P1:

The missing value is imputed using mean of the remaining value. A R package named "stat" is used for compute kmeans clustering. The default option of the distance measure is Euclidean distance.

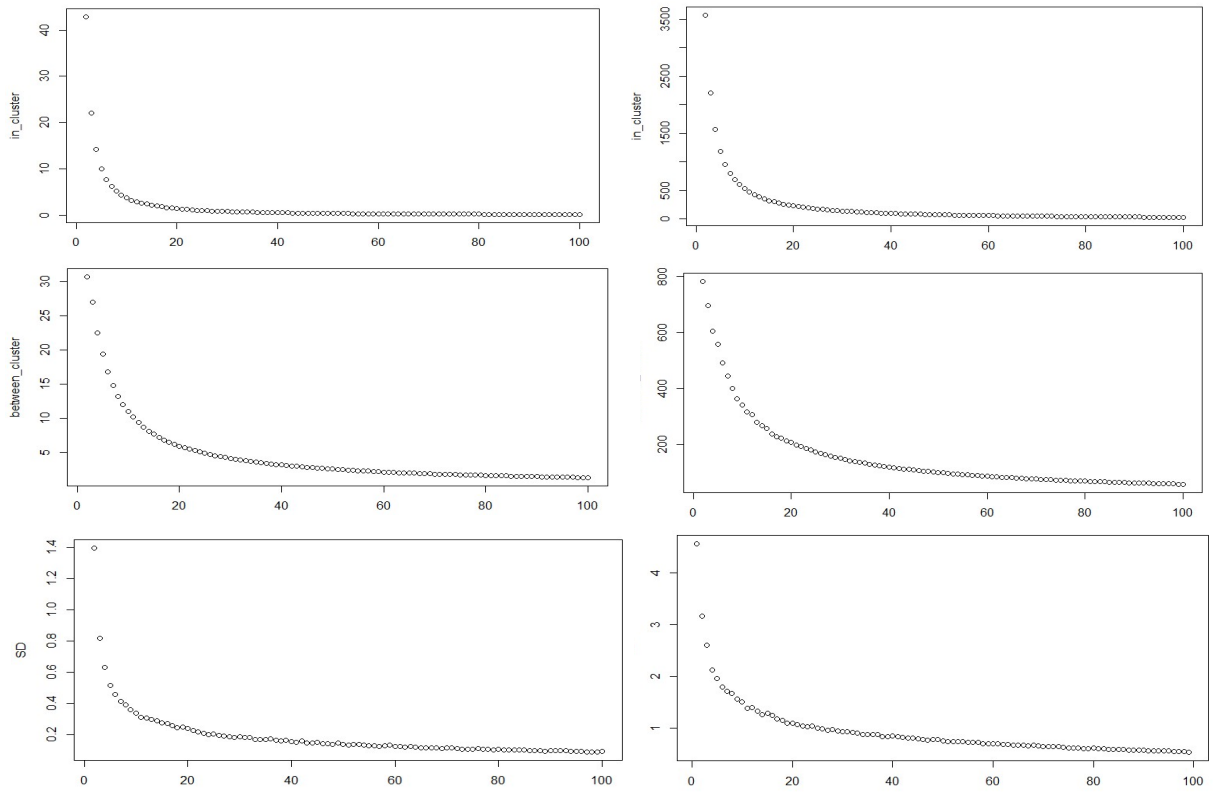
While the first part of problem required dot similarity, for normalized vectors dot product is the same as cosine similarity. Cosine similarity considers variability of data and features' relative frequencies while dot product is cheaper in terms of complexity and implementation. Given that we are interested in the relative variations in the data, we could use cosine similarity instead of dot product in the computation to make sure we are considering the relative abundance of a gene across different sample sets. Also, direct implementation of cosine similarity and implement dot product after normalizing the vector would yield same result so it does not violate the requirement in the question.

Given that the provided R package uses Euclidean distances as default and the first part of the problem require cosine similarity (dot product) as distance measure, the algorithm is slightly modified such that when recomputing the center of mass, rather than simply compute the mean, we will average the normalized vector being clustered then normalizing the result. The result of second part of the problem is directly computed by the default option.

