Dimenziócsökkentési eljárások

Contents

Motiváció

A valóságban előforduló adathalmazok jellemzően nehezen áttekinthetőek, hiszen kettőnél több változót (dimenziót) tartalmaznak, ezért az adathalmazban rejlő összefüggések könnyen rejtve maradhatnak. Az adatok megfelelő prezentálása, a helyes "rálátási szög" meghatározása kulcsfontosságú.

A dimenziócsökkentési eljárások kulcsfontosságúak a sok dimenzióval rendelkező adathalmazok áttekintéséhez. Tekintsük például az alábbi adathalmazt, amely az egyes országok étkezési szokásait jellemzi alapvető élelmiszerek fogyasztásának mérésével (1973-as adatok).

```
data.file <- read.table("protein.txt", sep="\t", header = TRUE)
data <- data.file[,-1]
rownames(data) <- data.file[,1]
head(data)
```

```
##               RedMeat WhiteMeat Eggs Milk Fish Cereals Starch Nuts Fr.Veg
## Albania          10.1       1.4  0.5  8.9  0.2    42.3    0.6  5.5    1.7
## Austria           8.9      14.0  4.3 19.9  2.1    28.0    3.6  1.3    4.3
## Belgium          13.5       9.3  4.1 17.5  4.5    26.6    5.7  2.1    4.0
```

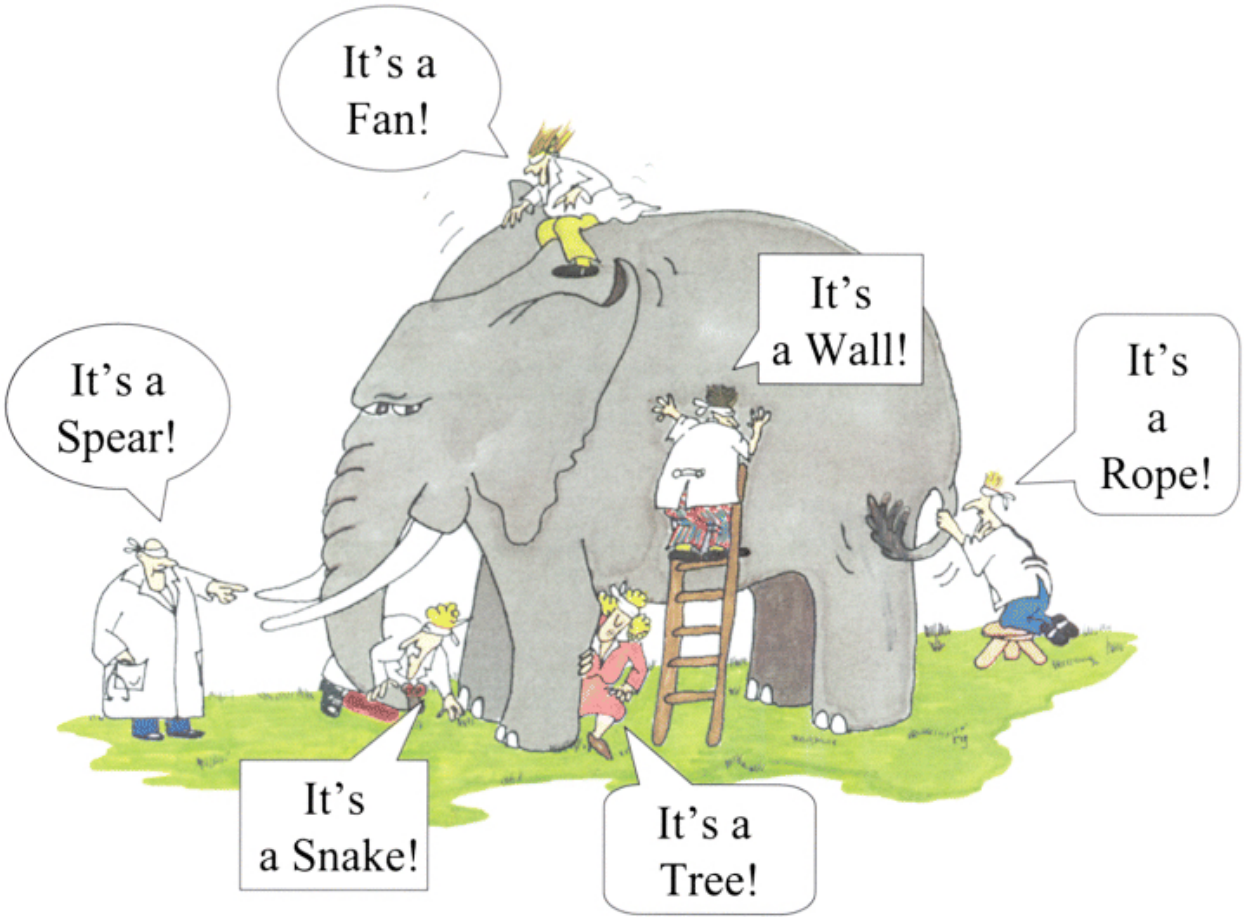


Figure 1: Az elefánt és a vakok esete

```
## Bulgaria           7.8       6.0  1.6  8.3  1.2    56.7    1.1  3.7    4.2
## Czechoslovakia     9.7      11.4  2.8 12.5  2.0    34.3    5.0  1.1    4.0
## Denmark           10.6      10.8  3.7 25.0  9.9    21.9    4.8  0.7    2.4
```

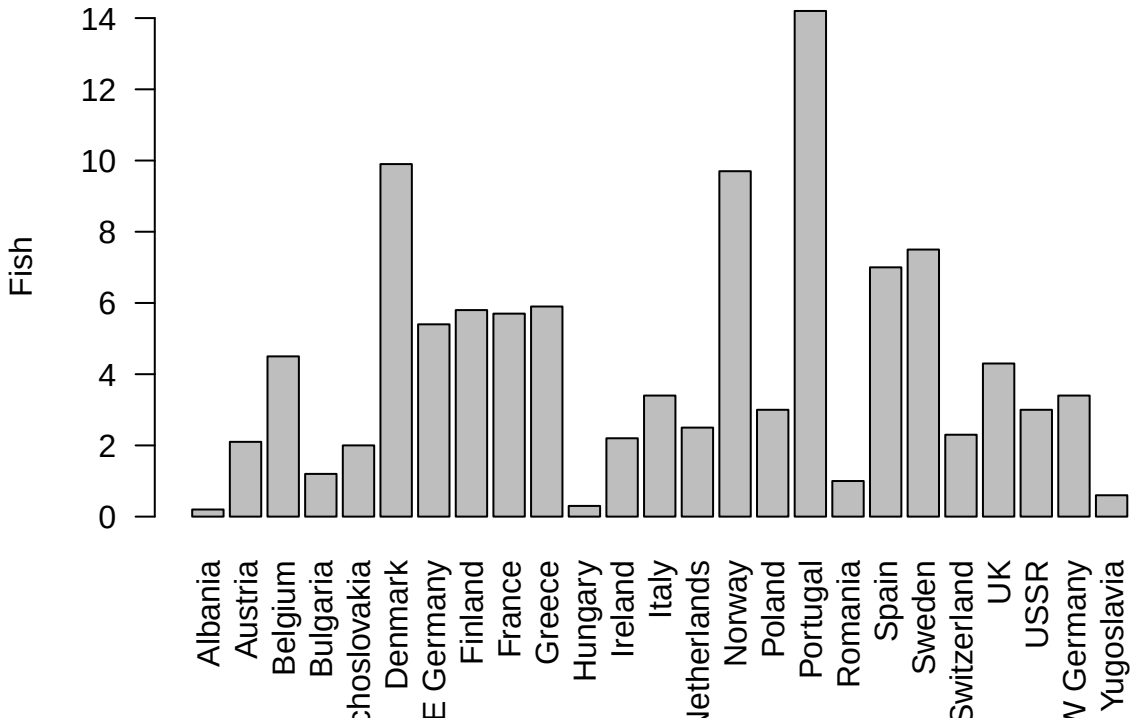
Jól látható, hogy összesen 9 élelmiszer típust sorol fel a táblázat, így azt is mondhatjuk, hogy az egyes országokhoz a 9 dimenziós térben elhelyezhetünk egy pontot, amely azt az országot jellemzi. Világos, hogy 9 dimenziós ábrázolás közvetlenül nem végezhető el, tehát valamiképpen csökkentenünk kell a dimenziók számát.

A dimenziócsökkentés feladatának számos motivációja lehet. Az egyik cél, hogy az adathalmazt kisebb helyen tudjuk tárolni, tehát tömörítsük. A másik - és itt elsősorban ezzel foglalkozunk -, hogy az adatok áttekintését segítse, az adathalmazban található információkat alacsonyabb dimenziós térben is megőrizhessük. Mind az adatfeldolgozás, mind az adatok ábrázolása a csökkentett dimenziós térben is megtörténhet. Itt elsősorban az ábrázolási és áttekintési feladatokban fogjuk kihasználni a dimenziócsökkentési lehetőségeket.

Projekció transzformáció nélkül

Számos esetben az adat projekció önmagában is elegendő a dimenziószám csökkentéséhez. Elképzelhető, hogy számos attribútum elhagyható, hiszen vagy nem képezi a vizsgálódásunk tárgyát, vagy éppen nem járul hozzá az adathalmazhoz információval (pl. az értéke konstans). Ha például csak egyetlen dimenzióra vagyunk kíváncsiak, akkor kiválasztjuk az egyik dimenziót (pl. a halakat), és ábrázolhatjuk az egyes pontok (sorok) értékeit:

```
barplot(data$Fish,names.arg=rownames(data), las=2, ylab="Fish")
```

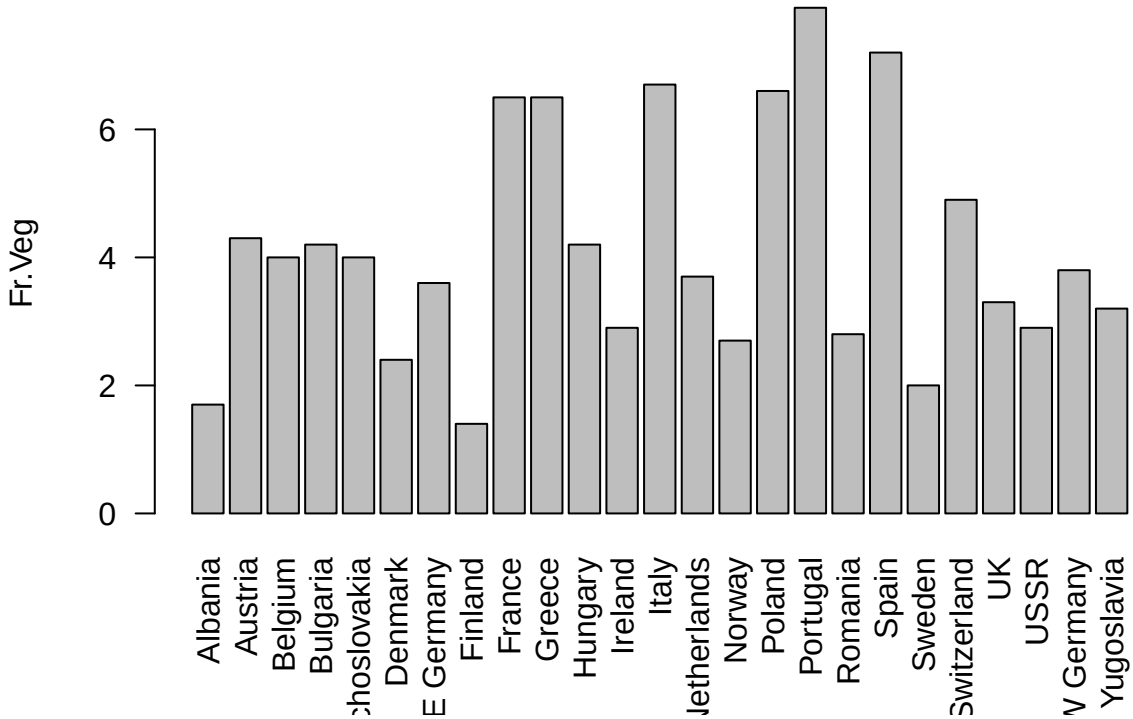


Jól látszik, hogy Magyarország, Románia, Albánia és Jugoszlávia nagyon kevés halat fogyasztanak. Az ábrázolást megtehetjük egyesével az összes dimenzióra:

```
for(name in names(data))
{
  barplot(data[[name]],names.arg=rownames(data), las=2, ylab=name)
}
```

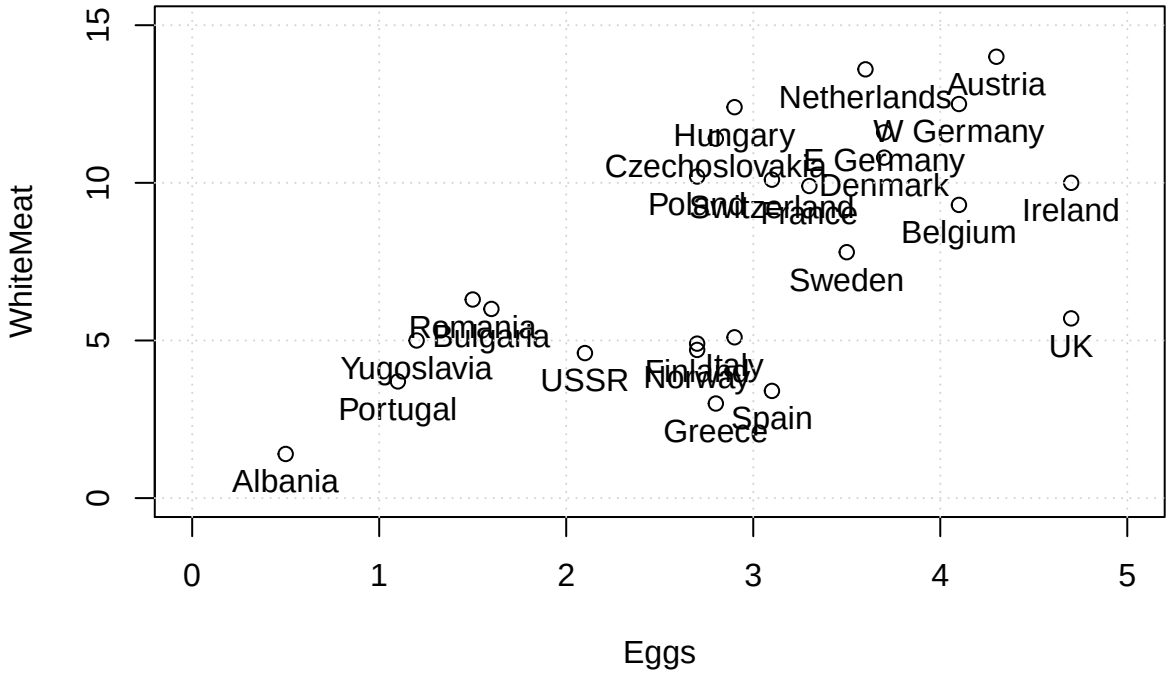
RedMeat
WhiteMeat
Eggs
Milk
Albania
Austria
Belgium
Bulgaria
Denmark
Finland
France
Greece
Hungary
Ireland
Italy
Norway
Poland
Portugal
Romania
Spain
Sweden
UK
USSR
Yugoslavia

Fish
0
2
4
6
8
10
12
14
Albania
Austria
Belgium
Bulgaria
choslovakia
Denmark
E Germany
Finland
France
Greece
Hungary
Ireland
Italy
Netherlands
Norway
Poland
Portugal
Romania
Spain
Sweden
Switzerland
UK
USSR
V Germany
Yugoslavia
Cereals
0
10
20
30
40
50
Albania
Austria
Belgium
Starch
0
1
2
3
4
5
6
Albania
Austria
Belgium
Bulgaria
choslovakia
Denmark
E Germany
Finland
France
Greece
Hungary
Ireland
Italy
Netherlands
Norway
Poland
Portugal
Romania
Spain
Sweden
Switzerland
UK
USSR
V Germany
Yugoslavia
Nuts
0
2
4
6
Albania
Austria
Belgium



