Yeast Data Set Multivariate Analysis

NAVEENSURYA V

6 March 2024

Loading data set

I choose a data set Yeast from UCL ML respiratory. I am loading csv file from the website directly it represents the classification of different types of yeast based on its different components namely "V2","V3",. . . ,"V9".

```
yeast <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/yeast/yeast.data")

head(yeast)
```

```
##           V1   V2   V3   V4   V5  V6  V7   V8   V9 V10
## 1 ADT1_YEAST 0.58 0.61 0.47 0.13 0.5 0.0 0.48 0.22 MIT
## 2 ADT2_YEAST 0.43 0.67 0.48 0.27 0.5 0.0 0.53 0.22 MIT
## 3 ADT3_YEAST 0.64 0.62 0.49 0.15 0.5 0.0 0.53 0.22 MIT
## 4 AAR2_YEAST 0.58 0.44 0.57 0.13 0.5 0.0 0.54 0.22 NUC
## 5 AATM_YEAST 0.42 0.44 0.48 0.54 0.5 0.0 0.48 0.22 MIT
## 6 AATC_YEAST 0.51 0.40 0.56 0.17 0.5 0.5 0.49 0.22 CYT
```

as we see here it contains 1484 rows with 10 columns. first column indicates name of particular yeast.

V1 represents Index. This is a unique identifier for each data point (gene) in the dataset.

V2 to V8 represents Mitochr, ER, Golgi, Vacuole, Cytosol, Nucleus, Lysosome, Peroxisome: These columns represent different cellular compartments (organelles) within the yeast cell. Each column contains a binary value (0 or 1), indicating whether the corresponding protein encoded by the gene is localized in that specific compartment.

last column indicates groups like "MIT" , "CYT", etc. . .

Meaning of each groups

1.MIT

Score of discriminant analysis of the amino acid content of the N-terminal region (20 residues long) of mitochondrial and non-mitochondrial proteins

2.MCH

McGeoch's method for signal sequence recognition.

3.GVH

von Heijne's method for signal sequence recognition.

4.ALM

Score of the ALOM membrane spanning region prediction program

5.ERL

Presence of HDEL substring (thought to act as a signal for retention in the endoplasmic reticulum lumen). Binary attribute.

6.POX

`Peroxisomal targeting signal in the C-terminus.`

7.VAC

Score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins.

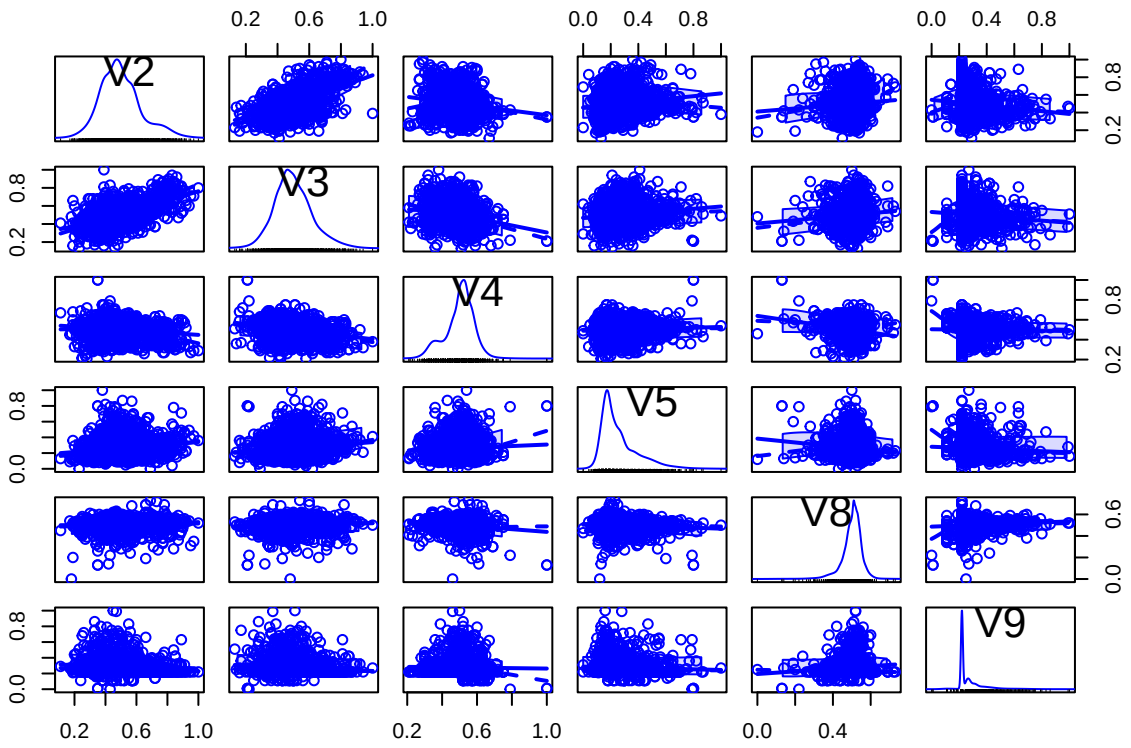
8.NUC

Score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins.

```
library(car)
```

```
## Loading required package: carData
```

```
scatterplotMatrix(yeast[c(2,3,4,5,8,9)])
```



Here,In this matrix scatterplot, the diagonal cells show histograms of each of the variables, in this case the concentrations of each variables (V2, V3, . . . , V9) here we can see all V2,..,V9 are concentrated on particular region. it is nor widely concentrated

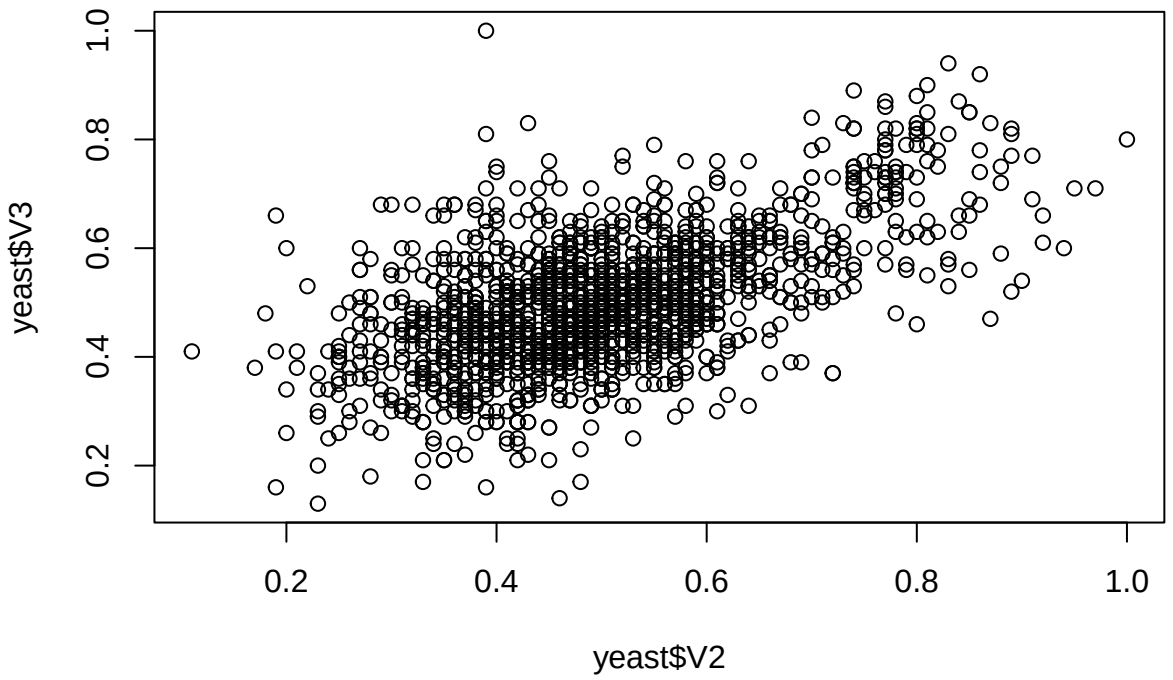
Each non diagonal element in scatterplot refers plot between corresponding Vi and Vj where i,j is its position.