The figures below summarize the result for both similarity measure. The 3 diagrams on the left is the result using cosine similarity (dot product), and on the right is result using Euclidean distances. From top to bottom is the plot for in- cluster similarity, between-cluster similarity and S/D ratio.

With Cosine Similarity

With Euclidean Distance

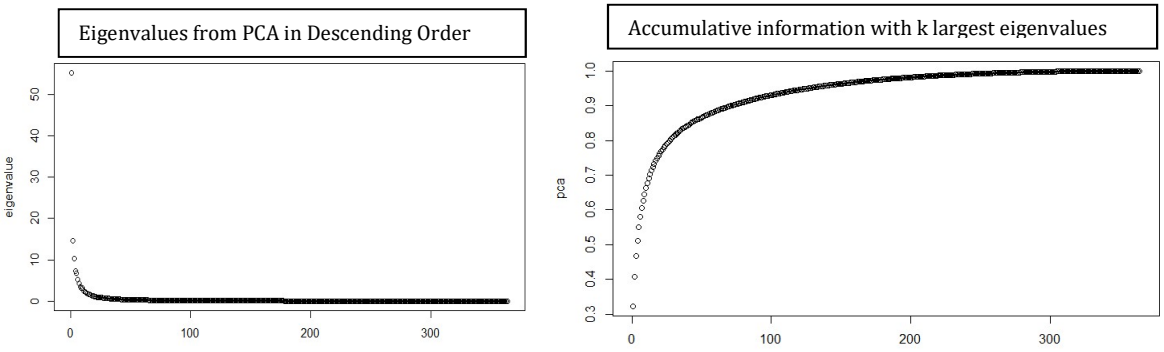


A popular method to choose optimum k is the elbow method. The method is based on the fact that one should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point, the marginal gain will drop, giving an angle in the graph, and we could choose this point as out optimal number of clusters since we do not want infinitely many clusters (or # of cluster equals to # points), although that would give 0 in-cluster distance, we loss the point of clustering. Although method is not always deterministic for smooth curves, according the plot the optimum number of cluster would be 5 for using cosine similarity and 7 using Euclidean distance.

The result is similar for both similarity measure and since there's no group provided in original data it is not possible to determined which method gives better result. One noticeable difference between them are the run time. It looks like Cosine Similarity is more complex and requires longer run time. For more complex data set, we may take that into account and choose Euclidean distance, or preprocess the data to reduce the computation complexity.

P2:

Data matrix is imputed using means and normalized to (1, -1) range by subtract minimum then divided by the difference of maximum and minimum for each column. The data matrix has 364 rows, which consists of all patients from AD and control data set, and 800 columns, which consists of the first 800 genes. PCA is performed using the prcomp() function in R "stat" package, the eigen value is reported directly using the stored information in model generated. The following is the plot of sorted eigen values and the plot of accumulative information.

