HW ____Term Paper_____
Winter 2018, DSPA (HS650)
Name: __Xinchun Li_____
SID: __85922126_____
UMich E-mail: __xincli___@umich.edu
I certify that the following paper represents my own independent work and conforms with the guidelines of academic honesty described in the UMich student handbook.

ABSTRACT
In this term paper, I will be examining a SOCR dataset related to the Modeling and Analysis of Clinical, Genetic and Imaging Data of Alzheimer's Disease. My goal is to interrogate the data using exploratory and quantitative data analysis methods and look for patterns, trends or relations that may provide cues to the diagnosis, cognitive state or prognosis of these subjects (patients and control). Specifically, I will investigate the following:

1. Identify clusters of data that agrees with specific phenotypes (SymptomSeverety)
2. Reduce the dimensionality of the data and identify linear sub-spaces that contain large between subject variations or expose common traits.

INTRODUCTION
There are many health-related issues that are awaiting for us to explore and study nowadays. There are a lot of challenges around these issues due to the vast number of variables involved and the convolutedness of their relations with each other. In this paper, I will try to examine in details how those variables that we assume would affect the probability of a person developing Alzheimer's Disease actually play out in this process by the application of statistical models, methods and algorithmic thinking.

PROBLEMS
The problems I'm trying to solve include:
Is there a relation between imaging, genetic and clinical covariates for the Alzheimer's Disease?
How can we predict subject diagnosis using model-based approaches?
What method is the best or the most plausible?
Among possible variables, what variables have more relevance to the Alzheimer Disease and what have less?

METHODOLOGIES

Before fitting a model, let's examine the independence of our potential predictors and the dependent variable. Multiple linear regressions assume that predictors are all independent with each other. Let's randomly choose a few variables to examine their relationship with each other.

```
cor(ad_data[c("adascog", "DXCURREN", "Weight_Kg","GLUCOSE", "CDHOME")])
```

```
##                adascog    DXCURREN   Weight_Kg      GLUCOSE       CDHOME
## adascog     1.000000000  0.68830833 -0.06816078 -0.001357867  0.589678873
## DXCURREN    0.688308334  1.00000000 -0.08051666  0.031479077  0.668148651
## Weight_Kg -0.068160782 -0.08051666  1.00000000  0.085728146 -0.055567938
## GLUCOSE   -0.001357867  0.03147908  0.08572815  1.000000000  0.007169041
## CDHOME     0.589678873  0.66814865 -0.05556794  0.007169041  1.000000000
```

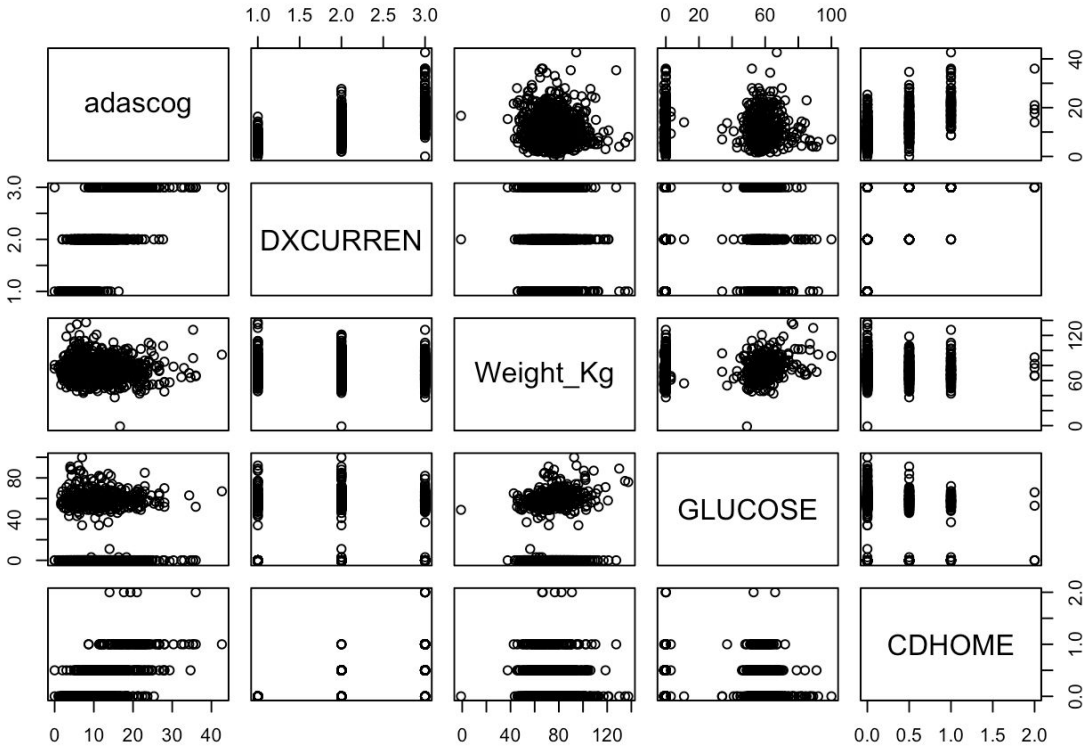
Here we can see that the the variables we randomly choose have more or less correlation with each other. From the above output we can see that there's a huge positive correlation between DXCURREN and adascog, DXCURREN and CDHOME, CDHOME and adascog, CDHOME and DXCURREN, etc. And there's a small negative correlation between adascog and Weight_Kg, adascog and GLUCOSE, DXCURREN and Weight_Kg, etc.

```
car::vif(lm(DXCURREN ~ adascog + Weight_Kg + GLUCOSE + CDHOME, data=ad_data))
```

```
##    adascog Weight_Kg    GLUCOSE     CDHOME
##   1.536061  1.012530  1.007559  1.533850
```

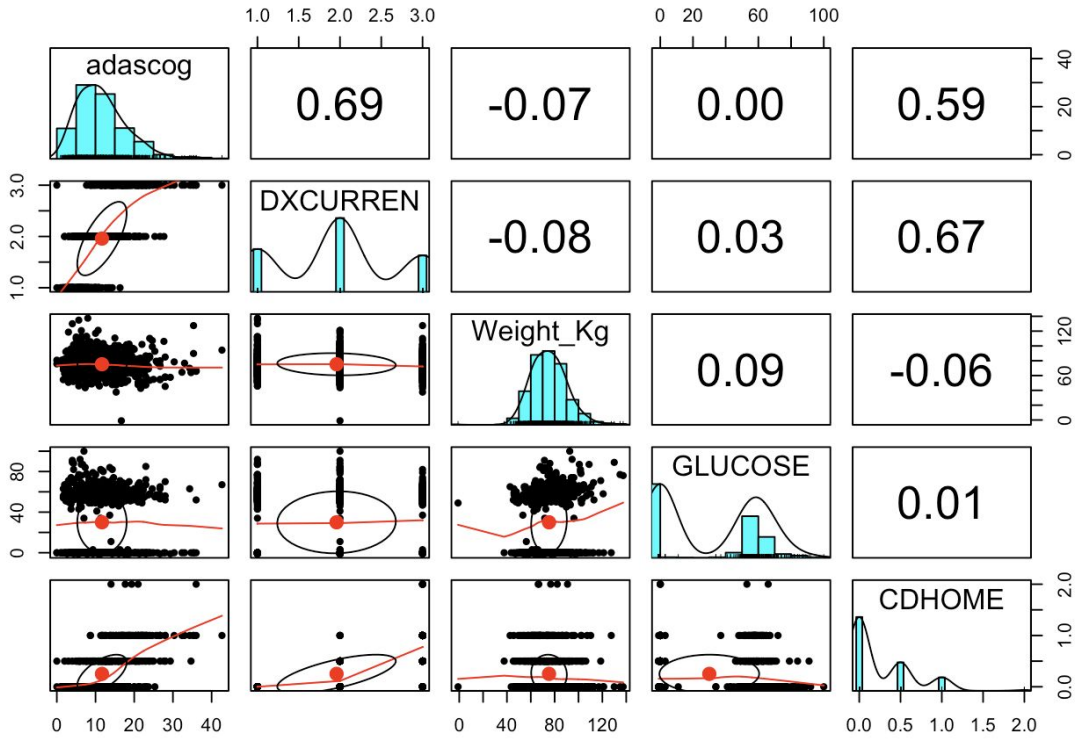
VIFk~1 implies that there is no multicollinearity involving the kth predictor and the remaining features and hence the variance estimate of βk is not inflated.

```
pairs(ad_data[c("adascog", "DXCURREN", "Weight_Kg","GLUCOSE", "CDHOME")])
```



From this pair chart, we still cannot get a very straightforward idea of how those relationship plays out.

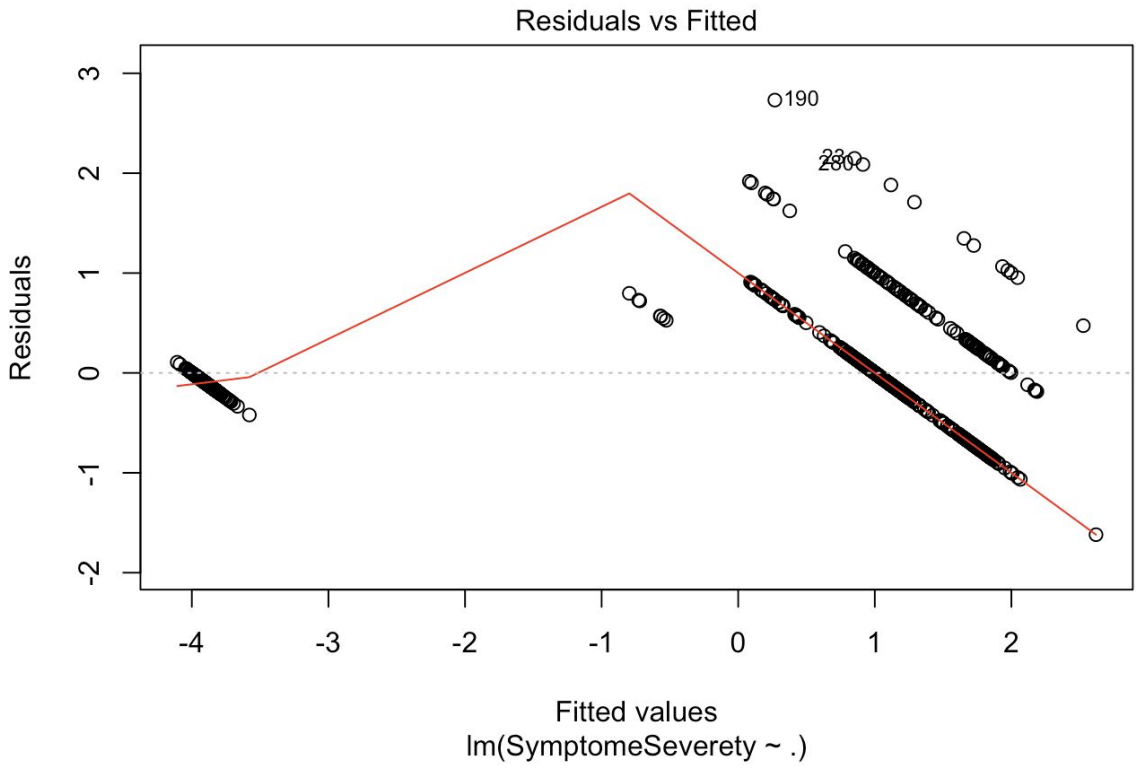
```
library(psych)
pairs.panels(ad_data[, c("adascog", "DXCURREN", "Weight_Kg","GLUCOSE", "CDHOME")])
```