Observation from the Scatterplot

1. V2 and V3 are positively correlated.
2. V9 has highly positive kurtosis.
3. V5,V8 and V9 has almost equal kurtosis.
4. V4, V5 and V9 have skewness towards RIGHT.

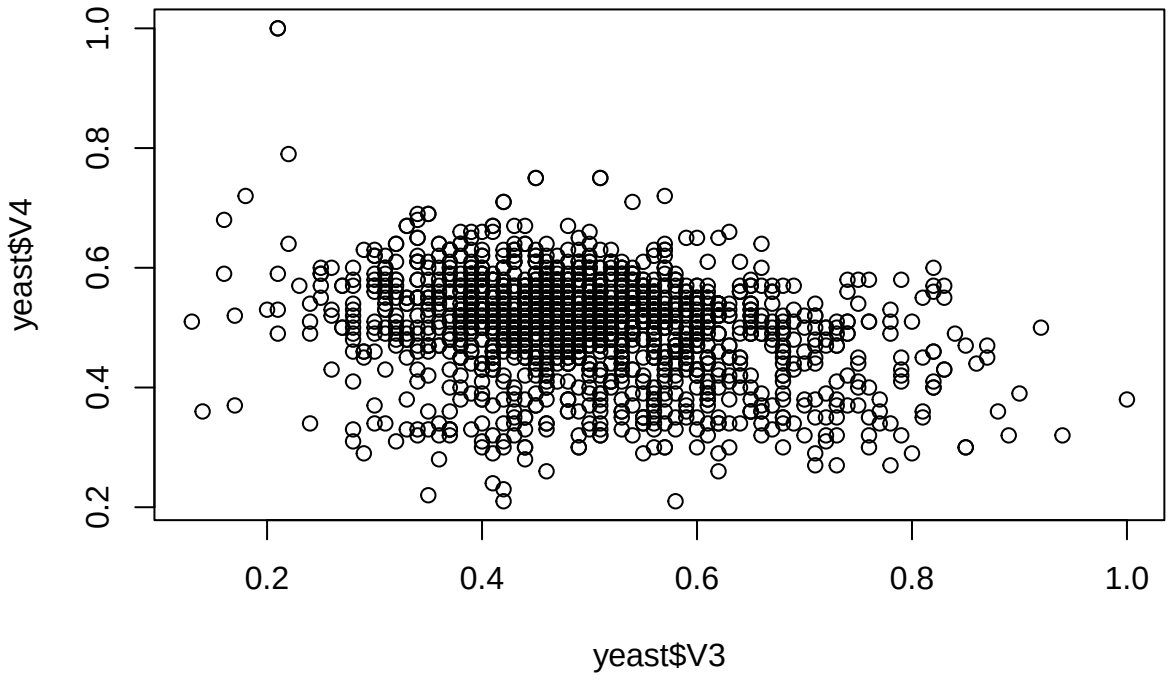
5. V8 have skewness towards LEFT.

```
plot(yeast$V2, yeast$V3)
```



here, we plotted V2 vs. v3 , if we can see it properly we can infer it is positively correlated as we see in scatter plot. remaining graphs we didn't see any direct relationship by looking into the scatter plot. In the same way we can plot each plots and check result separately which we identified from the scatter plot

```
plot(yeast$V3, yeast$V4)
```



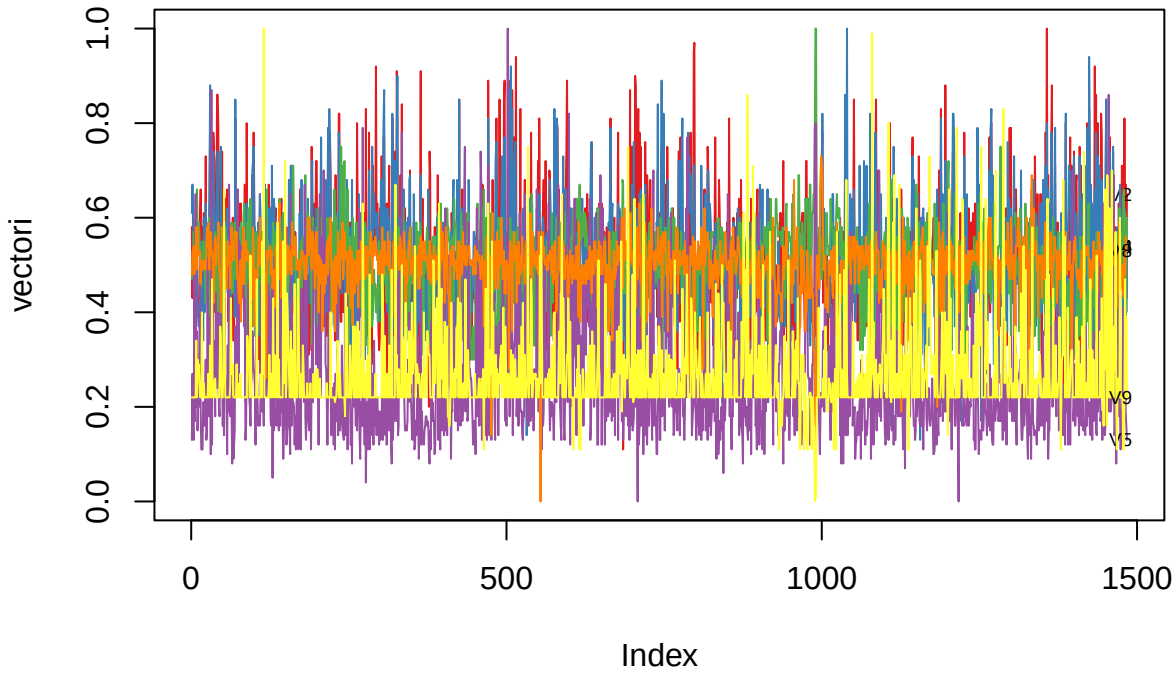
here we can see, both are almost spreaded and we can conclude about correlation here.

