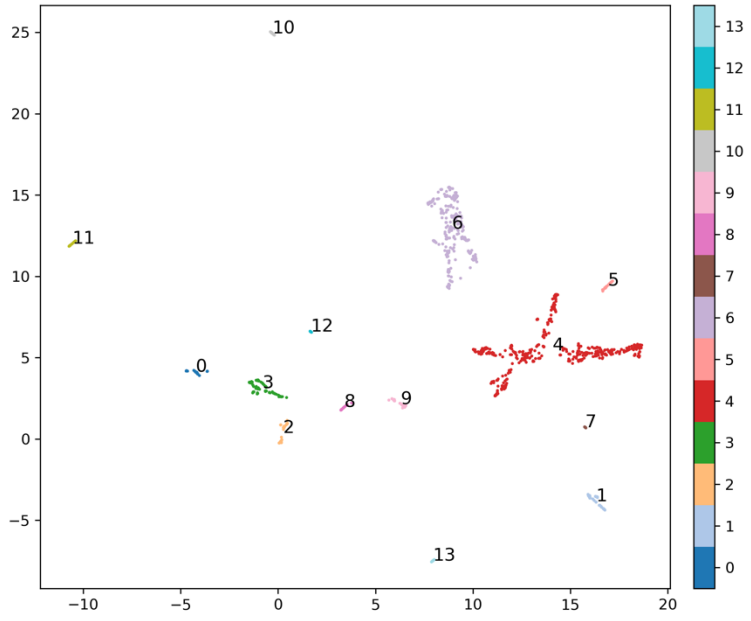
Epsilon 0.3



Epsilon 0.75



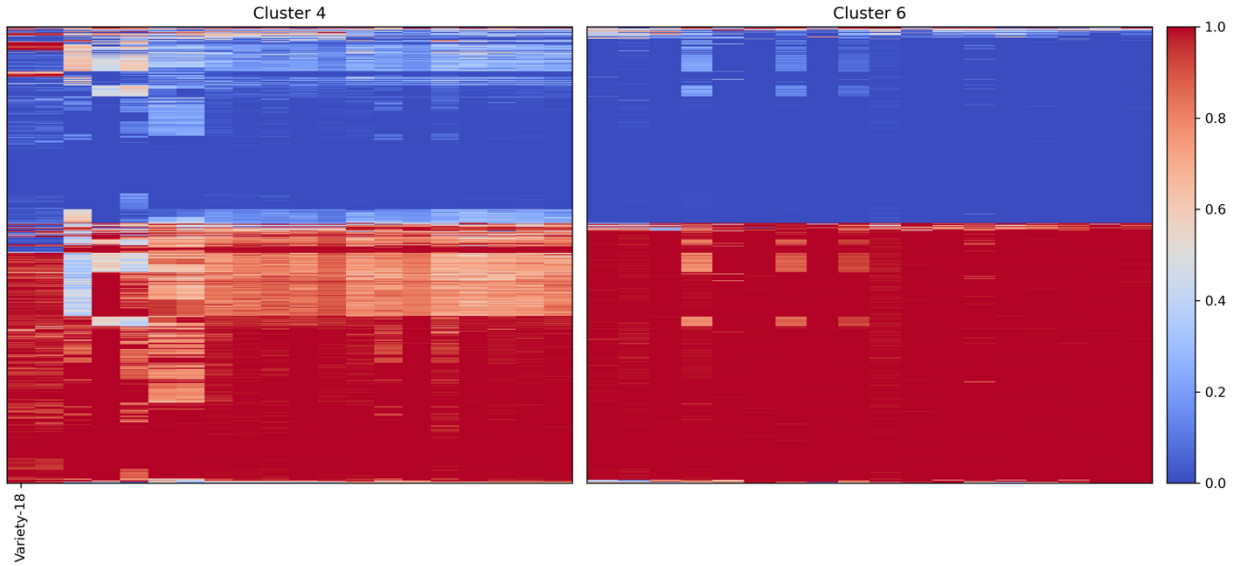
Epsilon 1.0



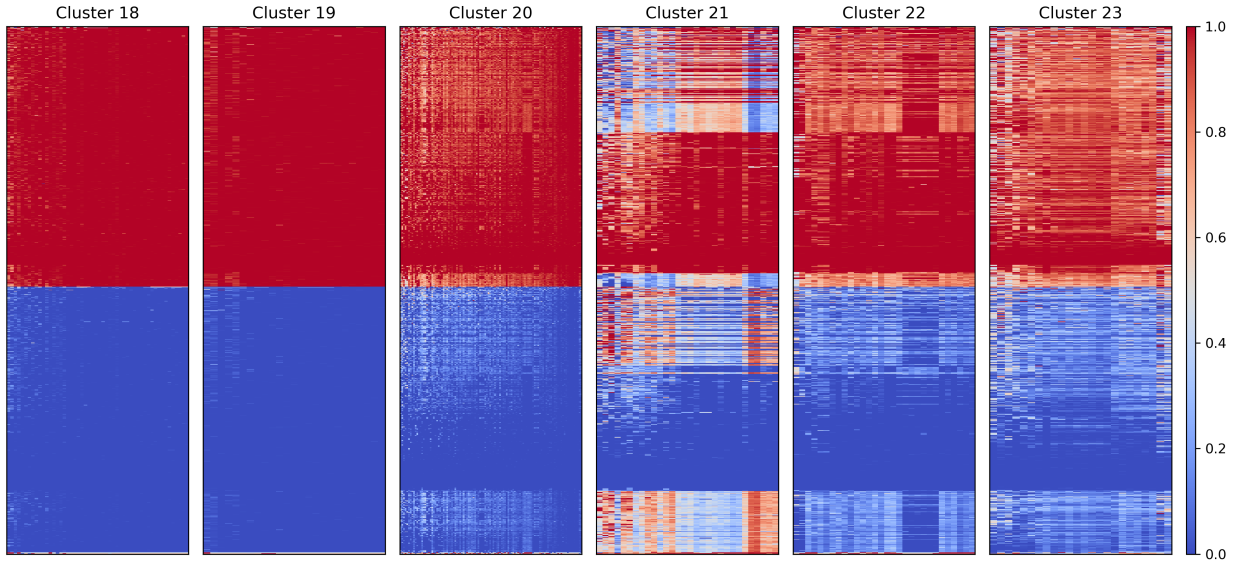
heatmapManyClusters will generate heatmaps for a list of clusters where the markers are in the same order for each subplot. This particular line of code will generate a subplot for clusters 2 and 3 and columns that correspond to references will be labeled on the x-axis.

plot.heatmapManyClusters(snpProportion, sampleMeta, db_communities, [2,3], tickType= referencesAll)

For example, cluster 2 for epsilon == 0.75 is split into two different clusters for epsilon == 0.3. Generating a heatmap for all of these smaller clusters using heatmapManyClusters results in the following figure. Both of these smaller clusters have a similar genetic fingerprint, suggesting that they should be combined and that an epsilon value of 0.75 is more appropriate than 0.3.