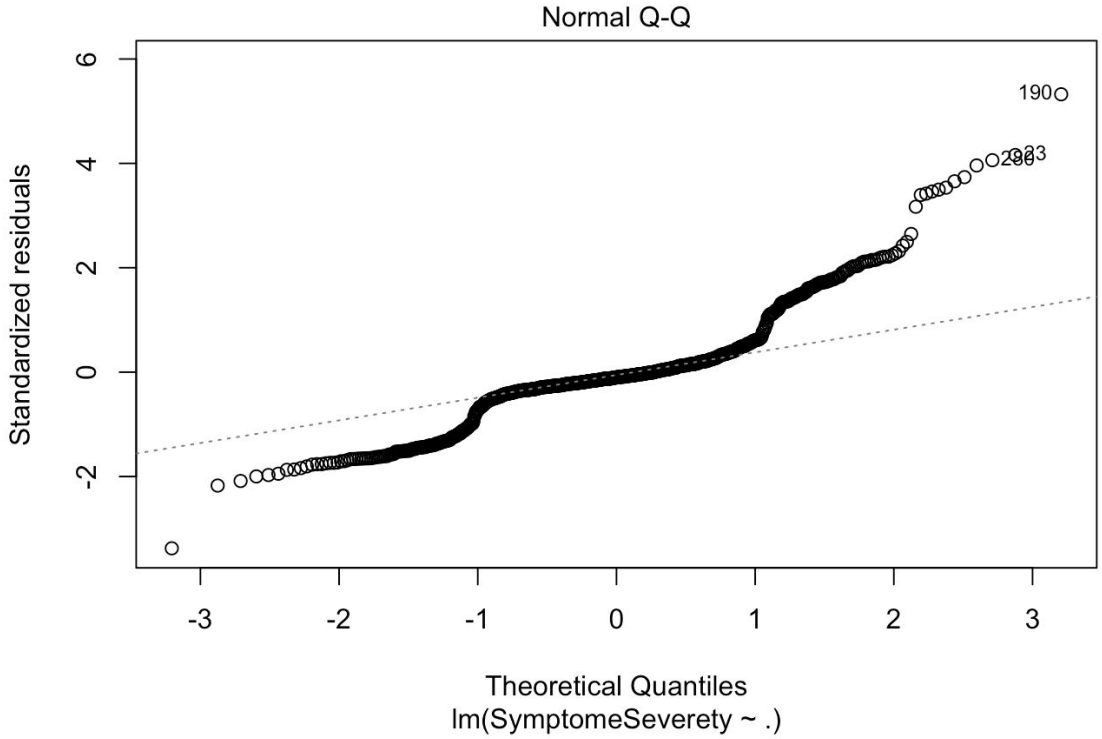


This plot give us much more information about the five variables. Above the diagonal, we have our correlation coefficients in numerical form. On the diagonal, there are histograms of variables. Below the diagonal, more visual information are presented to help us understand the trend. This specific graph shows us adascogt and CDHOME are positively and strongly correlated. Also the relationships between GLUCOSE and adascog, Weight_Kg and DXCURREN are very weak (horizontal red line in the below diagonal graphs indicates weak relationships).

```
fit <- lm(SymptomeSeverety ~., data = ad_data)
fit
```

```r
plot(fit, which = 1:2)
```





The summary shows us how well does the model fits the dataset.

Residuals: This tells us about the residuals. If we have extremely large or extremely small residuals for some observations compared to the rest of residuals, either they are outliers due to reporting error or the model fits data poorly. We have 73.649 as our maximum and −48.692 as our minimum. Their extremeness could be examined by residual diagnostic plot.

Coefficients: In this section, we look at the very right column that has stars. Stars or dots next to variables show if the variable is significant and should be included in the model. However, if nothing is next to a variable then it means this estimated covariance could be 0 in the linear model. Another thing we can look at is the Pr(>|t|) column. A number closed to 0 in this column indicates the row variable is significant, otherwise it could be deleted from the model.

Here only some of the teams and positions are not significant. Age and Height are significant.

R-squared: What percent in y is explained by included predictors. Here we have 38.58%, which indicates the model is not bad but could be improved. Usually a well-fitted linear regression would have over 70%.

The diagnostic plots also helps us understanding the situation.

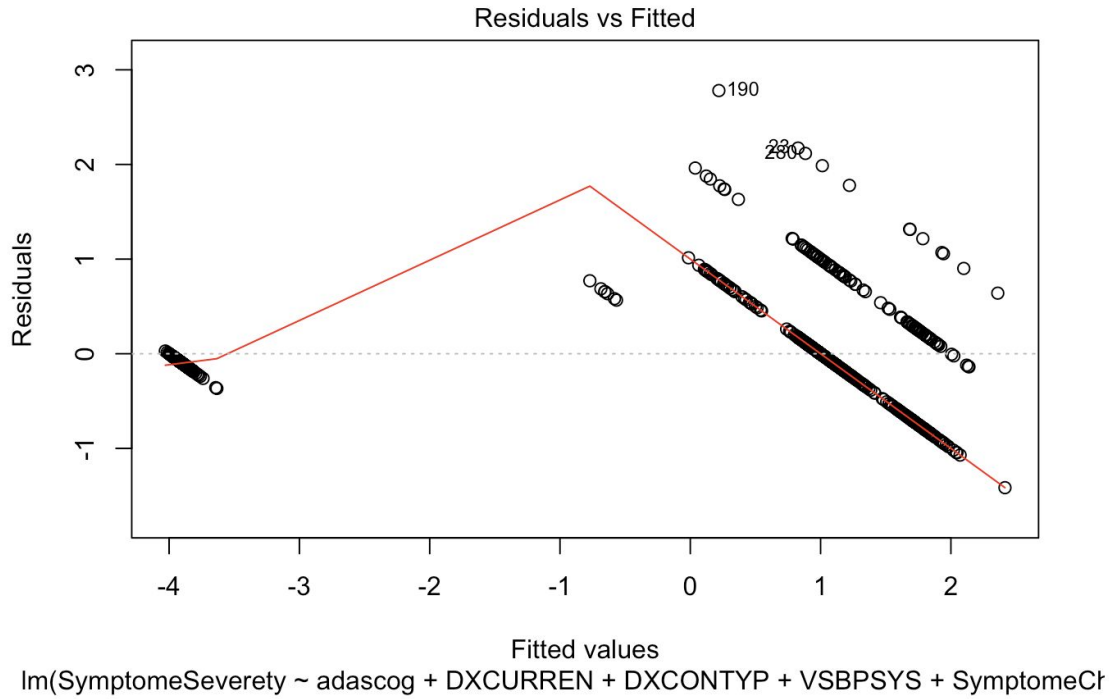
Residual vs Fitted: This is the residual diagnostic plot. We can see that the residuals of observations indexed 65, 160 and 237 are relatively far apart from the rest. They are potential influential points or outliers.

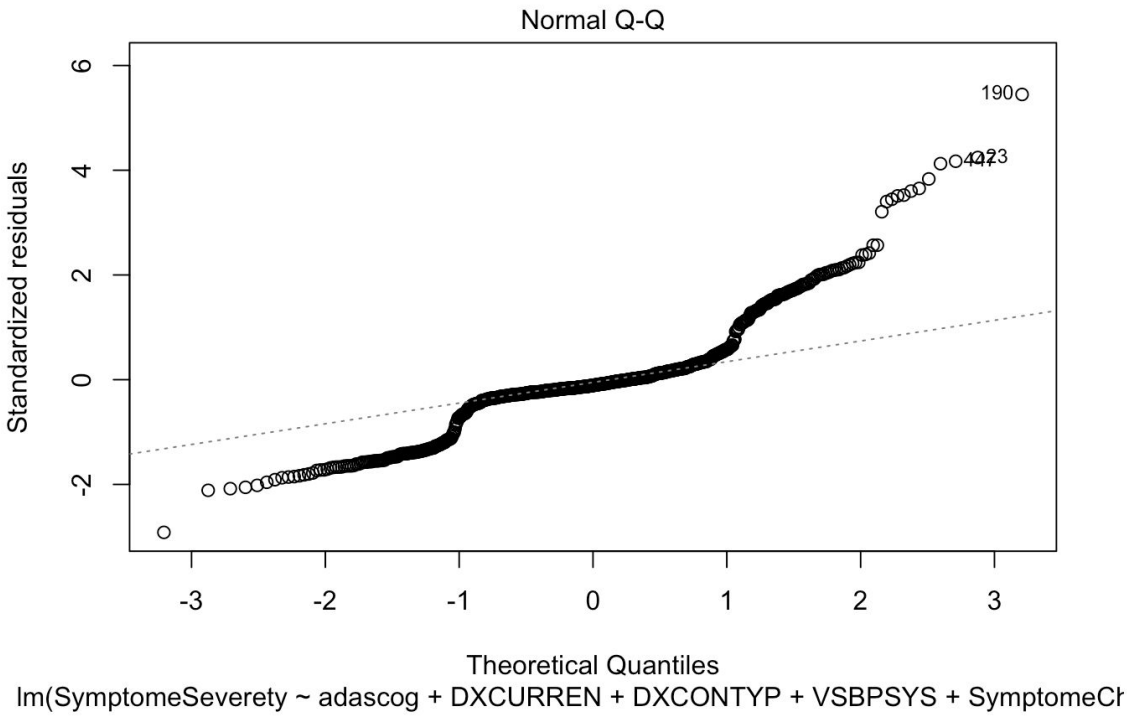
Normal Q-Q: This plot examines the normality assumption of the model. If these dots follows the line on the graph, the normality assumption is valid. In our case, it is relatively close to the line. So, we can say that our model is valid in terms of normality.