Ha esetleg az egyes adatsorok elhelyezkedését szeretnénk vizsgálni két kiválasztott dimenzió terében, akkor már két dimenziós projekciót végzünk. Például, ha a fehér húsok és tojás fogyasztásának arányára vagyunk kíváncsiak, kiválaszthatjuk az első két dimenziót, és ábrázolhatjuk:

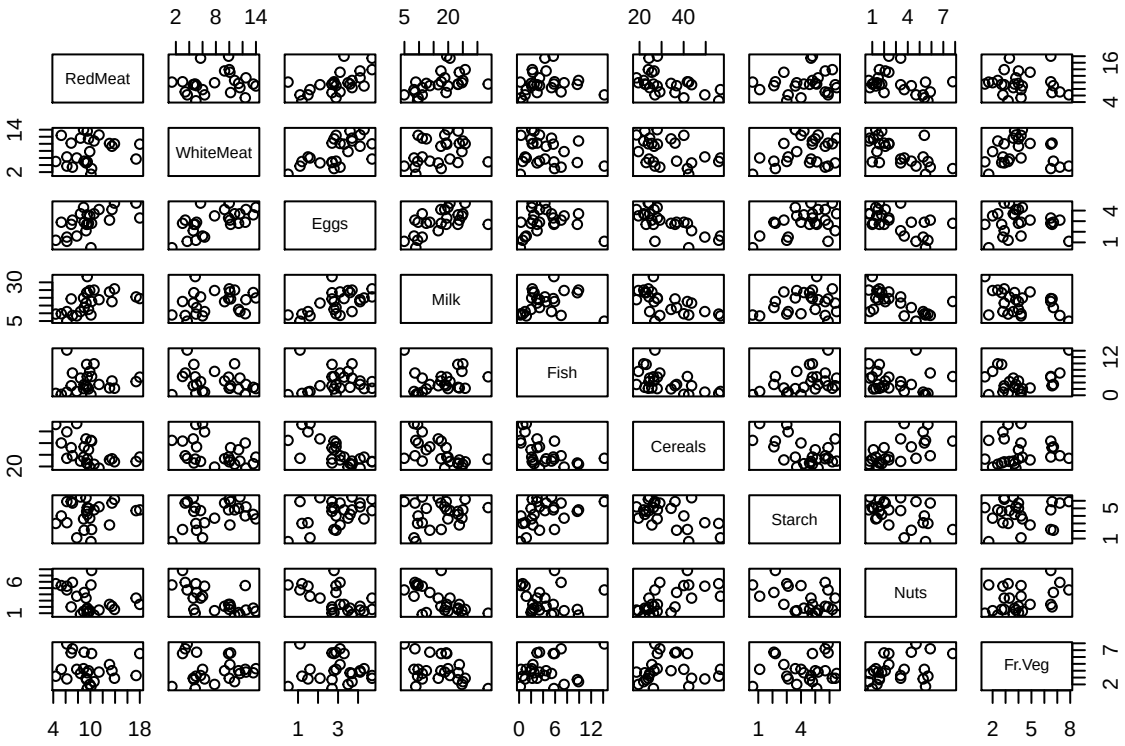
```
plot(WhiteMeat~Eggs,data=data,xlim=c(0,5),ylim=c(0,15))
text(WhiteMeat~Eggs,data=data,labels=rownames(data),pos=1)
grid()
```



Az érdekes, hogy ebben az esetben két, esetleg három jól elkülöníthető csoportot látunk az egyes országok között, hiszen a jobb felső országok (pl. Magyarország, Ausztria, stb.) esetében a magas tojás fogyasztás

magas fehér hús fogyasztással jár (nem biztos, hogy ok-okozati az összefüggés), míg például Portugália és Romániában mindkettő alacsony. Az összes dimenzióról kaphatunk némi áttekintést, ha egyszerűen az összes dimenziót az összes másik dimenzióval ábrázoljuk. Ez értelemszerűen N dimenzió esetén $\binom{N}{2}$ ábrát fog jelenteni legalább (jelen esetben 36 darab ábra).

```
plot(data)
```



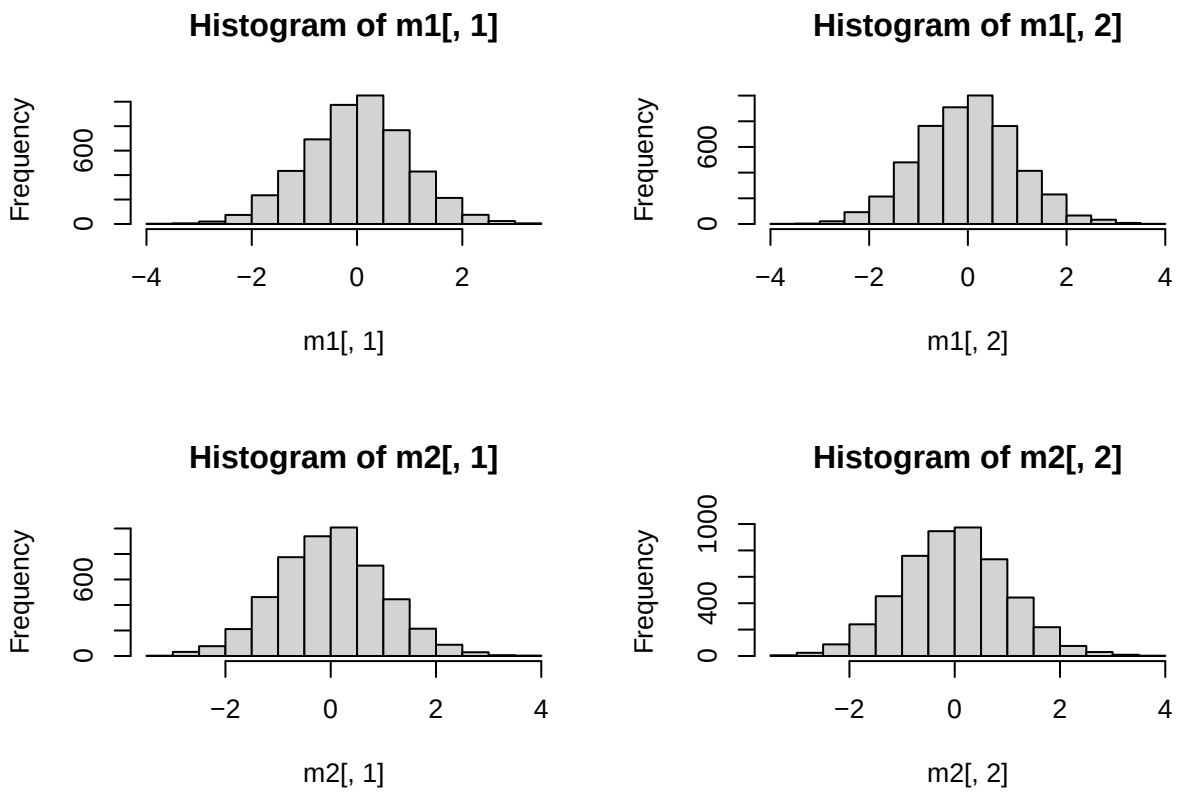
Főkomponens analízis

Az egyik leggyakoribb, és legalapvetőbb dimenziócsökkentési eljárás az ún. főkomponens analízis (PCA - Principal Component Analysis).

Motiváció

A PCA motivációjához tekintsük az alábbi két adathalmazt (m1,m2), melyek egy dimenziós projekcióit hisztogrammként ábrázoljuk.

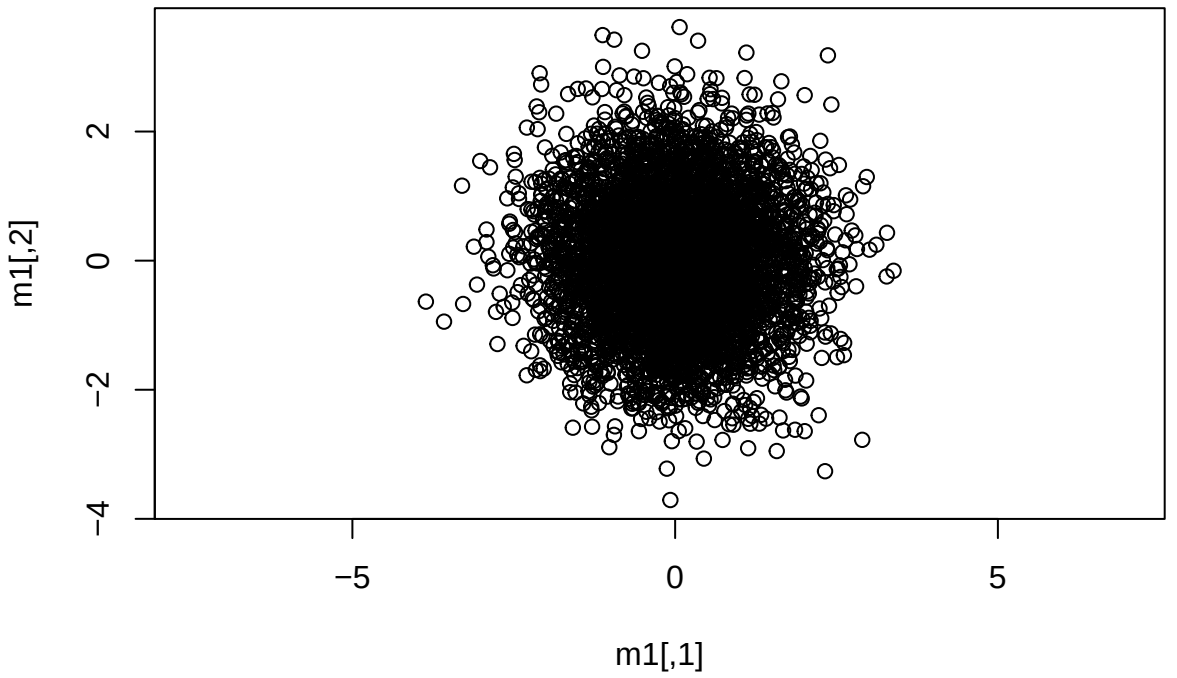
```
m1<-matrix(rnorm(10000,0,1), ncol=2)
m2<-matrix(rnorm(10000,0,1), ncol=2)
m2<-m2 %*% matrix(c(1,0,0.9,0.435),ncol=2)

par(mfrow=c(2,2))
hist(m1[,1])
hist(m1[,2])
hist(m2[,1])
hist(m2[,2])
```



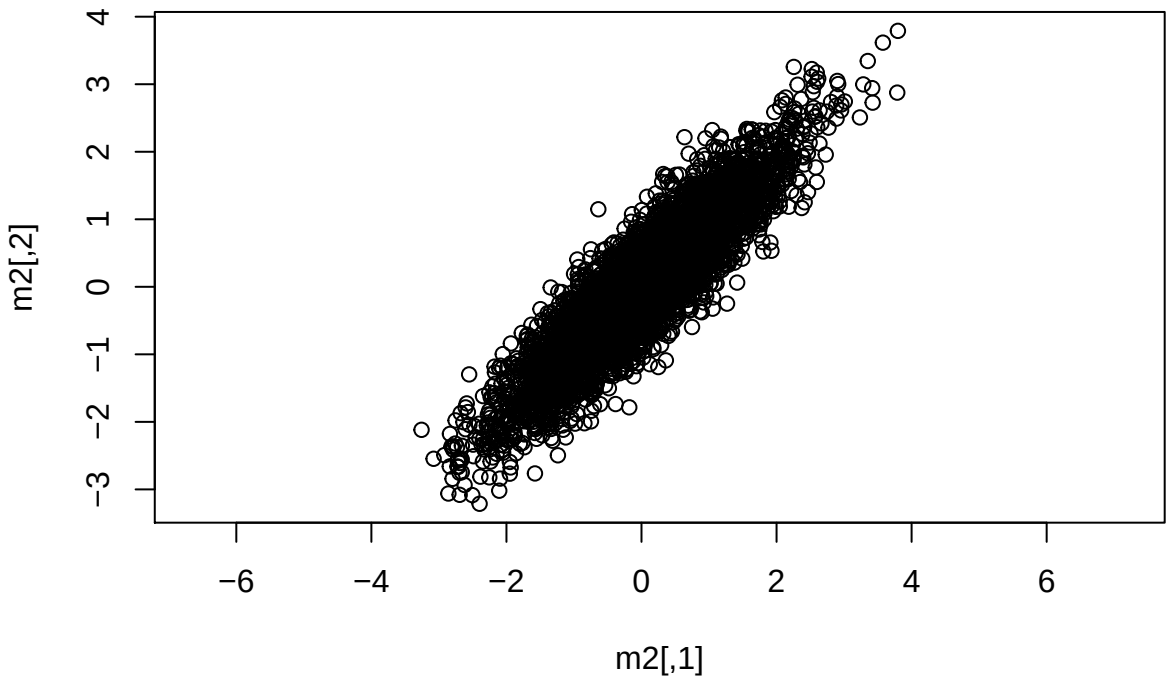

```
par(mfrow=c(1,1))
```

Habár a hisztogrammok hasonlónak tűnnek, a két adathalmaz nem azonos, és ez a különbség az egyes változók összefüggéseiben keresendők (pontosabban a kovarianciában).

```
plot(m1,asp=1)
```



```
plot(m2,asp=1)
```




```
cov(m1)
```

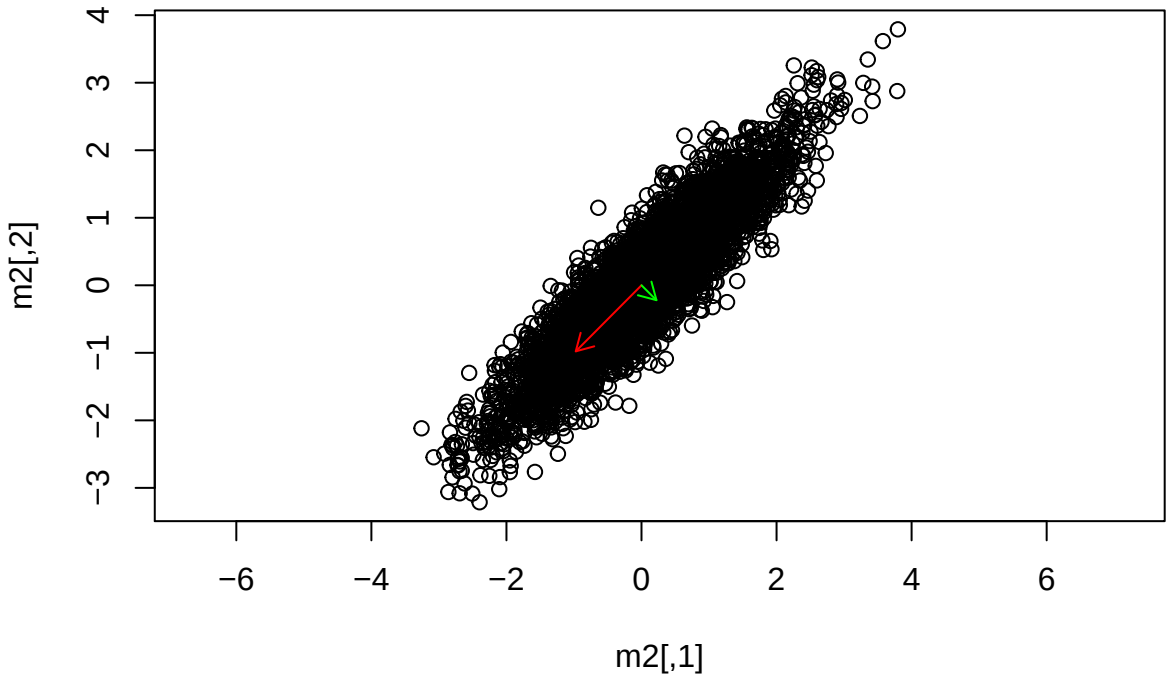
```
##             [,1]        [,2]
## [1,]  0.95162582 -0.01896358
## [2,] -0.01896358  0.99028535
```

```
cov(m2)
```

```
##           [,1]      [,2]
## [1,] 0.9969901 0.9034656
## [2,] 0.9034656 1.0051455
```

A főkomponens analízis alapötlete, hogy forgassuk el az adatokat úgy, hogy a transzformált adatpontokra a kovariancia mátrix diagonális legyen. Így az egyes komponensek - a főkomponensek (PC - principal component) - egymástól függetlenek lesznek. A főkomponensek az eredeti adathalmaz változóinak a lineáris kombinációi. Például, készítsük el a bemutatott adahalmaz (m2) főkomponens analízisét, és rajzoljuk ki nyilakkal a főkomponensek irányát:

```
p<-prcomp(m2)

plot(m2,asp=1)
arrows(0,0,p$sdev[1]*p$rotation[1,1],p$sdev[1]*p$rotation[2,1],col='red',length=0.1)
arrows(0,0,p$sdev[2]*p$rotation[1,2],p$sdev[2]*p$rotation[2,2],col='green',length=0.1)
```



Láthatjuk az ábrán, hogy az első főkomponens abba az irányba mutat, amerre a legnagyobb az adathalmaz szórása, a második komponens predig erre merőleges.

