Clustering Analysis

Lorenz Wolf

October 14, 2021

1 The Data

The train set contains data on 516 individuals. The column, denoted z, is a class indicator corresponding to different chromosome types. The variables, V2, V3,. . . V12 relate to the size and shape of the chromosome and the remainder relate to quantitative descriptions of the banding pattern. The feature vectors for each individual have been normalised, so that they can be compared across different individuals.

Firstly, note that all features have some missing values. Each feature is only missing very few values, thus performing listwise deletion would lead to a greater loss of data than necessary where necessary we will use stochastic regression imputation to address this issue. We note that both categories are roughly equally represented in the dataset. An inspection of the summary statistics uncovers a great variety in the sample standard deviations of the feature columns, taking values between 61 (V3) and 247 (V28), which could lead to problems in the model training. Furthermore, some of the features are strongly correlated. In particular, V2 and V3 have a correlation of 0.9949, but also the pairs 7 and 11 as well as 9 and 12 show a strong correlation.

Plotting the histograms for each feature and splitting by category does reveal some underlying structure in the distributions of some of the features, while for others nothing specific can be observed (see Figure 1). Specifically we note that for variables V2, V3, V5, V9, V12, V16, and V19 larger values tend to be more commonly associated with type 1 and rarely with type 0, while for V17 this pattern is reversed.