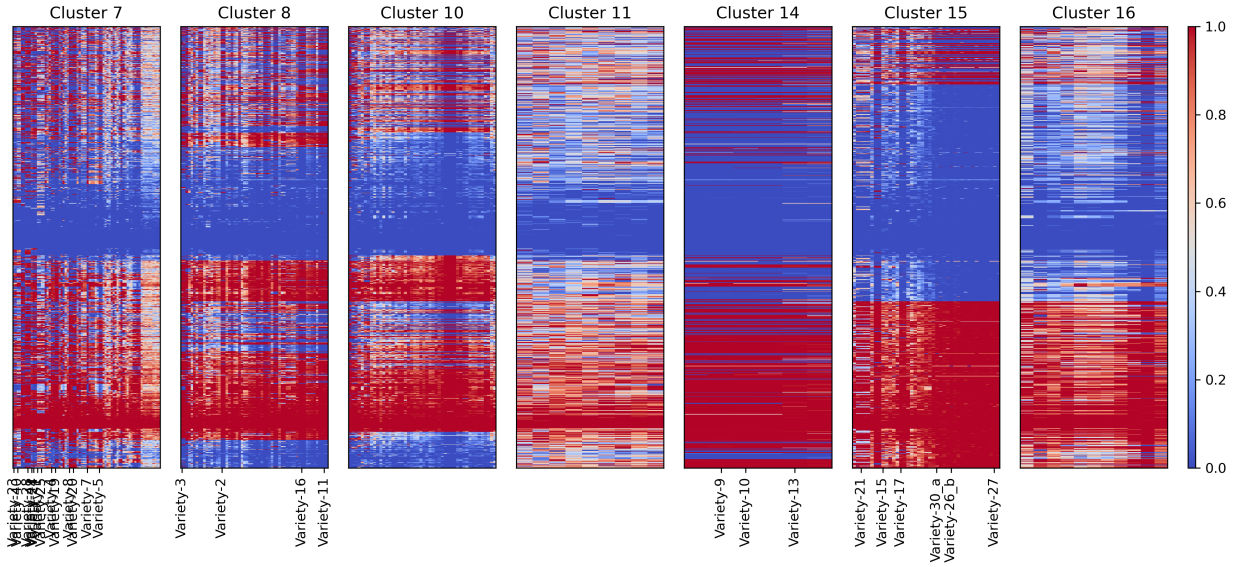


Cluster 7 for epsilon == 0.75 is split into six different clusters for epsilon == 0.3 and again all six of these smaller clusters have a similar genetic fingerprint, suggesting that 0.75 is a better value for epsilon.

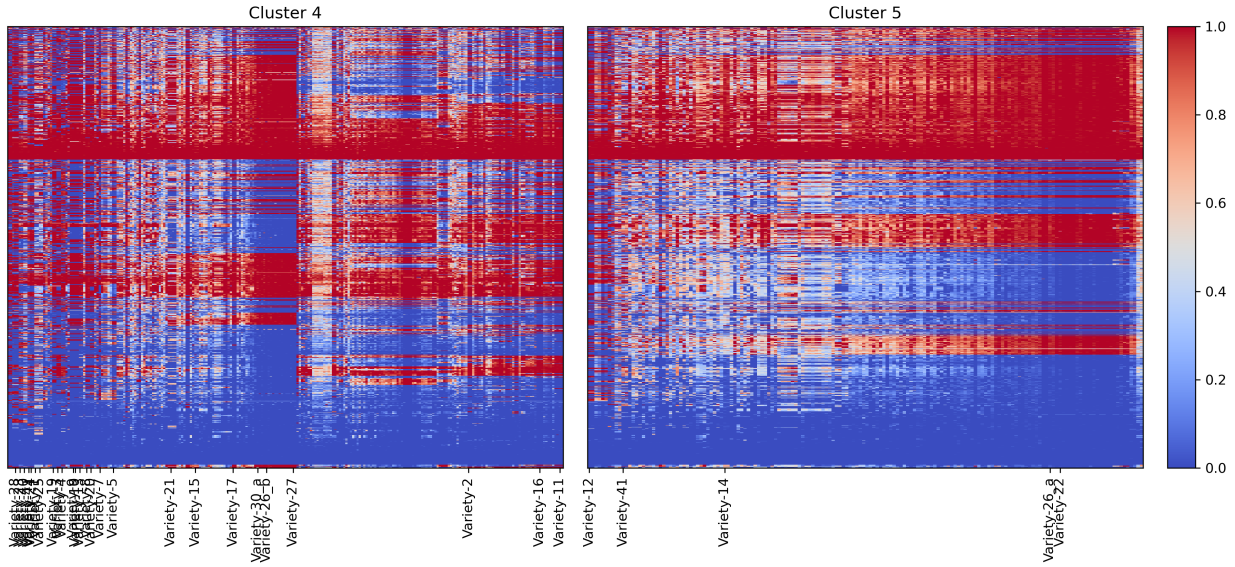


Cluster 4 for epsilon == 0.75 is split into seven different clusters for epsilon == 0.3. In this case, there appear to be at least two different very distinct genetics fingerprints, one fingerprint in cluster 7-11 and a second fingerprint in clusters 14-16. This suggests that neither 0.75 nor 0.3 is an optimal epsilon value, and in cases like this, the larger epsilon value should be used because the second stage of clustering is able to differentiate between multiple genetic fingerprints

whereas using a smaller epsilon value would mean that none of the field samples in cluster 16 could be labeled as the reference varieties in cluster 14 or 15.



Cluster 4 for epsilon == 1.0 is split into two different clusters for epsilon == 0.75. Both of these smaller clusters have distinct genetic fingerprints, which suggests that 0.75 is a sufficiently large value for epsilon.