Profile plot

```
makeProfilePlot <- function(mylist,names)
  {
     require(RColorBrewer)
     # find out how many variables we want to include
     numvariables <- length(mylist)
     # choose 'numvariables' random colours
     colours <- brewer.pal(numvariables,"Set1")
     # find out the minimum and maximum values of the variables:
     mymin <- 1e+20
     mymax <- 1e-20
     for (i in 1:numvariables)
     {
        vectori <- mylist[[i]]
        mini <- min(vectori)
        maxi <- max(vectori)
        if (mini < mymin) { mymin <- mini }
        if (maxi > mymax) { mymax <- maxi }
     }
     # plot the variables
     for (i in 1:numvariables)
     {
        vectori <- mylist[[i]]
        namei <- names[i]
        colouri <- colours[i]
```

```
        if (i == 1) { plot(vectori,col=colouri,type="l",ylim=c(mymin,mymax)) }
        else         { points(vectori, col=colouri,type="l")                                    }
        lastxval <- length(vectori)
        lastyval <- vectori[length(vectori)]
        text((lastxval-10),(lastyval),namei,col="black",cex=0.6)
    }
  }

library(RColorBrewer)
names <- c("V2", "V3", "V4", "V5" ,"V8", "V9")
mylist <- list(yeast$V2,yeast$V3,yeast$V4,yeast$V5,yeast$V8,yeast$V9)
makeProfilePlot(mylist,names)
```

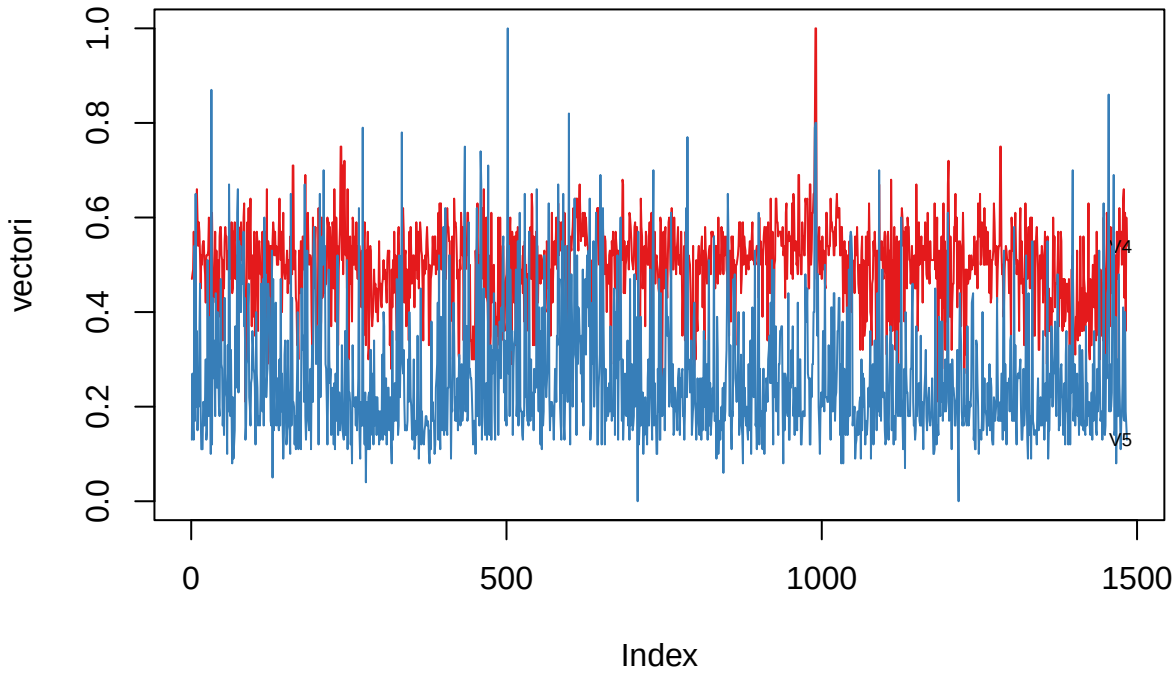


this profile plotting indicates all mean lies almost close to each other. Since it is densed we cant infer anything about standard deviation .

So, we can plot less in profile plot to see the hidden pattern

```
names23 <- c("V4", "V5")
mylist23 <- list(yeast$V4,yeast$V5)
makeProfilePlot(mylist23,names23)
```

```
## Warning in brewer.pal(numvariables, "Set1"): minimal value for n is 3, returning requested palette wi
```



Observation from profile plot V4 vs V5

Differences in Means:

The difference in means between V4 and V5 suggests that, on average, the values of V5 are higher or lower than those of V4. This difference in means could indicate a shift in the central tendency of the data between the two variables.

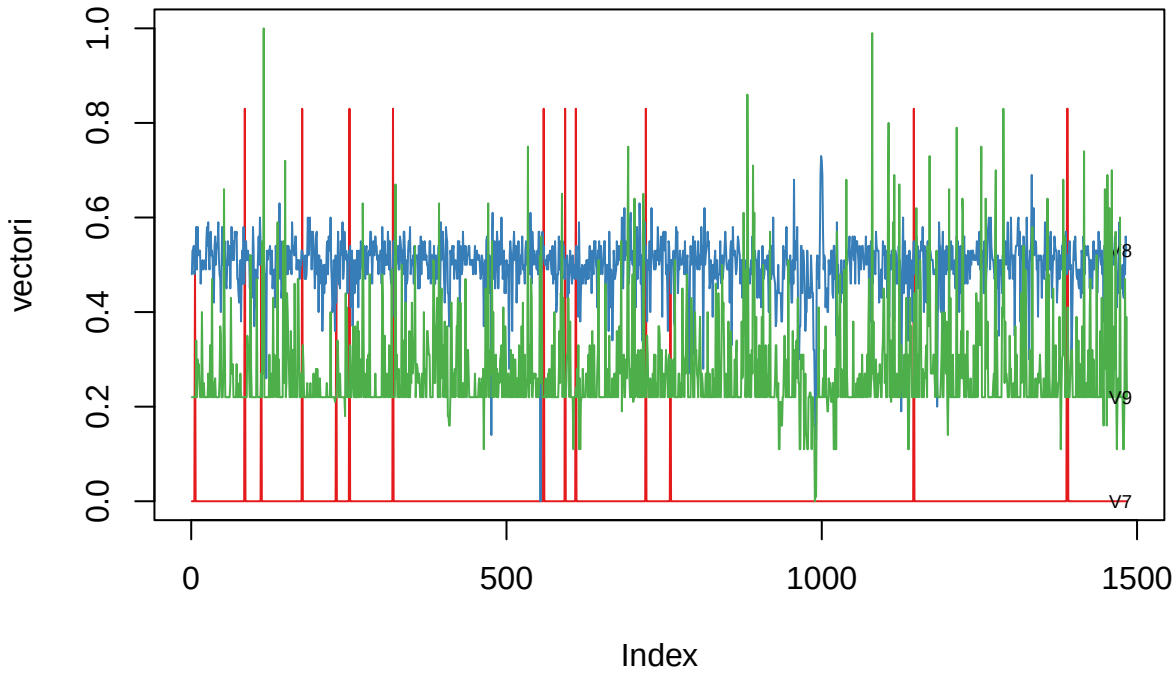
Variability:

The higher deviation of V5 compared to V4 implies that the data points of V5 are more spread out from the mean compared to V4. This could indicate greater variability or dispersion within the dataset of V5.

Skewness:

The higher deviation of V5 might also suggest that its distribution is more skewed compared to V4. If the distribution of V5 is more spread out towards one tail compared to the other, it could result in a higher standard deviation.

```
names3 <- c("V7" ,"V8", "V9")
mylist3 <- list(yeast$V7,yeast$V8,yeast$V9)
makeProfilePlot(mylist3,names3)
```



Observation from profile plot V4 vs V5

V7 Nearly Zero Everywhere with Few Red Areas:

The predominance of zero values in V7, except for a few areas highlighted in red, suggests that V7 is mostly close to zero but occasionally takes non-zero values in certain regions. These non-zero values, represented in red, could indicate specific instances or conditions where V7 deviates from its usual pattern of being close to zero.

Differences in Means between V8 and V9:

The presence of different means between V8 and V9 implies that, on average, the values of V9 are higher or lower than those of V8. This difference in means could indicate a shift in the central tendency of the data between the two variables.

Variability and Deviation:

The higher deviation of V9 compared to V8, indicated by the green color (V9) being highly deviated compared to the blue color (V8), suggests that the data points of V9 are more spread out from the mean compared to V8. This higher variability could indicate a wider range of values or greater dispersion within the dataset of V9.

