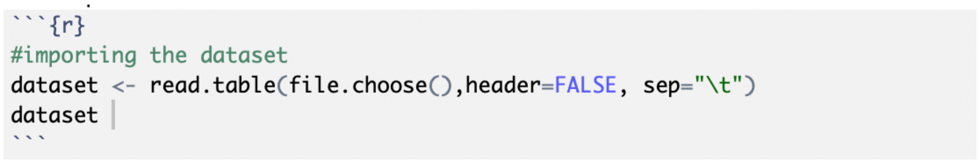
CSE 601: Data Mining and Bioinformatics

Fall 2018

Submitted by: Tanushri Nayak, Chandani Jaiswal, Sai Kalyan Katta

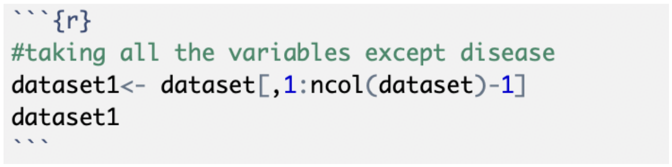
Workflow:
The first step in the dimensionality reduction is to import the dataset into our workspace. The code-snippet shown in the Fig 1 shows how to import the dataset from the text file.



~~~
```{r}
#importing the dataset
dataset <- read.table(file.choose(),header=FALSE, sep="\t")
dataset
```
~~~

Fig 1. Code for importing the dataset

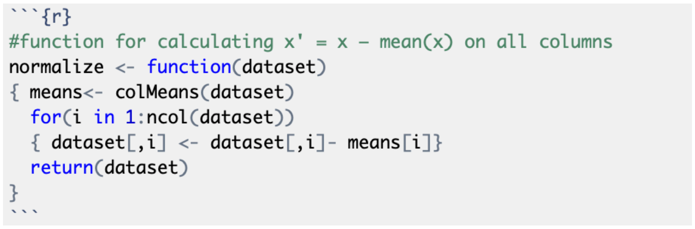
Now for our computational purposes we remove the last column from the dataset i.e. the diseases column from the dataset which is of type ***factor***. This can be seen in Fig 2.



~~~
```{r}
#taking all the variables except disease
dataset1<- dataset[,1:ncol(dataset)-1]
dataset1
```
~~~

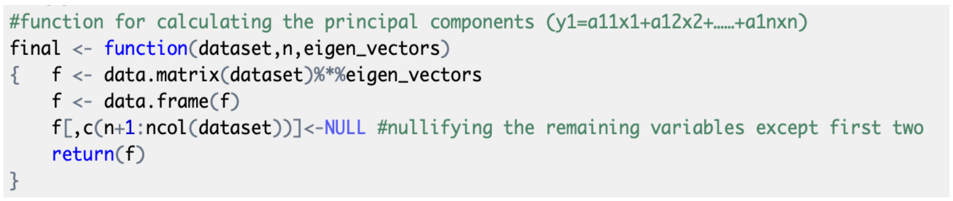
Fig 2. Removing the last column

We have defined two functions "***normalize***" and "***final***" to compute $x^I = x - mean(x)$ on all the columns and the principal components as shown in the Fig 3 and Fig 4 respectively.



~~~
```{r}
#function for calculating x' = x - mean(x) on all columns
normalize <- function(dataset)
{ means<- colMeans(dataset)
  for(i in 1:ncol(dataset))
  { dataset[,i] <- dataset[,i]- means[i]}
  return(dataset)
}
```
~~~

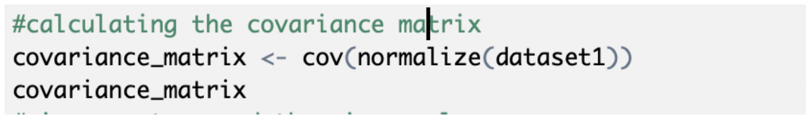
Fig 3. Function to compute $x^I = x - mean(x)$



```
#function for calculating the principal components (y1=a11x1+a12x2+……+a1nxn)
final <- function(dataset,n,eigen_vectors)
{   f <- data.matrix(dataset)%*%eigen_vectors
    f <- data.frame(f)
    f[,c(n+1:ncol(dataset))]<-NULL #nullifying the remaining variables except first two
    return(f)
}
```

