CM121: K-Means Clustering Project

In this project, I attempt to implement k-means clustering to data derived from DeRisi et al. This data is microarray counts for all *Saccharomyces cerevisiae* genes across different time points during the fermentation process to see what genes undergo upregulation and downregulation. The dataset contains 6400 genes across 7 time points: (0hr, 9.5hr, 11.5hr, 13.5hr, 15.5hr, 18.5hr, 20.5hr).

Implementation

In general, k-means attempts to cluster data across $n$ dimensions into $k$ clusters. In this case, we want to cluster 6400 genes across 7 dimensions into $k$ = 1:7 clusters. In this case, the Lloyd algorithm was implemented, which consists of the following steps:

(1) Initialize k centroids across 7 dimensions (time points)
(2) Calculate squared euclidean distances between each gene and the centroids
(3) Assign the genes membership to any of the k clusters depending on the centroid of cluster that is closest to the gene
(4) Calculate new centroids for the $k$ clusters by taking the mean values across all dimensions/all time points for the genes assigned to their respective clusters
(5) Repeat (2) - (4) until the assignments remain unchanged

In general, K-means seeks to minimize the loss function as defined below:

$$L(\boldsymbol{\mu}, \boldsymbol{\alpha}) = \sum_{k=1}^{K} \sum_{i=1}^{n} ||x_i - \mu_k||_2^2 \mathbb{1}\{\alpha_i = k\}$$

data — cluster center — 1 if data belongs to cluster k

We can evaluate the performance of a k-means run using Within Sum of Squares which measures the intercluster variation of a cluster k, where $\boldsymbol{D}$ = pairwise squared euclidean distance between all points belonging to cluster k and its centroid. WSS is given by:

$$W_k = \sum_{r=1}^{K} \frac{1}{n_r} D_r$$

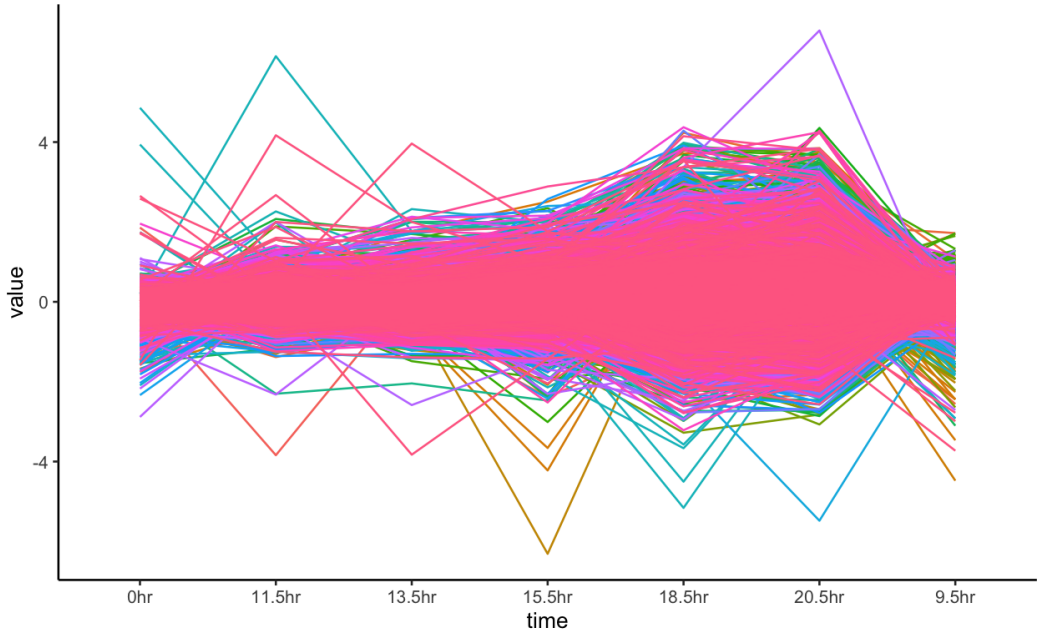
**The "within sum of squares"**

number of points in cluster r

As a result, we can evaluate and compare the performance of a k-means run that splits the data into $k$ categories by evaluating the average WSS as intuitively, we want the amount of variation within each cluster to be minimized. As the cluster numbers, WSS decreases as we have more clusters to capture the variation in the data. However, this provides us with diminishing returns as splitting our data into many clusters may not be biologically useful to observe changes in the data. We want to choose the simplest model/smallest value of k that explains the complex variation in the data. An important part of the implementation is centroid initialization. Simply using a random value for the centroids can result in unpredictable behavior and also increase the number of iterations needed for convergence. As a result, k++initialiser was implemented to initialize the clusters as detailed in the textbook.