Now I want to improve the model performance. I employ the step function to perform forward or backward selection of important features/predictors. It works for both lm and glm models. In most cases, backward-selection is preferable because it tends to retain much larger models. On the other hand, there are various criteria to evaluate a model. Commonly used criteria include AIC, BIC, Adjusted R2, etc. Let's compare the backward and forward model selection approaches. The step function argument direction allows this control (default is both, which will select the better result from either backward or forward selection).

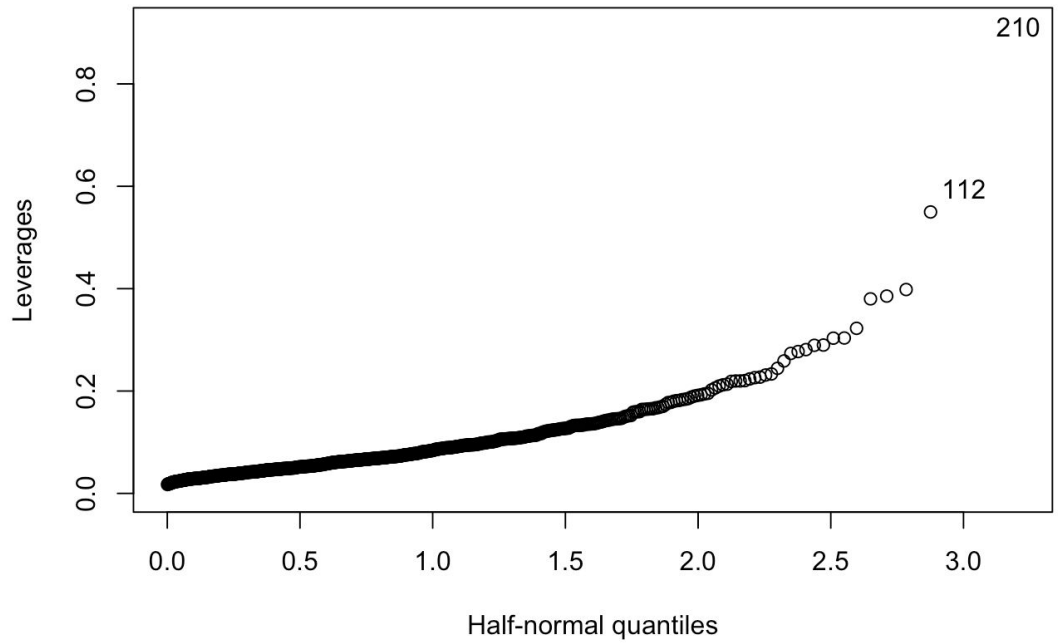
```
fit2 = step(fit,k=2,direction = "backward")
```

```
plot(fit2, which = 1:2)
```






```
halfnorm(lm.influence(fit)$hat, nlab = 2, ylab="Leverages")
```



Sometimes, simpler models are preferable even if there is a little bit loss of performance. In this case, we have a simpler model and R2=0.365. The whole model is still very significant. We can see that observations 112, 210 are relatively far from other residuals. They are potential influential points or outliers. Also, we can observe the leverage points. Leverage points are those either outliers or influential points or both.

```
# Train Data
set.seed(1234)
train_index <- sample(seq_len(nrow(ad_data)), size = 0.75*nrow(ad_data))
ad_train<-ad_data[train_index, ]
ad_test<-ad_data[-train_index, ]
#install.packages("rpart")
library(rpart)
```

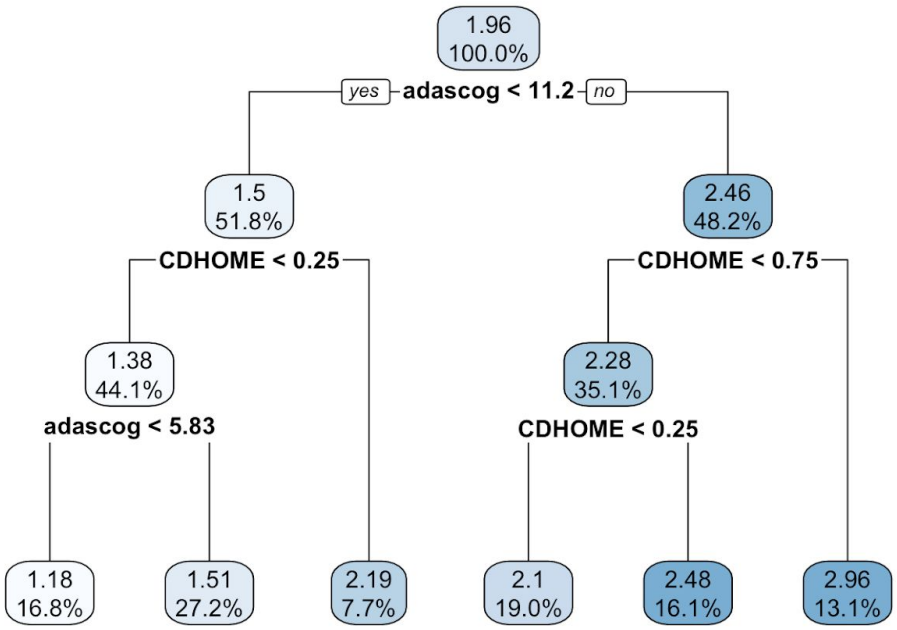
```
##
## Attaching package: 'rpart'
```

```
## The following object is masked from 'package:faraway':
##
##     solder
```

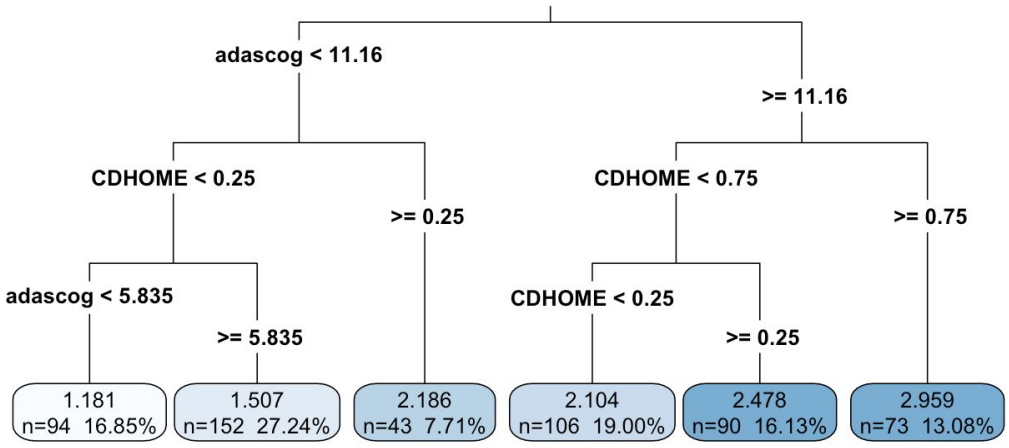
```
ad.rpart<-rpart(DXCURREN ~ adascog + Weight_Kg + GLUCOSE + CDHOME, data=ad_train)
ad.rpart
```

```
## n= 558
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 558 295.283200 1.964158
##    2) adascog< 11.165 289  92.249130 1.501730
##      4) CDHOME< 0.25 246  62.081300 1.382114
##        8) adascog< 5.835 94  13.925530 1.180851 *
##        9) adascog>=5.835 152  41.993420 1.506579 *
##      5) CDHOME>=0.25 43   6.511628 2.186047 *
##    3) adascog>=11.165 269  74.840150 2.460967
##      6) CDHOME< 0.75 196  47.122450 2.275510
##       12) CDHOME< 0.25 106  17.858490 2.103774 *
##       13) CDHOME>=0.25 90  22.455560 2.477778 *
##      7) CDHOME>=0.75 73   2.876712 2.958904 *
```

```
# install.packages("rpart.plot")
library(rpart.plot)
rpart.plot(ad.rpart, digits=3)
```




```
rpart.plot(ad.rpart, digits = 4, fallen.leaves = T, type=3, extra=101)
```



Firstly, I utilize PCA (Principal Component Analysis) to transform a number of possibly correlated variables into a smaller number of uncorrelated variables through a process known as

orthogonal transformation. The first step is to load the data into the software R and name it as ad_data:

```
#Load the data from website
site <- "http://wiki.socr.umich.edu/index.php/SOCR_Data_AD_BiomedBigMetadata"
wiki_url <- read_html(site)
html_nodes(wiki_url, "#content")
```

```
## {xml_nodeset (1)}
## [1] <div id="content" class="mw-body-primary" role="main">\n\t<a id="top ...
```

```
ad_data <- html_table(html_nodes(wiki_url,"table")[[1]])
head(ad_data)
```