Step 3: Hierarchical clustering

Many of the DBSCAN groupings still contain multiple reference samples, which suggests that finer scale clustering is required to get to the level of labeling samples as specific varieties. This second clustering step is done using hierarchical clustering with the correlation distance metric. The main parameter for hierarchical clustering is cut height, which determines what distance cutoff is used to “cut” a dendrogram into clades. If there are reference samples present in the

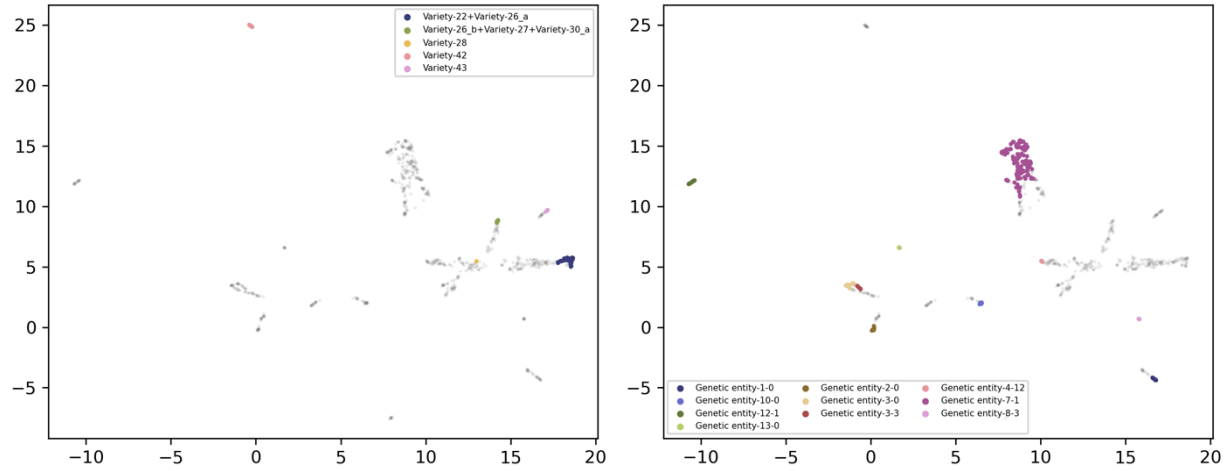
clade, all of the field samples will be labeled as those varieties, and otherwise the samples will be labeled as a new genetic entity.

To perform hierarchical clustering run:
output = labelSamples(snpProportion, sampleMeta, db_communities, embedding, cutHeight, admixedCutoff, filePrefix)

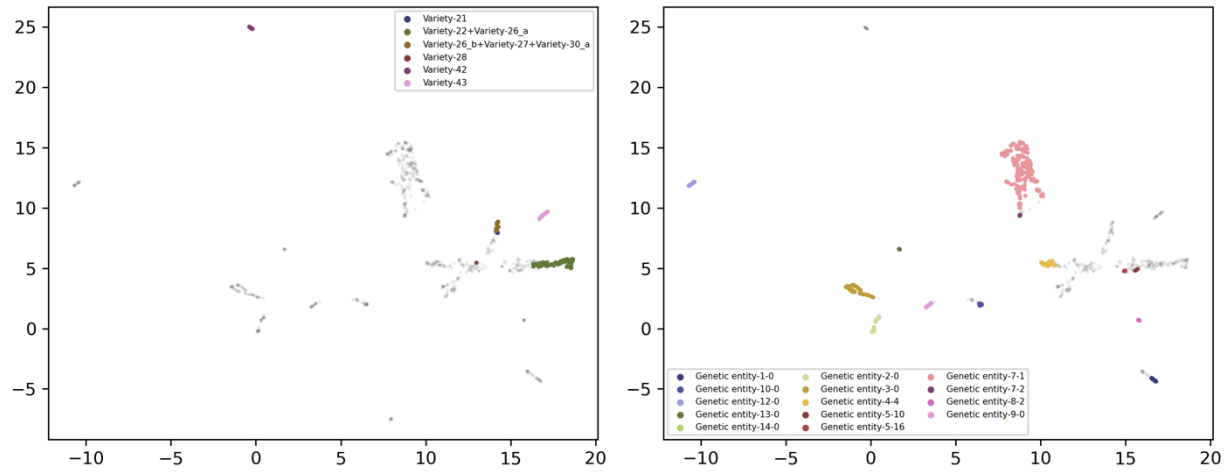
umapRefLandrace will generate a pair of scatterplots with the UMAP embedding where the samples are colored by the variety they have been labeled. The left subplot shows the reference varieties, and the right subplot shows genetic entities.

