Data description (and data preprocessing), PCA

Lúa Arconada Manteca, Nuria Errasti Domínguez, Alejandro Macías Pastor

2023-11-17

The purpose of this assignment is to reduce the dimension of our dataset, in other words, we want to capture the information contained in the dataset with fewer variables. For this task we are going to use a database obtained from the UCI Irvine Machine Learning Directory (https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease). It consists of a dataset with 25 variables and 400 observations. It contains 11 numeric variables and 14 nominal ones (both binary and multinomal) that can be used to predict chronic kidney disease.

Before starting with the analysis, we need to load the required packages.

```r
library(pander)
library(knitr)
library(latex2exp)
library(graphics)
library(kableExtra)
library(calibrate)
library(e1071)
library(Matrix)
library(ggplot2)
library(GGally)
library(car)
library(dplyr)
library(corrplot)
```

To start with, we load the dataset, add names to the different variables and omit the NA's.

```r
names = c('age', 'bp','sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',
'bu',
          'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm',
'cad',
          'appet', 'pe', 'ane', 'class')
data =read.table("chronic_kidney_disease.csv", sep=';', na.strings='?',
                 header=F, col.names=names)

data =na.omit(data)
```

The variables indicate the following:

- age - age in years
- bp - blood pressure in mm/Hg

- `sg` - specific gravity (1.005,1.010,1.015,1.020,1.025)
- `al` - albumin (0,1,2,3,4,5)
- `su` - sugar (0,1,2,3,4,5)
- `rbc` - red blood cells (normal,abnormal)
- `pc` - pus cell (normal,abnormal)
- `pcc` - pus cell clumps (present, not present)
- `ba` - bacteria (present, not present)
- `bgr` - blood glucose random in mgs/dl
- `bu` - blood urea in mgs/dl
- `sc` - serum creatinine in mgs/dl
- `sod` - sodium in mEq/L
- `pot` - potassium in mEq/L
- `hemo`- hemoglobin in gms
- `pcv` - packed cell volume
- `wc` - white blood cell count in cells/cmm
- `rc` - red blood cell count in millions/cmm
- `htn` - hypertension (yes, no)
- `dm` - diabetes mellitus (yes, no)
- `cad` - coronary artery disease (yes, no)
- `appet` - appetite (good, poor)
- `pe` - pedal edema (yes, no)
- `ane` - anemia (yes, no)
- `class` - class (chronic kidney disease,not chronic kidney disease)

Since this assignment consists of data description, data preprocessing and PCA, we are only going to use the numeric variables. Therefore, we create a new dataset with only those variables (`age`, `bgr`, `bu`, `sc`, `sod`, `pot`, `hemo`, `pvc`, `wc` and `rc`). We are not including variable `bp` since it is discrete and includes few values.

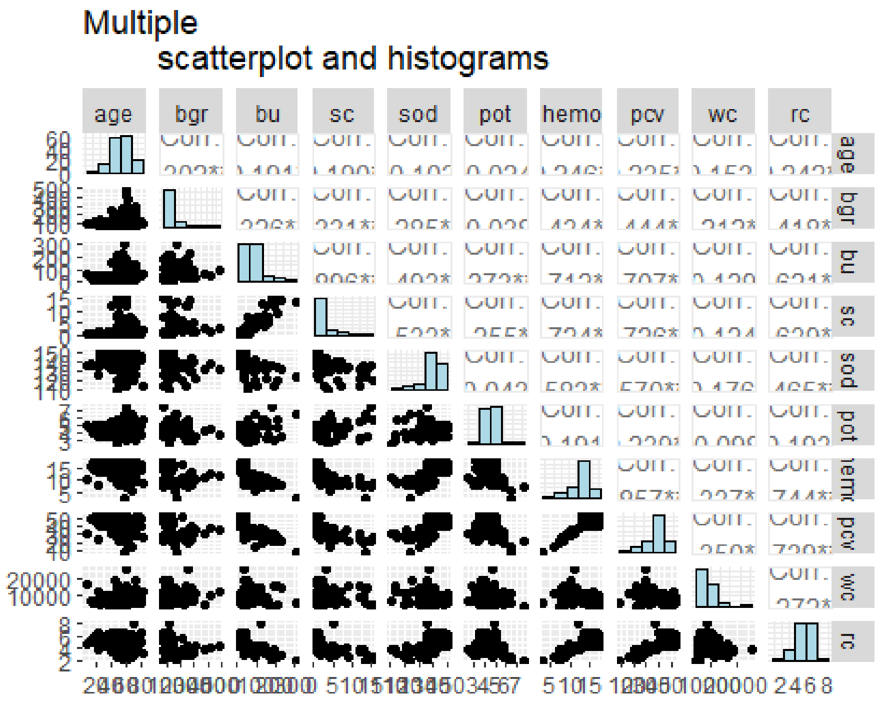
```r
numericdata<- data[c(1,10:18)]
pander(head(numericdata), row.names=FALSE)
```

age	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc
48	117	56	3.8	111	2.5	11.2	32	6700	3.9
53	70	107	7.2	114	3.7	9.5	29	12100	3.7
63	380	60	2.7	131	4.2	10.8	32	4500	3.8
68	157	90	4.1	130	6.4	5.6	16	11000	2.6
61	173	148	3.9	135	5.2	7.7	24	9200	3.2
48	95	163	7.7	136	3.8	9.8	32	6900	3.4