Data Imputation

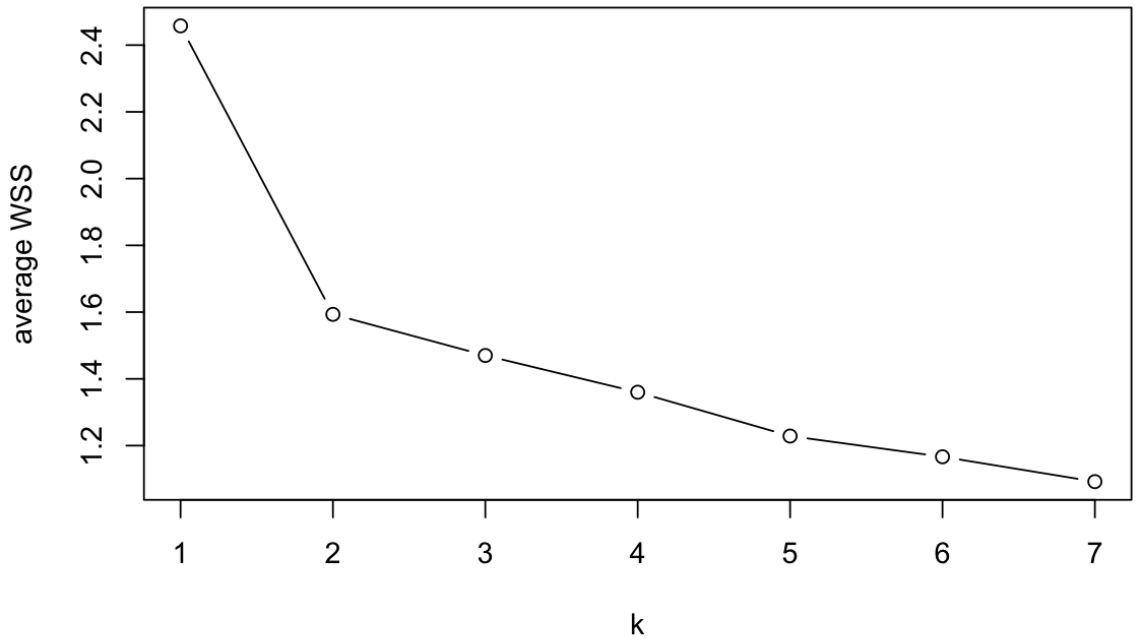
Several values in the dataset contain missing values (NAs). To deal with this, some form of imputation needs to be done. Simply removing genes that have any NAs reduces the genes that we can survey and therefore the possible conclusions that can be drawn from the data. Of 6400 genes, 72, 352, 220, 150, 115, 58, 256 genes contained missing values in the 0hr, 9.5hr, 11.5hr, 13.5hr, 15.5hr, 18.5hr, 20.5hr time points respectively. As can be seen from Appendix 1A, which details the patterns of missing values in the dataset - luckily there are very few genes that miss the majority of their values and therefore imputation seems like a valid approach. The R package, "mice" was used to impute values for the missing values using "Predictive Mean Matching. As can be seen in Appendix 1B, the distribution of imputed values (in pink) closely follow that of the observed data (in blue) giving confidence that the completed, imputed dataset would maintain its biological relevance. Below is a plot counts for the complete dataset:



## Results

K means was run on the completed dataset using k=1:7. K++initalizer was used to initialize the centroids

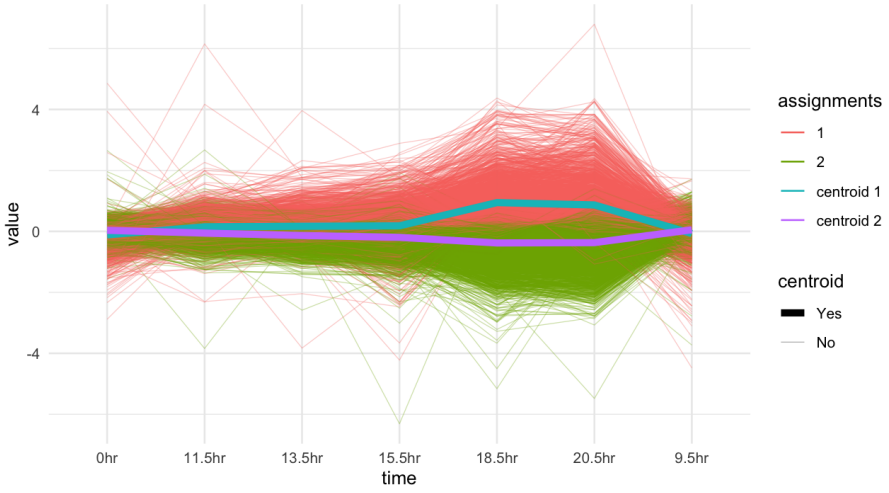
Finding Optimum K




As can be seen above, when plotting the average WSS for clusters after k-means with increasing values of K, the average WSS decreases as expected. However - there is a sharp elbow at k=2, indicating that this might be the optimum k. There is also a slight elbow at k=5 and thus, it may be investigated as well

For all k-means plots, see Appendix 2.