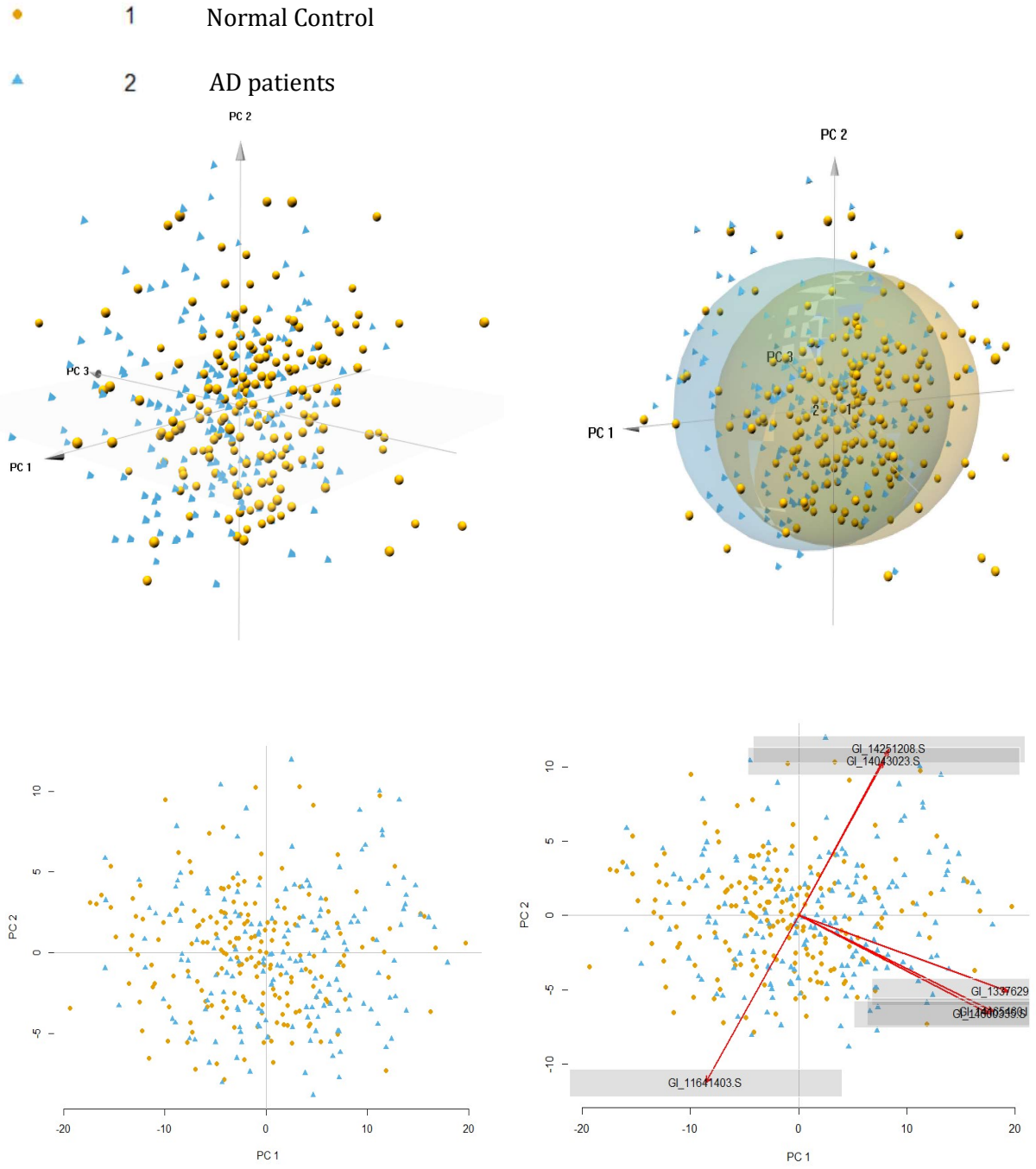


In our data set, we are looking at the case that the number of dimension, which is 364, is smaller than the number of the features, which is 800, therefore the maximum of number of principle component can be generated is the number of dimension since PCA requires a full rank matrix since the covariance matrix are being projected orthogonally down into a lower dimension space. Since we only have 364 observations here, the sample covariance necessarily lies on a subspace of dimension smaller or equal to 364. From the plot we could see there are 364 PCs in total. Since the eigenvalue reflects the variation explained in each principle component generated, we can see from the plot that the first several principle components are a lot more significant compared to the later principle component. First 10 principle components already explained 70% of the variations. If we want to reduce the complexity of computation, we could use the PCs according the desired number of PCs. Using similar criteria to find optimal k for the k means clustering problem, we could look at the accumulative information plot and choose the point with greatest change in slope. The result would be using 42 PCs, which gives 84% accumulative information.

It is hard to judge which dimension is effective, but from sorted eigen value there's a significant discrepancy at 327th PC where the eigen value drops from 10^{-3} to 10^{-18}. While other transitions are relative smooth compared to this one, we might say there are 327 effective dimensions. Another way to determine effective dimension is by looking at accumulative information. We can see that 245 PCs would provide more than 97% of accumulative information and it is unlikely we still need information from the rest of PCs, we could therefore state there are 245 effective PCs.