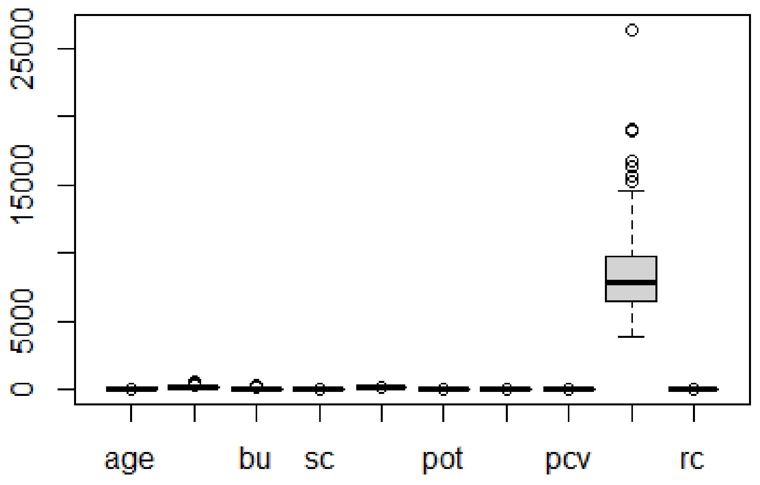
DATA VISUALIZATION AND TRANSFORMATION (SIMMETRY AND STANDARIZATION)

In order to perform PCA on our dataset we first need to understand and visualize it. Therefore, we start by computing scatter plots, histograms and boxplots.

```
diagonal_hist <- function(data, mapping) {
  ggplot(data = data, mapping = mapping) +
    geom_histogram(fill = "lightblue", color = "black", bins =5 ) +
    theme_minimal()
}
ggpairs(numericdata, diag = list(continuous = diagonal_hist), title =
"Multiple
        scatterplot and histograms", upper = list('cor', fontsize=5))
```



```
boxplot(numericdata)
```

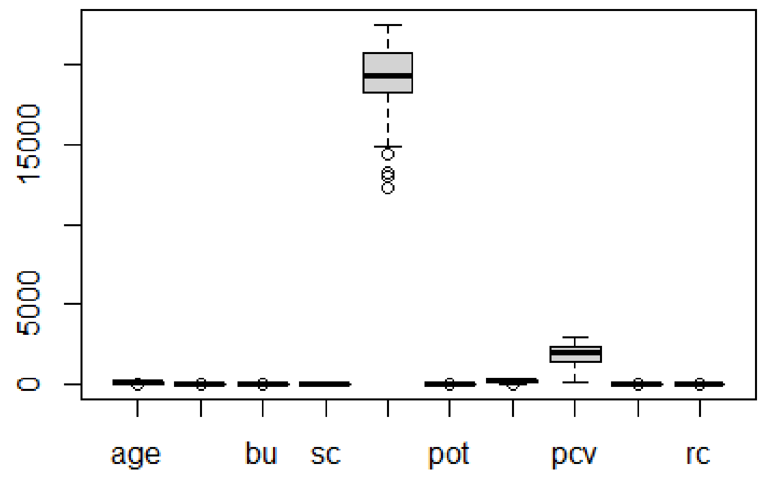


Next, we study the skewness of the variables. Since most of them are not symmetrical we apply transformations to those variables. Those that have excessively positive skewness will be transformed with the power -1 and those that have excessively negative skewness will be squared.

```
pander(apply(numericdata, MARGIN=2, FUN=skewness))
```

age	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc
-0.3222	2.688	2.571	2.697	-1.136	0.4893	-0.9711	-1.005	2.031	-0.3802

```
numericdata[,c(2,3,4,9)]<-1/(numericdata[,c(2,3,4,9)])
numericdata[,c(5,7,8)]<-(numericdata[,c(5,7,8)])^2
boxplot(numericdata)
```



```
pander(apply(numericdata, MARGIN=2, FUN=skewness))
```

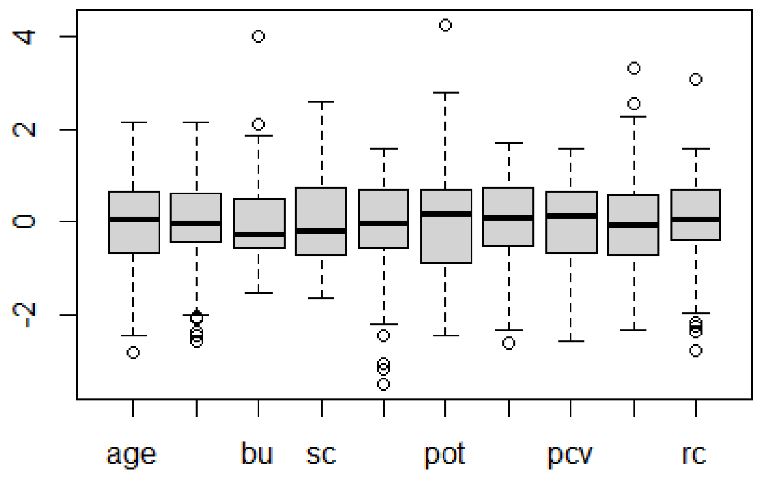
Table continues below

age	bgr	bu	sc	sod	pot	hemo	pcv
-0.3222	-0.3203	1.154	0.1793	-0.9045	0.4893	-0.4838	-0.5129

wc	rc
0.4855	-0.3802

Now, the variables are more symmetric but our problems do not end here. Since the variables have different scales (some of them have values between 0 and 15 while others are around the thousands) we are going standardize them.

```
numericdata<-as.data.frame(scale(numericdata,center=TRUE,scale=TRUE))
boxplot(numericdata)
```



Let’s see now the sample covariance matrix and the sample correlation matrix.