Here, I did a bit of data cleaning -- removing the unrelated subject ID variable and transform those factor variables into numeric variables to make the later computation work smoothly.

```
# 10.2 Step 2: Exploring and preparing the data
# To make sure that the data is ready for further modeling, we first should delete the subject ID variable as it
 is not needed in the dimension reduction procedures.
ad_data <- ad_data[ ,-1]

ad_data[ad_data == "."] <- 0
ad_data$SymptomeChronicity <- as.numeric(ad_data$SymptomeChronicity)
ad_data$FAQTOTAL <- as.numeric(ad_data$FAQTOTAL)
ad_data$DX_Confidence <- as.numeric(ad_data$DX_Confidence)
ad_data$SymptomeSeverety <- as.numeric(ad_data$SymptomeSeverety)
ad_data$CTWHITE <- as.numeric(ad_data$CTWHITE)
ad_data$CTRED <- as.numeric(ad_data$CTRED)
ad_data$PROTEIN <- as.numeric(ad_data$PROTEIN)
ad_data$GLUCOSE <- as.numeric(ad_data$GLUCOSE)
ad_data$DX_Conversion <- as.numeric(ad_data$DX_Conversion)
ad_data$adascog <- as.numeric(ad_data$adascog)
ad_data$FAQTOTAL <- as.numeric(ad_data$FAQTOTAL)

summary(ad_data)
```

Then, I subtract the average of all column means from each element in the column in preparation of the PCA analysis. I cast ad_data as a matrix and compute its variance covariance matrix, S. After these steps, I calculate the corresponding eigenvalues and eigenvectors of S.

```
mu<-apply(ad_data, 2, mean)
mean(mu)
```

```
## [1] 11.7855
```

```
ad.center<-as.matrix(ad_data)-mean(mu)
S<-cov(ad.center)
eigen(S)
```

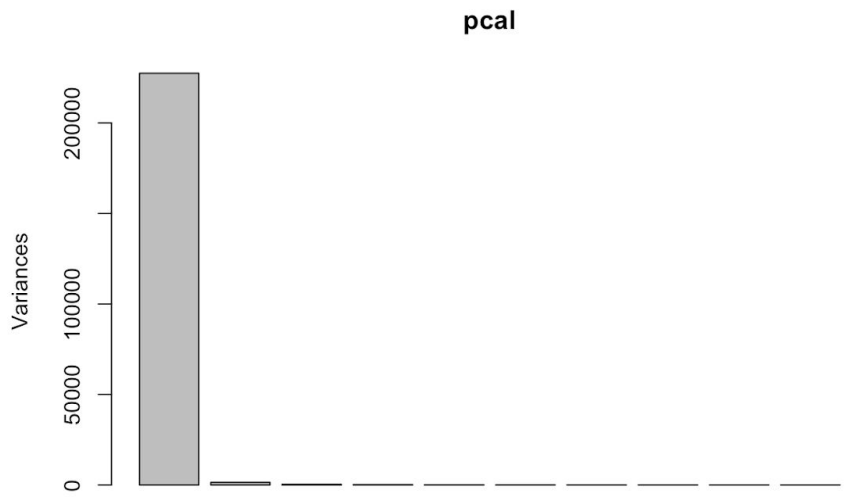
```
pcal <- prcomp(as.matrix(ad_data), center = T)

summary(pcal)
```

```
pcal$rotation
```

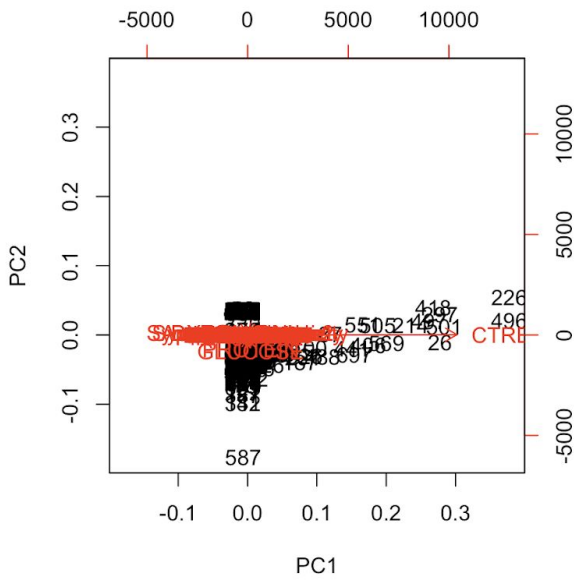
The eigenvalues correspond to the amount of the variation explained by each principal component (PC), which is the same as the eigenvalues of the S matrix.

I utilize the plot() function to visualize the PC loadings to give a better idea. From this plot, we can see that the pcal plot has a clear right skew.

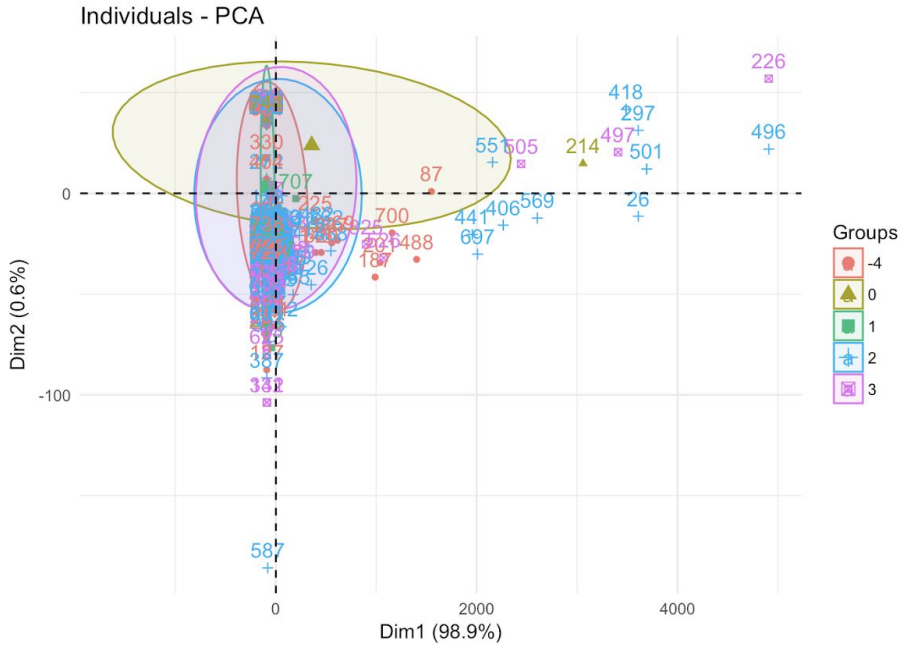
```
plot(pcal)
```




```
library(graphics)
biplot(pcal, choices = 1:2, scale = 1, pc.biplot = F)
```




```
# for plots of individuals
fviz_pca_ind(pcal, habillage = qualit_vars, addEllipses = TRUE, ellipse.level = 0.68) + theme_minimal()
```

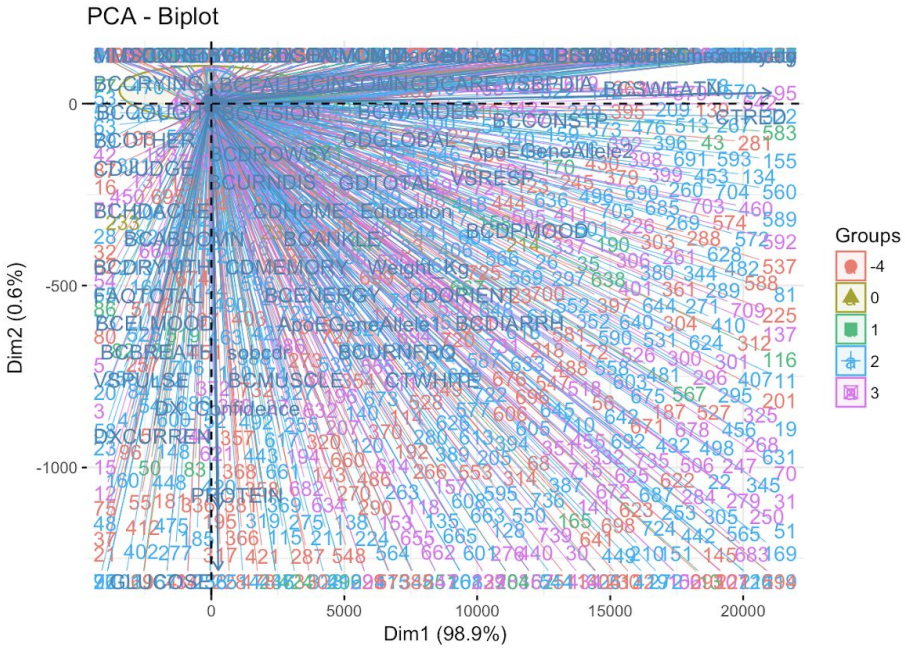


The histogram plot has a clear “elbow” point at the second PC, suggesting that the first two PCs explain about 95% of the variation in the original dataset. Thus, we say we can use the first 2 PCs to represent the data. In this case, the dimension of the data is substantially reduced.

Here, biplot uses PC1 and PC2 as the axes and red vectors to represent the direction of variables after adding loadings as weights. It help us to visualize how the loadings are used to rearrange the structure of the data.

Next, let's try to obtain a bootstrap test for the confidence interval of the explained variance.

```
# for Biplot of individuals and variables
fviz_pca_biplot(pcal, axes = c(1, 2), geom = c("point", "text"),
                col.ind = "black", col.var = "steelblue", label = "all",
                invisible = "none", repel = T, habillage = qualit_vars,
                palette = NULL, addEllipses = TRUE, title = "PCA - Biplot")
```



The histogram plot has a clear "elbow" point at the second PC, suggesting that the first two PCs explain about 95% of the variation in the original dataset. Thus, we say we can use the first 2 PCs to represent the data. In this case, the dimension of the data is substantially reduced.

Here, biplot uses PC1 and PC2 as the axes and red vectors to represent the direction of variables after adding loadings as weights. It help us to visualize how the loadings are used to rearrange the structure of the data.