Lineáris kombináció

Tételezzük fel, hogy van két számunk, például az 5 és a 8. Ezek lineáris kombinációja az alábbi kifejezés:

$$a \cdot 5 + b \cdot 8$$

Persze lineáris kombináció nem csak számokra, hanem vektorokra is értelmezhető, például:

$$a \begin{bmatrix} 3 \\ 5 \end{bmatrix} + b \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Tekintsük például az `iris` adathalmaz első sorát:

```
iris[1,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

Ezt felirhatjuk a szokásos *bázisvektorok* segítségével:

$$5.1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 3.5 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + 1.4 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + 0.2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{bmatrix}$$

A kérdés, van-e olyan bázis, amiben az egyes koordináták (innentől *score* vagy *pontszám*) jobban reprezentálják az adatokat? Például felírhatjuk a következőt is:

$$-1.2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 2.1 \begin{bmatrix} 3 \\ 1 \\ 0 \\ 0 \end{bmatrix} + 0.7 \begin{bmatrix} 0 \\ 2 \\ 2 \\ 0 \end{bmatrix} + 0.2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{bmatrix}$$

Az egyetlen kérdés innentől, hogy miként határozzuk meg a bázisvektorokat. A következőkben az ún. főkomponens analízist fogjuk alkalmazni, amely lehetővé teszi számunkra egy speciális bázisrendszer kiválasztását.

A főkomponens analízis számítása és értelmezése

A főkomponens analízist az `iris` adathalmazon mutatjuk be. A következő ábra mutatja az iris virág leveleinek elnevezését és méreteit.

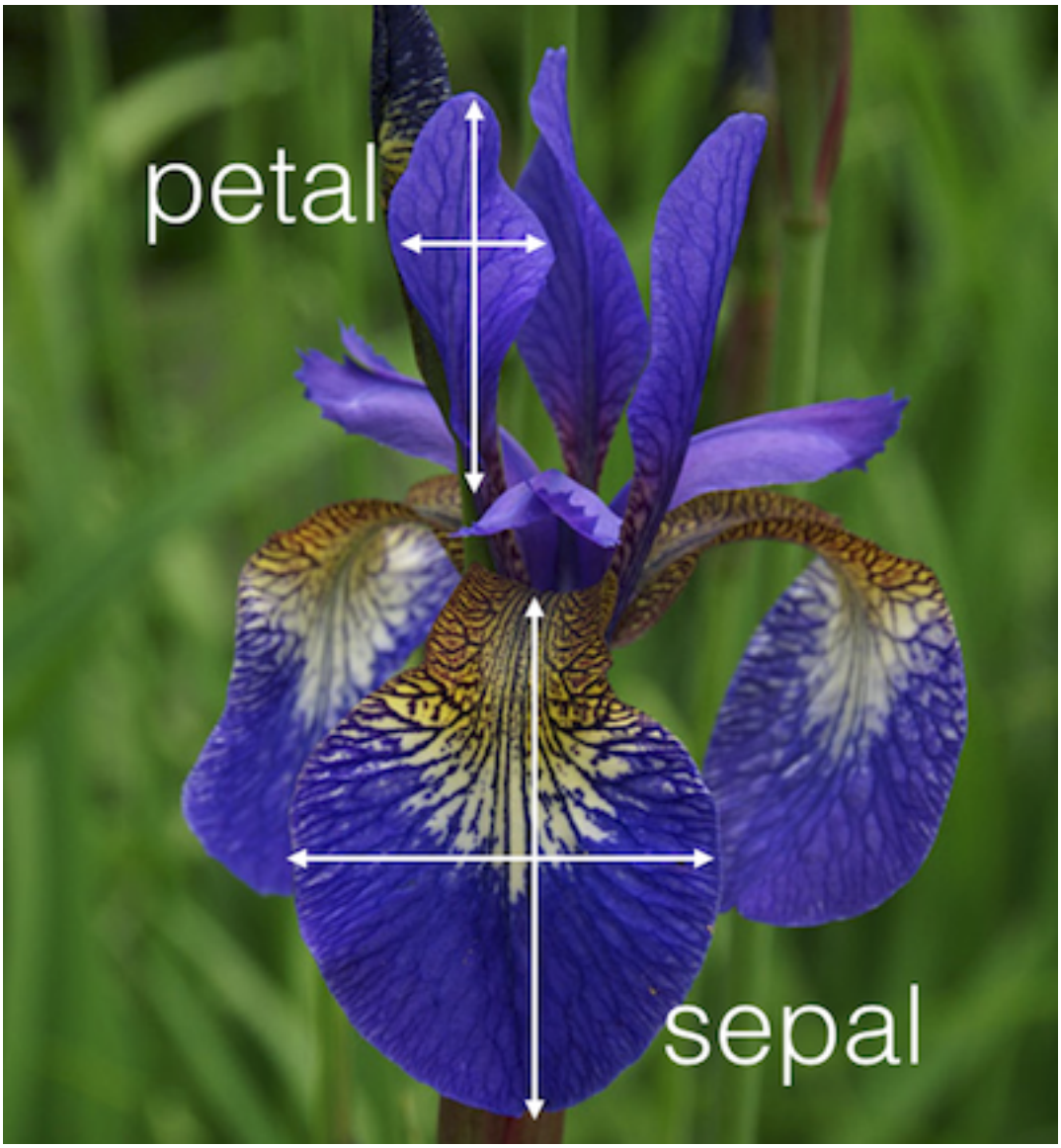


Figure 2: Az iris növény leveleinek elnevezése és méretei

Töltsük be az adathalmazt, és tekintsük át! Távolítsuk el a `Species` változót!

```
df<-iris
summary(df)
```

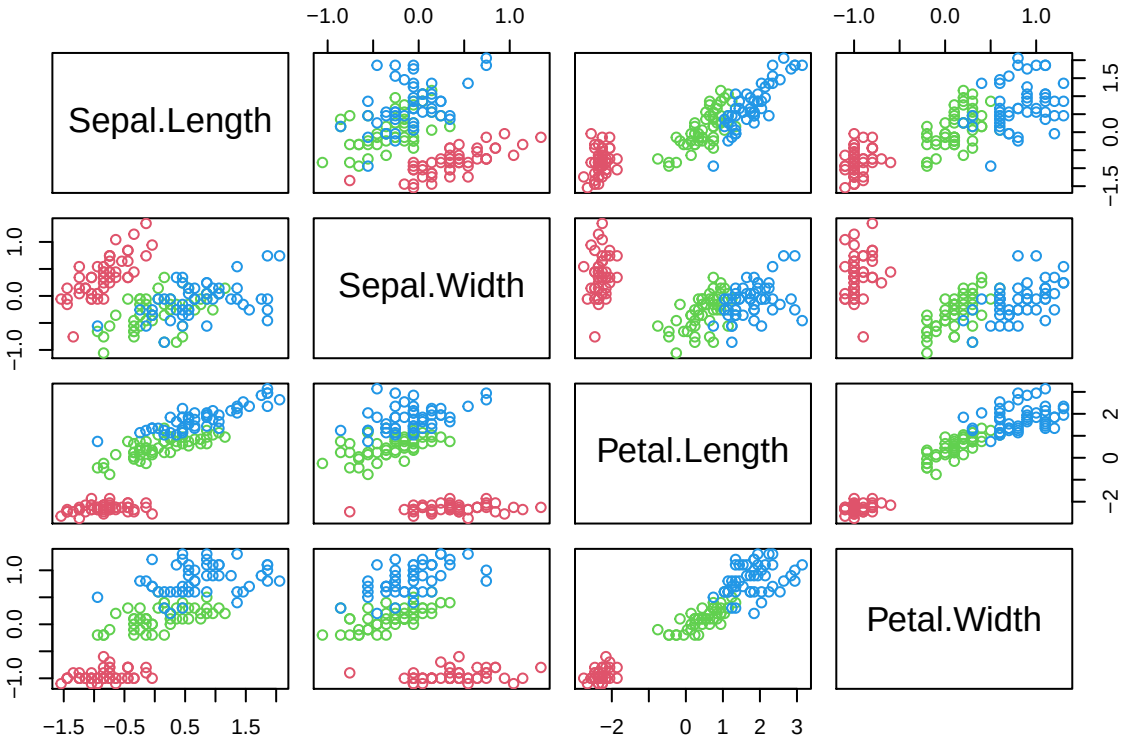
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
```

```
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
df<-df[,1:4]
```

A főkomponens analízis vizsgálatához az adathalmaz el kell tolni az origóba. Szerencsére, erre a `scale` függvényt tudjuk használni. Számos esetben célszerű az adathalmazt normalizálni, azaz az egyes változók szórását 1-re skálázni. Ennek megvalósítására ugyancsak a `scale` függvény használható. Jelen esetben ezt most nem tesszük meg, mivel az egyes változók *összemérhető* mennyiségek.

```
df<-as.data.frame(scale(df,scale=F,center=T))
plot(df,col=as.numeric(iris$Species)+1)
```



Végezzük el a főkomponens-analízist! Erre a `prcomp` függvényth használhatjuk:

```
p<-prcomp(df)
print(p)
```

```
## Standard deviations (1, .., p=4):
## [1] 2.0562689 0.4926162 0.2796596 0.1543862
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2         PC3        PC4
## Sepal.Length  0.36138659 -0.65658877  0.58202985  0.3154872
```

```
## Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
## Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
## Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

```
summary(p)
```

```
## Importance of components:
##                           PC1     PC2    PC3     PC4
## Standard deviation     2.0563 0.49262 0.2797 0.15439
## Proportion of Variance 0.9246 0.05307 0.0171 0.00521
## Cumulative Proportion  0.9246 0.97769 0.9948 1.00000
```

Láthatjuk, hogy visszakaptuk a főkomponenseket (PC1-4). Az egyes komponensek csökkenő fontosságban szerepelnek, az utolsó sorban láthatjuk a kommulatív arányát a lefedett szórásnégyzetnek. Már az első két főkomponens a szórásnégyzet 97.77 százalékát lefedi, azaz az első két főkomponens felel az adathalmazban található információk nagy részéért.

A főkomponensek tulajdonságai

Nézzük meg a főkomponensek pár fontos tulajdonságát! Az adathalmaz a főkomponensek terében (azaz az főkomponensekre forgatott adathalmaz) az `x` mezőben érhető el:

```
head(p$x)
```

```
##            PC1        PC2         PC3          PC4
## [1,] -2.684126 -0.3193972  0.02791483  0.002262437
## [2,] -2.714142  0.1770012  0.21046427  0.099026550
## [3,] -2.888991  0.1449494 -0.01790026  0.019968390
## [4,] -2.745343  0.3182990 -0.03155937 -0.075575817
## [5,] -2.728717 -0.3267545 -0.09007924 -0.061258593
## [6,] -2.280860 -0.7413304 -0.16867766 -0.024200858
```

A forgatásért felelős transzformációban a főkomponensek szerepelnek:

```
p$rotation
```

```
##                      PC1         PC2         PC3        PC4
## Sepal.Length  0.36138659 -0.65658877  0.58202985  0.3154872
## Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
## Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
## Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

A főkomponensek merőlegesek egymásra, és egységnyi hosszúságúak:

```
sum(p$rotation[,1] * p$rotation[,2])
```

```
## [1] -1.353084e-16
```

```
p$rotation %*% t(p$rotation)
```

```
##              Sepal.Length   Sepal.Width  Petal.Length  Petal.Width
## Sepal.Length 1.000000e+00  4.163336e-16  2.775558e-17 2.775558e-17
## Sepal.Width  4.163336e-16  1.000000e+00 -1.942890e-16 1.110223e-16
## Petal.Length 2.775558e-17 -1.942890e-16  1.000000e+00 0.000000e+00
## Petal.Width  2.775558e-17  1.110223e-16  0.000000e+00 1.000000e+00
```

```
apply(p$rotation^2,2,sum)
```

```
## PC1 PC2 PC3 PC4
##   1   1   1   1
```

A főkomponensek terében az egyes transzformált változók szórása megegyezik a főkomponensekhez tartozó szórással:

```
apply(p$x,2,sd)
```

```
##       PC1       PC2       PC3       PC4
## 2.0562689 0.4926162 0.2796596 0.1543862
```

A főkomponensek terében a kovarianciamátrix diagonális:

```
cov(p$x)
```

```
##              PC1           PC2           PC3           PC4
## PC1 4.228242e+00  5.493016e-16  1.997781e-16  7.969223e-16
## PC2 5.493016e-16  2.426707e-01 -5.859776e-17 -1.343738e-16
## PC3 1.997781e-16 -5.859776e-17  7.820950e-02 -1.628916e-17
## PC4 7.969223e-16 -1.343738e-16 -1.628916e-17  2.383509e-02
```

A főkomponensek értelmezése

A főkomponensek az eredeti változók lineáris kombinációja, és azt jelölik, hogy mekkora arányban vesz részt az adott főkomponensben az eredeti adathalmaz adott változója. Az egyes transzformált adatpontok koordinátáit *score*-nak hívjuk, és azt jelölik, hogy az adott adatpontban az adott főkomponens mekkora súllyal bír. Például tekintsük az első adatsort, és a főkomponenseket:

```
p$x[1,]
```

```
##          PC1          PC2          PC3          PC4
## -2.684125626 -0.319397247  0.027914828  0.002262437
```

```
p$rotation
```

```
##                     PC1         PC2         PC3        PC4
## Sepal.Length  0.36138659 -0.65658877  0.58202985  0.3154872
## Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
## Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
## Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

Az eredeti adatsor helyreállítható az egyes főkomponensek megfelelő lineáris kombinációjával:

```
p$x[1,1] * p$rotation[,1] + p$x[1,2] * p$rotation[,2] +
  p$x[1,3] * p$rotation[,3] + p$x[1,4] * p$rotation[,4]
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##   -0.7433333    0.4426667   -2.3580000   -0.9993333
```

```
p$x[1,] %*% t(p$rotation)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]   -0.7433333   0.4426667       -2.358  -0.9993333
```