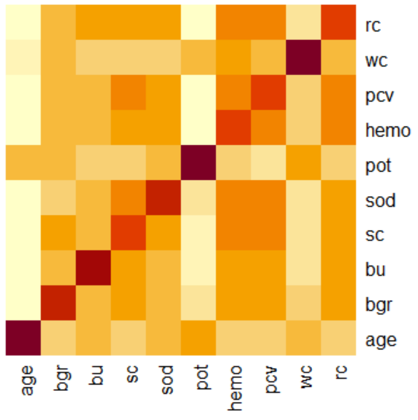
```
S<-cov(numericdata)
S

##               age         bgr          bu         sc         sod
pot
## age   1.00000000 -0.31736415 -0.16907027 -0.2643181 -0.09747152 -
0.02418725
## bgr  -0.31736415  1.00000000  0.25923986  0.4796316  0.24032586 -
0.02736274
## bu   -0.16907027  0.25923986  1.00000000  0.4224052  0.34349066 -
0.14285523
## sc   -0.26431805  0.47963159  0.42240524  1.0000000  0.53171826 -
0.13446826
## sod  -0.09747152  0.24032586  0.34349066  0.5317183  1.00000000
0.03677428
## pot  -0.02418725 -0.02736274 -0.14285523 -0.1344683  0.03677428
1.00000000
## hemo -0.24498927  0.43456292  0.45485101  0.6137707  0.56141857 -
0.15261291
## pcv  -0.23394985  0.46160197  0.43814788  0.6500094  0.54624432 -
0.21188926
## wc   -0.14182524  0.11517842  0.06249647  0.1040663  0.04358823
0.13537317
## rc   -0.24223469  0.41430843  0.44702215  0.5286104  0.45967013 -
0.19296605
##            hemo        pcv          wc         rc
## age  -0.2449893 -0.2339498 -0.14182524 -0.2422347
## bgr   0.4345629  0.4616020  0.11517842  0.4143084
## bu    0.4548510  0.4381479  0.06249647  0.4470221
## sc    0.6137707  0.6500094  0.10406625  0.5286104
## sod   0.5614186  0.5462443  0.04358823  0.4596701
## pot  -0.1526129 -0.2118893  0.13537317 -0.1929660
## hemo  1.0000000  0.7872312  0.25356284  0.7090112
## pcv   0.7872312  1.0000000  0.22391478  0.7046015
## wc    0.2535628  0.2239148  1.00000000  0.1300432
## rc    0.7090112  0.7046015  0.13004321  1.0000000
```

```
R<-cor(numericdata)
R
```

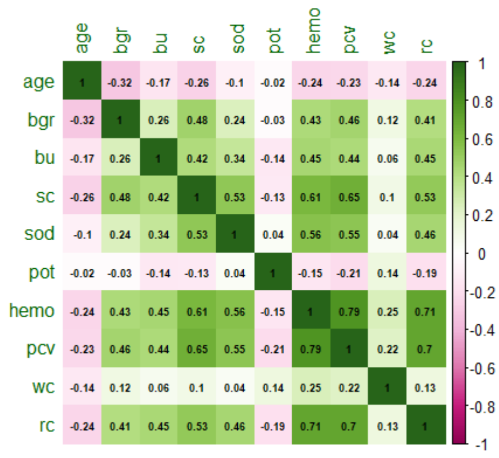
```
##              age         bgr          bu          sc         sod
pot
## age   1.00000000 -0.31736415 -0.16907027 -0.2643181 -0.09747152 -
0.02418725
## bgr  -0.31736415  1.00000000  0.25923986  0.4796316  0.24032586 -
0.02736274
## bu   -0.16907027  0.25923986  1.00000000  0.4224052  0.34349066 -
0.14285523
## sc   -0.26431805  0.47963159  0.42240524  1.0000000  0.53171826 -
0.13446826
## sod  -0.09747152  0.24032586  0.34349066  0.5317183  1.00000000
0.03677428
## pot  -0.02418725 -0.02736274 -0.14285523 -0.1344683  0.03677428
1.00000000
## hemo -0.24498927  0.43456292  0.45485101  0.6137707  0.56141857 -
0.15261291
## pcv  -0.23394985  0.46160197  0.43814788  0.6500094  0.54624432 -
0.21188926
## wc   -0.14182524  0.11517842  0.06249647  0.1040663  0.04358823
0.13537317
## rc   -0.24223469  0.41430843  0.44702215  0.5286104  0.45967013 -
0.19296605
##            hemo        pcv          wc         rc
## age  -0.2449893 -0.2339498 -0.14182524 -0.2422347
## bgr   0.4345629  0.4616020  0.11517842  0.4143084
## bu    0.4548510  0.4381479  0.06249647  0.4470221
## sc    0.6137707  0.6500094  0.10406625  0.5286104
## sod   0.5614186  0.5462443  0.04358823  0.4596701
## pot  -0.1526129 -0.2118893  0.13537317 -0.1929660
## hemo  1.0000000  0.7872312  0.25356284  0.7090112
## pcv   0.7872312  1.0000000  0.22391478  0.7046015
## wc    0.2535628  0.2239148  1.00000000  0.1300432
## rc    0.7090112  0.7046015  0.13004321  1.0000000
```

The matrices have rank 10, therefore there are no linearly dependent variables. We can also compute the heatmap of our variables. The more intense the colour is, the more intercorrelated those variables are. Moreover, we are also going to compute a correlation plot.

```r
heatmap(R,Colv=NA,Rowv=NA)
```



```r
corrplot(R, method="color",  col = COL2('PiYG'), tl.col = "darkgreen",
         addCoef.col ='black',number.cex = 0.55)
```



Let's compute now some overall intercorrelation measures using the eigenvalues of the sample correlation matrix. The closer the value is to 1, the greater the amount of intercorrelation between variables.

```r
intercorrelation <- function(X){
  q<-rep(0,6)
  p<-ncol(X)
  R<-cor(X)
  q[1]<-(1-min(eigen(R)$value)/(max(eigen(R)$value)))^(p+2)
  q[2]<- 1-p/(sum(1/eigen(R)$value))
  q[3]<-1-sqrt(abs(det(R)))
  q[4]<-(max(eigen(R)$value)/p)^(3/2)
  q[5]<-(1-min(eigen(R)$value)/p)^5
  R_1=solve(R)
  q[6]<-sum((1-(1/diag(R_1)))/p)
  cbind(c("q1", "q2", "q3", "q4", "q5", "q6"), q)
}

kable(intercorrelation(numericdata))
```

	q
q1	0.550505372161797
q2	0.48797614147615
q3	0.859231019478683
q4	0.277649425255441
q5	0.900915086484089
q6	0.399075203586763

As mentioned in the beginning, only the numerical variables can be used for PCA, so the categorical ones have been mostly ignored. Although not the focus in this project, it could be interesting to group the numerical variables according to said categories and compare between different groups. The graph below shows and example of the difference in intercorrelations when the numerical data is split according to the variable appet.