Calculating Summary Statistics for Multivariate Data

```
sapply(yeast[2:9],mean)
```

```
##        V2        V3        V4        V5        V6        V7        V8        V9
## 0.5001213 0.4999326 0.5000337 0.2611860 0.5047170 0.0075000 0.4998854 0.2761995
```

Here we can see mean values for each of following columns V2,V3,V4,V6 and V8 have mean very close to each other as we see from profile plot.

Observation of means

Consistency: The means of V2, V3, and V4 are close to each other, around 0.5, indicating that the data represented by these variables have similar average values. This suggests consistency or similarity in those datasets.

Outlier: V7 has a mean of 0.4998854, which is very close to 0.5. This might indicate that the dataset represented by V7 has values very close to 0.5, suggesting potential clustering or a dominant value around this mean.

Skewness: The mean of V5 is noticeably lower than the others, at 0.261186. This might suggest that the data represented by V5 is skewed towards lower values, as the mean is pulled towards the lower end of the distribution.

```
sapply(yeast[2:9],sd)
```

```
##         V2         V3         V4         V5         V6         V7         V8
## 0.13729930 0.12392435 0.08667025 0.13709763 0.04835097 0.07568267 0.05779659
##         V9
## 0.10649053
```

Here we can see Standard Deviation of each columns

Observation of Standard deviations

Consistency: If the standard deviations of the values are similar, it suggests that the datasets have similar levels of variability. Conversely, if there are large disparities in the standard deviations, it indicates that some datasets have more variability than others.

Outliers: A very large standard deviation relative to the mean might suggest the presence of outliers or extreme values in the dataset. These outliers can significantly impact the standard deviation, pulling it away from the mean.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(gplots) # Load the gplots library for heatmap function
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
# Compute correlation matrix
correlation_matrix <- cor(yeast[2:9])
correlation_matrix
```

```
##              V2           V3           V4           V5           V6
## V2  1.000000000  0.5816314195 -0.163951300  0.158175472  0.064921537
## V3  0.581631420  1.0000000000 -0.271800035  0.140313937  0.060823302
## V4 -0.163951300 -0.2718000351  1.000000000  0.059668297 -0.008083462
## V5  0.158175472  0.1403139371  0.059668297  1.000000000 -0.005930705
## V6  0.064921537  0.0608233024 -0.008083462 -0.005930705  1.000000000
## V7  0.005596973  0.0003918342  0.009377914 -0.009039816 -0.009674234
## V8  0.075042684  0.0887593904 -0.185805390 -0.103591413  0.043626910
## V9 -0.124540384 -0.1029839928 -0.022042837 -0.054796535  0.002829256
##               V7          V8           V9
## V2  0.0055969730  0.07504268 -0.124540384
```

```
## V3  0.0003918342  0.08875939 -0.102983993
## V4  0.0093779144 -0.18580539 -0.022042837
## V5 -0.0090398164 -0.10359141 -0.054796535
## V6 -0.0096742336  0.04362691  0.002829256
## V7  1.0000000000  0.02089969 -0.035658650
## V8  0.0208996906  1.00000000  0.089690412
## V9 -0.0356586498  0.08969041  1.000000000
```

it gives correlation matrix for each data.

Observation from Correlation Matrix

Strength of Correlation:

1. Variables with correlation coefficients close to 1 or -1 have a strong linear relationship. For example, V2 and V3 have a correlation coefficient of 0.5816, indicating a moderately positive correlation.
2. Variables with correlation coefficients close to 0 suggest a weak or no linear relationship. For example, the correlation coefficient between V6 and V7 is very close to 0, indicating a weak linear relationship between these variables.

Direction of Correlation:

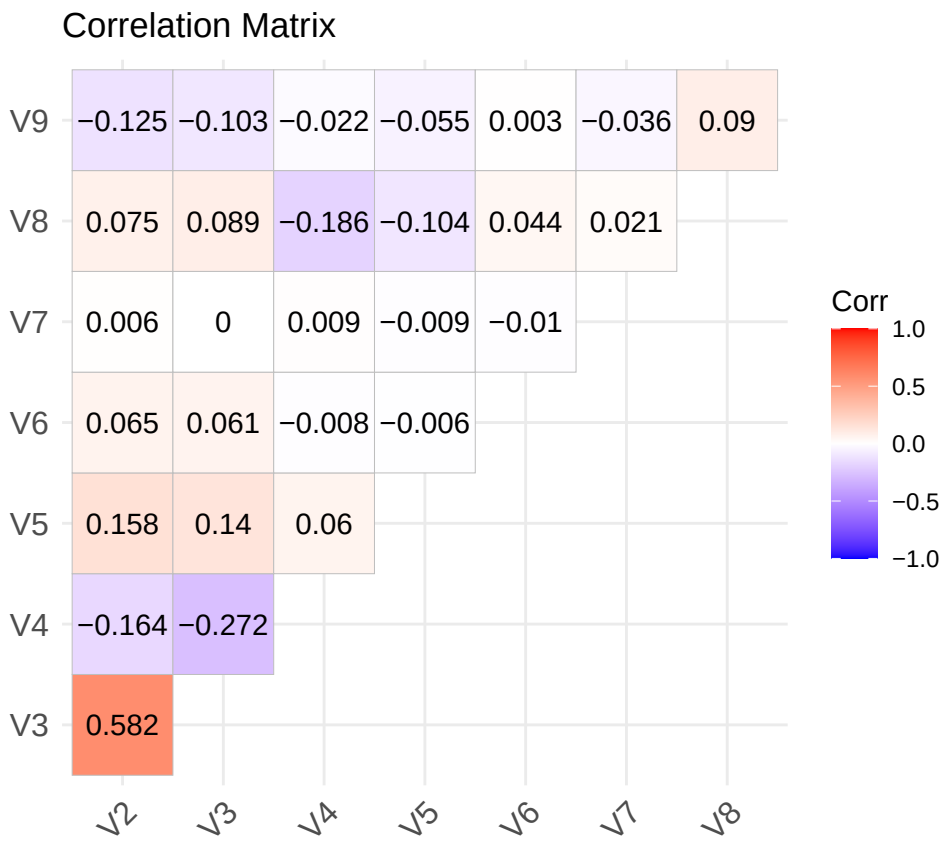
1. Positive correlation coefficients (closer to 1) indicate that when one variable increases, the other variable tends to increase as well. For example, V2 and V5 have a positive correlation coefficient of 0.1582.
2. Negative correlation coefficients (closer to -1) indicate that when one variable increases, the other variable tends to decrease. For example, V2 and V4 have a negative correlation coefficient of -0.1639.

we can plot using ggcorrplot to get analyse

```
library(ggcorrplot)
```

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

```
ggcorrplot(correlation_matrix,lab = T,type = "upper",digits = 3,title = "Correlation Matrix ")
```



as we see above, V2 and V3 are highly correlated and we can get top 10 highly correlated by using this code. We can see the correlation matrix above, 1. we can clearly see V2 is highly positively correlated with V2 as we saw from scatter plot

```
mosthighlycorrelated <- function(mydataframe, numtoreport) {
  # find the correlations
  cormatrix <- cor(mydataframe)

  # set the correlations in the upper triangle and diagonal to zero,
  # so they will not be reported as the highest ones:
  cormatrix[upper.tri(cormatrix, diag = TRUE)] <- 0

  # flatten the matrix into a dataframe for easy sorting
  indices <- which(cormatrix != 0, arr.ind = TRUE)
  fm <- data.frame(First.Variable = rownames(cormatrix)[indices[, 1]],
                   Second.Variable = colnames(cormatrix)[indices[, 2]],
                   Correlation = cormatrix[indices])