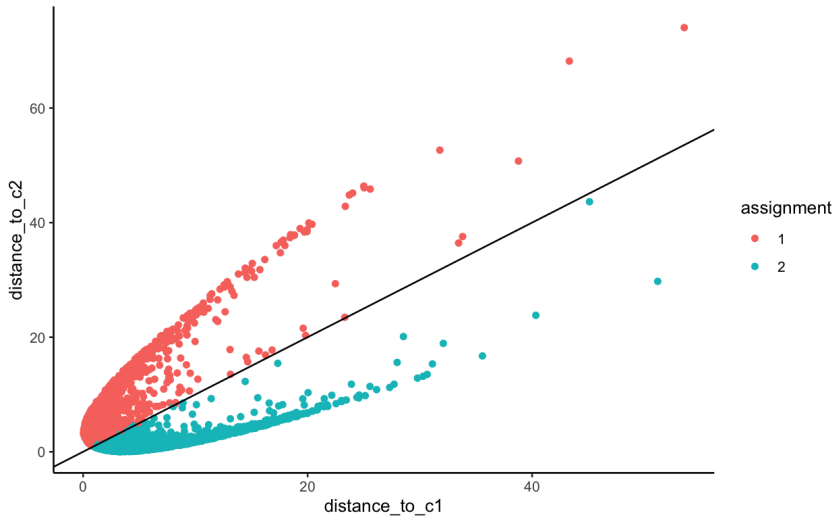
*k=2*
Cluster 1: 2910 genes
Cluster 2: 3490 genes
Average WSS: 1.593279




k=2 seems to partition the data quite intuitively, especially when looking at the genes that do not change drastically across time points. The clustering generally splits the data into genes that are upregulated during fermentation and those that are downregulated. This cluster seems biologically useful as we can look at the large set of genes that are upregulated and downregulated. However, as can be seen from the data, several genes

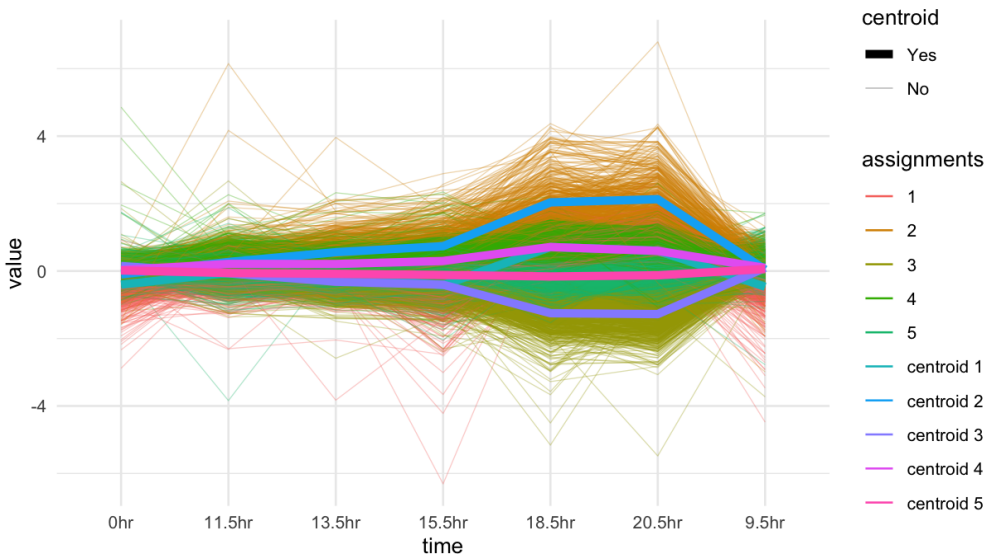
undergo drastic changes in expression between time points and the cluster centroids don't provide an intuitive way to explain the separation of these genes. Both centroids seem quite dissimilar to the expression pattern of these genes and so their classification seems to be unmeaningful as they are clumped with genes that have more standard expression patterns.

k=2 outlier plot




Here, the data is plotted according to its distance to c2 vs distance to c1. By definition of the centroids, genes fall on either side of the y=x line according to their assignment. However, it is clear to see that many outlier genes are far from both c2 and c1 in the upper right quadrant. In the context of the research questions i.e, what are the broad patterns of gene expression in the diauxic shift, we need to classify these highly changing genes as well, and thus, it would be more appropriate to use a larger number of clusters that captures these outliers, rapidly changing genes.