Next, let's try to obtain a bootstrap test for the confidence interval of the explained variance.

```
library(scatterplot3d)

#Fit linear model
lm.fit <- lm(BCABDOMN ~  DXCURREN + CDGLOBAL, data = ad_data)

#plot results
myPlot <- scatterplot3d(ad_data$BCABDOMN, ad_data$DXCURREN, ad_data$CDGLOBAL)

# Get the ranges of the variable.names
summary(ad_data$DXCURREN)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   1.958   2.000   3.000
```

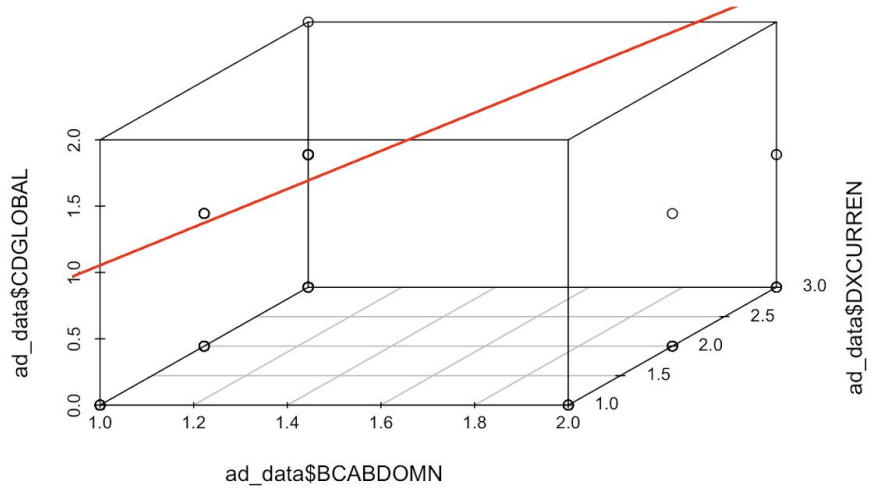
```
summary(ad_data$BCABDOMN)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.074   1.000   2.000
```

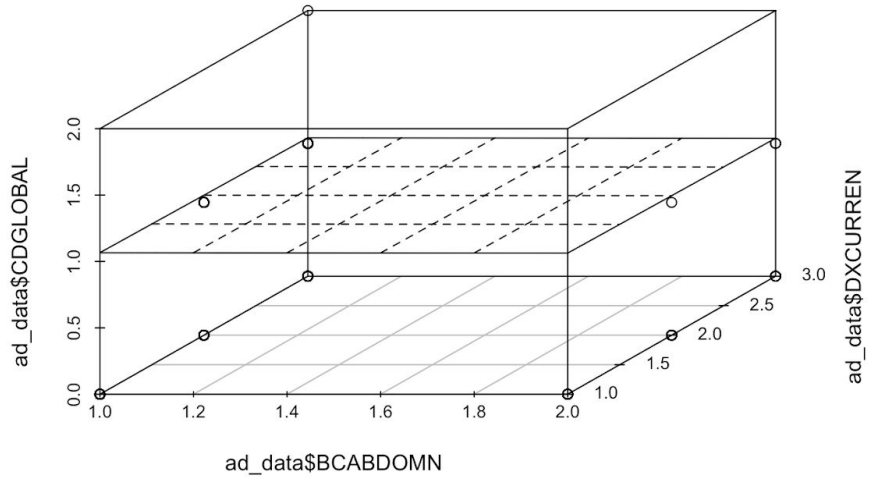
```
summary(ad_data$CDGLOBAL)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.0672  0.0000  2.0000
```

```
# Plot the linear model (line in 3D)
myCoef <- lm.fit$coefficients
plotX <- seq(0.93, 1.4,length.out = 100)
plotY <- seq(0,6,length.out = 100)
plotZ <- myCoef[1] + myCoef[2]*plotX + myCoef[3]*plotY # linear model
#Add the linear model to the 3D scatterplot
myPlot$points3d(plotX,plotY,plotZ, type = "l", lwd=2, col = "red")
```



```
myPlot <- scatterplot3d(ad_data$BCABDOMN, ad_data$DXCURREN, ad_data$CDGLOBAL)

# Static Plot
myPlot$plane3d(lm.fit, lty.box = "solid")
```



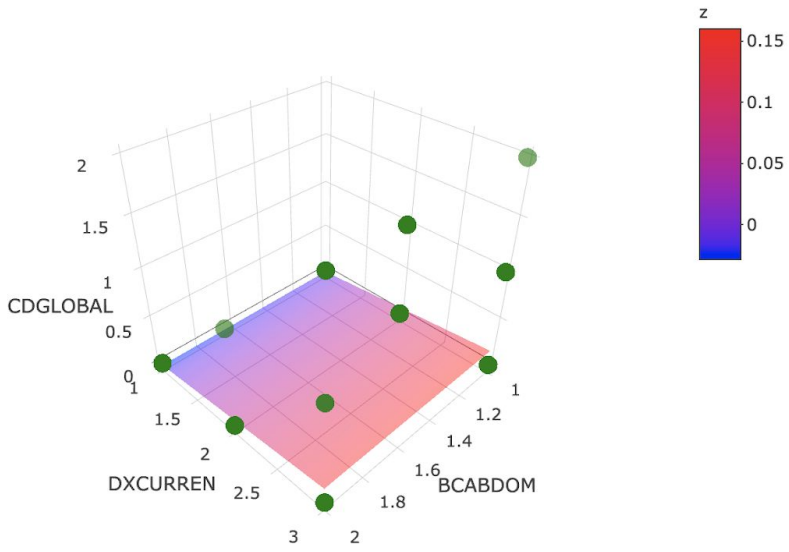

```
pca1 <- prcomp(as.matrix(cbind(ad_data$BCABDOMN, ad_data$DXCURREN, ad_data$CDGLOBAL)), center = T)
summary(pca1)
```

```
## Importance of components:
##                           PC1    PC2     PC3
## Standard deviation     0.7218 0.2619 0.24654
## Proportion of Variance 0.8011 0.1055 0.09345
## Cumulative Proportion  0.8011 0.9065 1.00000
```

```
normVec = c(pca1$rotation[,1][2]*pca1$rotation[,2][3]-
              pca1$rotation[,1][3]*pca1$rotation[,2][2],
            pca1$rotation[,1][3]*pca1$rotation[,2][1]-
              pca1$rotation[,1][1]*pca1$rotation[,2][3],
            pca1$rotation[,1][1]*pca1$rotation[,2][2]-
              pca1$rotation[,1][2]*pca1$rotation[,2][1]
)

# install.packages("rgl")
# Interactive RGL 3D plot with PCA Plane
library(rgl)
```

```
# Compute the 3D point representing the gravitational balance
dMean <- apply(cbind(ad_data$BCABDOMN, ad_data$DXCURREN, ad_data$CDGLOBAL), 2, mean)
# then the offset plane parameter is (d):
d <- as.numeric((-1)*normVec %*% dMean)  # force the plane to go through the mean

# Plot the PCA Plane
plot3d(ad_data$BCABDOMN, ad_data$DXCURREN, ad_data$CDGLOBAL, type = "s", col = "red", size = 1)
planes3d(normVec[1], normVec[2], normVec[3], d, alpha = 0.5)

# Define the 3D features
x <- ad_data$BCABDOM
y <- ad_data$DXCURREN
z <- ad_data$CDGLOBAL
myDF <- data.frame(x, y, z)

### Fit a (bivariate-predictor) linear regression model
lm.fit <- lm(z ~ x+y)
coef.lm.fit <- coef(lm.fit)

### Reparameterize the 2D (x,y) grid, and define the corresponding model values z on the grid
x.seq <- seq(min(x),max(x),length.out=100)
y.seq <- seq(min(y),max(y),length.out=100)
z.seq <- function(x,y) coef.lm.fit[1]+coef.lm.fit[2]*x+coef.lm.fit[3]*y
# define the values of z = z(x.seq, y.seq), as a Matrix of dimension c(dim(x.seq), dim(y.seq))
z <- t(outer(x.seq, y.seq, z.seq))

# First draw the 2D plane embedded in 3D, and then add points with "add_trace"
# install.packages('plotly')
library(plotly)
```

```
myPlotly <- plot_ly(x=~x.seq, y=~y.seq, z=~z,
                    colors = c("blue", "red"),type="surface", opacity=0.7) %>%

  add_trace(data=myDF, x=x, y=y, z=ad_data$CDGLOBAL, mode="markers",
            type="scatter3d", marker = list(color="green", opacity=0.9,
                                            symbol=105)) %>%

  layout(scene = list(
    aspectmode = "manual", aspectratio = list(x=1, y=1, z=1),
    xaxis = list(title = "BCABDOM"),
    yaxis = list(title = "DXCURREN"),
    zaxis = list(title = "CDGLOBAL"))
  )
# print(myPlotly)
myPlotly
```




```r
# install.packages("Rtsne")
library(Rtsne)

# If working with post-processed AD data above: remove duplicates (after stripping time)
ad_data <- unique(ad_data[,])

# Run the t-SNE, tracking the execution time (artificially reducing the sample-size to get reasonable calculation
 time)
execTime_tSNE <- system.time(tsne_digits <- Rtsne(ad_data, dims = 2, perplexity=30, verbose=TRUE, max_iter = 1000
))
```

```
## Read the 744 x 50 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 744
## Done in 0.10 seconds (sparsity = 0.168679)!
## Learning embedding...
## Iteration 50: error is 60.285453 (50 iterations in 0.32 seconds)
## Iteration 100: error is 57.846652 (50 iterations in 0.32 seconds)
## Iteration 150: error is 57.537668 (50 iterations in 0.33 seconds)
## Iteration 200: error is 57.398163 (50 iterations in 0.46 seconds)
## Iteration 250: error is 57.315573 (50 iterations in 0.38 seconds)
## Iteration 300: error is 1.146274 (50 iterations in 0.26 seconds)
## Iteration 350: error is 1.023349 (50 iterations in 0.26 seconds)
## Iteration 400: error is 0.995633 (50 iterations in 0.26 seconds)
## Iteration 450: error is 0.975695 (50 iterations in 0.25 seconds)
## Iteration 500: error is 0.966837 (50 iterations in 0.24 seconds)
## Iteration 550: error is 0.958256 (50 iterations in 0.26 seconds)
## Iteration 600: error is 0.950938 (50 iterations in 0.24 seconds)
## Iteration 650: error is 0.947155 (50 iterations in 0.25 seconds)
## Iteration 700: error is 0.943608 (50 iterations in 0.31 seconds)
## Iteration 750: error is 0.942047 (50 iterations in 0.27 seconds)
## Iteration 800: error is 0.939308 (50 iterations in 0.31 seconds)
## Iteration 850: error is 0.936660 (50 iterations in 0.27 seconds)
## Iteration 900: error is 0.934960 (50 iterations in 0.25 seconds)
## Iteration 950: error is 0.931646 (50 iterations in 0.27 seconds)
## Iteration 1000: error is 0.929727 (50 iterations in 0.24 seconds)
## Fitting performed in 5.75 seconds.
```

```
execTime_tSNE
```

```
##    user  system elapsed
##   6.010   0.079   6.661
```

```
table(ad_data$DXCURREN)
```