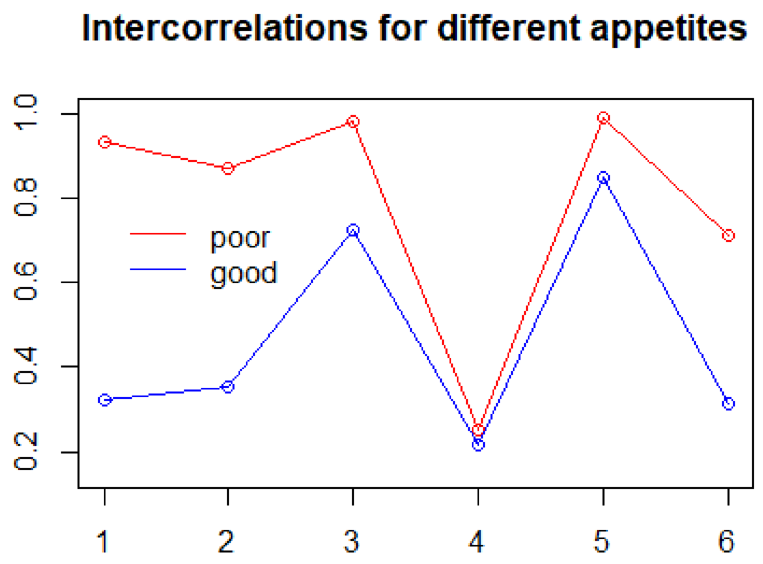
```
appet<-split(numericdata,data$appet)
poor<-appet$poor
good<-appet$good
interpoor<-intercorrelation(poor)
kable(interpoor)
```

	q
q1	0.932871878980613
q2	0.872284091760635
q3	0.98281193413251
q4	0.249678242098102
q5	0.988605288876243
q6	0.712910386545258

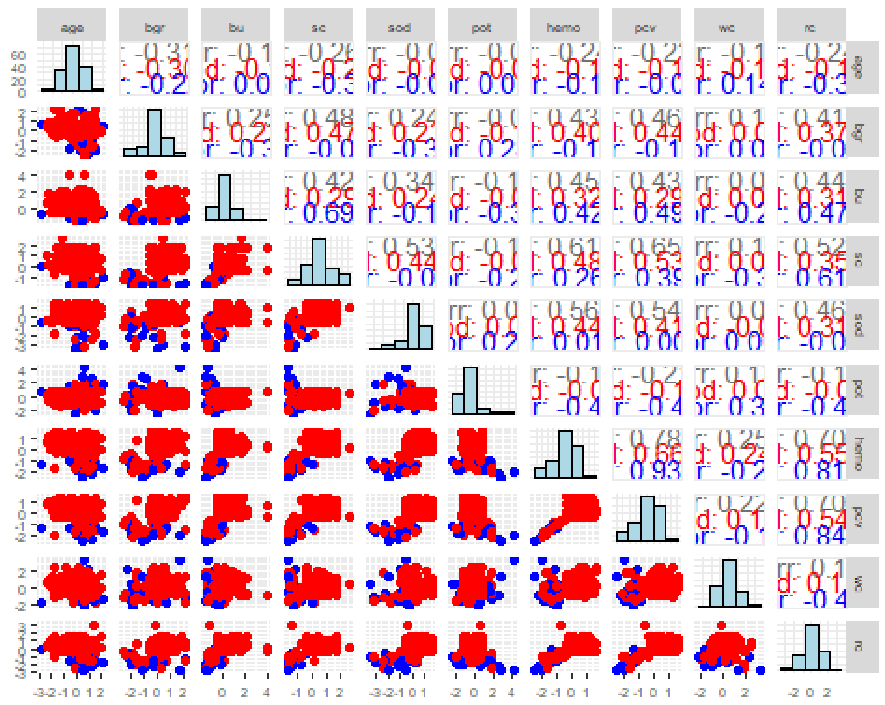
```
intergood<-intercorrelation(good)
kable(intergood)
```

	q
q1	0.324144766871978
q2	0.354730219488259
q3	0.726550014061951
q4	0.214327046666936
q5	0.849512693421607
q6	0.311988679218254

```r
plot(1:6,interpoor[,2],col='red',ylim=c(0.15,1), xlim=c(1,6), xlab='',
ylab='',
     main='Intercorrelations for different appetites')
lines(1:6,interpoor[,2],col='red')
points(1:6,intergood[,2],col='blue')
lines(1:6,intergood[,2],col='blue')
legend(1,0.8,legend=c('poor','good'),col=c('red','blue'),lty=1:1,cex=1,bt
y='n')
```



```r
ggpairs(numericdata,
        diag = list(continuous = diagonal_hist),
        aes(color = data$appet))+
  scale_color_manual(values = c("red", "blue"))+
  theme(text = element_text(size = 7))
```



PRINCIPAL COMPONENT ANALYSIS (PCA)

As mentioned before, our aim is to reduce the dimension of our dataset. Since we have intercorrelated variables we should be able to replace them with a smaller number of new variables. Since our variables are not commensurate we are going to use the correlation matrix to compute the PCA. We have already centered our data and standardized it. The following step is computing the eigenvalues and eigenvectors of the sample correlation matrix R.

```
eigenlista<-eigen(R)
lambda<-eigenlista$values
lambda

##  [1] 4.2559631 1.2147874 0.9782375 0.8946471 0.7003015 0.6222941
0.4922552
##  [8] 0.3616883 0.2732999 0.2065261

V<-eigenlista$vectors
pander(V)
```