plot.umapRefLandrace(snpProportion, outputL, sampleMeta, 5, noRef=True)

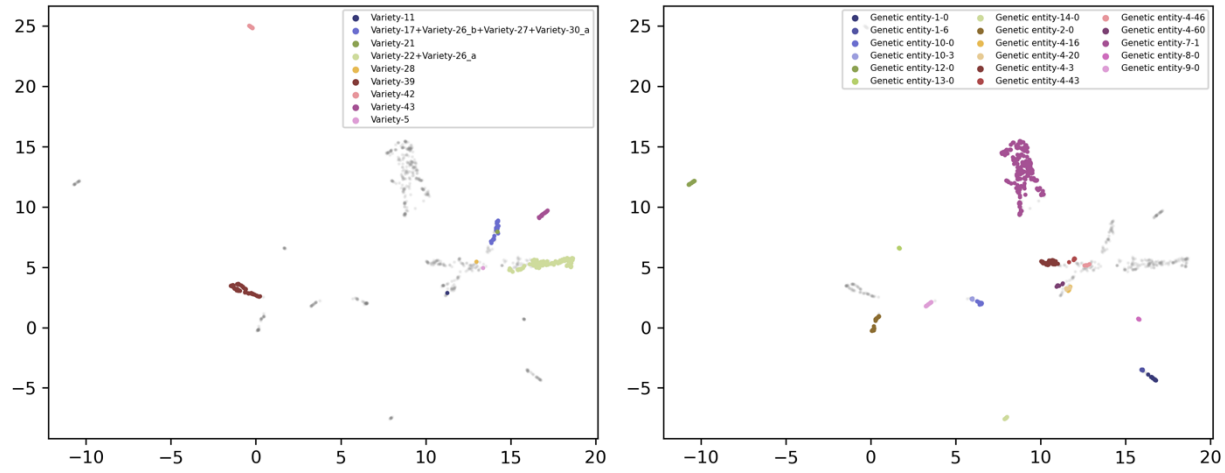
cutHeight 0.01



cutHeight 0.043



cutHeight 0.1

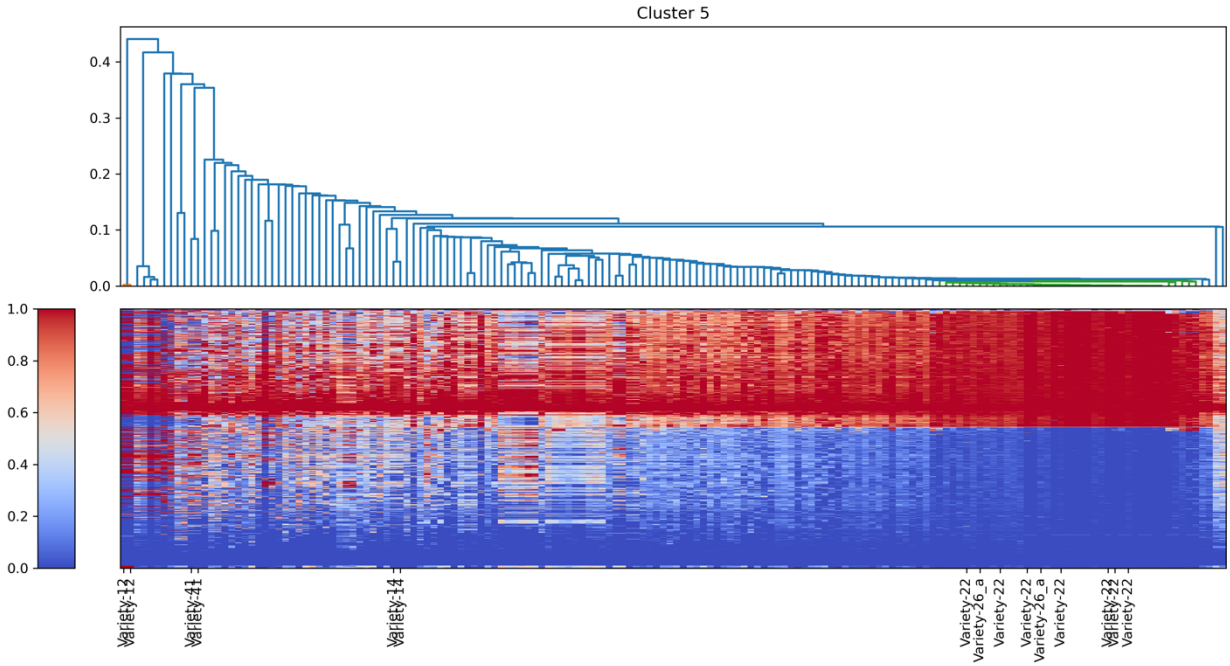


As the cutHeight value increases, one of the largest changes is how many field samples in cluster 5 are labeled as Variety-22+Variety-26_a. An appropriate cutHeight value will generate clades of samples with similar genetic profiles, and