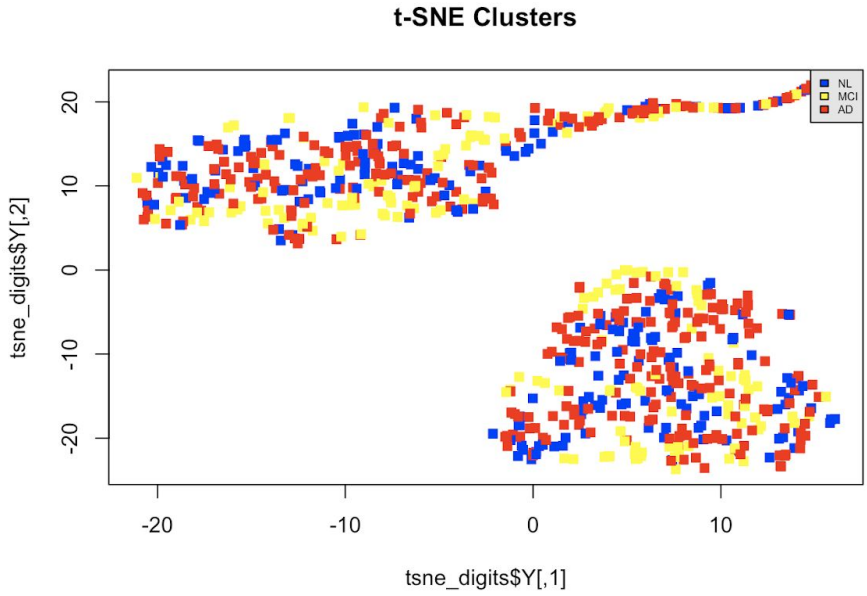
```
##
##   1   2   3
## 208 359 177
```

```
CharToColor = function(input_char){
  mapping = c("1"="blue", "2"="red", "3"="yellow")
  mapping[input_char]
}
ad_data$DXCURREN.col = sapply(ad_data$DXCURREN, CharToColor)

plot(tsne_digits$Y, main="t-SNE Clusters", col=ad_data$DXCURREN.col, pch = 15)
legend("topright", c("NL", "MCI", "AD"), fill=unique(ad_data$DXCURREN.col), bg='gray90', cex=0.5)
```

Now, I want to use t-SNE technique as a machine learning strategy for nonlinear dimensionality reduction, because it is useful for embedding (e.g., scatter-plotting) of high-dimensional data into lower-dimensional (1D, 2D, 3D) spaces. For each object (point in the high-dimensional space), the method models similar objects using nearby and dissimilar objects using remote distant objects. The two steps in t-SNE include (1) construction of a probability distribution over pairs of the original high-dimensional objects where that similar objects have a high probability of being picked and correspondingly, dissimilar objects have a small probability of being selected;

and (2) defining a similar probability distribution over the points in the derived low-dimensional embedding minimizing the Kullback-Leibler divergence between the high- and low-dimensional distributions relative to the locations of the objects in the embedding map. Either Euclidean or non-Euclidean distance measures between objects may be used as similarity metrics.

Here's how the t-SNE Clusters looks like:



I choose a few variables: gender, married, education, age, weight_kg to study their relationship with SymptomeSeverety. And I choose to divide the whole observations into 10 clusters.

```
# choose a few variables for us to study the relationship with SymptomeSeverety
ad_data_sub <- ad_data[c(11, 12, 13, 14, 15, 21)]
summary(ad_data_sub)
```

```
di_z<- as.data.frame(lapply(ad_data_sub, scale))
str(di_z)
```

```
## 'data.frame':    744 obs. of  6 variables:
##  $ Gender          : num  -0.828 -0.828 -0.828 1.206 1.206 ...
##  $ Married         : num  3.314 -0.301 -0.301 -0.301 3.314 ...
##  $ Education       : num  0.122 0.799 0.122 -0.895 0.799 ...
##  $ Age             : num  0.514 0.806 -0.217 0.66 1.244 ...
##  $ Weight_Kg       : num  0.9161 -0.0854 0.8494 -0.3926 -1.0202 ...
##  $ SymptomeSeverety: num  0.477 0.912 1.346 0.477 0.912 ...
```

```r
library(stats)
set.seed(321)
diz_clussters<-kmeans(di_z, 10)
diz_clussters$size
```

```
##  [1]  98  37  88  80 101  25  80 125  44  66
```

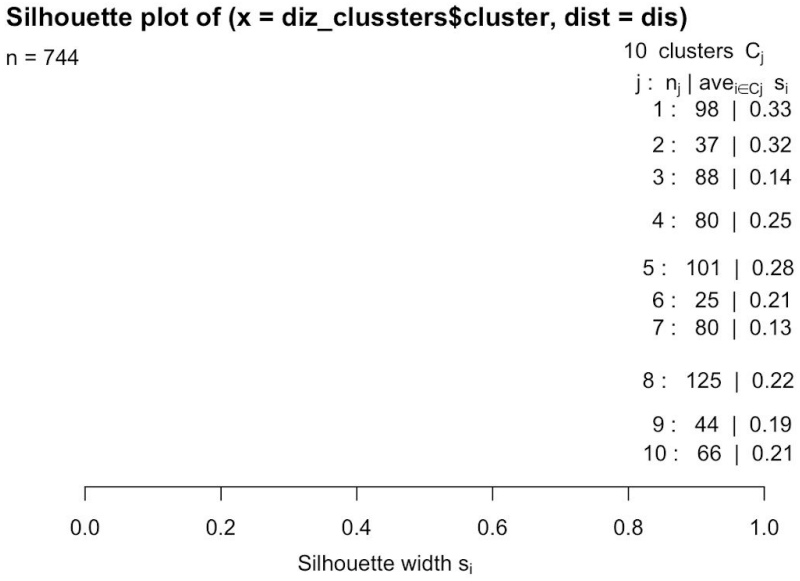
```r
require(cluster)
```

```
## Loading required package: cluster
```

```r
dis = dist(di_z)
sil = silhouette(diz_clussters$cluster, dis)
summary(sil)
```

```
## Silhouette of 744 units in 10 clusters from silhouette.default(x = diz_clussters$cluster, dist = dis) :
##  Cluster sizes and average silhouette widths:
##        98        37        88        80       101        25        80
## 0.3288310 0.3172919 0.1387517 0.2537492 0.2753873 0.2106932 0.1251885
##       125        44        66
## 0.2173494 0.1875424 0.2108584
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1165  0.1201  0.2276  0.2270  0.3392  0.5418
```

```r
plot(sil)
```



```
head(ad_data_sub)
```

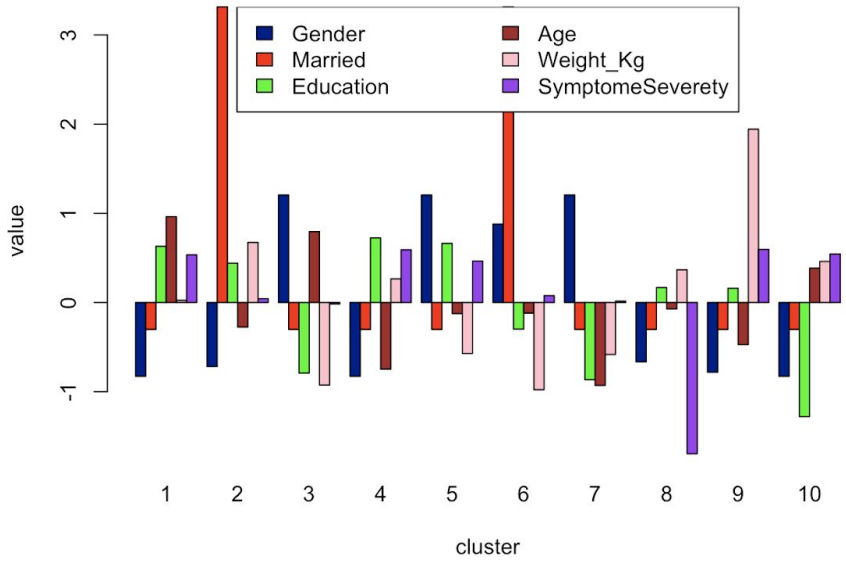
```
##   Gender Married Education Age Weight_Kg SymptomeSeverety
## 1      1       2        16  79  89.00000                1
## 2      1       1        18  81  74.00000                2
## 3      1       1        16  74  88.00000                3
## 4      2       1        13  80  69.39972                1
## 5      2       2        18  84  60.00000                2
## 6      2       1        12  74  68.00000                2
```

```
par(mfrow=c(1, 1), mar=c(4, 4, 4, 2))
myColors <- c("darkblue", "red", "green", "brown", "pink", "purple")
barplot(t(diz_clussters$centers), beside = TRUE, xlab="cluster",
        ylab="value", col = myColors)
legend("top", ncol=2, legend = c("Gender", "Married", "Education", "Age", "Weight_Kg", "SymptomeSeverety"), fill
 = myColors)

ad_data_sub$clusters<-diz_clussters$cluster
ad_data_sub[1:5, ]
```

```
##   Gender Married Education Age Weight_Kg SymptomeSeverety clusters
## 1      1       2        16  79  89.00000                1        2
## 2      1       1        18  81  74.00000                2        1
## 3      1       1        16  74  88.00000                3        4
## 4      2       1        13  80  69.39972                1        3
## 5      2       2        18  84  60.00000                2        6
```

```
require(ggplot2)
```