```
print(df[1,])
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1   -0.7433333   0.4426667       -2.358  -0.9993333
```

A főkomponensek ábrázolása

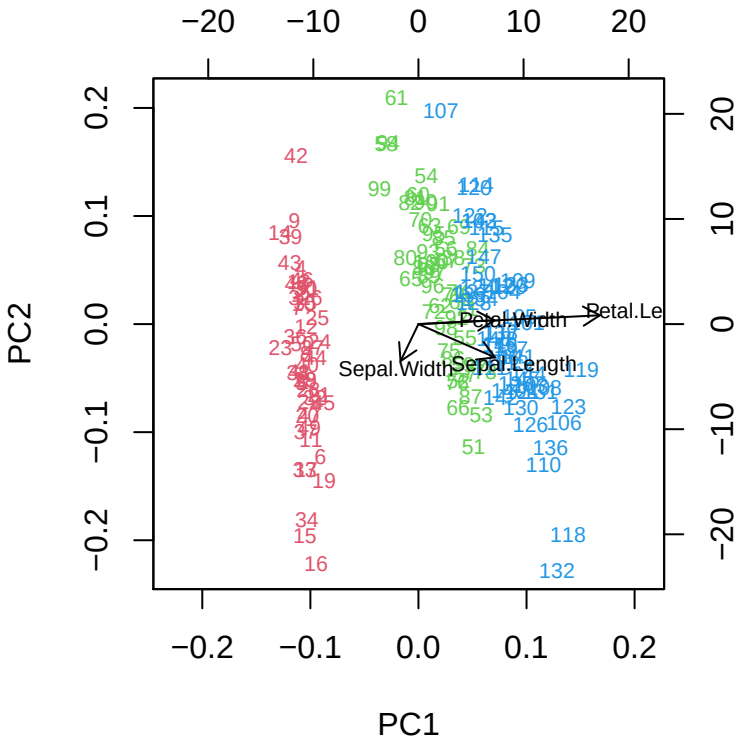
A főkomponensek, és a hozzá tartozó transzformált adatpontok ábrázolása ún. *biplot* segítségével történhet. Fontos megemlíteni, hogy a biplot két dimenziós ábrázolás, és tipikusan az első két főkomponens információit

tartalmazza. A biplot akkor tekinthető helyesnek, ha az első két főkomponens a szórásnégyzet legalább 90-95 százalékát lefedi, egyébként erősen torzít:

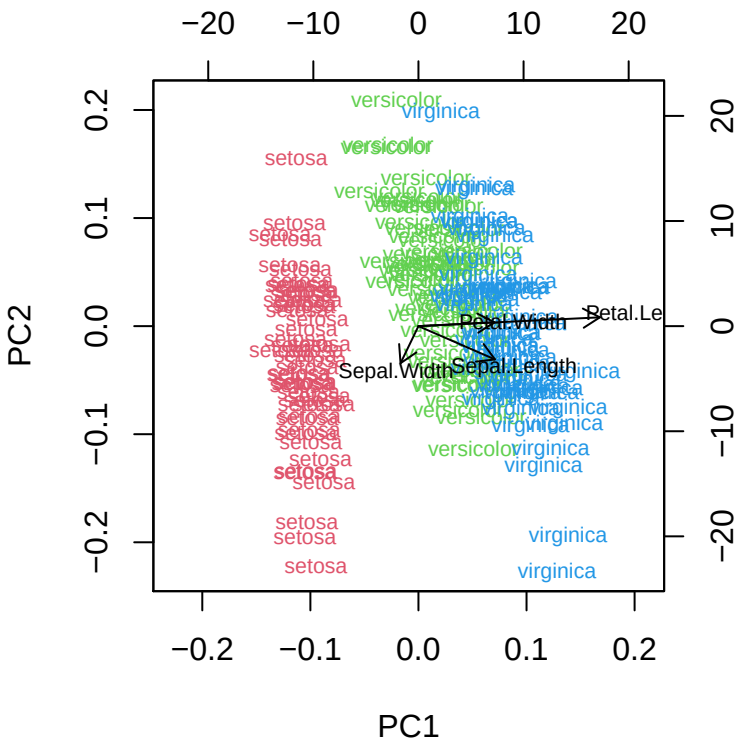
```
library(compositions)
```

```
## Welcome to compositions, a package for compositional data analysis.
## Find an intro with "? compositions"

##
## Attaching package: 'compositions'

## The following objects are masked from 'package:stats':
##
##     cor, cov, dist, var

## The following objects are masked from 'package:base':
##
##     %*%, norm, scale, scale.default
```

```
coloredBiplot(p,pch=as.numeric(iris$Species),col=1,cex=0.7,
              xlabs.col=as.numeric(iris$Species)+1)
```




```
coloredBiplot(p,pch=as.numeric(iris$Species),col=1,
              xlabs=iris$Species,cex=0.7,xlabs.col=as.numeric(iris$Species)+1)
```



Az egyes nyilak a változókat, a pontok az adatpontokat mutatják az első két főkomponens terében. A következő megjegyzéseket kell tennünk:

- A nyilak hossza a változó szórásával arányos
- A nyilak iránya mutatja az egyes változók főkomponensekben elfoglalt fontosságát
- Az egyes nyilak bezárt szögének koszinusza az egyes változók korrelációját mutatja
- Két nyíl skaláris szorzata a két változó kovarianciáját mutatja
- Az egyes adatpontok távolsága arányos a két adatsor Mahalanobis-távolságával
- Egy nyíl és egy adatpont skaláris szorzata arányos az eredeti adatsorral

Adattömörítés

A főkomponens analízis nagy előnye, hogy lehetőséget ad az adatok tömörítésére is. Jelen esetben a célunk ezzel, hogy a főkomponens-analízis értelmezését megerősítsük. A kulcsmozzanat az adattömörítésben, hogy az adatokat le tudjuk írni hatékonyan a főkomponensek egy alhalmazával is. Mint ahogy említettük, az adathalmaz helyreállítható a transzformáció (forgatás) ellentettjével:

```
head(p$x %*% t(p$rotation))
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]   -0.7433333  0.44266667       -2.358  -0.9993333
## [2,]   -0.9433333 -0.05733333       -2.358  -0.9993333
## [3,]   -1.1433333  0.14266667       -2.458  -0.9993333
## [4,]   -1.2433333  0.04266667       -2.258  -0.9993333
## [5,]   -0.8433333  0.54266667       -2.358  -0.9993333
## [6,]   -0.4433333  0.84266667       -2.058  -0.7993333
```

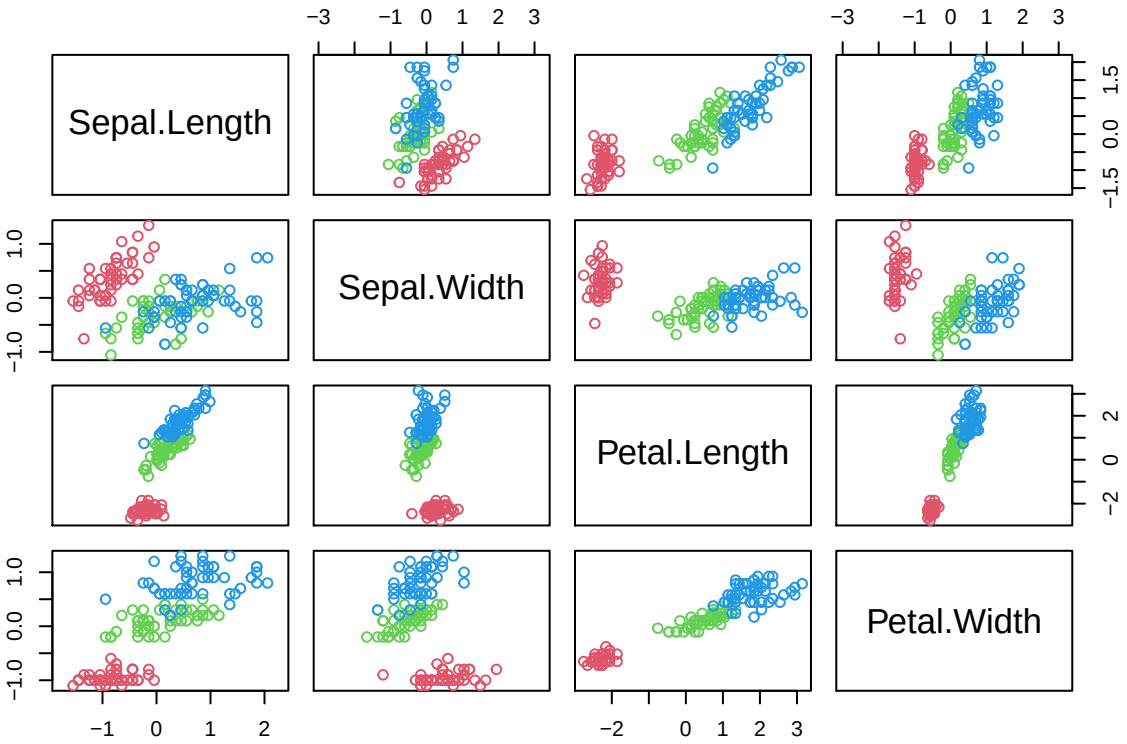
```
head(df)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1   -0.7433333  0.44266667       -2.358  -0.9993333
## 2   -0.9433333 -0.05733333       -2.358  -0.9993333
## 3   -1.1433333  0.14266667       -2.458  -0.9993333
```

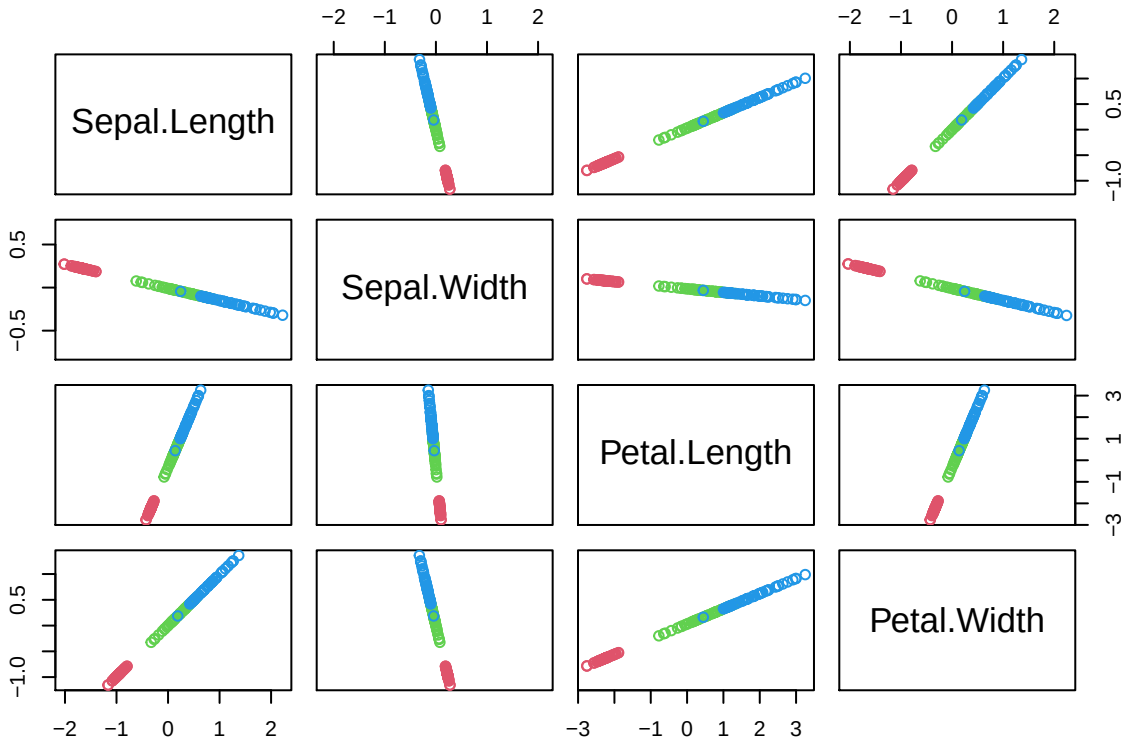
```
## 4   -1.2433333  0.04266667       -2.258  -0.9993333
## 5   -0.8433333  0.54266667       -2.358  -0.9993333
## 6   -0.4433333  0.84266667       -2.058  -0.7993333
```

A helyreállítás során nem feltétlenül szükséges az összes főkomponenst használnunk, az adatok egyfajta közelítését érhetjük el, ha csak az első pár főkomponenst használjuk. (Itt közelítés alatt az ún. Frobenius-normát minimalizáljuk az eredeti adathalmaz és a helyreállított adathalmaz között, azaz másnéven a négyzetes eltérést minimalizáljuk.)

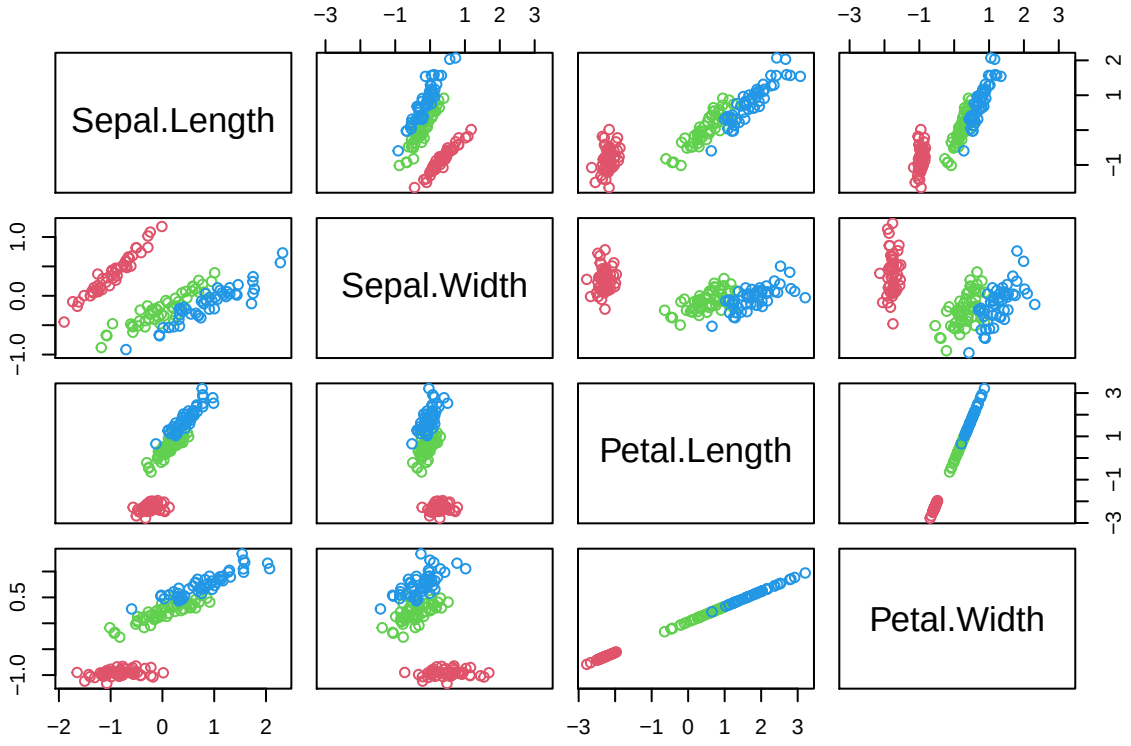
```
df<-as.data.frame(scale(iris[,1:4],scale=F))
plot(df,asp=1,col=as.numeric(iris$Species)+1)
```



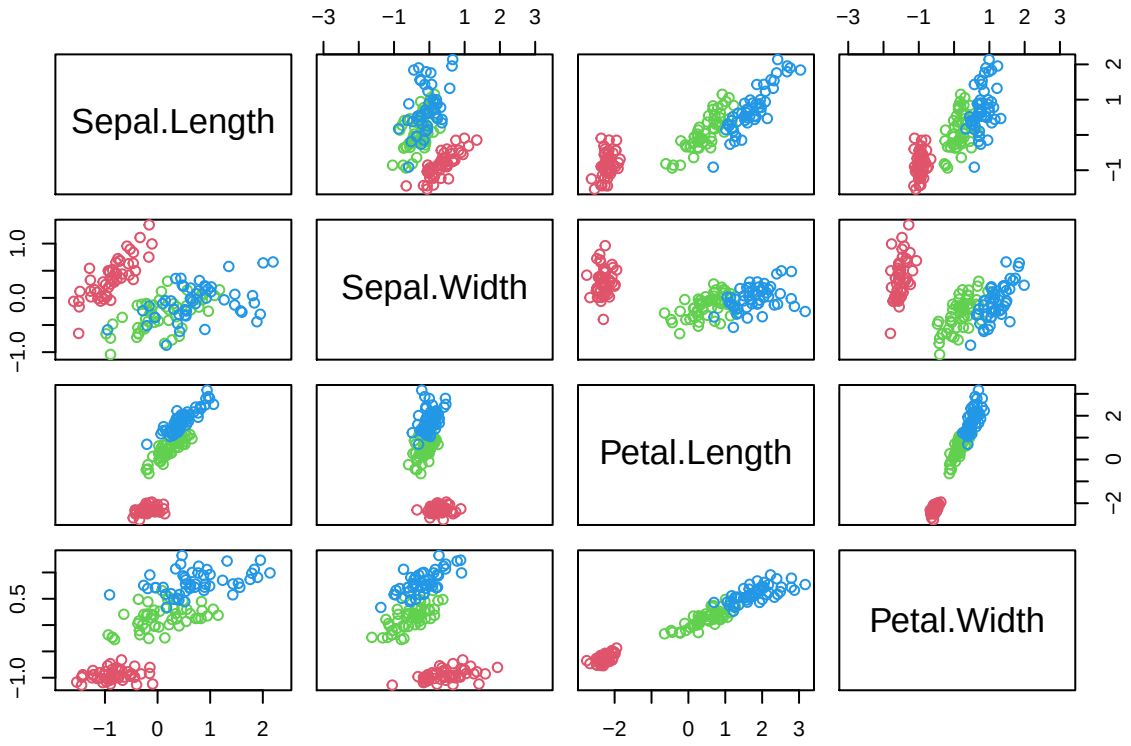