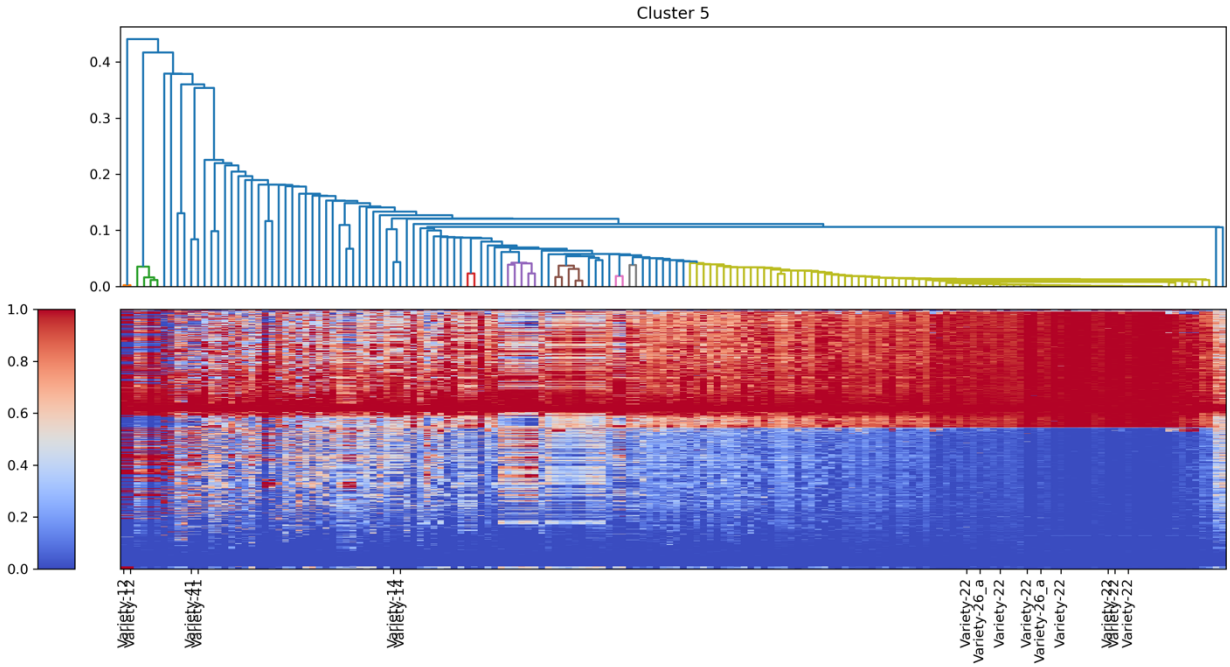
heatmapDendrogram will generate a paired heatmap and dendrogram for a specific cluster clusters where the samples are aligned in the heatmap and dendrogram. Groups of samples within the same clade are shown in the same color, and samples labeled blue are in a clade of size zero. This particular line of code will generate a figures for clusters 5.

plot.heatmapDendrogram(snpProportion, sampleMeta, db_communities, 5, cutHeight)