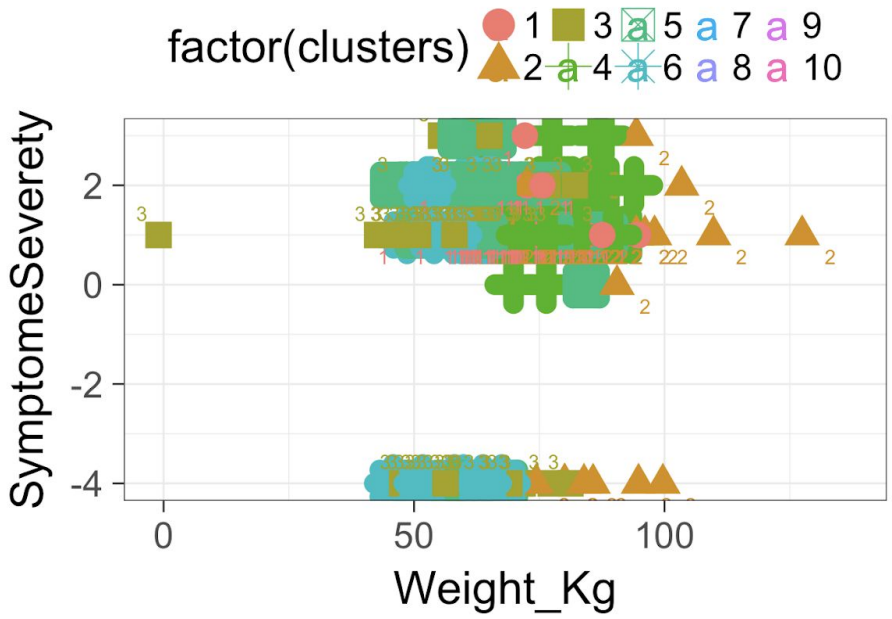
```
## Loading required package: ggplot2
```



```
ggplot(ad_data_sub, aes(Weight_Kg, SymptomeSeverety), main="Weight vs SymptomeSeverety") +
  geom_point(aes(colour = factor(clusters), shape=factor(clusters), stroke = 8), alpha=1) +
  theme_bw(base_size=25) +
  geom_text(aes(label=ifelse(clusters%in%1, as.character(clusters), ''), hjust=2, vjust=2, colour = factor(cluste
rs)))+
  geom_text(aes(label=ifelse(clusters%in%2, as.character(clusters), ''), hjust=-2, vjust=2, colour = factor(clust
ers)))+
  geom_text(aes(label=ifelse(clusters%in%3, as.character(clusters), ''), hjust=2, vjust=-1, colour = factor(clust
ers))) +
  guides(colour = guide_legend(override.aes = list(size=8))) +
  theme(legend.position="top")
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have
## 10. Consider specifying shapes manually if you must have them.
```

```
## Warning: Removed 315 rows containing missing values (geom_point).
```

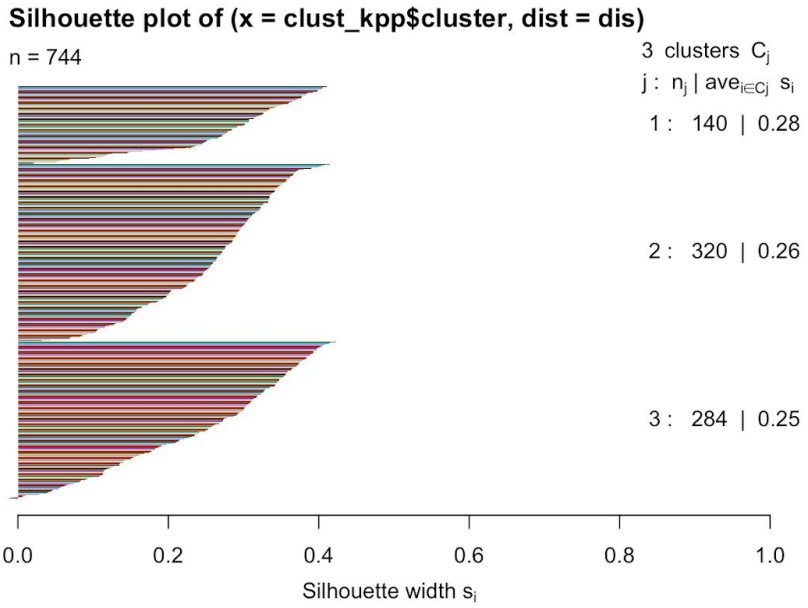


```r
# install.packages("matrixStats")
library(matrixStats)
kpp_init = function(dat, K) {
  x = as.matrix(dat)
  n = nrow(x)
  # Randomly choose a first center
  centers = matrix(NA, nrow=K, ncol=ncol(x))
  set.seed(123)
  centers[1,] = as.matrix(x[sample(1:n, 1),])
  for (k in 2:K) {
    # Calculate dist^2 to closest center for each point
    dists = matrix(NA, nrow=n, ncol=k-1)
    for (j in 1:(k-1)) {
      temp = sweep(x, 2, centers[j,], '-')
      dists[,j] = rowSums(temp^2)
    }
    dists = rowMins(dists)
    # Draw next center with probability proportional to dist^2
    cumdists = cumsum(dists)
    prop = runif(1, min=0, max=cumdists[n])
    centers[k,] = as.matrix(x[min(which(cumdists > prop)),])
  }
  return(centers)
}

clust_kpp = kmeans(di_z, kpp_init(di_z, 3), iter.max=100, algorithm='Lloyd')
clust_kpp$centers
```

```
##       Gender     Married  Education         Age  Weight_Kg
## 1 -0.5813629 -0.09469700  0.1482486 -0.15336050  0.3621335
## 2 -0.8156303  0.07156080  0.1703375  0.10079627  0.4849434
## 3  1.2056074 -0.03395027 -0.2650099 -0.03797302 -0.7249316
##   SymptomeSeverety
## 1       -1.6957441
## 2        0.5655158
## 3        0.1987293
```

```
## Silhouette of 744 units in 3 clusters from silhouette.default(x = clust_kpp$cluster, dist = dis) :
##  Cluster sizes and average silhouette widths:
##       140       320       284
## 0.2781701 0.2556724 0.2479231
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.01182  0.18425  0.27691  0.25695  0.33447  0.42241
```

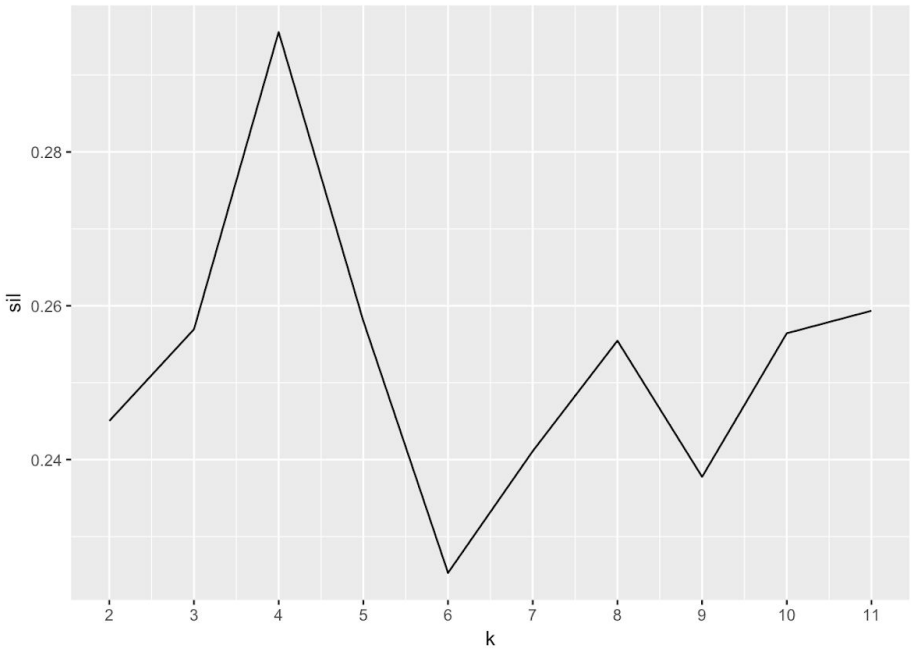
```
plot(sil2, col=1:length(diz_clussters$size), border=NA)
```



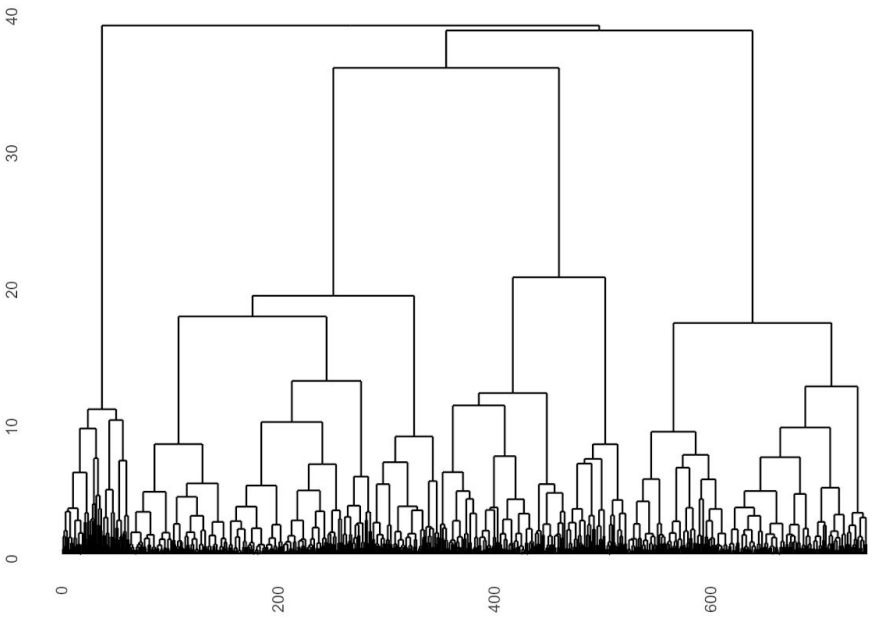

```
mat = matrix(0,nrow = 11)
for (i in 2:11){
  set.seed(321)
  clust_kpp = kmeans(di_z, kpp_init(di_z, i), iter.max=100, algorithm='Lloyd')
  sil = silhouette(clust_kpp$cluster, dis)
  mat[i] = mean(as.matrix(sil)[,3])
}
mat
```

```
##            [,1]
##  [1,] 0.0000000
##  [2,] 0.2450268
##  [3,] 0.2569478
##  [4,] 0.2955618
##  [5,] 0.2580041
##  [6,] 0.2252852
##  [7,] 0.2411053
##  [8,] 0.2554575
##  [9,] 0.2377648
## [10,] 0.2564237
## [11,] 0.2593522
```

```
ggplot(data.frame(k=2:11,sil=mat[2:11]),aes(x=k,y=sil))+geom_line()+scale_x_continuous(breaks = 2:11)
```




```
library(cluster)
pitch_sing = agnes(di_z, diss=FALSE, method='single')
pitch_comp = agnes(di_z, diss=FALSE, method='complete')
pitch_ward = agnes(di_z, diss=FALSE, method='ward')
sil_sing = silhouette(cutree(pitch_sing, k=3), dis)
sil_comp = silhouette(cutree(pitch_comp, k=3), dis)
# try 10 clusters, see plot above
sil_ward = silhouette(cutree(pitch_ward, k=10), dis)

# install.packages("ggdendro")
library(ggdendro)
ggdendrogram(as.dendrogram(pitch_ward), leaf_labels=FALSE, labels=FALSE)
```