Fig 4. Function to calculate the principal components

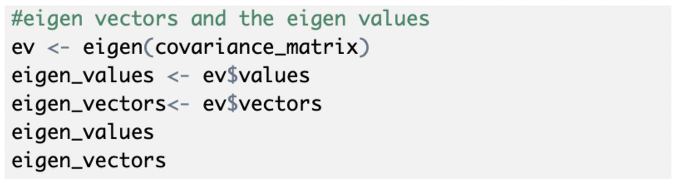
Now, using the in-built function "***cov***" we calculate the covariance_matrix using the normalized matrix (normalized matrix is calculated by calling the "***normalize***" function that we defined before) as shown in the Fig 5.



```
#calculating the covariance matrix
covariance_matrix <- cov(normalize(dataset1))
covariance_matrix
```

Fig 5. Calculating the covariance_matrix

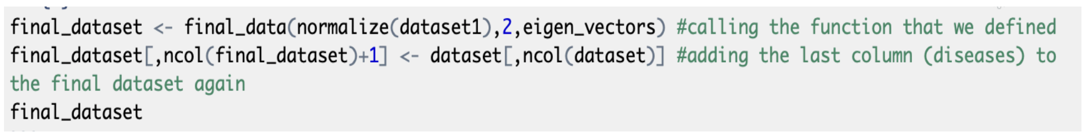
Using the in-built function "***eigen***" we calculated the eigen_values and the eigen_vectors from the covariance_matrix as shown in the Fig 6.



```
#eigen vectors and the eigen values
ev <- eigen(covariance_matrix)
eigen_values <- ev$values
eigen_vectors<- ev$vectors
eigen_values
eigen_vectors
```

Fig 6. Calculating the eigen_values and the eigen_vectors

The final dataset after the dimensionality reduction using PCA is calculated by calling the function "***final***" by passing the normalized matrix and the eigen_vectors. (Here, 2 denotes the number of final variables that the original dataset is to be reduced to)



```
final_dataset <- final_data(normalize(dataset1),2,eigen_vectors) #calling the function that we defined
final_dataset[,ncol(final_dataset)+1] <- dataset[,ncol(dataset)] #adding the last column (diseases) to
the final dataset again
final_dataset
```

Fig 7. Calculating the final dataset after dimensionality reduction

Scatter plots:
The Figures below, show the scatter plots of the final datasets that are obtained after dimensionally reducing the datasets provided with all the three algorithms (implemented PCA, SVD, T-SNE).

For the dataset ***pca_a.txt***:

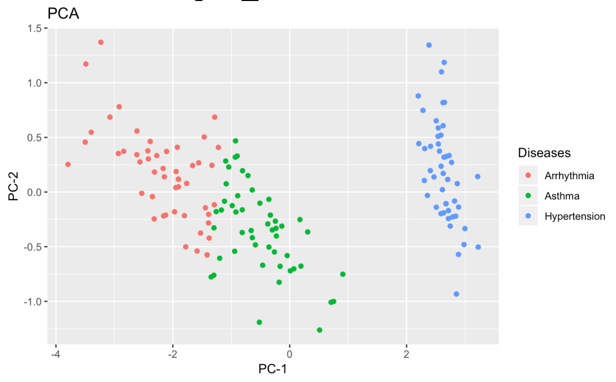


Fig 8. Scatter plot of the data points in *pca_a.txt* after PCA

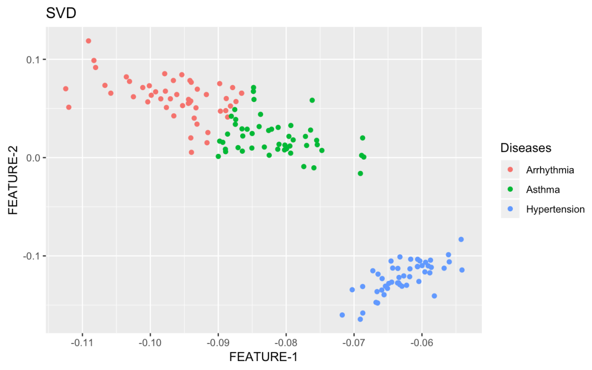


Fig 9. Scatter plot of the data points in *pca_a.txt* after SVD

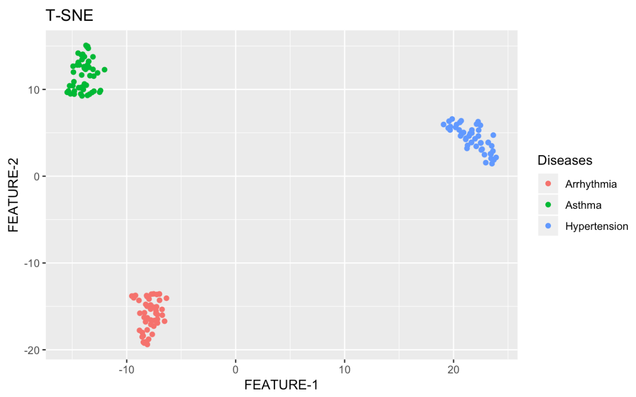
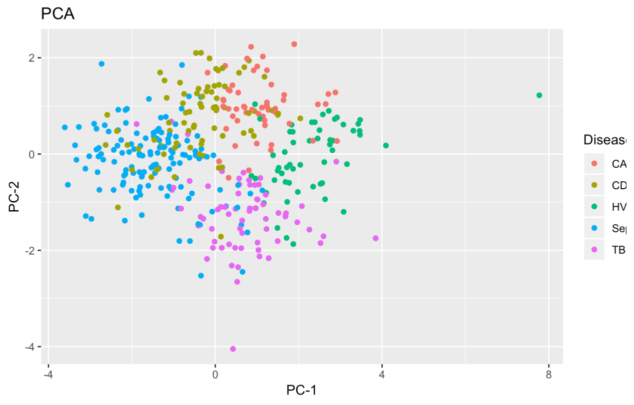


Fig 10. Scatter plot of the data points in *pca_a.txt* after T-SNE

For the dataset ***pca_b.txt***:



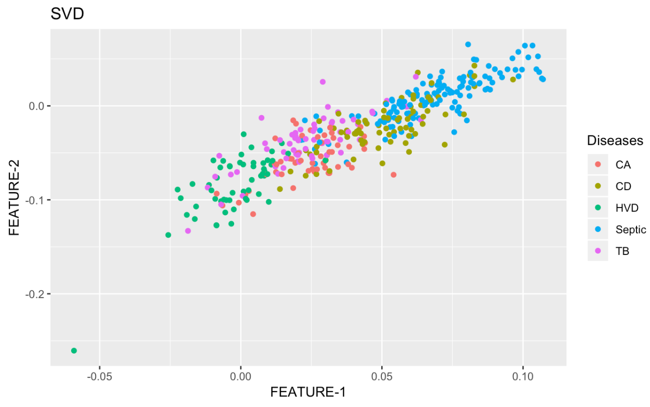


Fig 11. Scatter plot of the data points in *pca_b.txt* after PCA

Fig 12. Scatter plot of the data points in *pca_b.txt* after SVD

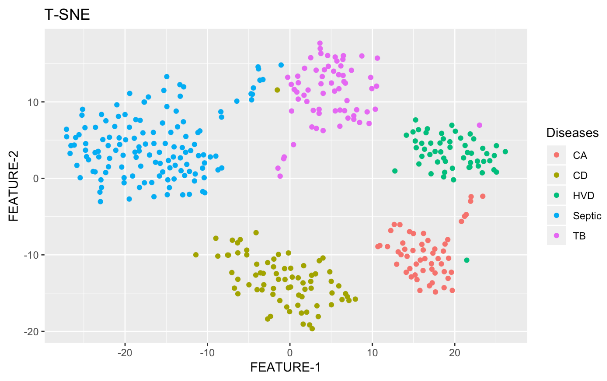


Fig 13. Scatter plot of the data points in *pca_b.txt* after T-SNE

For the dataset ***pca_c.txt***:

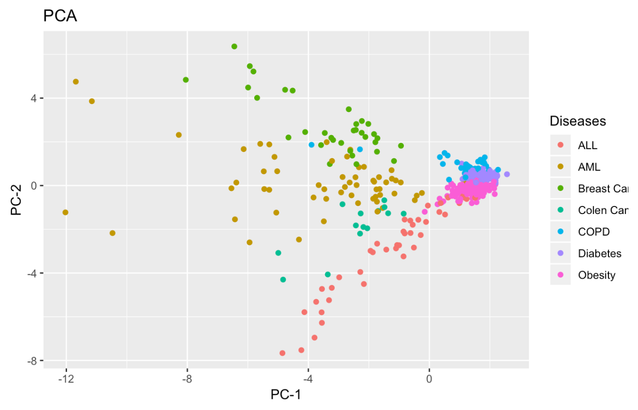


Fig 14. Scatter plot of the data points in *pca_c.txt* after PCA

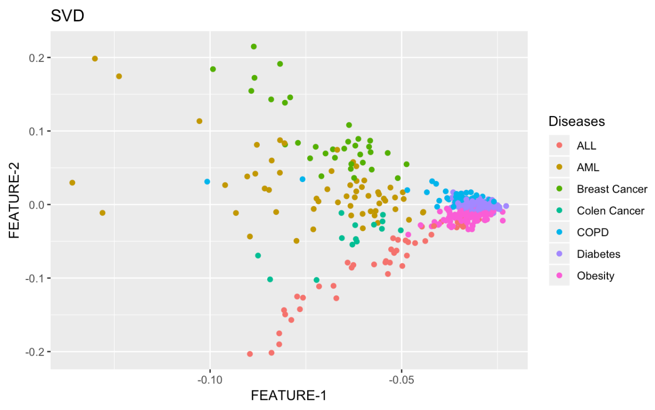


Fig 15. Scatter plot of the data points in *pca_c.txt* after SVD

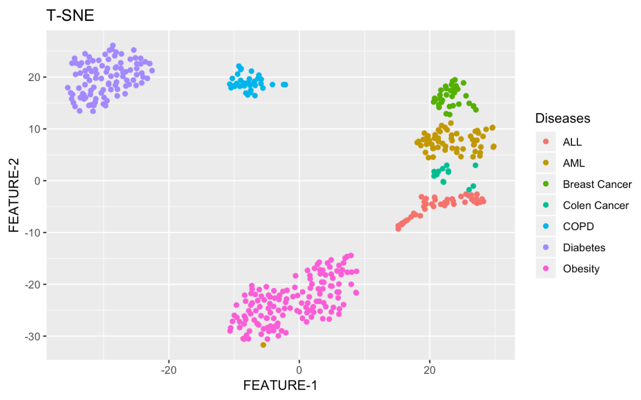


Fig 16. Scatter plot of the data points in *pca_c.txt* after T-SNE

Observations:

From the results and the scatter plots that we have obtained during our implementation, we have observed that the algorithms PCA and SVD are two eigenvalue methods and are closely related to eachother while giving the similar kind of results on the datasets. It can be furthur extended that the SVD algorithm when imposed upon the normalised or adjusted data (subtracting means of corresponding columns) the results are the same as that of the PCA algorithm.

From the observations, it can also be seen that the T-SNE algorithms has performed well in reducing the dimensionality of the dataset and seggregating the data into well separated clusters. While the T-SNE algorithm tries to understand the data, while the PCA algorithm doesn't. It checks for the data points that are similar to the neighbours to form the clusters.