  # sort and print the top n correlations
  fm <- fm[order(abs(fm$Correlation), decreasing = TRUE), ]
  head(fm, n = numtoreport)
}
```

```
mosthighlycorrelated(yeast[2:8], 10)
```

```
##    First.Variable Second.Variable Correlation
## 1              V3              V2  0.58163142
## 7              V4              V3 -0.27180004
```

```
## 15             V8             V4 -0.18580539
## 2              V4             V2 -0.16395130
## 3              V5             V2  0.15817547
## 8              V5             V3  0.14031394
## 18             V8             V5 -0.10359141
## 11             V8             V3  0.08875939
## 6              V8             V2  0.07504268
## 4              V6             V2  0.06492154
```

here we see most highly correlated values. as we see from correlation ploting V2, V3 are highly correlated.

Mean and Standard Deviations by each Group

According to "V10" column we get there are multiple groups. we can calculate the mean and standard deviations within each groups

```
printMeanAndSdByGroup <- function(variables,groupvariable)
  {
     # find the names of the variables
     variablenames <- c(names(groupvariable),names(as.data.frame(variables)))
     # within each group, find the mean of each variable
     groupvariable <- groupvariable[,1] # ensures groupvariable is not a list
     means <- aggregate(as.matrix(variables) ~ groupvariable, FUN = mean)
     names(means) <- variablenames
     print(paste("Means:"))
     print(means)
     # within each group, find the standard deviation of each variable:
     sds <- aggregate(as.matrix(variables) ~ groupvariable, FUN = sd)
     names(sds) <- variablenames
     print(paste("Standard deviations:"))
     print(sds)
     # within each group, find the number of samples:
     samplesizes <- aggregate(as.matrix(variables) ~ groupvariable, FUN = length)
     names(samplesizes) <- variablenames
     print(paste("Sample sizes:"))
     print(samplesizes)
}

printMeanAndSdByGroup(yeast[2:8],yeast[10])
```

```
## [1] "Means:"
##    V10        V2        V3        V4        V5        V6          V7        V8
## 1  CYT 0.4807127 0.4695032 0.5354212 0.2271058 0.5043197 0.001079914 0.4993952
## 2  ERL 0.7920000 0.7720000 0.4820000 0.3360000 1.0000000 0.000000000 0.5460000
## 3  EXC 0.7354286 0.7168571 0.4925714 0.2920000 0.5000000 0.000000000 0.4557143
## 4  ME1 0.7886364 0.7565909 0.3761364 0.3118182 0.5000000 0.000000000 0.5129545
## 5  ME2 0.7215686 0.6031373 0.4149020 0.2825490 0.5098039 0.000000000 0.5101961
## 6  ME3 0.4308589 0.4895092 0.3642945 0.2134969 0.5030675 0.000000000 0.5101227
## 7  MIT 0.5214344 0.5332377 0.5173770 0.4044262 0.5000000 0.008852459 0.5016803
## 8  NUC 0.4524476 0.4561305 0.5293240 0.2283450 0.5034965 0.000000000 0.4941026
## 9  POX 0.5210000 0.5080000 0.5065000 0.2475000 0.5000000 0.423500000 0.5030000
## 10 VAC 0.5476667 0.5260000 0.4653333 0.2010000 0.5000000 0.000000000 0.5250000
## [1] "Standard deviations:"
##    V10         V2         V3         V4         V5         V6         V7
## 1  CYT 0.10744237 0.09211542 0.06470200 0.11516525 0.04632281 0.02323697
## 2  ERL 0.06534524 0.13535139 0.09203260 0.06655825 0.00000000 0.00000000
```

```
## 3  EXC 0.11059579 0.10747835 0.05462939 0.07086856 0.00000000 0.00000000
## 4  ME1 0.06711985 0.07329276 0.05969753 0.12730247 0.00000000 0.00000000
## 5  ME2 0.16036674 0.12039087 0.07553470 0.12547260 0.07001400 0.00000000
## 6  ME3 0.09889762 0.11423824 0.05434196 0.08018908 0.03916302 0.00000000
## 7  MIT 0.09723752 0.09851453 0.07292595 0.16585009 0.00000000 0.08136386
## 8  NUC 0.11093201 0.10994239 0.05833163 0.10989720 0.04171429 0.00000000
## 9  POX 0.13317578 0.11353645 0.05081183 0.11111279 0.00000000 0.40477772
## 10 VAC 0.14119401 0.13410238 0.09205446 0.09422460 0.00000000 0.00000000
##            V8
## 1  0.06402838
## 2  0.02073644
## 3  0.07009597
## 4  0.06094525
## 5  0.06071211
## 6  0.03527647
## 7  0.04650816
## 8  0.06066569
## 9  0.04027537
## 10 0.04932300
## [1] "Sample sizes:"
##    V10  V2  V3  V4  V5  V6  V7  V8
## 1  CYT 463 463 463 463 463 463 463
## 2  ERL   5   5   5   5   5   5   5
## 3  EXC  35  35  35  35  35  35  35
## 4  ME1  44  44  44  44  44  44  44
## 5  ME2  51  51  51  51  51  51  51
## 6  ME3 163 163 163 163 163 163 163
## 7  MIT 244 244 244 244 244 244 244
## 8  NUC 429 429 429 429 429 429 429
## 9  POX  20  20  20  20  20  20  20
## 10 VAC  30  30  30  30  30  30  30
```

From, this we can get group-wise Mean , standard Deviation.

Observation from group-wise mean

V2-V5: Show variability in mean values across groups. V2 and V3 have higher means in the ERL group, while V4 has the highest mean in the CYT group. V5 varies, with the highest mean in the MIT group.

V6-V7: Consistently close to zero across groups, indicating low variability and little influence on group differences.

V8: Shows variability in mean values across groups. Highest mean in ERL, lowest in EXC.

Observations from group-wise Standard deviation

V2-V5:

Standard deviation values for V2-V5 vary across different groups, indicating variability in these variables among the groups. V2 and V3 tend to have relatively low standard deviations across groups, suggesting consistency in their measurements. V4 and V5 show slightly higher standard deviations, implying more variability in their measurements across groups.

V6-V7: Standard deviations for V6 and V7 are consistently close to zero across groups, suggesting low variability in these variables and little influence on group differences.

V8: Standard deviation values for V8 vary across groups, indicating variability in this variable among the groups. Some groups, such as EXC and MIT, have relatively higher standard deviations for V8, suggesting more variability in their measurements compared to other groups

Within group variance

```
calcWithinGroupsVariance <- function(variable,groupvariable)
  {
     # find out how many values the group variable can take
     groupvariable2 <- as.factor(groupvariable[[1]])
     levels <- levels(groupvariable2)
     numlevels <- length(levels)
     # get the mean and standard deviation for each group:
     numtotal <- 0
     denomtotal <- 0
     for (i in 1:numlevels)
     {
        leveli <- levels[i]
        levelidata <- variable[groupvariable==leveli,]
        levelilength <- length(levelidata)
        # get the standard deviation for group i:
        sdi <- sd(levelidata)
        numi <- (levelilength - 1)*(sdi * sdi)
        denomi <- levelilength
        numtotal <- numtotal + numi
        denomtotal <- denomtotal + denomi
     }
     # calculate the within-groups variance
     Vw <- numtotal / (denomtotal - numlevels)
     return(Vw)
}

calcWithinGroupsVariance(yeast[2],yeast[10])
```

```
## [1] 0.01174351
```

The value we got, **0.01174351**, appears to be the within-group variance. Within-group variance measures the variability or spread of data within each group or category in a dataset.