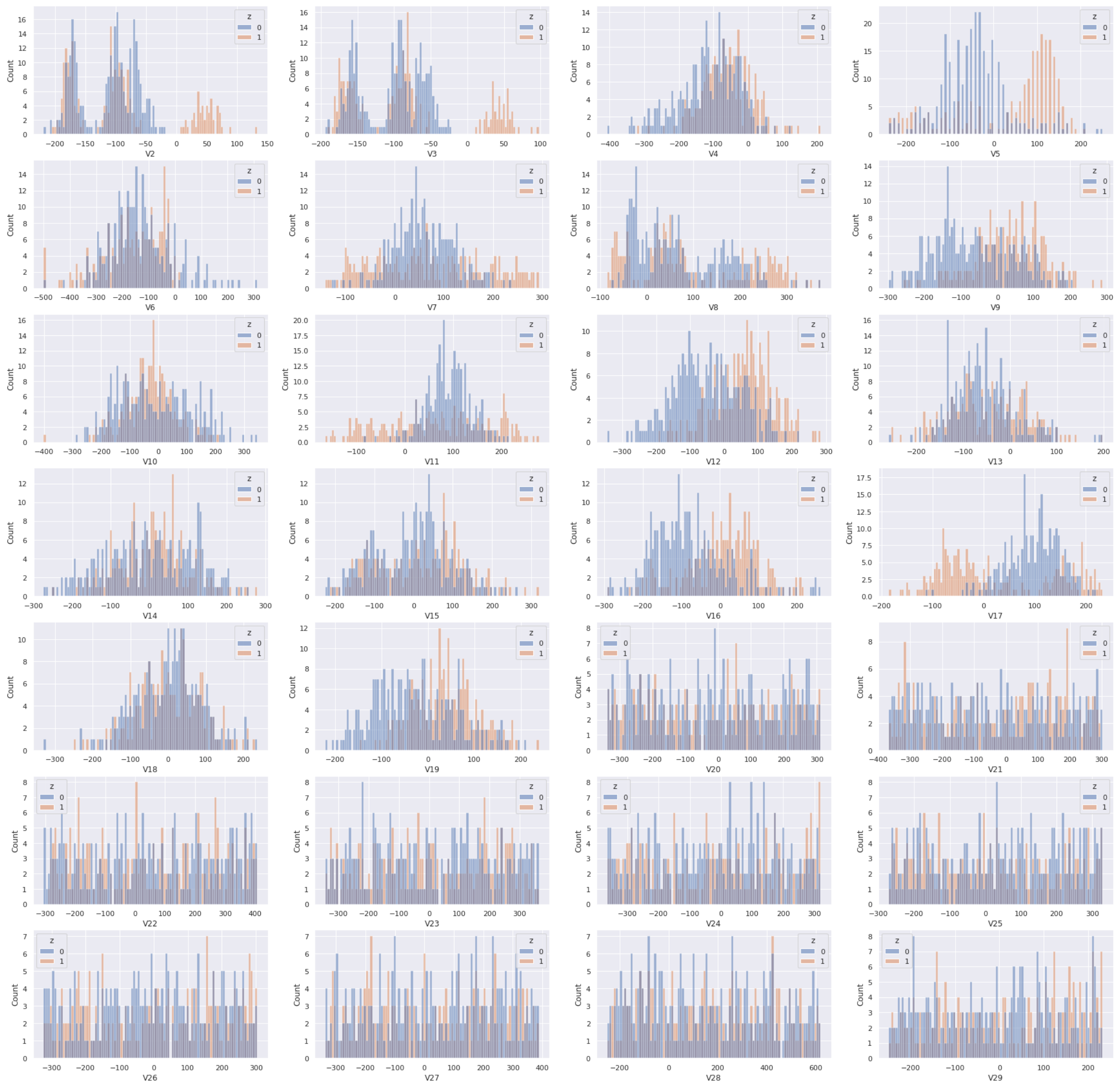

Figure 1: Histograms of the data

2 Clustering

2.1 Hierarchical clustering

Hierarchical clustering is a popular approach to clustering a set of observations. In the following paragraphs we briefly describe the method and its dependencies on distance measures.

There are two different approaches, namely agglomerative and divisive. The first of these assigns each observation its own clusters and then recursively combines the most similar clusters, while the second approach starts with one cluster and recursively divides a cluster.

Regardless of the approach, a distance measure to evaluate the similarity/dissimilarity between two data points x_j and x_i must be chosen. The choice of distance measure directly influences the resulting clusters and should be chosen based on the data at hand and underlying theoretical restrictions/assumptions. It is common to use the Euclidean distance but any other metric distance can be chosen. For example if we consider the a two dimensional space with observations being locations but we can only move along the axes, then instead of the Euclidean distance one should choose the Manhattan distance to incorporate this restriction.

For features, which have different scale factors or are correlated, a more suitable distance is the Mahalanobis distance given by

$$D_\Sigma(p,q) = \sqrt{(p-q)^\top \Sigma^{-1}(p-q)},$$

where Σ is the covariance matrix of the data set. [1]

Another aspect to consider is whether the features are categorical or continuous. For categorical features a common metric is the Hamming distance which counts the disagreements of features between x_j and x_i.

Finally, a linkage criterion must be chosen. The linkage criterion determines on which points of a cluster the distance measure between clusters is evaluated. For example, complete-linkage considers the maximum distance between two points in different cluster, while single-linkage looks at the minimum distance. The third is average-linkage which means the distance between two clusters is the average distance between any point in the first cluster and any point in the second.

Single-linkage has the disadvantage that a chain of individual points can connect two clusters so that they will be merged to one. With average linkage the downside is that it can cause elongated clusters to split and portions of adjacent elongated cluster to merge. Complete-linkage generally leads to more compact clusters than single-linkage.

Ward's method aims to minimise the total within-cluster variance by combining clusters leading to the smallest increase in within cluster variance. This method aims to find compact and spherical clusters in the data.

We now aim to cluster the features. First, remove the label column from the data. In order for hierarchical clustering to give sensible results we normally scale the data before clustering it, so the distance measure is not affected by different variances and means. Furthermore, to cluster the features we want to transpose the data so that each row corresponds to a feature and the columns correspond to individuals. Scaling after transposing would mean that each observation is expected to have mean 0 and variance 1 which is not reasonable and we do not want the features to influence each other. The data for different individuals are already scaled so that they are comparable, hence we can normalise them to prevent certain features from dominating by taking into account different means and variances as observed in the initial EDA.