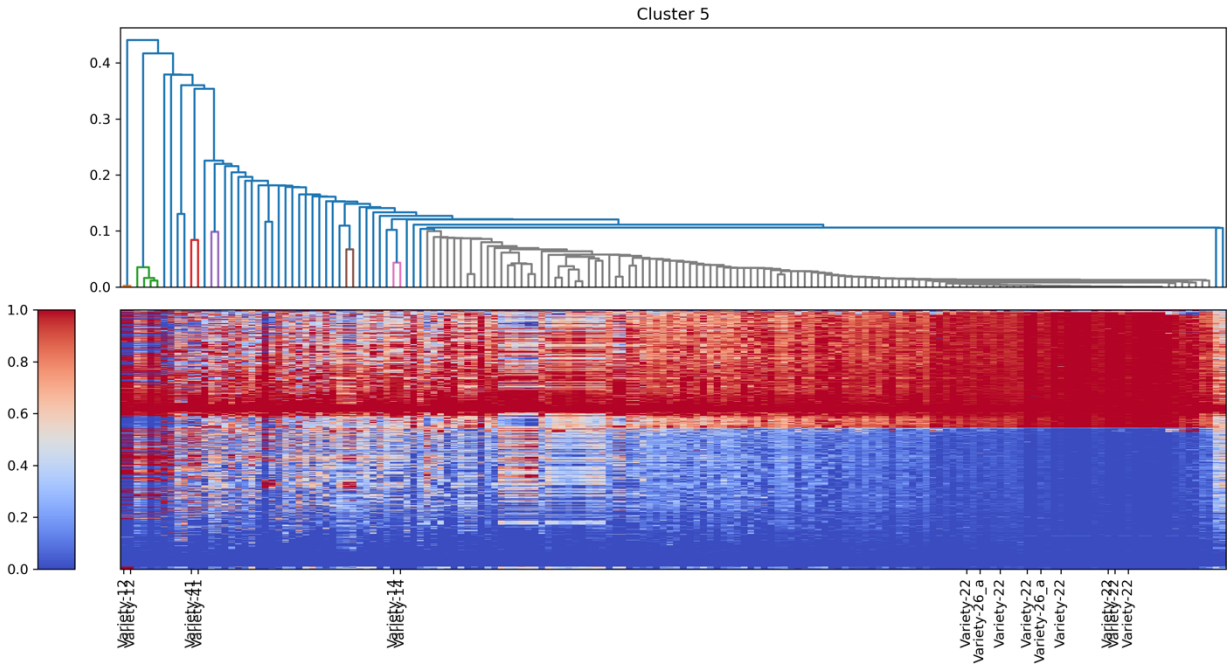
cutHeight 0.01



cutHeight 0.043



cutHeight 0.1



In cluster 5, for a cutHeight of 0.01, the clade of samples labeled as Variety-22+Variety-26_a (shown in green) includes all of the Variety-22+Variety-26_a references, and all of the samples in the clade visually have a similar fingerprint. However, there are additional samples to the left of the clade that also have a similar fingerprint. Increasing the cutHeight to 0.043 results in a clade that includes most of these samples, while increasing cutHeight further to 0.1 starts to include samples with a noticeably different fingerprint.

From only looking at cluster 5, it is not possible to determine the optimal cutHeight value or exactly how large the Variety-22+Variety-26_a clade should be. A cutHeight of 0.043 is roughly an appropriate value, but the optimal cutHeight might be a bit higher or lower than that (although 0.01 is clearly too tow and 0.1 is clearly too high). The next step would be to repeat this process for additional clusters, and find a cutHeight value that generates reasonable clades across all of the clusters. Some good starting points would be the clusters that contain: the most prevalent varieties, clades with multiple reference varieties, or varieties at a very different prevalence from what was expected based on prior knowledge of the crop.

There is no single optimal value for the cutHeight. One reason is that the level of intra variety genetic variability will depend on the crop. For vegetatively propagated crops, the variability will be low, but for open pollinated varieties the variability can be quite high. If the samples are pools of multiple plants, this will also increase the variability (for example if multiple varieties were pooled together). There can also be use cases where it is important to differentiate between varieties that are genetically similar (so a more stringent cutHeight value should be used), or where the goal is to identify improved vs unimproved samples and the specific variety label is less important (so a less stringent cutHeight value should be used). Overall however,

this approach should result in a reasonable starting point for a cutHeight value, which can then be refined through discussion with end users.