```
calcBetweenGroupsVariance <- function(variable,groupvariable)
  {
     # find out how many values the group variable can take
     groupvariable2 <- as.factor(groupvariable[[1]])
     levels <- levels(groupvariable2)
     numlevels <- length(levels)
     # calculate the overall grand mean:
     grandmean <- as.numeric(sapply(variable, mean, na.rm = TRUE))
     # get the mean and standard deviation for each group:
     numtotal <- 0
     denomtotal <- 0
     for (i in 1:numlevels)
     {
        leveli <- levels[i]
        levelidata <- variable[groupvariable==leveli,]
        levelilength <- length(levelidata)
        # get the mean and standard deviation for group i:
        meani <- mean(levelidata)
        sdi <- sd(levelidata)
        numi <- levelilength * ((meani - grandmean)^2)
        denomi <- levelilength
```

```
        numtotal <- numtotal + numi
        denomtotal <- denomtotal + denomi
    }
    # calculate the between-groups variance
    Vb <- numtotal / (numlevels - 1)
    Vb <- Vb[[1]]
    return(Vb)
}

calcBetweenGroupsVariance(yeast[2],yeast[10])
```

```
## [1] 1.182916
```

The value we got, **1.1829161**, appears to be the between-group variance.

```
calcSeparations <- function(variables,groupvariable)
  {
    # find out how many variables we have
    variables <- as.data.frame(variables)
    numvariables <- length(variables)
    # find the variable names
    variablenames <- colnames(variables)
    # calculate the separation for each variable
    for (i in 1:numvariables)
    {
        variablei <- variables[i]
        variablename <- variablenames[i]
        Vw <- calcWithinGroupsVariance(variablei, groupvariable)
        Vb <- calcBetweenGroupsVariance(variablei, groupvariable)
        sep <- Vb/Vw
        print(paste("variable",variablename,"Vw=",Vw,"Vb=",Vb,"separation=",sep))
    }
}
```

```
calcSeparations(yeast[2:9], yeast[10])
```

```
## [1] "variable V2 Vw= 0.0117435078018214 Vb= 1.18291640747013 separation= 100.729392565879"
## [1] "variable V3 Vw= 0.0106880775751165 Vb= 0.780062990637085 separation= 72.9844057693958"
## [1] "variable V4 Vw= 0.00409075979748818 Vb= 0.567790930429589 separation= 138.798403851095"
## [1] "variable V5 Vw= 0.0143737593958499 Vb= 0.743021257664542 separation= 51.6928965625425"
## [1] "variable V6 Vw= 0.00151267315657012 Vb= 0.137477877699013 separation= 90.8840598525164"
## [1] "variable V7 Vw= 0.00337258733495328 Vb= 0.391470140919874 separation= 116.074130049266"
## [1] "variable V8 Vw= 0.00326362269977999 Vb= 0.0159222962367519 separation= 4.87871843697658"
## [1] "variable V9 Vw= 0.0100576047561455 Vb= 0.221406172112244 separation= 22.0138072115985"
```

Here for each variables, we can get Variance within the group, and Variance between the group and separation.
Observation from the separations

Variability in Vw and Vb: There is significant variability in the values of Vw and Vb across the different variables (V2 through V9). For instance, Vw ranges from as low as 0.0015 to as high as 0.0144, and Vb ranges from 0.0159 to 1.1829. This suggests that different variables may have different characteristics or behaviors.

Relationship between Vw, Vb, and Separation: There seems to be a relationship between Vw, Vb, and the separation value. While the exact nature of this relationship would require further analysis, it appears that changes in Vw and Vb are associated with changes in separation. This could indicate some form of dependency or correlation between these variables.

Differences in Separation: The separation values (ranging from around 5 to 139) indicate differences in the degree of separation between Vw and Vb for each variable. Variables with higher separation values have more distinct values for Vw and Vb compared to those with lower separation values.

Potential Patterns or Trends: Further analysis could reveal potential patterns or trends in the data. For example, it may be interesting to investigate if there is any relationship between the magnitude of Vw/Vb and the separation value across variables. Additionally, clustering analysis or visualization techniques could help identify any inherent groupings or similarities among the variables.

Between-groups Covariance and Within-groups Covariance for Two Variables

```
calcWithinGroupsCovariance <- function(variable1,variable2,groupvariable)
{
  # find out how many values the group variable can take
  groupvariable2 <- as.factor(groupvariable[[1]])
  levels <- levels(groupvariable2)
  numlevels <- length(levels)
  # get the covariance of variable 1 and variable 2 for each group:
  Covw <- 0
  for (i in 1:numlevels)
  {
    leveli <- levels[i]
    levelidata1 <- variable1[groupvariable==leveli,]
    levelidata2 <- variable2[groupvariable==leveli,]
    mean1 <- mean(levelidata1)
    mean2 <- mean(levelidata2)
    levelilength <- length(levelidata1)
    # get the covariance for this group:
    term1 <- 0
    for (j in 1:levelilength)
    {
      term1 <- term1 + ((levelidata1[j] - mean1)*(levelidata2[j] - mean2))
    }
    Cov_groupi <- term1 # covariance for this group
    Covw <- Covw + Cov_groupi
  }
  totallength <- nrow(variable1)
  Covw <- Covw / (totallength - numlevels)
  return(Covw)
}
```

```
calcWithinGroupsCovariance(yeast[5],yeast[8],yeast[10])
```

```
## [1] -0.0008503556
```

the value we got , *-0.0008503556* is the covariance with in the group

Next, we are going to use multigroup library to get calculate varience between and within groups

```
library(multigroup)

result <- TBWvariance(yeast[,2:9], yeast[,10])

# Extract the desired outputs
within_var <- result$Within.Var
between_var <- result$Between.Var
```

```r
# Print the extracted outputs
print(data.frame(within_var))
```

```
##               V2            V3            V4            V5            V6
## V2  1.167224e-02  4.428958e-03 -6.271420e-04  6.735261e-04 -3.797072e-05
## V3  4.428958e-03  1.062321e-02 -1.144545e-03  1.499089e-04 -4.013858e-05
## V4 -6.271420e-04 -1.144545e-03  4.065934e-03  4.611370e-04 -6.356417e-06
## V5  6.735261e-04  1.499089e-04  4.611370e-04  1.428653e-02 -7.737837e-05
## V6 -3.797072e-05 -4.013858e-05 -6.356417e-06 -7.737837e-05  1.503493e-03
## V7 -8.558647e-05 -8.065075e-05 -1.260958e-05 -2.127704e-04 -1.456391e-06
## V8  5.689093e-04  5.938327e-04 -6.381078e-04 -8.451949e-04  4.376013e-05
## V9 -4.501581e-04 -1.804957e-04 -6.002053e-04  3.621689e-04  4.738137e-05
##               V7            V8            V9
## V2 -8.558647e-05  5.689093e-04 -4.501581e-04
## V3 -8.065075e-05  5.938327e-04 -1.804957e-04
## V4 -1.260958e-05 -6.381078e-04 -6.002053e-04
## V5 -2.127704e-04 -8.451949e-04  3.621689e-04
## V6 -1.456391e-06  4.376013e-05  4.738137e-05
## V7  3.352120e-03  7.118197e-05  1.131335e-05
## V8  7.118197e-05  3.243816e-03  6.410087e-04
## V9  1.131335e-05  6.410087e-04  9.996567e-03
```

here we can see the Dataframe which contains varience within group