As different distance measures we consider the Manhattan distance and Euclidean distance. The link functions considered here are

- the maximum or complete-linkage clustering: $\max\{d(a,b), a \in Ab \in B\}$
- the minimum or single-linkage clustering: $\min\{d(a,b), a \in Ab \in B\}$
- the average linkage clustering: $\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$
- Ward linkage: $ESS(XY) - [ESS(X) + ESS(Y)]$ where XY is the combined cluster of X and Y and ESS(.) is the within cluster error sum of squares.

To determine the number of clusters as well as the best distance measure and linkage we use the silhouette score.[2] In particular, the average silhouette width is computed for each combination of linkage, distance and cluster number and the combination with the highest score is chosen. As possible number of clusters we consider $k = 2, \ldots, 27$. We use the silhouette(.) functions in the 'cluster' package in R and huclust(.) from the 'stats' package. It turns out that Ward's linkage with Manhattan distance and 14 clusters gives the highest average width of 0.1651. The average width for this combination of linkage and distance measure is plotted against k in Figure 2. We note that based on the plot a slightly smaller number of clusters could have also been reasonable, since the increase in the average width is very small between 10 and 14 clusters.

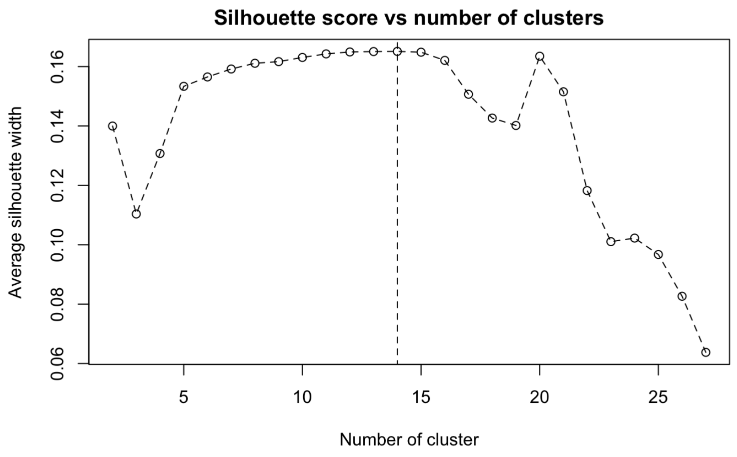

Figure 2: Average silhouette width against number of clusters for Ward's linkage with Manhattan distance.

Figure 3 shows the produced dendogram with the final choice of measure and linkage. The rectangles mark the 14 clusters.

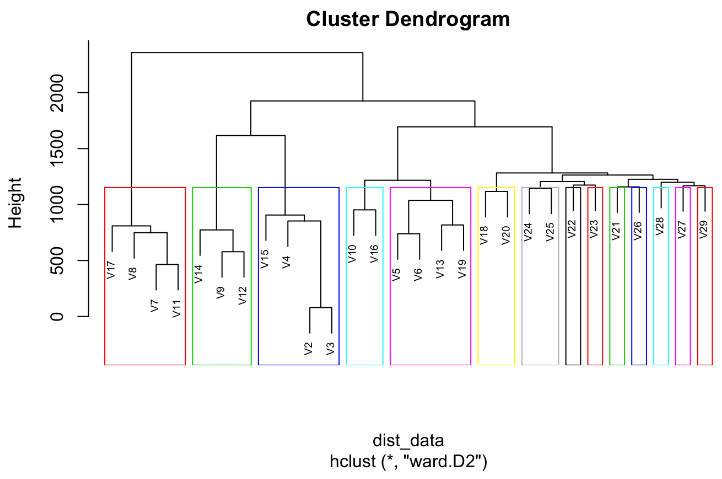

Figure 3: Dendogram for Ward's linkage with Manhattan distance. The 14 clusters are marked by the rectangles.

In Figure 4 the dendograms for all combinations of distance measure and linkage are visualised. First, note that the choice of distance measure seems to have a small effect on the clusterings for any linkage. The key difference between single linkage and the remaining linkage functions is that single linkage splits of individual features first while the others produce more balanced cluster sizes. However, for 14 clusters there are great similarities between all the different methods used. For all methods except Ward's method the features V20-V29 are in individual clusters and some of the variables stick together in the same cluster regardless of method or distance.