Table continues below

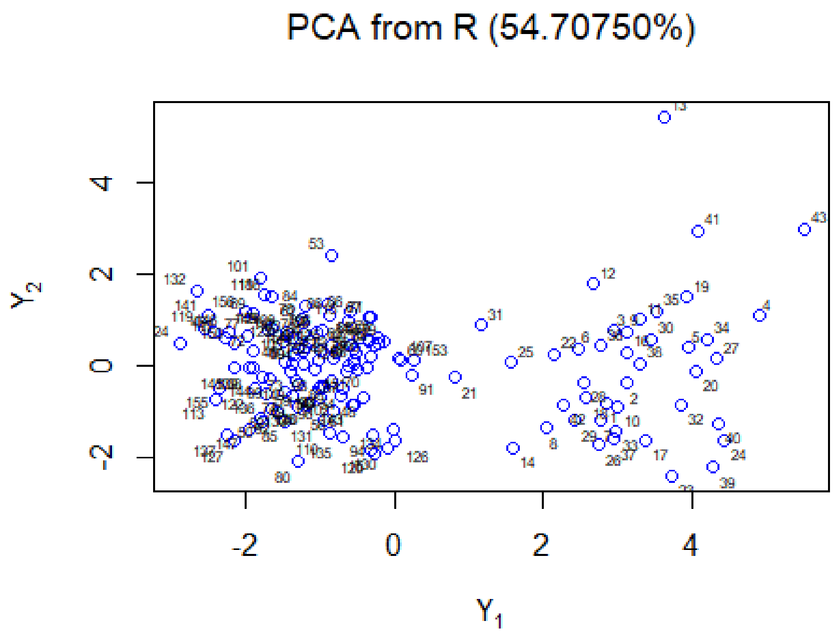
0.1813	-0.3938	0.6367	-0.2455	-0.3081	0.4951	0.02201
-0.2942	0.1945	-0.3464	0.2474	-0.4555	0.5949	-0.02265
-0.2943	-0.1627	0.02561	0.02966	0.8024	0.4773	-0.06666
-0.3859	-0.0334	0.01413	0.1679	-0.1323	0.01701	-0.5691

```
 -0.3224    -0.04981    0.4954     0.2761    -0.01633    -0.3514    -0.2797
 0.09269     0.6473     0.4485     0.4483     0.08877     0.1434     0.2666
 -0.425     0.001165    0.1137    -0.1428    -0.06481    -0.1115     0.2117
 -0.4276    -0.05094    0.06222   -0.1419    -0.1471     -0.08828    0.1136
 -0.115      0.5883     0.1021    -0.7242     0.04374     0.06121   -0.2308
 -0.3947    -0.09671   -0.001117  -0.08522   -0.02036    -0.06977    0.6406
```

```
 -0.06693   -0.03093    0.02091
  0.3626    -0.02203    0.02359
  0.07845    0.04401   -0.03551
 -0.6462    -0.2294     0.09836
  0.5792    -0.1837    -0.02744
 -0.2483     0.08128   -0.05055
 -0.03723    0.5159     0.676
 -0.1214     0.4538    -0.7264
  0.1123    -0.1846    -0.009848
 -0.1203    -0.6304    -0.005313
```

In order to visualize our results, we are going to keep only the first two components, even though it explains little variability.

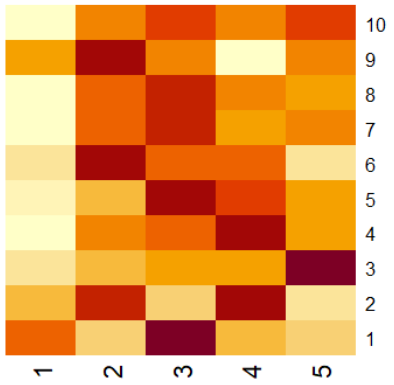
```r
matriz<-as.matrix(numericdata)
Y<-matriz %*% V
plot(Y[,1],Y[,2],xlab=TeX('$Y_1$'),ylab=TeX('$Y_2$'),
     main=TeX('PCA from $R$ (54.70750%)'),col='blue')
textxy(Y[,1],Y[,2],labs=1:156,cex=0.5,offset=1)
```



METHODS FOR DISCARDING COMPONENTS

After computing the principal components we need to decide how many to keep and how many we are going to discard. This can be done following different criteria.

```
heatmap(V[,1:5],Colv=NA,Rowv=NA)
```



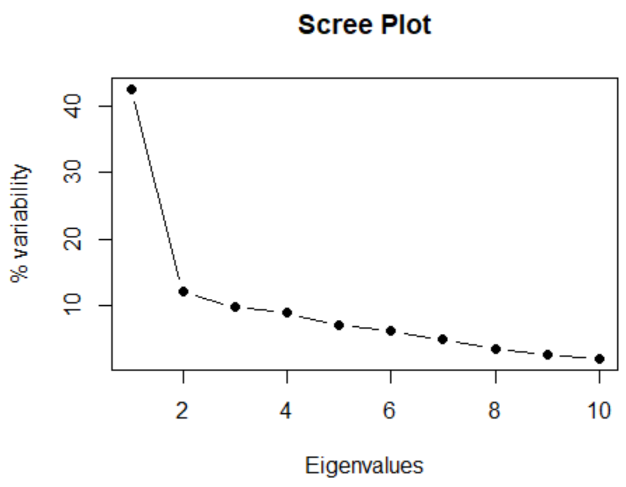
1. Percentage of explained variability

We can now compute the percentage of explained variability by each component and the cumulative explained variability. Following this criterion we should keep enough components such that between 70% and 90% of the variability is explained.

```
varexpm<-lambda*(100/sum(lambda))
varexp<-diag(varexpm)
varexpm

##  [1] 42.559631 12.147874  9.782375  8.946471  7.003015  6.222941
4.922552
##  [8]  3.616883  2.732999  2.065261

varexpacum<-cumsum(varexp)
vectorcum<- varexpacum[c(10,20,30,40,50,60,70,80,90,100)]
vectorcum

##  [1]  42.55963  54.70750  64.48988  73.43635  80.43937  86.66231
91.58486
##  [8]  95.20174  97.93474 100.00000
```

We can see that the first 5 components explain 80.44% of the variability. The same can also be seen in the following scree graph.