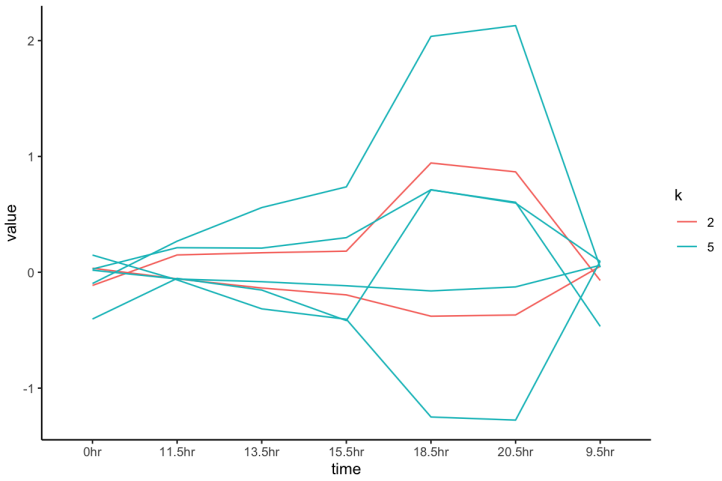
*k=5*
Cluster 1: 925 genes
Cluster 2: 450 genes
Cluster 3: 769 genes
Cluster 4: 1710 genes
Cluster 5: 2546 genes
Average WSS: 1.228657




K=5 does a great job of not only capturing the largely invariable up-regulated and down-regulated genes but also the patterns of genes that undergo rapid changes. These are represented by cluster 1 which undergoes sharp downregulation at 15.5hr and 9.5hr, cluster 2 which undergoes rapid upregulation in the 18.5 and 20.5 hr time points, and finally, cluster 3 which undergoes rapid downregulation in the 18.5 and 20.5 time points. This represents a much more biologically meaningful clustering as it can capture most of the broad patterns in the

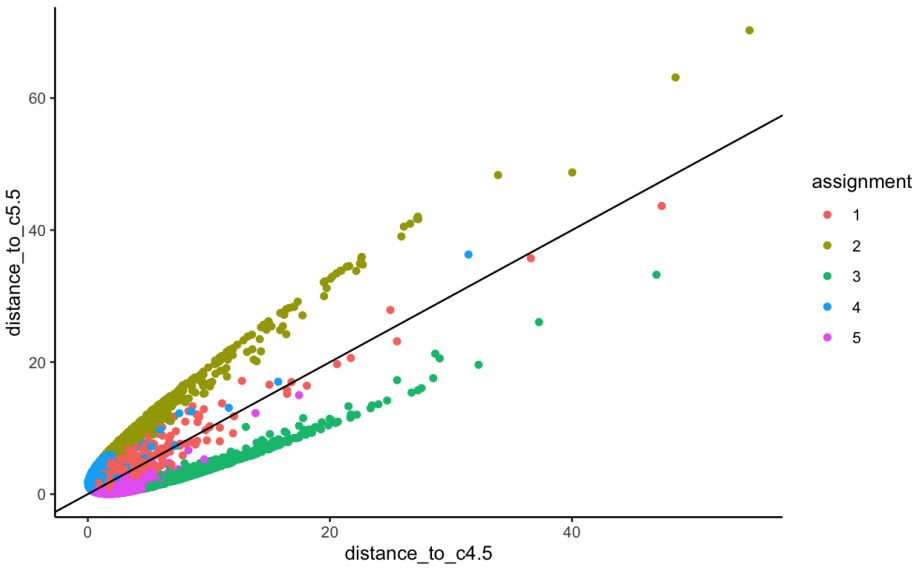
data. As is clear from the plot below, k=5 centroids can capture the highly variable genes much better than k=2 is and this is further reflected in the outlier plot

Centroids for k=2 and k=5

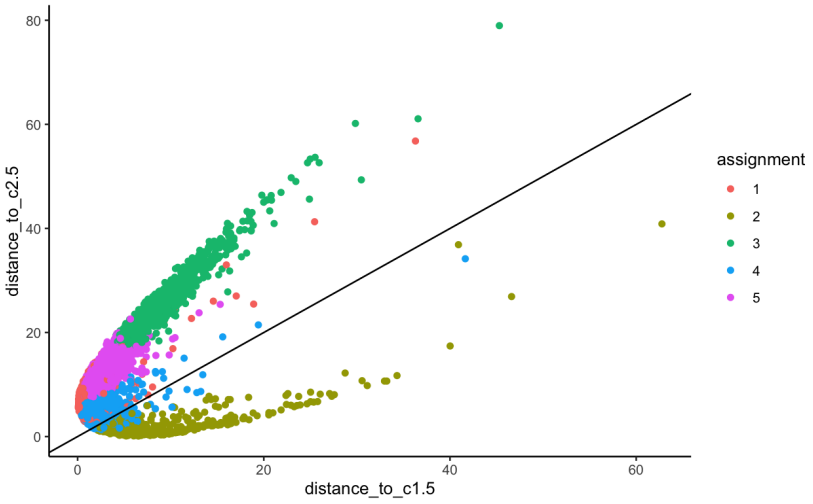



Pairwise plots for k=5 to see cluster outliers:

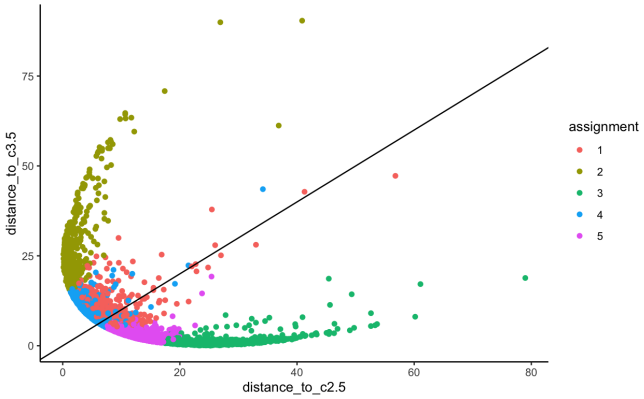
The plots above show that centroid 4 and centroid 5 most closely match centroid 1 and centroid 2 of the k=2 case. Thus, we expect these centroids to explain variation for the majority of the genes. When comparing the distance to c4 vs distance to c5 of the data points, we wee that as expected, it is separating c4 and c5 points very well (blue and purple) but most notably, there are few to none c4 or c5 outliers demonstrating that we are capturing the largely invariant genes effectively. This invariance is also demonstrated by the points' general closeness to the y=x line. The outliers represent the highly variable genes, as they are assigned to clusters 1,2, and 3.




In this plot, we are comparing distances to cluster 1 vs distances to cluster 2. When looking at the cluster plot, we can see that clusters 1 and 2 both capture genes that undergo drastic upregulation. Furthermore, compared to clusters 4 and 5, these genes (red and yellow) are quite different from each other and so, we see that the plot is generally more open i.e, fewer points that fall on the y=x line. This demonstrates that centroid 1 and centroid 2 are capturing unique patterns of expression.



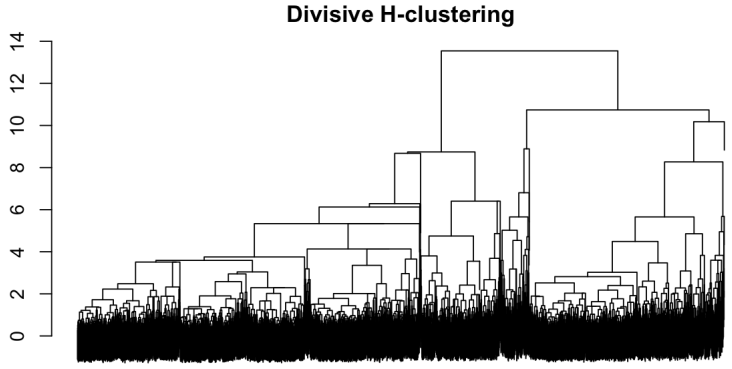
When comparing the distance to centroid 2 vs distance to centroid 3, we see a much wider plot. This is because these centroids are representing genes with opposite expression patterns. Genes assigned 2 and 3 are the furthest away from each other in this plot and notably, there are no outliers in the upper right quadrant. This indicates that these genes in clusters 2 and 3 are being accurately captured by their respective centroids.




**Thus, it seems like k=5 is doing a great job of capturing the data.**

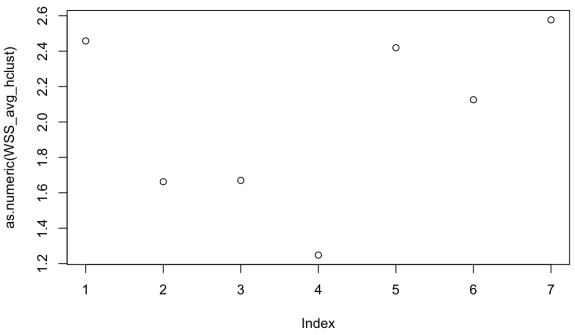
## Hierarchical Clustering

To compare clustering methods, I clustered the data using R's cluster package. Below is the hierarchical clustering of the euclidean distance matrix of the data using a divisive (top to bottom) method.