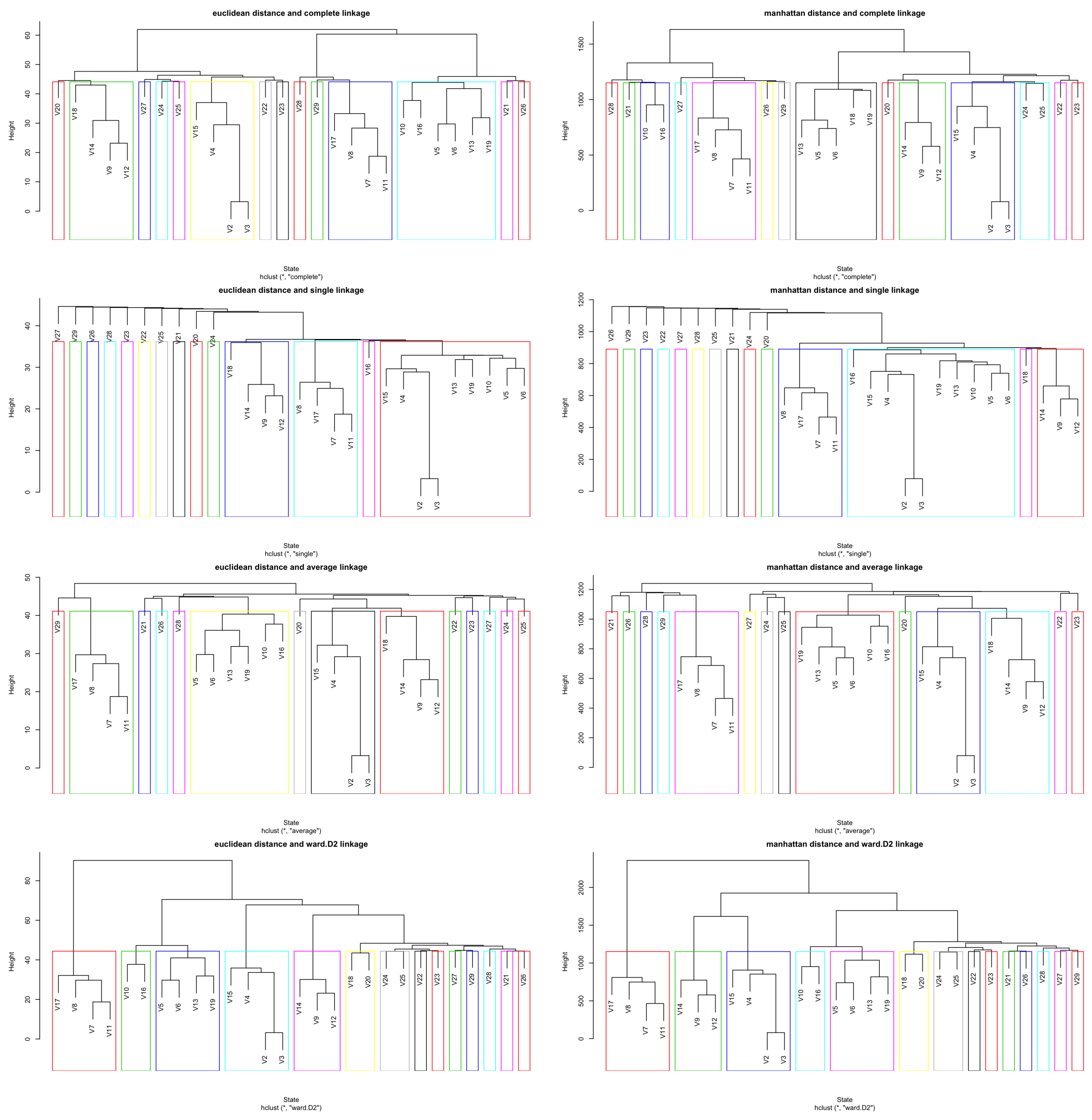

Figure 4: Dendograms for the different combinations of distance measure and linkage.