```r
print(data.frame(between_var))
```

```
##               V2            V3            V4            V5            V6
## V2  7.178859e-03  5.467342e-03 -1.323840e-03  2.303875e-03  4.689558e-04
## V3  5.467342e-03  4.734030e-03 -1.774738e-03  2.233988e-03  4.045834e-04
## V4 -1.323840e-03 -1.774738e-03  3.445798e-03  2.478587e-04 -2.751806e-05
## V5  2.303875e-03  2.233988e-03  2.478587e-04  4.509232e-03  3.806489e-05
## V6  4.689558e-04  4.045834e-04 -2.751806e-05  3.806489e-05  8.343229e-04
## V7  1.437456e-04  8.432573e-05  7.412340e-05  1.189741e-04 -3.394482e-05
## V8  2.658676e-05  4.189794e-05 -2.926368e-04  2.435989e-05  7.815618e-05
## V9 -1.370756e-03 -1.178560e-03  3.967596e-04 -1.162176e-03 -3.281376e-05
##               V7            V8            V9
## V2  1.437456e-04  2.658676e-05 -1.370756e-03
## V3  8.432573e-05  4.189794e-05 -1.178560e-03
## V4  7.412340e-05 -2.926368e-04  3.967596e-04
## V5  1.189741e-04  2.435989e-05 -1.162176e-03
## V6 -3.394482e-05  7.815618e-05 -3.281376e-05
## V7  2.375746e-03  2.023745e-05 -2.987038e-04
## V8  2.023745e-05  9.662891e-05 -8.898318e-05
## V9 -2.987038e-04 -8.898318e-05  1.343665e-03
```

here we can see the Dataframe which contains varience between groups

Calculating Correlations for Multivariate Data

```r
cor.test(yeast$V2, yeast$V3)
```

```
##
##  Pearson's product-moment correlation
##
## data:  yeast$V2 and yeast$V3
## t = 27.526, df = 1482, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
##  0.5469332 0.6143348
## sample estimates:
##       cor
## 0.5816314
```

Correlation Coefficient (r):

The correlation coefficient (r) between V2 and V3 is approximately 0.5816. This indicates a moderately strong positive linear relationship between V2 and V3.

Test Statistic: The test statistic (t) is approximately 27.526. Degrees of freedom (df) is 1482.

Significance: The p-value is extremely small (less than 2.2e-16), suggesting strong evidence against the null hypothesis. Therefore, we reject the null hypothesis, indicating that there is a significant correlation between V2 and V3 in the yeast dataset.

Confidence Interval:

The 95% confidence interval for the correlation coefficient ranges from approximately 0.5469 to 0.6143. This interval indicates that we are 95% confident that the true correlation between V2 and V3 falls within this range.

Standardising Variables

```
standardisedconcentrations <- as.data.frame(scale(yeast[2:9]))
```

```
sapply(standardisedconcentrations,mean)
```

```
##            V2            V3            V4            V5            V6
##  2.819374e-16  4.832804e-17  1.617596e-17  9.448340e-17 -1.123165e-15
##            V7            V8            V9
##  8.747955e-18  3.829206e-17  8.868248e-17
```

```
sapply(standardisedconcentrations,sd)
```

```
## V2 V3 V4 V5 V6 V7 V8 V9
##  1  1  1  1  1  1  1  1
```

here we see we standarised all the variables.

Principal Component Analysis

we are going to perform Principal Component Analysis. we are using 8 variables. using PCA we can reduce the dimensionality of data while retaining the majority of variation.

```
yeast.pca <- prcomp(standardisedconcentrations)
```

```
yeast.pca
```

```
## Standard deviations (1, .., p=8):
## [1] 1.3469070 1.1270872 1.0105646 0.9970301 0.9670743 0.8955673 0.8696945
## [8] 0.6376238
##
## Rotation (n x k) = (8 x 8):
##             PC1         PC2         PC3          PC4         PC5         PC6
## V2 -0.609203749  0.12825311 -0.01883607  0.001786777 -0.03808517  0.06472147
## V3 -0.633179046  0.04252125 -0.01907814 -0.049126448 -0.01973746  0.15725521
## V4  0.355869732  0.43740649 -0.05327289  0.220533671 -0.13131668 -0.30438894
## V5 -0.189395266  0.50718077 -0.20518660 -0.080713751 -0.55316955 -0.36413257
## V6 -0.106544272 -0.08471644 -0.30261295  0.924761327  0.02624505  0.07815282
## V7 -0.008311823  0.01251881  0.85807725  0.273439017 -0.40759883  0.14540916
```

```
## V8 -0.170863076 -0.59545725  0.08224573  0.037461826 -0.09890576 -0.76721056
## V9  0.157563628 -0.41386395 -0.34598578 -0.105251099 -0.70591173  0.36051774
##            PC7          PC8
## V2  0.39357528 -0.671954679
## V3  0.21333394  0.723787756
## V4  0.70495371  0.153957023
## V5 -0.46869413 -0.004193061
## V6 -0.16708218 -0.000455977
## V7 -0.03775672  0.001591472
## V8  0.09478381  0.026274605
## V9  0.21125153 -0.013955335
```

Observations from the above data

Standard Deviations:

The standard deviations for the original variables (V2 through V9) range from approximately 0.6376 to 1.3469. These values indicate the variability or spread of the data along each principal component (PC) axis.

Rotation Matrix:

The rotation matrix represents the loadings of each original variable on the principal components. Each row corresponds to an original variable (V2 through V9), and each column represents a principal component (PC1 through PC8). Larger absolute values in the rotation matrix indicate stronger contributions of the corresponding original variable to the principal component.

Interpretation:

PC1: V2, V3, V5, and V8 have relatively higher loadings, suggesting that they contribute more to the variance captured by PC1. PC2: V4, V5, V6, and V9 have higher loadings, indicating their importance in capturing variance along PC2. PC3: V7 has the highest loading, suggesting it contributes significantly to the variance along PC3. PC4 to PC8: Each subsequent PC captures decreasing amounts of variance, with varying contributions from different variables.

Explained Variance:

The variance explained by each principal component can be obtained from the squared standard deviations. PC1 likely explains the most variance since it has the largest standard deviation, followed by PC2, and so on. Cumulatively, the explained variance of all principal components can be calculated to assess how much of the total variance in the data is captured.

The summary of PCA is given below

```
summary(yeast.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5    PC6     PC7
## Standard deviation     1.3469 1.1271 1.0106 0.9970 0.9671 0.8956 0.86969
## Proportion of Variance 0.2268 0.1588 0.1277 0.1243 0.1169 0.1003 0.09455
## Cumulative Proportion  0.2268 0.3856 0.5132 0.6375 0.7544 0.8546 0.94918
##                           PC8
## Standard deviation     0.63762
## Proportion of Variance 0.05082
## Cumulative Proportion  1.00000
```

```
yeast.pca$sdev
```

```
## [1] 1.3469070 1.1270872 1.0105646 0.9970301 0.9670743 0.8955673 0.8696945
## [8] 0.6376238
```

interpreting these metrics, we can derive the following insights:

1. PC1 explains the largest proportion of variance (22.68%) and has the highest standard deviation (1.3469).

2.PC2 also captures a substantial amount of variance (15.88%), followed by PC3 (12.77%) and PC4 (12.43%).

3.Together, the first four principal components (PC1 to PC4) account for over 63% of the total variance in the data.

4.As we add more components, the cumulative proportion of explained variance increases, reaching 100% when all eight components are included.

Based on this information, a common approach might be to retain the first few principal components that capture a significant portion of the total variance while discarding those that contribute less. In this case, PC1 to PC4 seem particularly important, as they collectively explain over 63% of the variance, making them potentially valuable for data reduction or analysis

```
screeplot(yeast.pca, type="lines")
```

the above figure shows the amount of variance explained by Principal Components where PCA is applied to all variables.

The screen plot of PCA which is given above represents first 6 Principal components explains around 85% of variation in data which would be enough for the Classification and Statistical Analysis.

first 4 Principal components explains around 63% of variation in data. we can also take 4 components which would also be enough for further calculation. it preservers around 63% of variance

```
(yeast.pca$sdev)^2
```

```
## [1] 1.8141585 1.2703256 1.0212408 0.9940690 0.9352328 0.8020408 0.7563685
## [8] 0.4065641
```

```
sum((yeast.pca$sdev)^2)
```

```
## [1] 8
```

This contains a matrix with the loadings of each principal component, where the first column in the matrix contains the loadings for the first principal component, the second column contains the loadings for the second principal component, and so on.