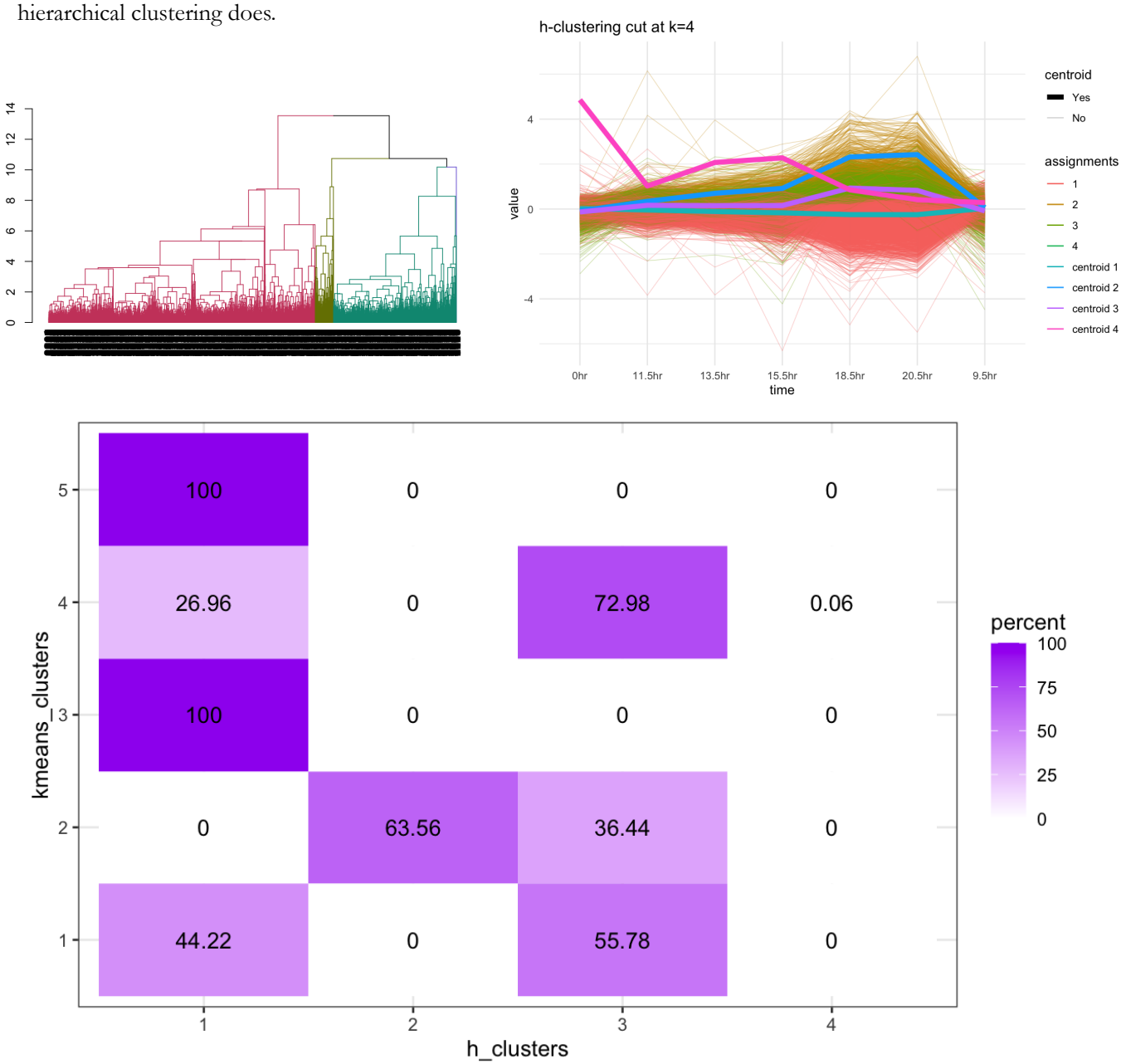



If we were to evaluate the WSS of the hierarchically-assigned clusters using centroids calculated by average expression values, we see something strange. We see the predicted decrease in average WSS values as we increase k=4 but after k=4, the WSS avg becomes unpredictably larger. This is because after k=3, very few genes are assigned to the clusters (see Appendix 3). And so, when calculating the squared euclidean distance using the centroid that captures only 2 points, the distance is very large. However, this represents an important distinction between the two clustering algorithms: WSS is guaranteed to converge in k-means, not hierarchical.

```
        [,1]
[1,] 2.457688
[1] 2.457688
        [,1]     [,2]
[1,] 1.399587 1.925992
[1] 1.662789
        [,1]     [,2]     [,3]
[1,] 1.399587 2.458948 1.151642
[1] 1.670059
        [,1]     [,2]     [,3] [,4]
[1,] 1.399587 2.458948 1.134533    0
[1] 1.248267
        [,1]     [,2]     [,3]     [,4] [,5]
[1,] 1.399587 2.211412 1.134533 7.350009    0
[1] 2.419108
         [,1]     [,2]     [,3]     [,4]     [,5] [,6]
[1,] 0.8389826 1.21933 2.211412 1.134533 7.350009    0
[1] 2.125711
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6] [,7]
[1,] 0.8030954 1.21933 2.211412 1.134533 5.318718 7.350009    0
[1] 2.576728
```