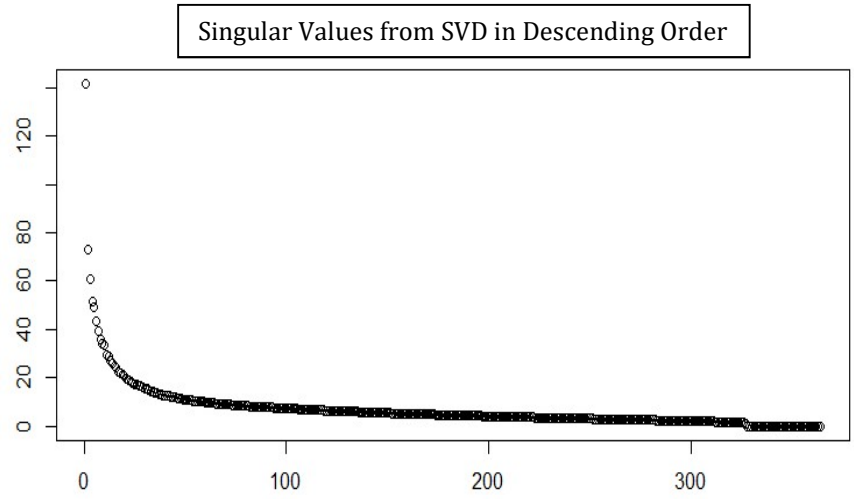
If want to retain 80% of accumulative information, from the graph we obtain that specifically, choosing 27th PC would give 79.98% and 28th PC would give 80.41% of accumulative information, both choices could be reasonable here.

The following is the plot of first 3 PCs and first 2 PCs. The major separation is along the PC 1 axis, yet the separation along other two axis are not so clear. It is reasonable given that the first 3 component contained less than 50% of accumulative information.



P3:

The data preprocessed in P2 is used for this analysis. The svd() function in R package "stat" is used to compute the model. Singular values are stored in the model and are reported directly after sorting.



Similar to that in problem 2, since the number of features is greater than the number of samples, there are 364 dimensions resulted from analysis, and there is also a significant drop at 327th singular values, at which the eigen value drops from 10^0 to 10^{-8}. Therefore, we could say that there are again 327 effective dimensions, which is same as using PCA. If computing the accumulative information as explained in previous problem for PCA, for each dimension, a reasonable cut off for effective dimension could be 200, which has already explained more than 95% of information, which suggest we could safely ignore the rest PCs.

Another thing to observe is that compared to PCA, although the first several eigen value is large, the eigen value for following around 20 dimensions decreases slower than that in PCA. That means that the first few (less than 10) dimensions are less effective explaining variations in data compared to that in PCA since variations are more evenly distributed in larger number of dimensions.

Singular values are related to the eigenvalues of covariance matrix via $\lambda_i = s^2_i/(n-1)$, where λ_i represents the eigenvalues for i^{th} principle component that shows variances of the respective PCs from PCA, and the left singular vectors correspond to PCs. Explanation is given below.

Given a data matrix n*p, PCA first compute the covariance matrix S such that $\mathbf{C} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$, then decompose the covariance matrix to $\mathbf{C} = \mathbf{E}\mathbf{L}\mathbf{E}^T$, where $\mathbf{L} = \mathbf{diag}(\lambda_1, \lambda_2, \lambda_3 \ldots\ldots \lambda_p)$ and E is a matrix of corresponding eigen vectors. Performing a SVD, the decomposition is given by $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where S is the diagonal matrix of singular values. Then we could compute that:

$$\mathbf{C} = \frac{\mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}\mathbf{U}^T}{n-1} = \mathbf{U}\frac{\mathbf{S}^2}{n-1}\mathbf{U}^T$$

Therefore $\mathbf{L} = \frac{\mathbf{S}^2}{n-1}$, which suggests that $\lambda_i = s^2_i/(n-1)$, and the left singular vector forms the eigen vectors of S.