```
yeast.pca$rotation[,1]
```

```
##           V2           V3           V4           V5           V6           V7
## -0.609203749 -0.633179046  0.355869732 -0.189395266 -0.106544272 -0.008311823
##           V8           V9
## -0.170863076  0.157563628
```

Another way to approach the computation of the first principal component is by creating a custom function that computes it based on the loadings and the values of the input variables

```
calcpc <- function(variables,loadings)
  {
     # find the number of samples in the data set
     as.data.frame(variables)
     numsamples <- nrow(variables)
     # make a vector to store the component
     pc <- numeric(numsamples)
     # find the number of variables
     numvariables <- length(variables)
     # calculate the value of the component for each sample
     for (i in 1:numsamples)
     {
        valuei <- 0
        for (j in 1:numvariables)
        {
           valueij <- variables[i,j]
           loadingj <- loadings[j]
           valuei <- valuei + (valueij * loadingj)
        }
        pc[i] <- valuei
     }
     return(pc)
  }
```

We can then use the function to calculate the values of the first principal component for each sample in our yeast data:

```
head(calcpc(standardisedconcentrations, yeast.pca$rotation[,1]))
```

```
## [1] -0.8720424 -0.8132078 -1.2826824  0.2297805  0.1811439  0.7413331
```

the values of the first principal component are stored in the variable yeast.pca$x[,1] that was returned by the "prcomp()" function, so we can compare those values to the ones that we calculated

```
head(yeast.pca$x[,1])
```

```
## [1] -0.8720424 -0.8132078 -1.2826824  0.2297805  0.1811439  0.7413331
```

Similarly, we can obtain the loadings for the second principal component

```
yeast.pca$rotation[,2]
```

```
##          V2          V3          V4          V5          V6          V7
##  0.12825311  0.04252125  0.43740649  0.50718077 -0.08471644  0.01251881
##          V8          V9
## -0.59545725 -0.41386395
```

Scatterplots of the Principal Components

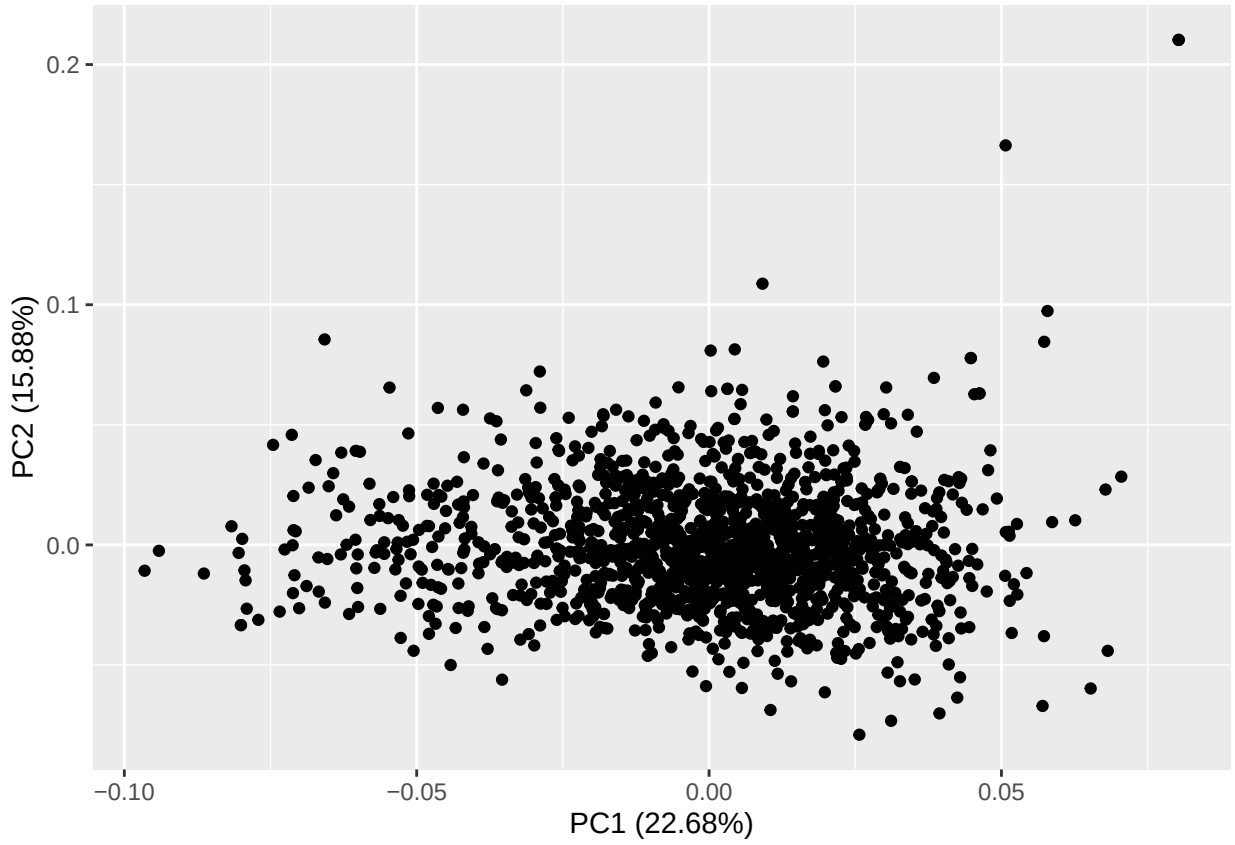
we can plot the principal conmponent of principal components like this

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.3.3
```

```
df <- yeast[2:9]
pca_res <- prcomp(df, scale. = TRUE)

autoplot(pca_res)
```




```
library(rgl)
```

```
## Warning: package 'rgl' was built under R version 4.3.3
```

```
scores = as.data.frame(yeast.pca$x)

plot3d(scores[,1:3],
       size=5,
       col = seq(nrow(scores)))
```

```
text3d(scores[,1:3],
       texts=c(rownames(scores)),
       cex= 0.7, pos=3)
```

3d plot is not visible here. we can use R to see 3 d graph between PCA1, PCA2 and PCA3.

Conclusion of PCA analysis:

Significant Variance Explained: The first four principal components (PC1 to PC4) collectively explain over 63% of the total variance in the data. This indicates that these components capture the most significant patterns and variability within the dataset.

Incremental Variance Explanation: As more principal components are considered, the cumulative proportion of explained variance increases, reaching 100% when all eight components are included. However, the additional variance explained by including more components diminishes compared to the initial ones.

Selection of Components: A common approach in PCA is to retain principal components that capture a substantial portion of the total variance while discarding less informative components. In this case, selecting the first four principal components seems reasonable, as they capture a significant amount of variance.

Scree Plot Analysis: The scree plot visually confirms the findings from the PCA results, showing that the first six principal components explain around 85% of the variation in the data. This suggests that these six components might be sufficient for analysis purposes.