Thus, as per this elbow plot, the most efficient hierarchical clustering is at k=4. In the plot below, the different colored branches correspond to the 4 different clusters. Cluster 4 only represents one gene and this can be reasoned when looking at centroid 4 - it is quite dissimilar from the rest of the data and as a result, is only capturing 1 gene. I argue that k-means does a better job of explaining the different gene expression patterns than hierarchical clustering does.



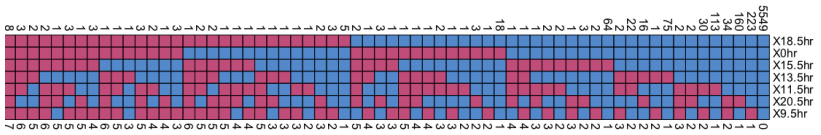

In the heatmap, we can see a proportion table of how the k_means clusters are distributed across the h_clust clusters. If we remember from our k-means = 4, clusters 4 and 5 represented the largely invariant genes. These are distributed across h_clusters 1 and 3 (which makes sense as these account for most of the genes). The highly upregulated genes in k_cluster 2 are distributed across h_clusters 2 and 3 and the highly downregulated genes in k_cluster 1 and 3 are distributed across h_clusters 1 and 3. Thus h_clusters is doing a bad job of explaining the variation in the data as h-cluster 3 contains genes that undergo little change, and rapid downregulation. However, h_cluster 3 seems to represent only downregulated genes.

# Conclusion

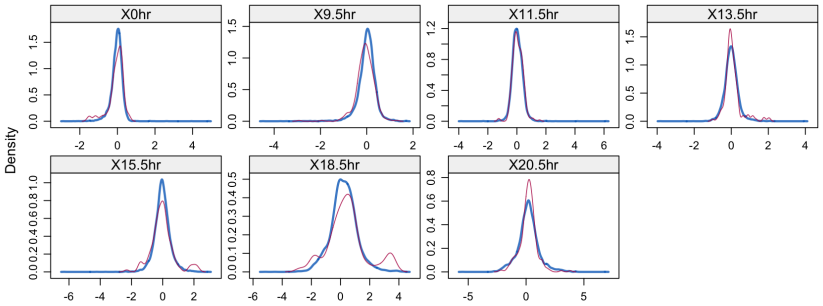
To summarise, k-means was implemented on yeast microarray data to uncover general patterns of gene regulation during the diauxic shift. K-means with a value of 5 seemed to explain the complex variation of the data the best. Similarly, hierarchical clustering was also implemented but, using our k-means implementation as a benchmark, performed badly as measured by its ability to capture broad variation in the data.