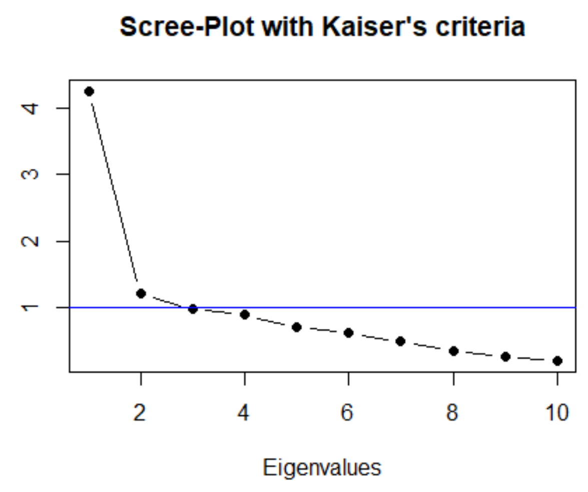
```
plot(1:length(lambda), diag(varexp), type = "b", pch = 19, xlab =
"Eigenvalues",
     ylab = "% variability", main = "Scree Plot")
```



2. Kaiser's criterion

We are going to exclude those components whose eigenvalues are smaller than $\bar{\lambda}$ (the mean of the eigenvalues) or that are smaller than $\lambda = 1$. Therefore, we would be keeping the first two principal components.

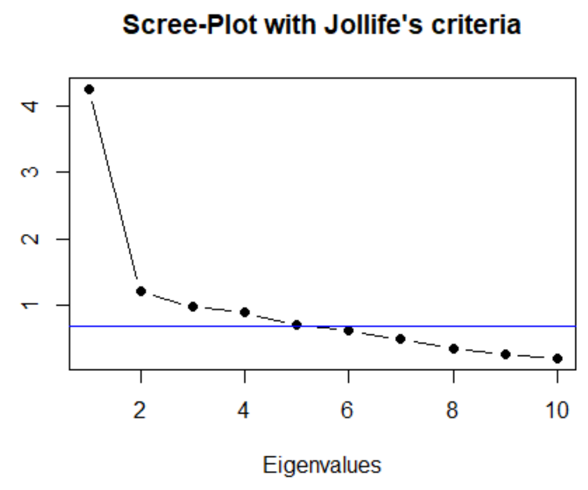
```
plot(1:length(lambda), lambda, type = "b", pch = 19, xlab =
"Eigenvalues",
     ylab='', main = "Scree-Plot with Kaiser's criteria")
abline(h = mean(lambda), col = "red")
abline(h=1, col="blue")
```



3. Jollife's criterion

This criterion will be more useful than the previous one since we have a small number of variables (less than 20). It takes into account those components whose eigenvalues are bigger than $0.7\bar{\lambda}$ or bigger than $\lambda = 0.7$. Therefore, we would be keeping the first 5 principal components.

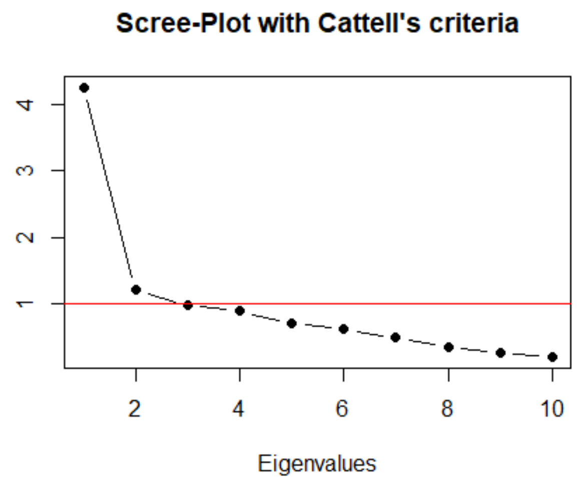
```r
plot(1:length(lambda), lambda, type = "b", pch = 19, xlab =
"Eigenvalues",
     ylab='',  main = "Scree-Plot with Jollife's criteria")
abline(h=0.7*mean(lambda), col = "red")
abline(h=0.7, col="blue")
```



4. Cattell's scree graph

Following this criterion we are going to keep the first $q < p$ components with the steepest slopes. That is, looking at our graph, we would be keeping the first 2 principal components.

```
plot(1:length(lambda), lambda, type = "b", pch = 19, xlab =
"Eigenvalues",
     ylab='', main = "Scree-Plot with Cattell's criteria")
abline(h=1, col = "red")
```



5. Dimensionality test

This criterion only applies when the covariance matrix is used for the PCA. As the aspect of our data favored the use of the correlation matrix, this does not apply.