2.2 K-Means

In this part we will use the kmeans algorithm to cluster the observations into the two clusters given by the 'z'-indicator. First, we note that kmeans is not able to handle missing data. Since listwise deletion would lead to a loss of roughly 10% of data due to only very few missing values in each column we turn to a different method. Mean imputation is common, however since we have a significant amount of information about each observation that is contained in the other features we use stochastic regression imputation. Essentially, a regression model is fit to the complete data and the missing data points are predicted. Some noise is added to the predictions to reflect the original characteristics of the data. This is done using the 'mice' package in R. For best practice several data sets should be produced to assess the effect of the imputation on the final result, but in clustering we do not have a ground truth which is why we simply produce one completed data set. Again the data is normalised before the imputation to ensure equal scales when computing distances.

Within the kmeans algorithm the goodness of clustering is assessed with

$$\mathcal{E}_K = \sum_{k=1}^{K} \sum_{n=1}^{N} z_{kn} \left\| x^{(n)} - \mathbf{m}_k \right\|^2$$

measuring the compactness of the clusters. In the above expression m_k is the within cluster mean and z_{kn} is the binary indicator which is 1 if point n is contained in cluster k.

Kmeans is used to cluster the data set into 2 clusters. Here the kmeans(.) function in R's 'stats' package is used. An issue with kmeans is the initialisation of the algorithm. The produced clusters can heavily rely on the initialisation of the cluster centers. To address this issue we initialise the algorithm 10 times and choose the best clustering.

To assess the performance the Rand index is used. Let C_1 be the clustering obtained with some method, in this case the k-means algorithm, and C_2 the true underlying clustering structure, here given by the 'z'-indicator. Furthermore, let

- a, the number of pairs of observations assigned to the same cluster in C_1 and that are in the same cluster in C_2
- b, the number of pairs of observations assigned to the same cluster in C_1 and that are in different clusters in C_2
- c, the number of pairs of observations assigned to different clusters in C_1 and that are in the same cluster in C_2
- d, the number of pairs of observations assigned different clusters in C_1 and that are in different clusters in C_2

The Rand index is then defined to be

$$RAND = \frac{a+d}{a+b+c+d}.$$

For perfectly agreeing clusterings the Rand index is 1, overall it takes values between 0 and 1. [1]
We use the 'fossil' package in R to compute the Rand index and the adjusted Rand index. On the data set a Rand index of only 0.5024 is obtained. To interpret this result we compare the kmeans clusters to randomly sampled clusters. The distribution of rand scores obtained by comparing the true labels with 400 sets of randomly sampled labels is visualised in Figure 5. It can be observed that in terms of Rand index the kmeans clustering is poor as it compares to randomly assigning labels to each observation. Computing the adjusted rand index, which accounts for agreement by chance and is 0 for a random classifier, yields a value of 0.004. This confirms that kmeans is only insignificantly better than a random clustering.