# Appendix

1A)imputation patterns




1B) imputation density plot

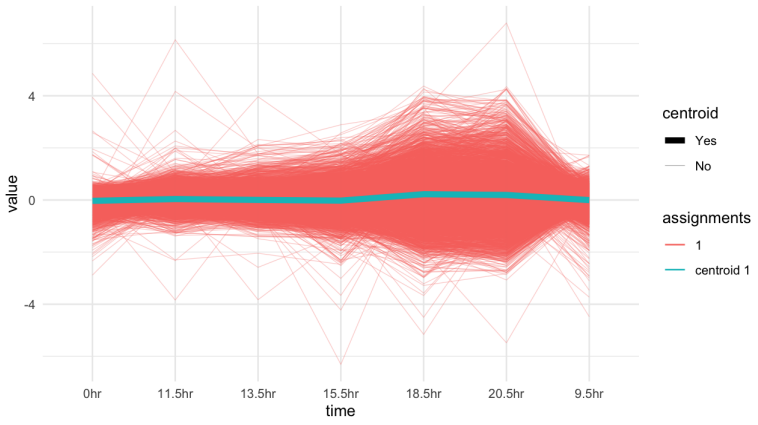



3) H-clustering assignments

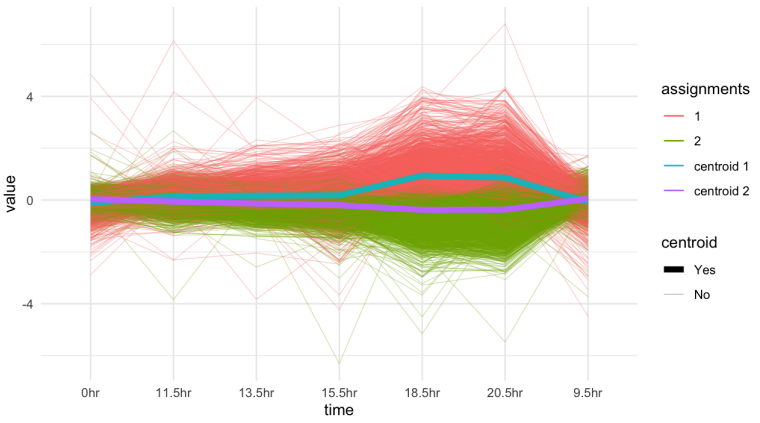
```
# A tibble: 1 × 2
  `cut_tree[[i]]` total
            <int> <int>
1               1  6400
# A tibble: 2 × 2
  `cut_tree[[i]]` total
            <int> <int>
1               1  4185
2               2  2215
# A tibble: 3 × 2
  `cut_tree[[i]]` total
            <int> <int>
1               1  4185
2               2   286
3               3  1929
# A tibble: 4 × 2
  `cut_tree[[i]]` total
            <int> <int>
1               1  4185
2               2   286
3               3  1928
4               4     1
# A tibble: 5 × 2
  `cut_tree[[i]]` total
            <int> <int>
1               1  4185
2               2   284
3               3  1928
4               4     2
5               5     1
# A tibble: 6 × 2
  `cut_tree[[i]]` total
            <int> <int>
1               1  3396
2               2   789
3               3   284
4               4  1928
5               5     2
6               6     1
# A tibble: 7 × 2
  `cut_tree[[i]]` total
            <int> <int>
1               1  3390
2               2   789
3               3   284
4               4  1928
5               5     6
6               6     2
7               7     1
```

2) K-means plots for different values of k

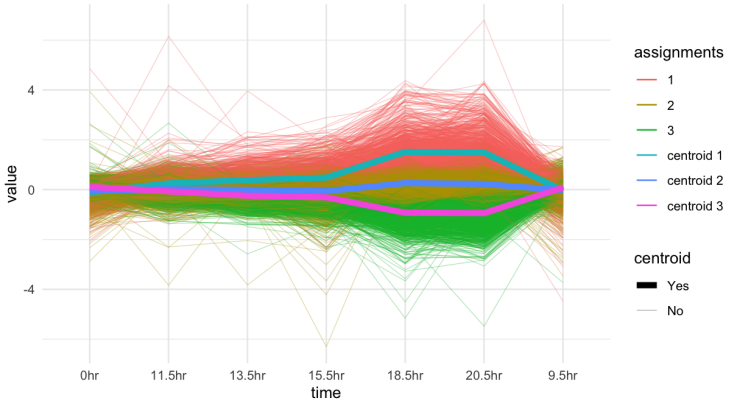
k = 1



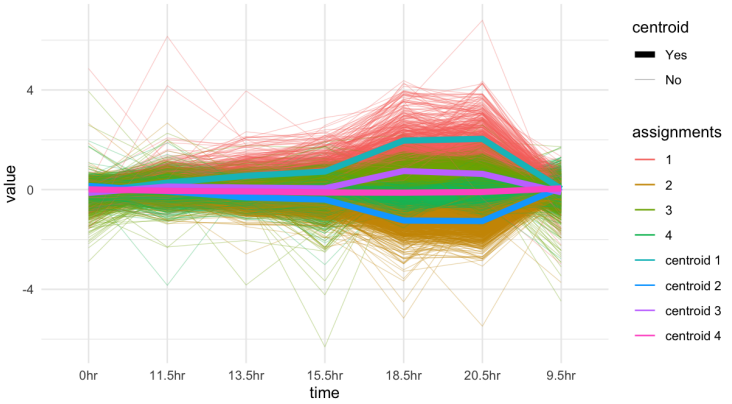

k = 2



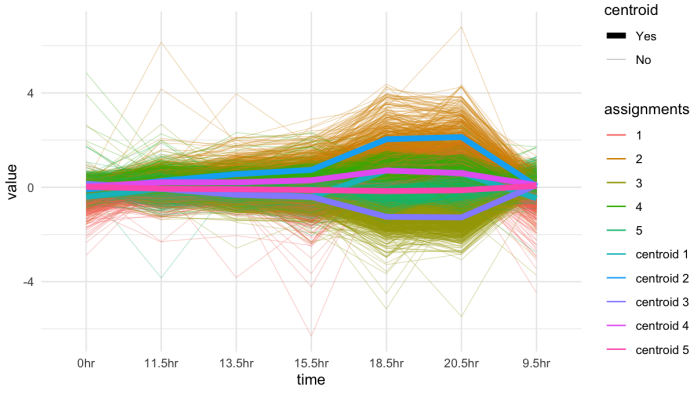

k = 3




k = 4




k = 5




k=6