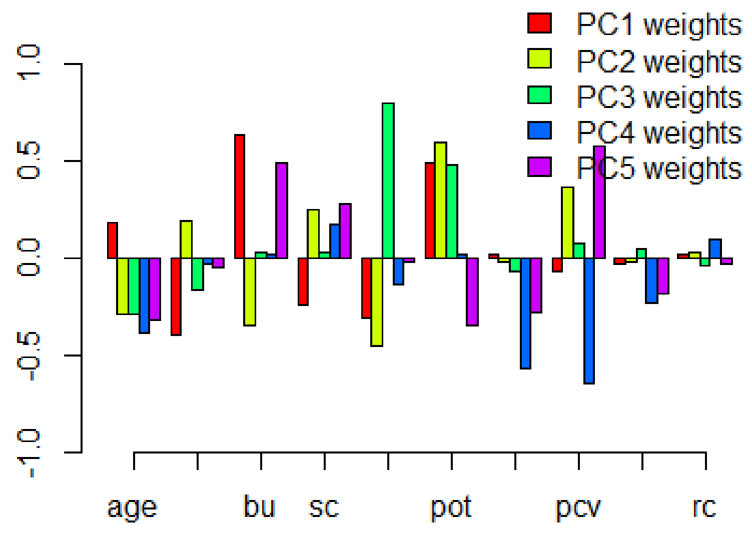
Criteria conclusion

Since we have a dataset with free variables, Jollife's criterion applies best. We can further analyse the five principal components selected with this criterion by looking at their weights:

```
varexpm<-lambda*(100/sum(lambda))
varexp<-diag(varexpm)
varexpacum<-cumsum(varexp)
vectorcum<- varexpacum[c(10,20,30,40,50,60,70,80,90,100)]
summary<-data.frame(rbind(lambda,diag(varexp),vectorcum),
                   row.names=c('eigenvalues','% variability',
                               'cumulated variability'))
colnames(summary)<-
c('$Y_1$','$Y_2$','$Y_3$','$Y_4$','$Y_5$','$Y_6$','$Y_7$',
                   '$Y_8$','$Y_9$','$Y_{10}$')
kable(summary, digits=2)
```

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}
eigenvalues	4.26	1.21	0.98	0.89	0.70	0.62	0.49	0.36	0.27	0.21
% variability	42.56	12.15	9.78	8.95	7.00	6.22	4.92	3.62	2.73	2.07
cumulated variability	42.56	54.71	64.49	73.44	80.44	86.66	91.58	95.20	97.93	100.00

```r
names=c('PC1 weights','PC2 weights','PC3 weights','PC4 weights','PC5
weights')
barplot(V[1:5,],beside=TRUE, col = rainbow(5),legend.text = names,
        args.legend = list(x = "topright", inset=c(-0,-0.2), y=0.7, bty =
"n"),
        ylim=c(-1,1), axis.lty=1,
        names.arg=names(numericdata))
```



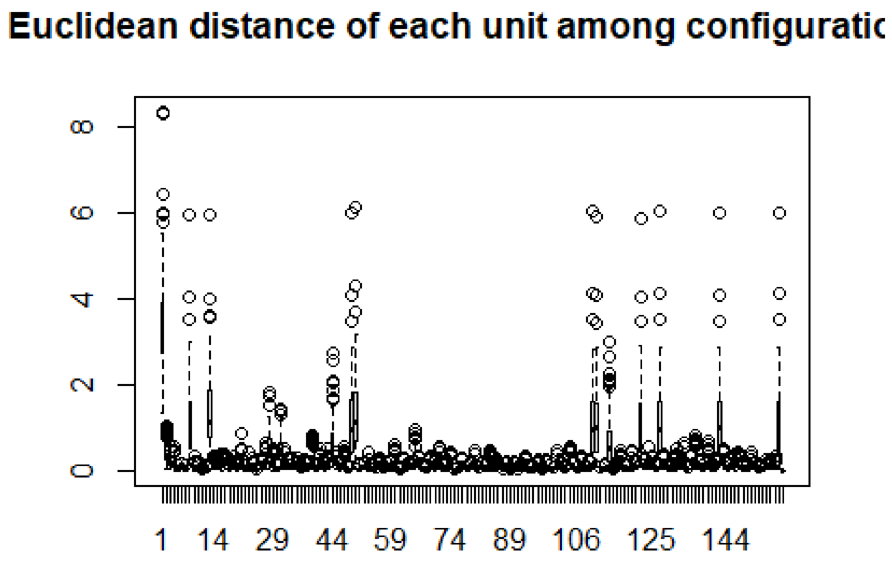
Looking at the first component, we see that the variables ‘bu’ and ‘pot’ have the greater contribution, so if this principal component is high it is because that person has high blood urea and potassium. In the second component, this happens with the variables ‘sod’ and ‘pot’ so it represents people with high scores in those variables. In the third component, the ones with the greater contribution are ‘sod’ and ‘pot’ again’, but being the sign of the second one negative, so this represents people with high sodium and low potassium and high potassium and low sodium.

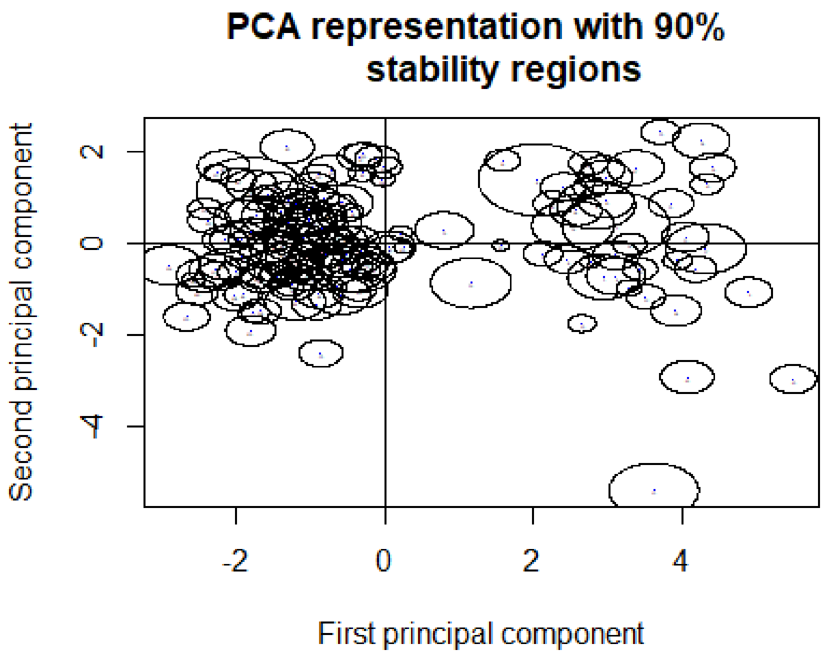
Now, in the fourth principal component the ones with greater contribution are new variables, ‘hemo’ and ‘pcv’, both with negative sign, so it represents people with high hemoglobin and packed cell volume. Finally, the fifth component has ‘sod’ and ‘pot’ once again as the variables with the greater contribution with 'sod' having a negative sign, so it conrresponds to individuals with high sodium and low potassium and viceversa.