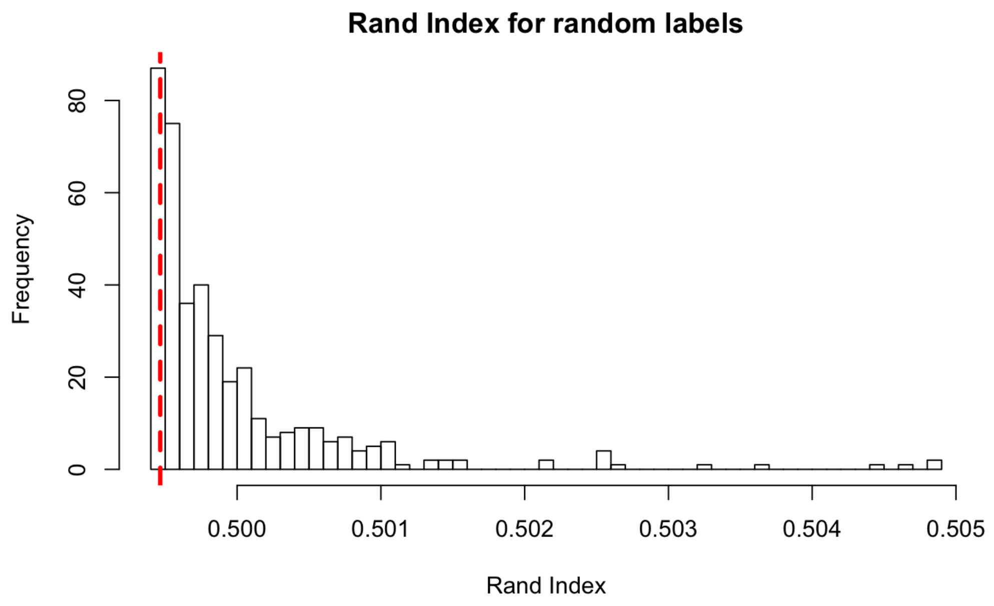

Figure 5: Histograms rand index based on randomly assigned cluster labels, repeated 400 times. Red line indicates Rand index of kmeans clustering.

To visualise the clusters in a 2D graph principal component analysis is performed and the second principal component is plotted against the first where points are colour coded based on the assigned cluster (Figure 7).

We compare this to the true clustering visualised in Figure 6. Besides missing data and the initialisation of the algorithm, an issue could be the structure of the data in the sense that simple kmeans is not able to separate it. Indeed, it can be observe that at least in the 2 dimensional projection kmeans is not able to pick up the structure in the data. This can be caused by the use of the euclidean distance in the feature space. In order to improve we try kernel kmeans (kkmeans(.) in kernlab package) with an RBF kernel. The kernel parameter $\sigma = 0.0222$ is used. The 2D visualisation is shown in Figure 8 and does indeed seem to have improved the clustering. The Rand index and adjusted Rand index of the kernel kmeans are 0.6069 and 0.2136 respectively. This shows indeed an improvement on the vanilla kmeans.

Finally, we mention a slight issue that arises when visualising the clusters. It might not always be clear which cluster corresponds to which true label. However, note that often when clustering is performed true labels are not available. Furthermore, the Rand index does not require knowledge about the specific label cluster assignment hence this issue might only confuse the reader looking at the visualisations.

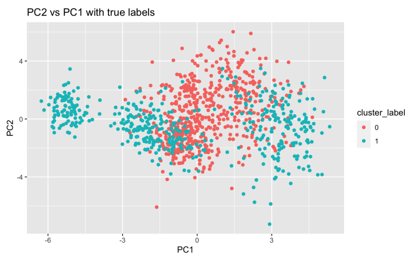

Figure 6: Second principal component against first principal component with true labels.

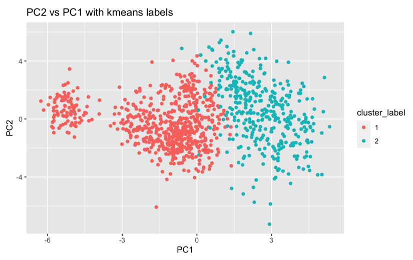

Figure 7: Second principal component against first principal component with kmeans labels.

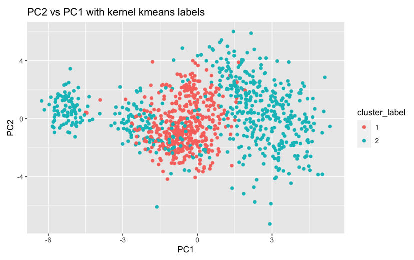

Figure 8: Second principal component against first principal component with kernel kmeans labels.

References

[1] Lior Rokach and Oded Maimon. “Clustering Methods”. In: Jan. 2005, pp. 321–352. DOI: 10.1007/0-387-25465-X_15.

[2] Peter J. Rousseeuw. “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.