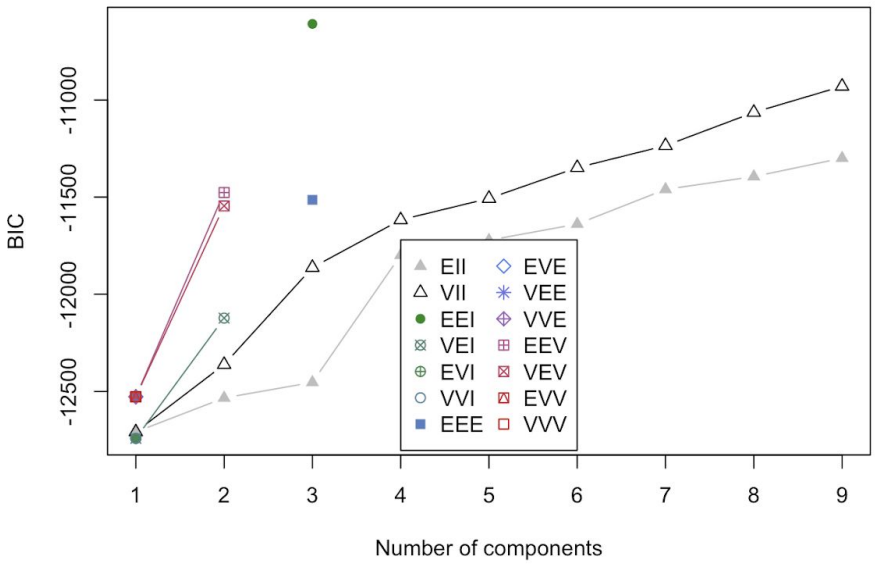


```r
library(mclust)
```

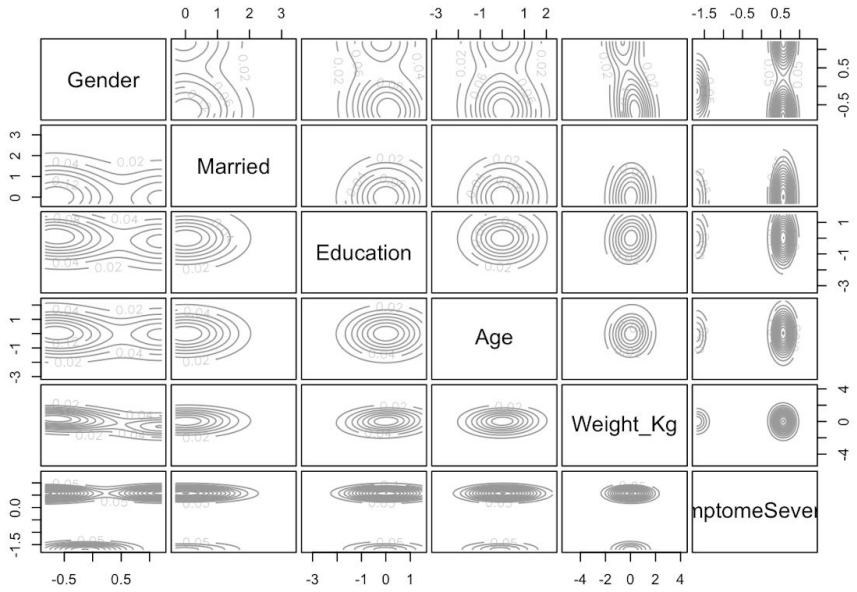
```
## Package 'mclust' version 5.4
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
set.seed(1234)
gmm_clust <- Mclust(di_z)
summary(gmm_clust, parameters = TRUE)
```

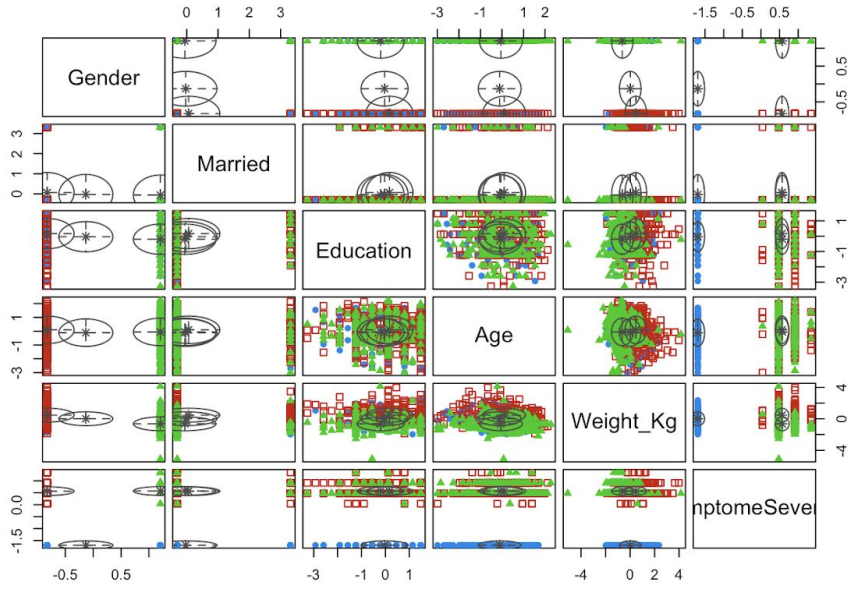
```r
plot(gmm_clust$BIC, legendArgs = list(x = "bottom", ncol = 2, cex = 1))
```



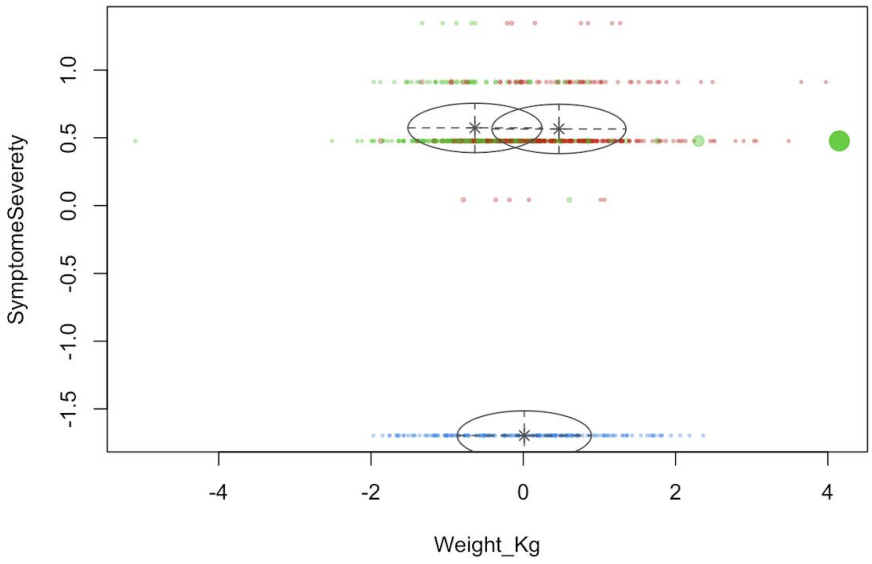
```
plot(gmm_clust, what = "density")
```




```
plot(gmm_clust, what = "classification")
```



```
plot(gmm_clust, what = "uncertainty", dimens = c(5,6), main = "Weight vs. SymptomSeverety")
```



SUMMARY:

From the above analysis, I first fit several Linear Models to examine the correlations between certain variables and how I can make accurate predictions with these correlations uncovered. Secondly, I use Dimensionality Reduction methods, mainly Principle Component Analysis (PCA), Independent Component Analysis (ICA), and Factor Analysis (FA) to reduce the dimensionality of the dataset so that I can extract a set of "uncorrelated" principal variables and reduce the complexity of the data. I also use Decision Tree Classification method to train and test data. This has proven to be a very effective method in evaluating and improving my model performance and has 82.5% specificity rate and 75.3% accuracy rate. Finally, I also use K-Means Clustering and Feature Selection method to place datasets into smaller clusters and analyze specific features/parameters and understand their correlation with the each other and with other variables to be examined.

ACKNOWLEDGEMENT

I would like to sincerely thank Professor Ivo Dinov for all his support and guidance with which I obtained the right direction to chose the topic, pursue my idea and finish the project. It was a pleasure to attend his course on Data Science and Predictive Analytics at the University of Michigan. The notes of this course significantly helped me understand a wide range of topics in Data Science and their application in Healthcare domain with statistical analysis and programming language R. The course assignments were fantastic and helped me to understand and use different R packages to achieve various kinds of analysis and interpretation. I'd like to also thank all of my classmates for their support and constructive feedback about my homework

assignments during this semester. It helped me to understand the according details and techniques more thoroughly, which enabled me to finish my final project in an efficient manner.

REFERENCE

1. Class notes: http://www.socr.umich.edu/people/dinov/courses/DSPA_Topics.html
2. Data Source: http://wiki.socr.umich.edu/index.php/SOCR_Data_AD_BiomedBigMetadata
3. Statistical computing and R code (co-submitted):
   HS_650_term_paper(1)
   HS_650_term_paper(2)
   HS_650_term_paper(3)
   HS_650_term_paper(4)