ANALYSIS OF STABILITY

Finally, we need to assess the stability and robustness of our PCA configuration. To do so we are going to analyze the contribution of each observation to the final model. We are going to study our PCA discarding components one at a time, that is, we are going to study the PCA configuration without the first component, then we will only discard the second and so on. Then we will compare this new configurations to the original one by using the distance of each observation in our original PCA to its equivalents in the new ones.

```r
PCAcovstability <- function(X) {
  n <- nrow(X)
  p <- ncol(X)
  H <- diag(n) - matrix(1, n, n) / n
  X <- H %*% X
  S <- cov(X)
  eigen_result <- eigen(S)
  V <- eigen_result$vectors[,order(-eigen_result$values)]

# Control of the signs of the components
for (j in 1:p) {
  if (V[1, j] < 0) {
    V[, j] <- -V[, j]
    }
  }

Y <- X %*% V

# Leave-one-out
v <- matrix(0, n, n)
for (i in 1:(n-1)) {
  Xi <- X[-i, , drop = FALSE]
  Si <- cov(Xi)
  eigen_result_i <- eigen(Si)
  Vi <- eigen_result_i$vectors[,order(-eigen_result_i$values)]
  # Control of the signs of the components
  for (j in 1:p) {
    if (Vi[1, j] < 0) {
      Vi[, j] <- -Vi[, j]
    }
  }

  Yi <- Xi %*% Vi
  y <- X[i, ] %*% Vi
  Yplusi <- rbind(Yi[1:(i-1), , drop = FALSE], y, Yi[(i):(n-1), , drop =
FALSE])

  # Euclidean distance of each unit among configurations
  for (k in 1:n) {
    v[k, i] <- sum((Yplusi[k, , drop = FALSE] - Y[k, , drop = FALSE])^2)
    }
}

# i-th row in matrix v contains the configuration for the i-th unit
v <- sqrt(v)

# Summary statistics for the rows of v
Rad <- matrix(0, n, 6)
for (i in 1:n) {
  Rad[i, 1] <- min(v[i, ])
  Rad[i, 2] <- max(v[i, ])
  Rad[i, 3] <- quantile(v[i, ], 0.50)
  Rad[i, 4] <- quantile(v[i, ], 0.75)
  Rad[i, 5] <- quantile(v[i, ], 0.90)
  Rad[i, 6] <- quantile(v[i, ], 0.95)
}

# Some summary statistics by rows
rowsummary <- rbind(mean(Rad), sd(Rad), apply(Rad, 2, median),
                    apply(Rad, 2, mad))
rowsummary<- as.matrix(rowsummary)
colnames(rowsummary)<- c("min", "max", "median", "75th-p", "90-th p",
"95-th p")
rownames(rowsummary)<- c("mean", "median", "std", "mad")

# PCA unit stability
boxplot(v, main='Euclidean distance of each unit among configurations')

lab <- sprintf('%3g', 1:n)
radius <- Rad[, 5]  # it could also be Rad[, 6]

theta <- seq(0, 2 * pi, by=0.01)

plot(Y[, 1], Y[, 2], col='blue', pch='.', main="PCA representation with
90%
     stability regions", xlab='First principal component', ylab='Second
principal component')
abline(h=0, v=0, col='black')  # línea base
text(Y[, 1], Y[, 2], labels=lab, cex=0.01)


for (i in 1:n) {
  circle <- matrix(0, length(theta), 2)
  for (j in 1:length(theta)) {
    circle[j, 1] <- Y[i, 1] + cos(theta[j]) * radius[i]
    circle[j, 2] <- Y[i, 2] + sin(theta[j]) * radius[i]
  }
  points(circle[, 1], circle[, 2], col='black', pch='.', cex=0.1)
}
return(rowsummary)
}

PCAcorrstability <- function(X) {
  Z <- scale(X)
  rowsummary <- PCAcovstability(Z)
  return(rowsummary)
}

print(kable(PCAcorrstability(numericdata)))

##
##
## |       |       min|       max|    median|    75th-p|   90-th p|   95-
th p|
## |:------|---------:|---------:|---------:|---------:|---------:|------
---:|
## |mean   | 0.8532617| 0.8532617| 0.8532617| 0.8532617| 0.8532617|
0.8532617|
## |median | 1.3449164| 1.3449164| 1.3449164| 1.3449164| 1.3449164|
1.3449164|
## |std    | 0.0000000| 3.2321595| 0.0840705| 0.1441742| 0.2960716|
0.9544121|
## |mad    | 0.0000000| 0.8594960| 0.0228733| 0.0399472| 0.0930160|
0.6828519|
```





Finding higher values in the Euclidean distance indicates more unstable observations. Thus, from the boxplot above we can conclude that observation number 1 is the most unstable, with a few others also standing out. These lower stabilities are also indicated by having larger 90% stability regions.

REFERENCES

Rubin, L., Soundarapandian, P., and Eswaran,P.. (2015). Chronic_Kidney_Disease. UCI Machine Learning Repository. https://doi.org/10.24432/C5G020.

Grané, A. (2023). Multidimensional datasets. Universidad Carlos III de Madrid.

Härdle, W., Simar, L. Applied Multivariate Statistical Analysis. Springer, Second Edition. New York, USA. 2007.