```
reconst<-as.data.frame(p$x[,1:1] %*% t(p$rotation[,1:1]))
plot(reconst,asp=1,col=as.numeric(iris$Species)+1)
```



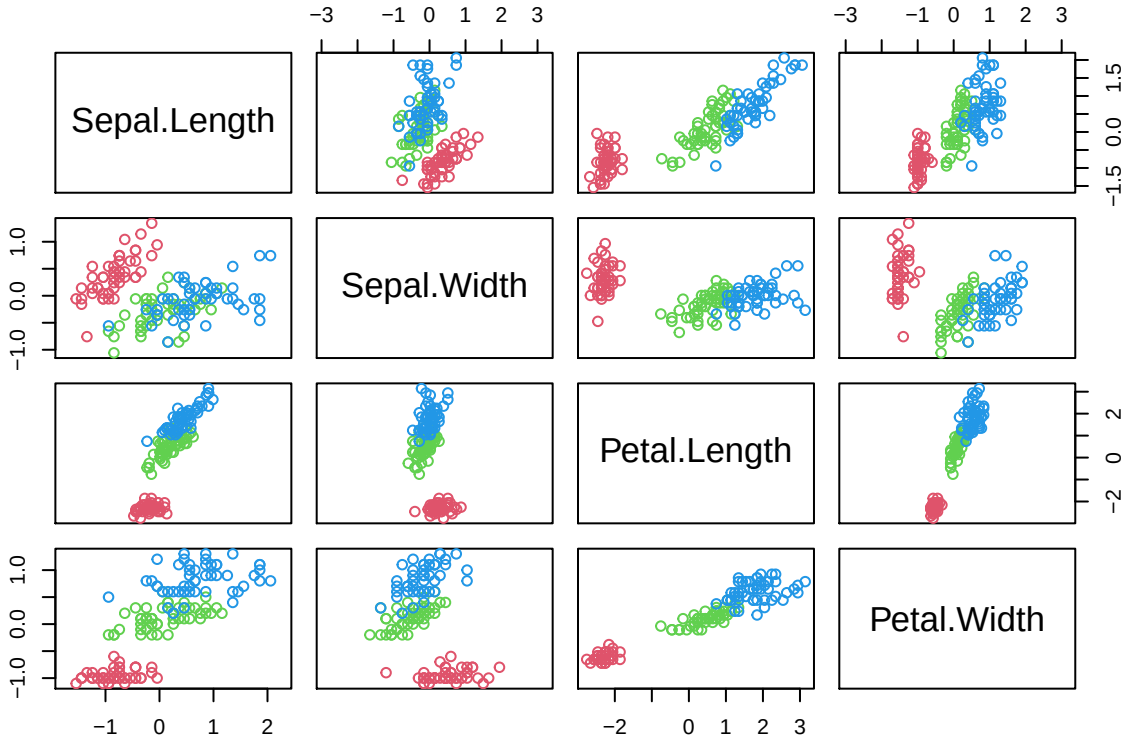

```
reconst<-as.data.frame(p$x[,1:2] %*% t(p$rotation[,1:2]))
plot(reconst,asp=1,col=as.numeric(iris$Species)+1)
```



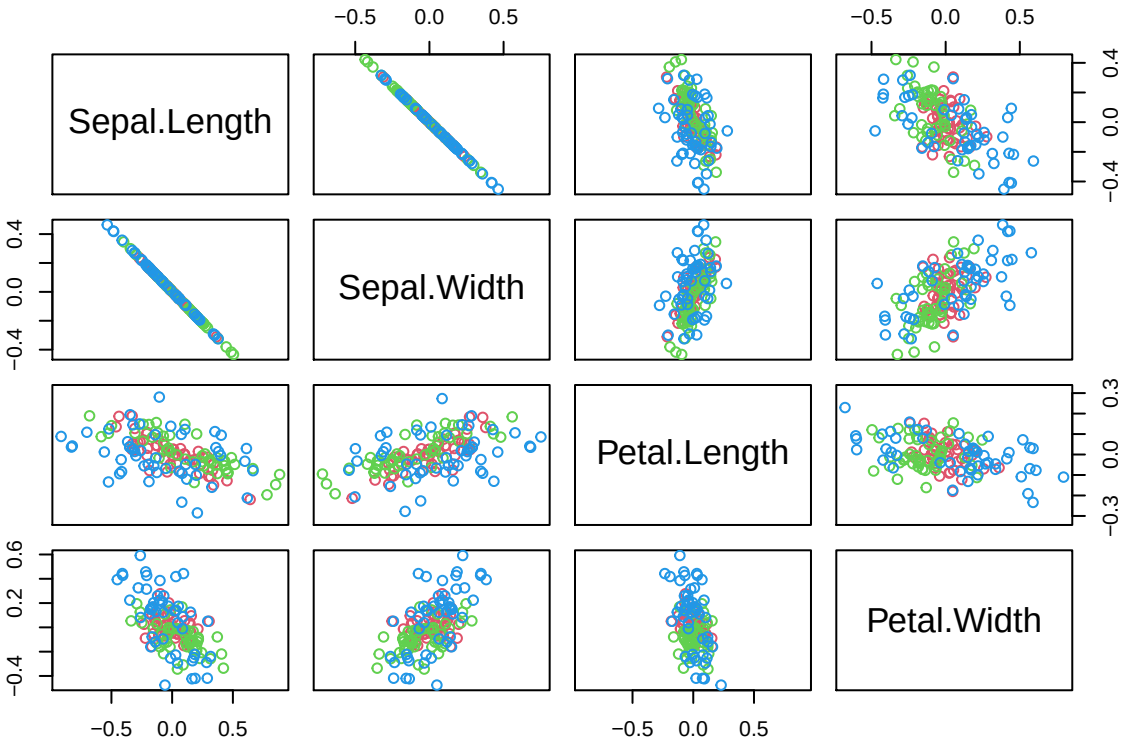

```
reconst<-as.data.frame(p$x[,1:3] %*% t(p$rotation[,1:3]))
plot(reconst,asp=1,col=as.numeric(iris$Species)+1)
```




```
reconst<-as.data.frame(p$x[,1:4] %*% t(p$rotation[,1:4]))
plot(reconst,asp=1,col=as.numeric(iris$Species)+1)
```




```
reconst<-as.data.frame(p$x[,3:4] %*% t(p$rotation[,3:4]))
plot(reconst,asp=1,col=as.numeric(iris$Species)+1)
```



Összefoglalás

A főkomponens analízis célja többdimenziós adathalmazok áttekintésének elősegítése, dimenziószámának csökkentése, az adatok jobb megértése. A főkomponens-analízis számos értelmezéssel bír, itt felsoroljuk a legfontosabbakat:

- A főkomponens-analízis megkeresi azt az ortogonális bázisrendszert (a főkomponenseket), amelyekben az adathalmaz dekorrelálttá válik, azaz kovarianciamátrixsza diagonálissá válik, az egyes főkomponensek függetlenek lesznek.
- A főkomponens-analízist tekinthetjük egy iteratív folyamatnak is. Az első főkomponens megkeresi az adathalmazban a legnagyobb szórás irányát, majd megkeresi az erre merőleges legnagyobb szórás irányát, és így tovább.
- A főkomponens analízis megkeresi a megfigyelt változók (adatoszlopok) mögött rejlő, rejtett (látens) okokat, ún. *látens változókat* (a főkomponenseket). Ezek a látens változók azt mutatják, hogy az adathalmaz alakulását milyen háttérfolyamatok vezérlik, azok az információtartalomhoz mennyire járulnak hozzá. Emiatt a főkomponens-analízis tekinthető egyfajta felügyelet nélküli tanulási módszernek is.

Főkomponens analízis, mint generatív modell (haladó)

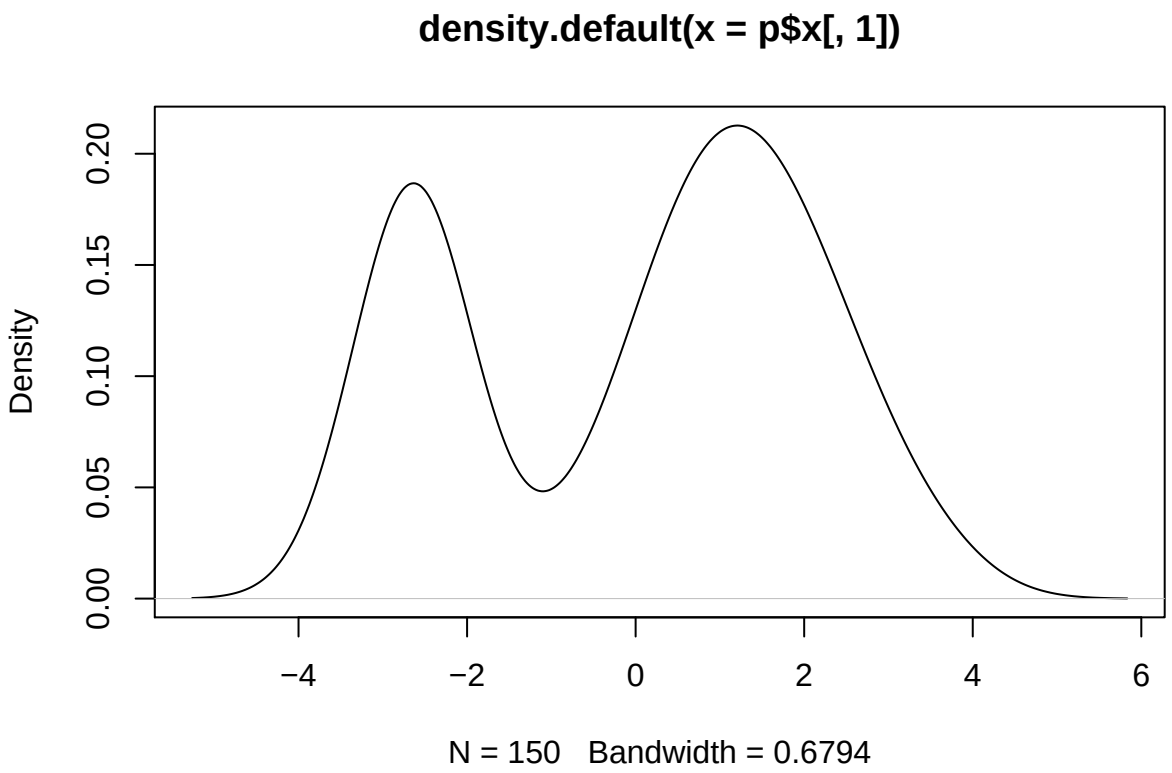
A főkomponens analízis lehetőséget nyújt számunkra új adatpontok generálására. Ehhez vegyük észre, hogy az egyes főkomponensek függetlenek, és tekinthetjük őket egyfajta látens változónak. Az egyes látens változók eloszlásának ismeretében tudunk új mintákat generálni a látens változókra, majd ezeket a főkomponens mintákat az eredeti adatok terébe vissza tudjuk transzformálni.

Készítsük el az iris adathalmaz főkomponens analízisét:

```
df<-as.data.frame(scale(iris[,1:4],scale=F))
p<-prcomp(df)
```

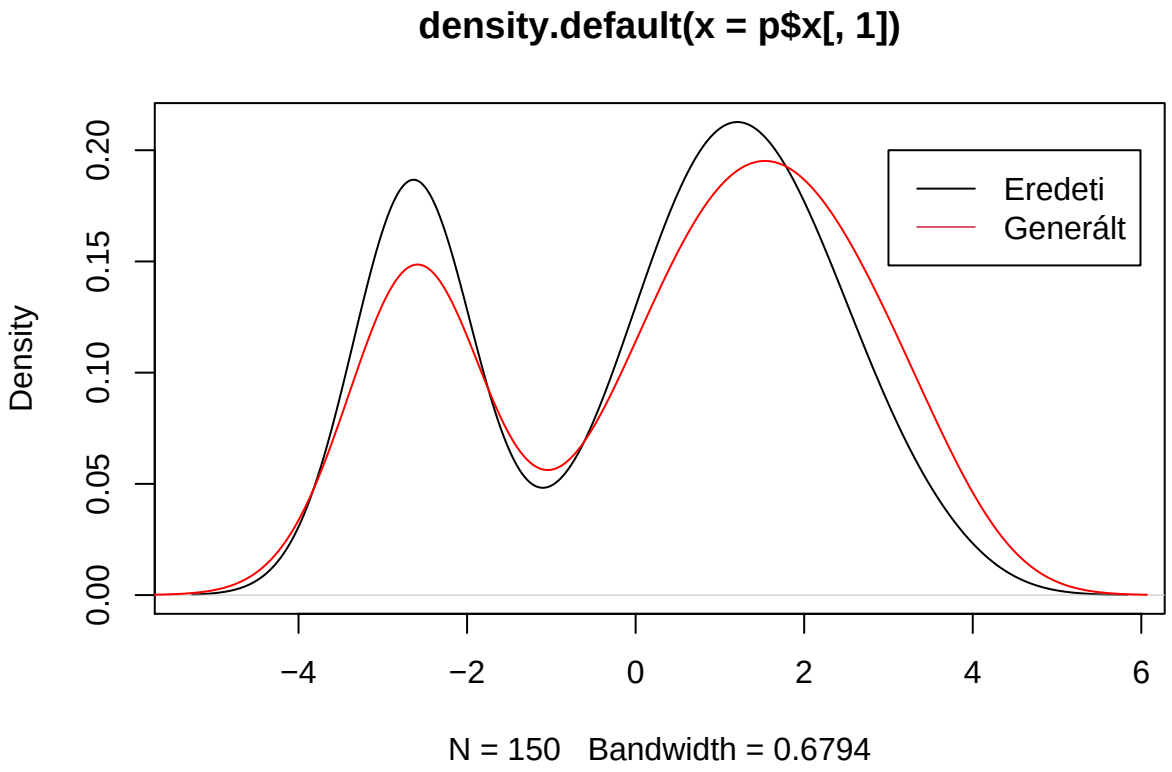
Tekintsük meg például az első főkomponens pontjainak eloszlását:

```
plot(density(p$x[,1]))
```



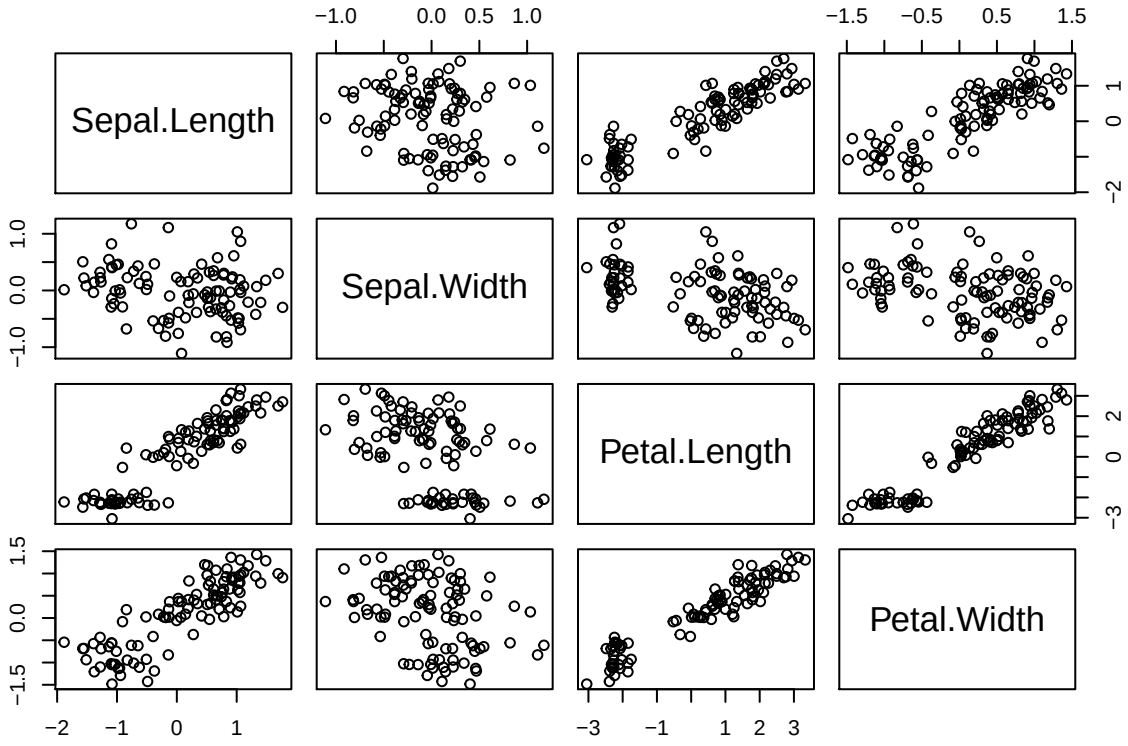
Kíséreljük meg 100 új pont generálását a főkomponensek pontjainak eloszlása alapján! Ábrázoljuk az első főkomponens eredeti és generált mintáinak eloszlását:

```
count<-100
pc_samples<-sapply(1:4, function(idx) {
  s<-sample(p$x[,idx],count,replace=T)+rnorm(count,0,sd=bw.nrd0(p$x[,idx]/5))
})
plot(density(p$x[,1]))
lines(density(pc_samples[,1]),col='red')
legend(3,0.2,legend=c('Eredeti','Generált'),col=1:2,lty=1)
```

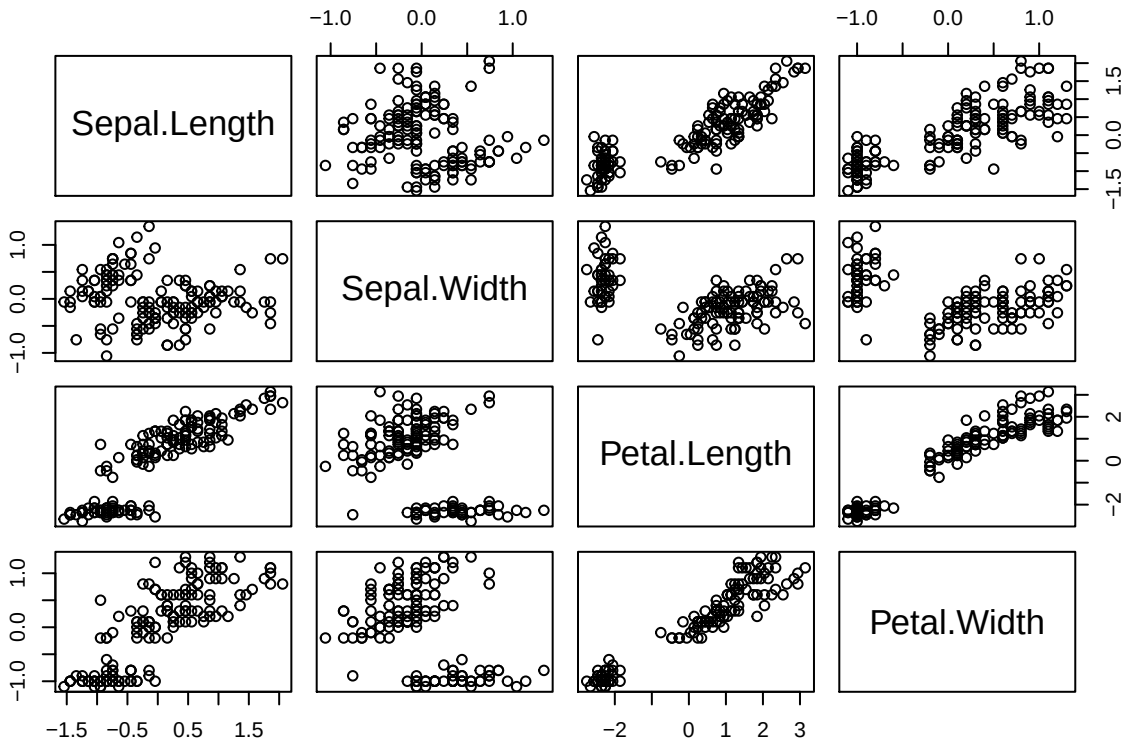


A generált pontokat transzformáljuk vissza az eredeti adatok terébe! Hasonlítsuk össze az eredeti `iris` adathalmazzal a transzformált pontokat!

```
gen_samples<-pc_samples %*% t(p$rotation)
gen_samples<-as.data.frame(gen_samples)
plot(gen_samples)
```



```
plot(df)
```



Főkomponens analízis matematikai háttere (haladó)

A főkomponens analízis számos, ekvivalens módon levezethető matematikai formában, itt most az egyik legegyszerűbbel foglalkozunk. A főkomponens analízis alapkérdése, hogy találhatunk-e egy olyan T transzformációt, amellyel az adatokat transzformálva (tehát $X' = XT$) a kovarianciamátrix diagonálmátrixszá alakul:

$$\mathrm{cov}(X') = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_N^2 \end{bmatrix}$$

Ennél a pontnál két dolgot kell kiemelnünk. A T transzformáció ortogonális transzformáció, tehát ortogonális mátrix, amelynek következménye, hogy az egyes bázisvektorokat úgy transzformálja, hogy azok egymásra merőlegesek maradnak. Speciálisan, két és három dimenzióban forgatásról beszélhetünk, azaz a transzformáció az adathalmazt elforgatja. Másrészt, vegyük észre, hogy a diagonális kovarianca mátrix legfontosabb jelentése, hogy az egyes változók korrelálatlanná válnak, azaz, közöttük nincs összefüggés. *Pongyolán fogalmazva, forgassuk el az adatokat úgy, hogy a két tengelyünk irányában az adatok korrelálatlanok legyenek!*

Szerencsére a matematika erre kész megoldással jelentkezik, hiszen az ún. *spektrálfelbontás* azt mondja, hogy minden kvadratikus mátrixhoz létezik olyan bázis (transzformáció), amelyben felírható diagonálmátrixként. Azaz, ha létezik egy $N \times N$ méretű A mátrix, akkor

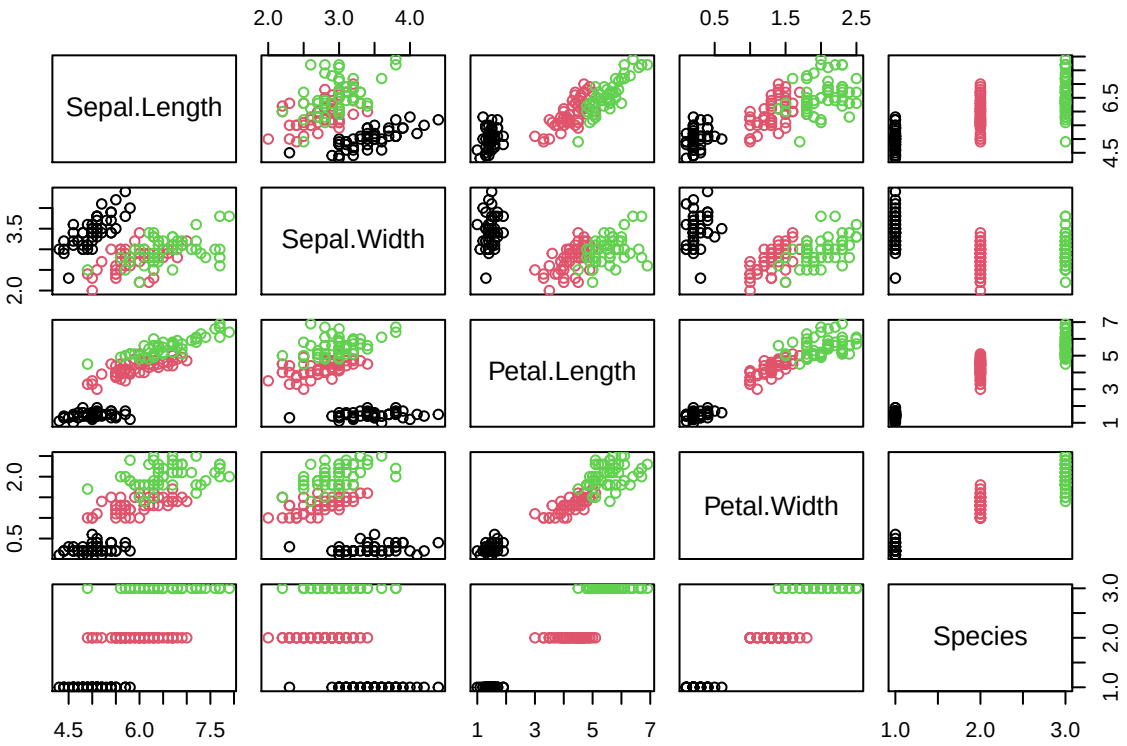
$$A = UDU^{-1} = U \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} U^{-1}$$

Ebben a felírásban a λ_i értékek az ún. **sajátértékek** és az U mátrix oszlopaiban az adott sajátértékhez tartozó **sajátvektorok** állnak. A sajátértékeket és a sajátvektorokat általában nagyság szerinti sorrendben szokták szerepeltetni, azaz $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$.

Többdimenziós-skálázás (Multi Dimensional Scaling - MDS)

Némely esetben szükség lehet olyan reprezentációra, amikor az egyes távolságokat szeretnénk megőrizni az adatpontok között. Az MDS algoritmus egy olyan kisebb dimenziós ponthalmazt generál, amelyben az egyes pontok távolságát próbálja megőrizni az egyes pontok között.

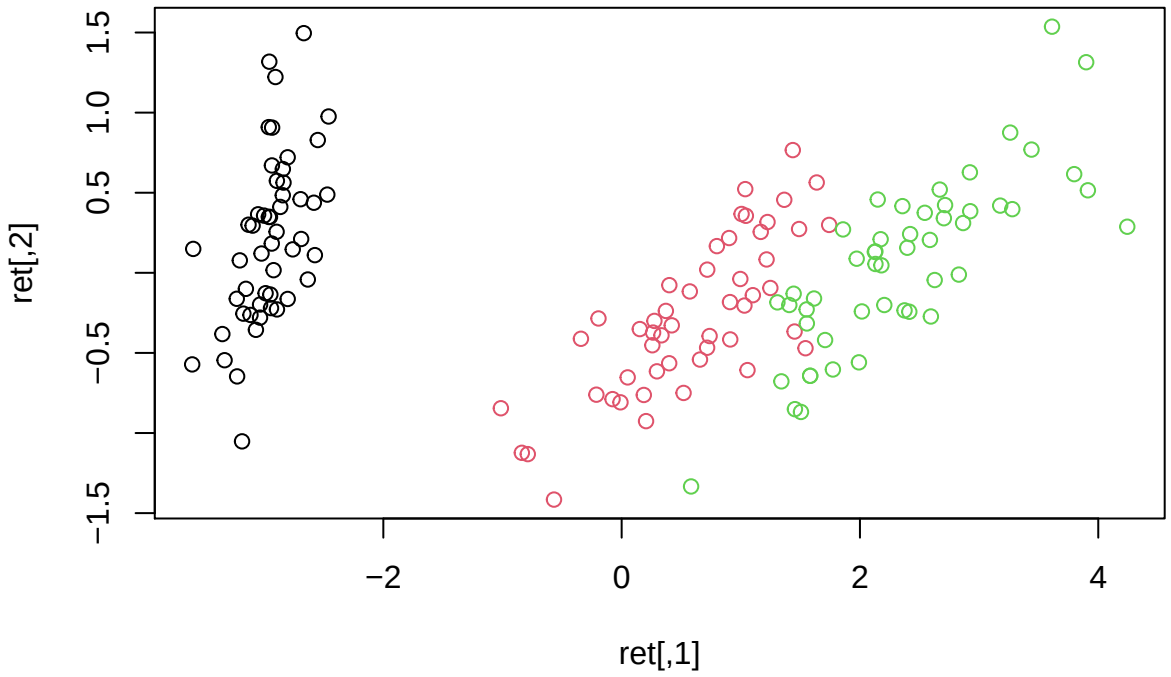
```
plot(iris,col=iris$Species)
```




```
d<-dist(iris)
```

```
## Warning in stats::dist(cdt(x), ...): NAs introduced by coercion
```

```
ret<-cmdscale(d,2)
plot(ret,col=iris$Species)
```



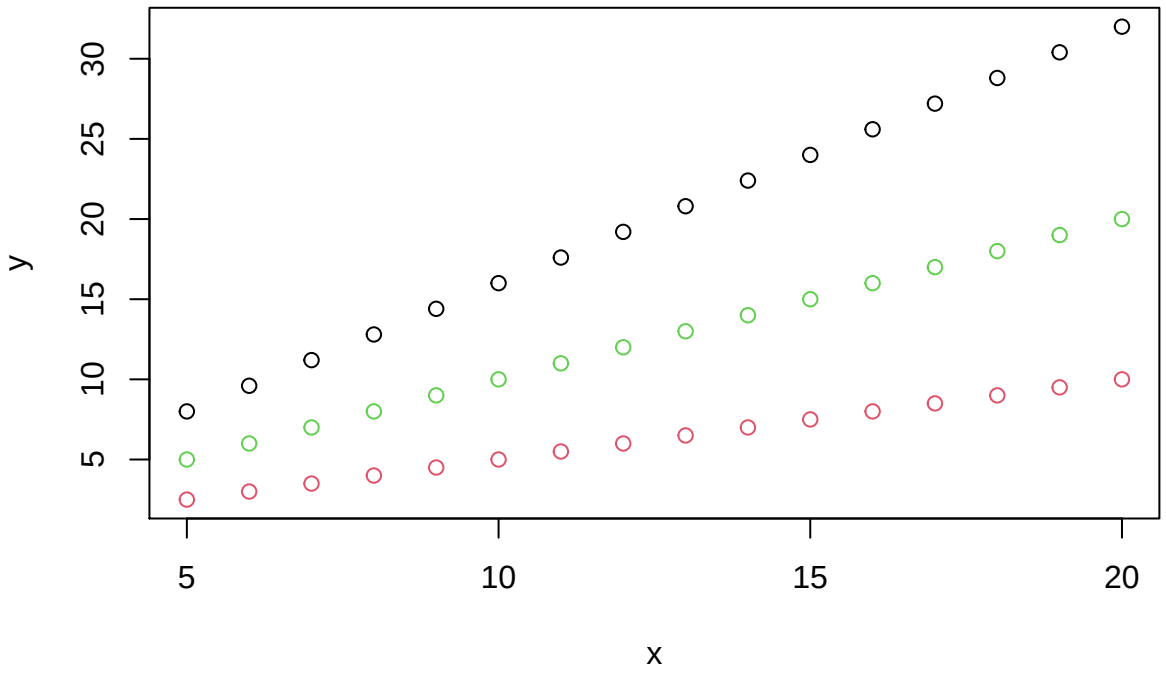
Lineáris diszkriminancia-analízis (haladó)

A lineáris diszkriminancia-analízis (LDA - Linear Discriminant Analysis) egy speciális módszer az adatok transzformálására, amely lehetővé teszi egyes különálló osztályok lehető legnagyobb mértékű elkülönítését. Az LDA feltételezi tehát, hogy az adatainkban meg tudunk különböztetni osztályokat (azaz csoportokat) az adatok között. Tekintsük például az alábbi adathalmazt:

```
d<-read.csv("lda2.csv")
head(d)
```

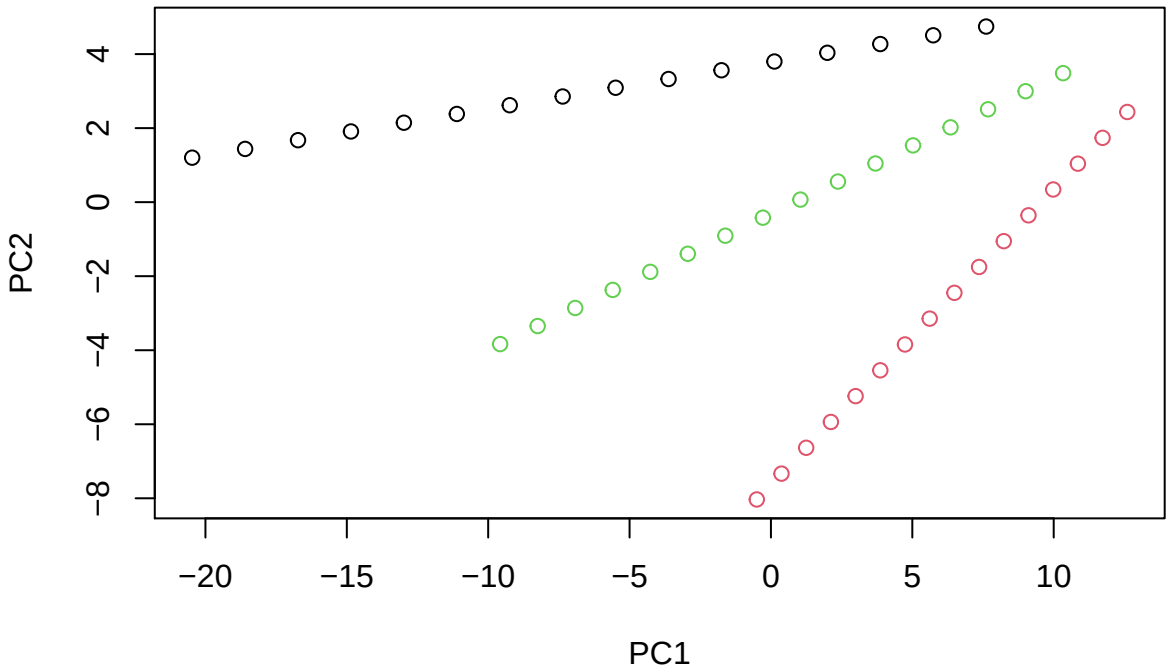
```
##    x    y cl
## 1  5  8.0  1
## 2  6  9.6  1
## 3  7 11.2  1
## 4  8 12.8  1
## 5  9 14.4  1
## 6 10 16.0  1
```

```
plot(y~x,data=d,col=d$cl)
```

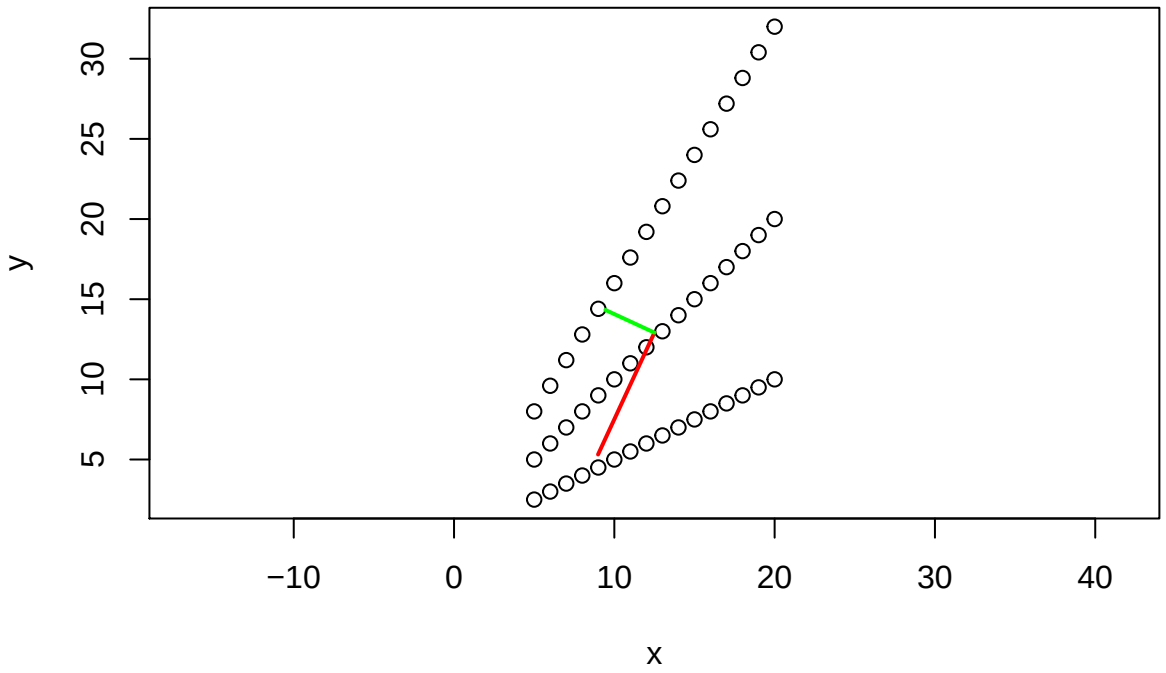


Ha erre az adathalmazra főkomponens analízis futtatunk, akkor az alábbi transzformációt kapjuk:

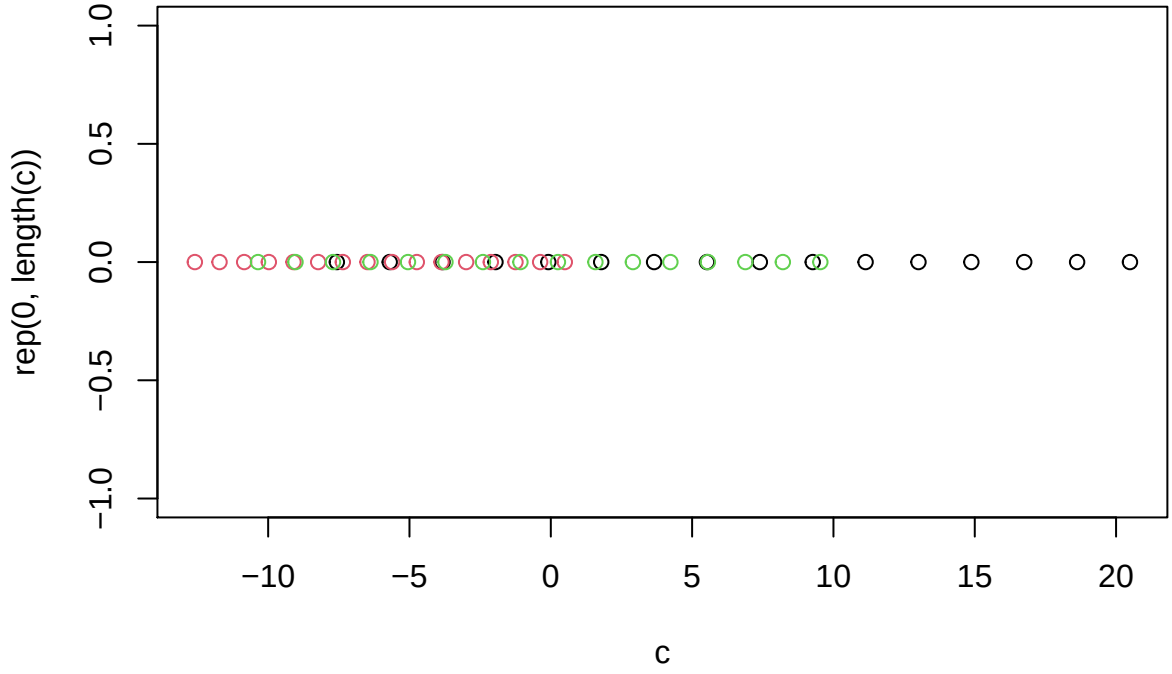
```
p<-prcomp(d[1:2])
plot(p$x,col=d$cl)
```




```
plot(d[1:2],asp=1)
m <- apply(d[1:2],2,mean)
lines(t(matrix(c(0,0,p$rotation[,1]*p$sdev[1])+m,2,2)),
      col="red",lwd=2)
lines(t(matrix(c(0,0,p$rotation[,2]*p$sdev[2])+m,2,2)),
      col="green",lwd=2)
```

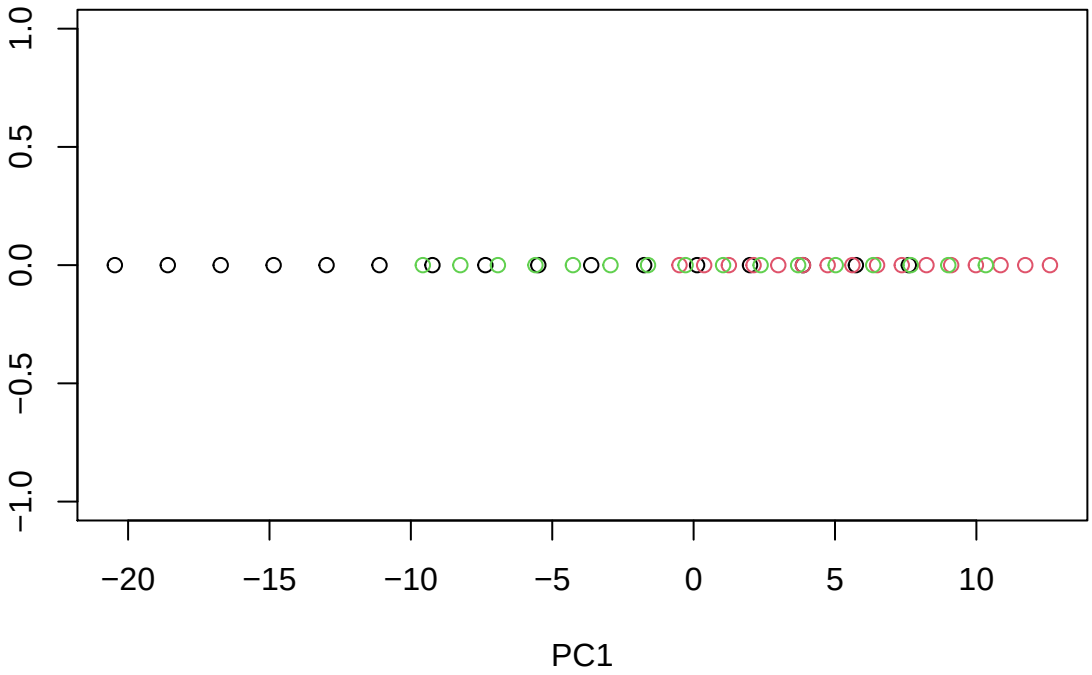



```
distance<-dist(d)
c<-cmdscale(distance,1)
plot(c,rep(0,length(c)),col=d$cl)
```



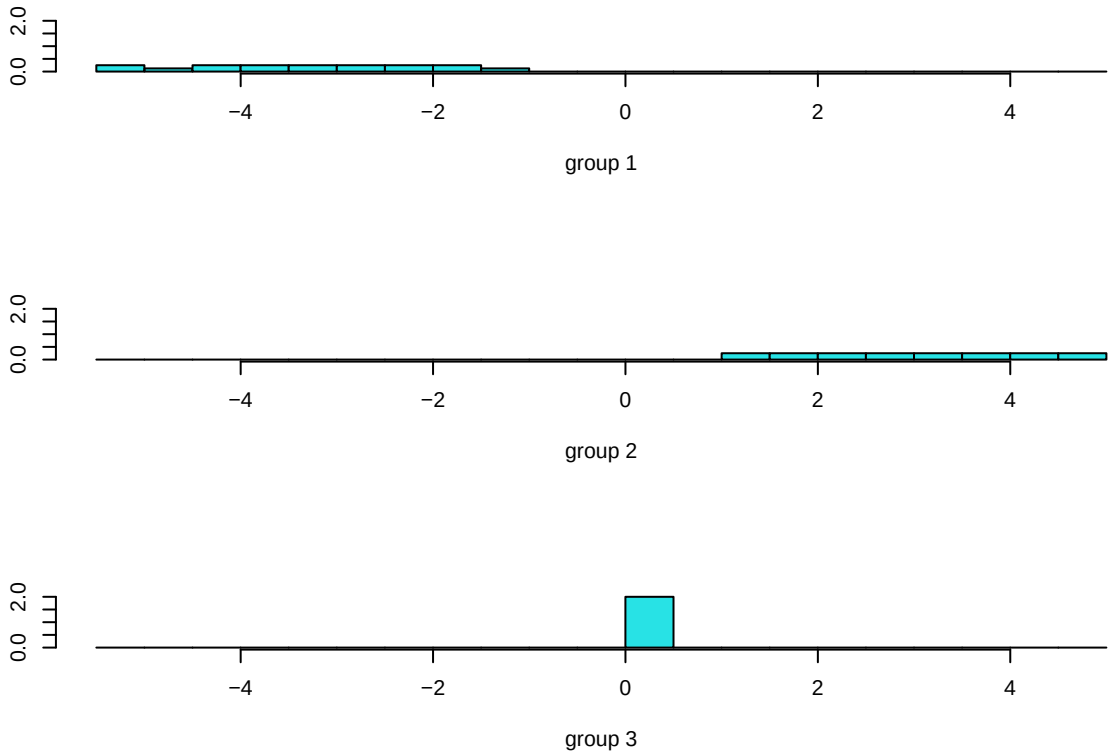
Ha ezt az adathalmazt levetítjük az első komponensre (elvégezzük a dimenziócsökkentést), akkor sajnos az egyes osztályok nem különülnek el, ahogy azt az ábrán látjuk:

```
plot(p$x[,1],rep(0,dim(p$x)[1]),col=d$cl,ylab="",xlab="PC1")
```

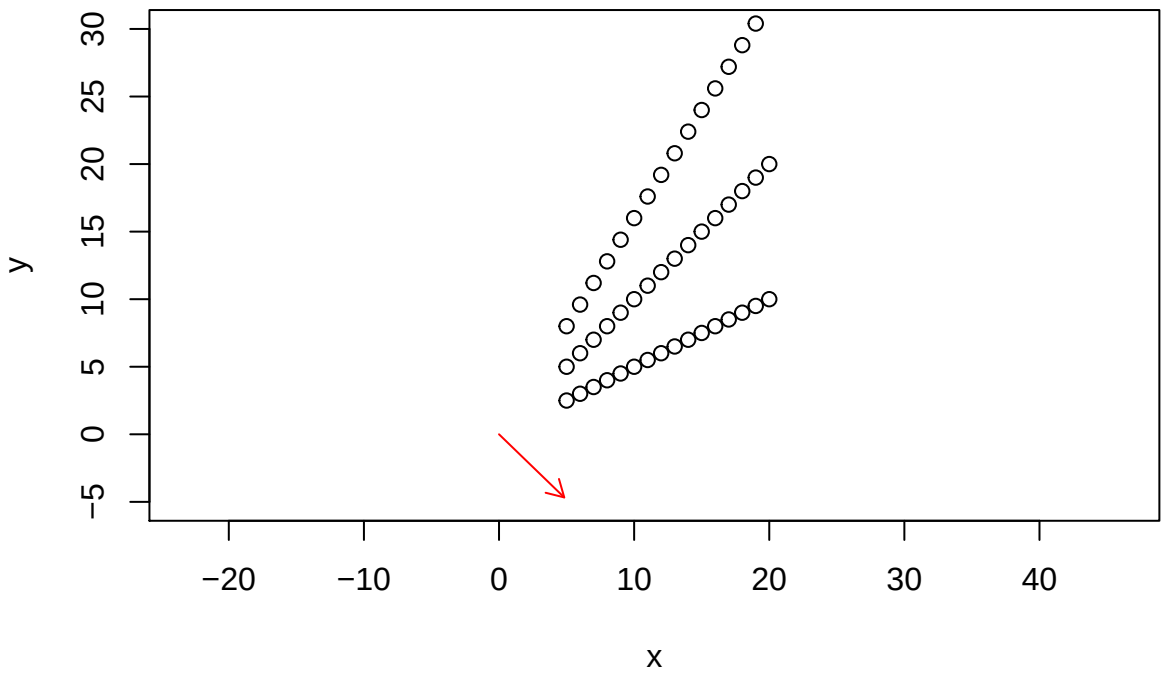


Az ún. lineáris diszkriminancia-analízis ezzel szemben egy olyan tengelyt keres az adathalmazban, amely mentén az egyes osztályok távolsága maximális lesz, azaz elsősorban akkor hasznos ez a megoldás, ha az egyes adatcsomoprtokat próbáljuk elkülöníteni (diszkriminálni). Az LDA algoritmust részletesen nem mutatjuk be itt, arról a (Fisher 1936) műben olvashatunk. Az LDA algoritmus futtatására a `lda` függvény használható:

```
g<-lda(cl ~ ., data=d)
plot(g)
```



```
c<-coef(g)
plot(d[1:2],xlim=c(-2,25),ylim=c(-5,30),asp=1)
arrows(0,0,c[1]*10,c[2]*10,length=0.1,col="red")
```



Példa

A következőkben egy példát tekintünk meg a dimenziócsökkentés témakörében.

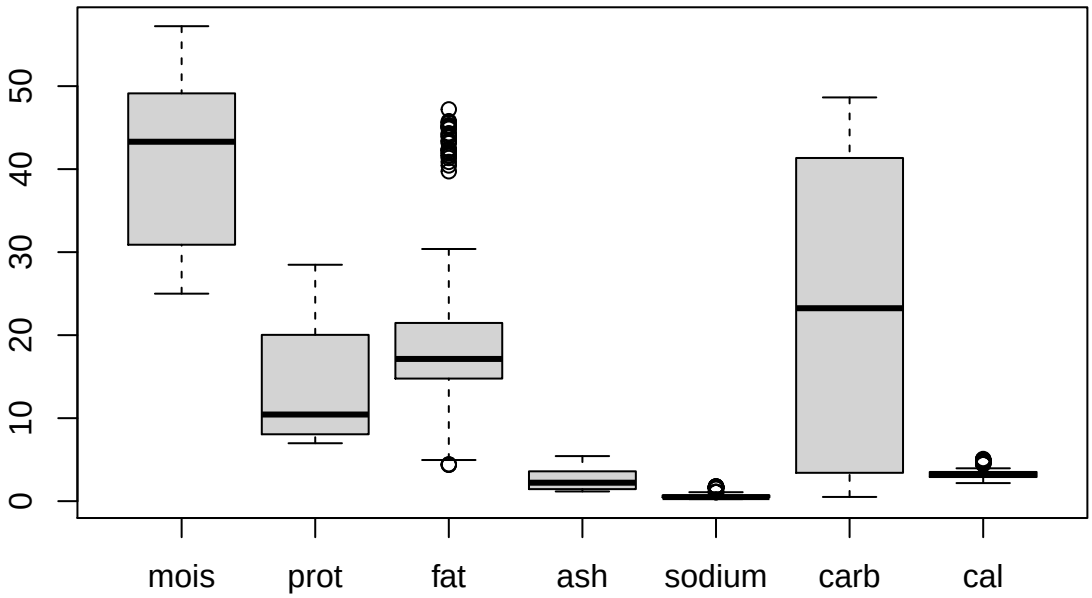
Pizza értékelés

Az adathalmaz az egyes pizzatípusok tápanyagtartalmát mutatja, 100 grammra vetítve. Töltsük be az adathalmazt!

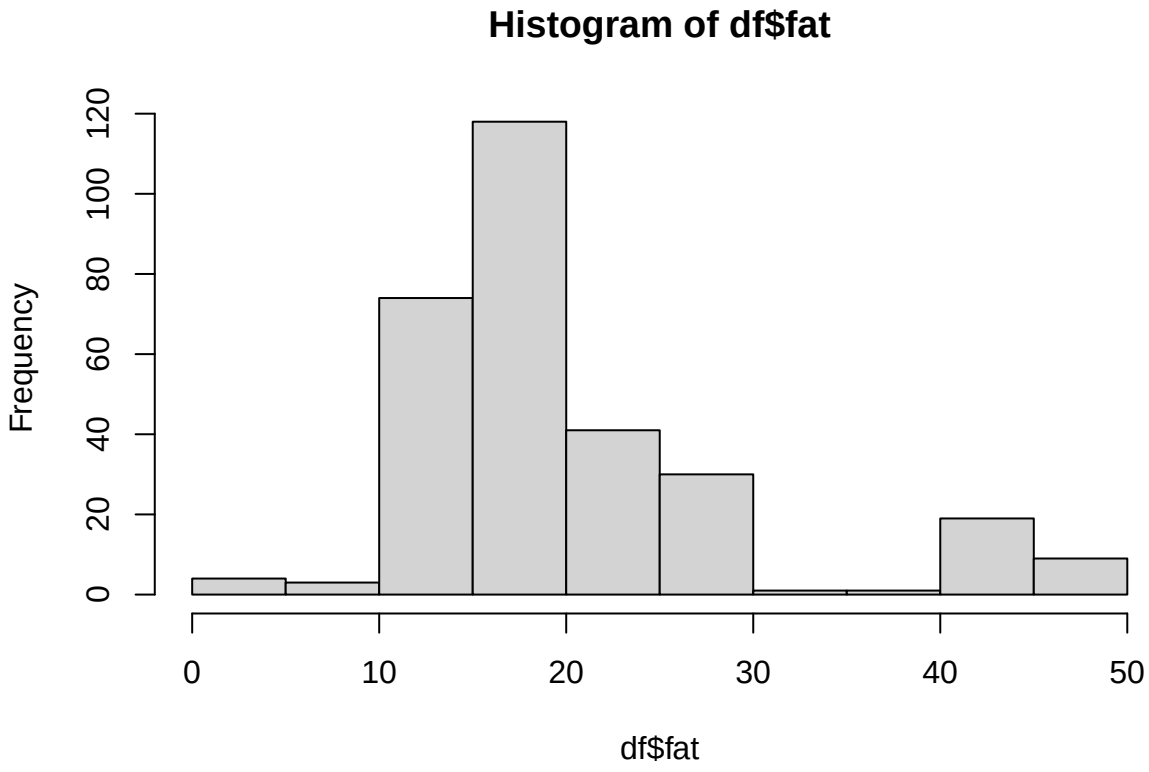
```
df<-read.csv("Pizza.csv",stringsAsFactors = T)
summary(df)
```

```
##      brand           id             mois            prot            fat
##  H      : 33   Min.   :14003   Min.   :25.00   Min.   : 6.98   Min.   : 4.38
##  D      : 32   1st Qu.:14094   1st Qu.:30.90   1st Qu.: 8.06   1st Qu.:14.77
##  J      : 32   Median :24020   Median :43.30   Median :10.44   Median :17.14
##  B      : 31   Mean   :20841   Mean   :40.90   Mean   :13.37   Mean   :20.23
##  F      : 30   3rd Qu.:24110   3rd Qu.:49.12   3rd Qu.:20.02   3rd Qu.:21.43
##  A      : 29   Max.   :34045   Max.   :57.22   Max.   :28.48   Max.   :47.20
##  (Other):113
##       ash             sodium            carb             cal
##  Min.   :1.170   Min.   :0.2500   Min.   : 0.510   Min.   :2.180
##  1st Qu.:1.450   1st Qu.:0.4500   1st Qu.: 3.467   1st Qu.:2.910
##  Median :2.225   Median :0.4900   Median :23.245   Median :3.215
##  Mean   :2.633   Mean   :0.6694   Mean   :22.865   Mean   :3.271
##  3rd Qu.:3.592   3rd Qu.:0.7025   3rd Qu.:41.337   3rd Qu.:3.520
##  Max.   :5.430   Max.   :1.7900   Max.   :48.640   Max.   :5.080
##
```

```
boxplot(df[,-(1:2)])
```

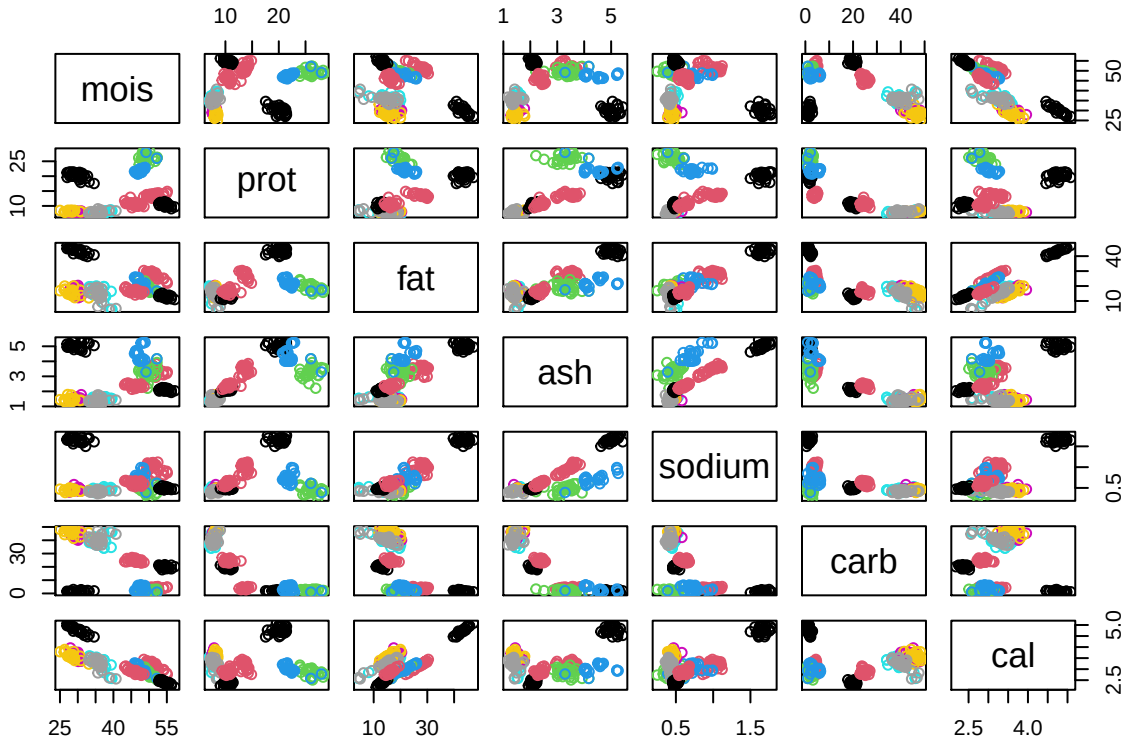



```
hist(df$fat)
```



Ábrázoljuk a tápanyagtartalmat márkánként külön színnel! Ha megviszgáljuk a korrelációs mátrixot, láthatjuk, hogy az egyes adatok között erős korrelációk találhatóak:

```
plot(df[,3:9],col=as.factor(df$brand))
```




```
cor(df[,3:9])
```

```
##              mois       prot        fat        ash     sodium        carb
## mois    1.0000000  0.3602477 -0.1713182  0.2655555 -0.1022789 -0.59180165
## prot    0.3602477  1.0000000  0.4980017  0.8238437  0.4291295 -0.85354226
## fat    -0.1713182  0.4980017  1.0000000  0.7916340  0.9333252 -0.64023817
## ash     0.2655555  0.8238437  0.7916340  1.0000000  0.8081221 -0.89898837
## sodium -0.1022789  0.4291295  0.9333252  0.8081221  1.0000000 -0.62017634
## carb   -0.5918017 -0.8535423 -0.6402382 -0.8989884 -0.6201763  1.00000000
## cal    -0.7644405  0.0702581  0.7645671  0.3264685  0.6719575 -0.02348458
##                cal
## mois   -0.76444054
## prot    0.07025810
## fat     0.76456710
## ash     0.32646845
## sodium  0.67195750
## carb   -0.02348458
## cal     1.00000000
```

Végezzünk főkomponens analízist, de figyeljünk oda, hogy normalizáljuk az adatokat, hiszen az egyes tápanyagok abszolút mértéke hibás következtetésekre sarkallhat minket:

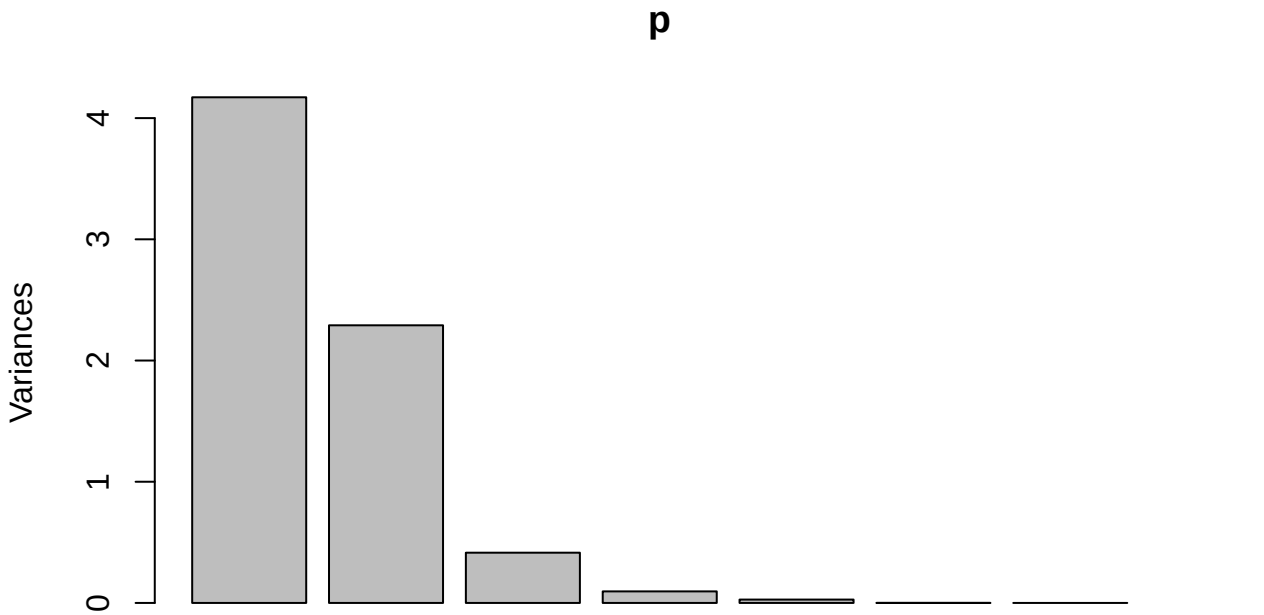
```
p<-prcomp(df[,3:9],scale=T)
summary(p)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6       PC7
## Standard deviation     2.042 1.5134 0.64387 0.3085 0.16636 0.01837 0.003085
## Proportion of Variance 0.596 0.3272 0.05922 0.0136 0.00395 0.00005 0.000000
## Cumulative Proportion  0.596 0.9232 0.98240 0.9960 0.99995 1.00000 1.000000
```

```
print(p)
```

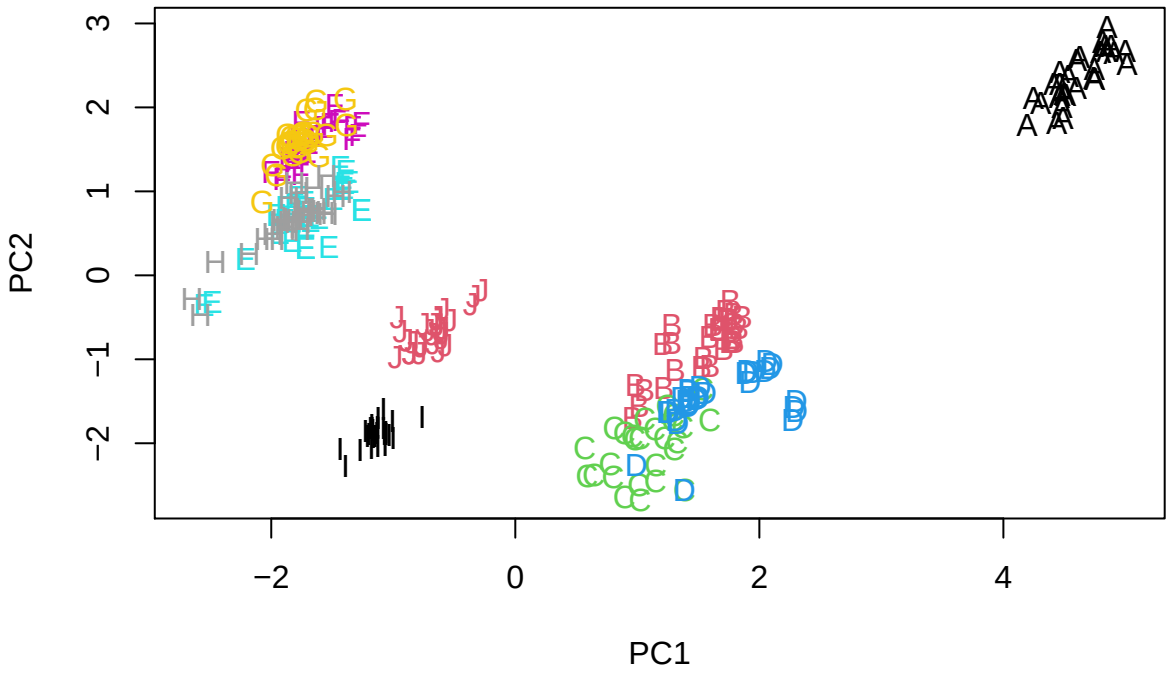
```
## Standard deviations (1, .., p=7):
## [1] 2.042494038 1.513425713 0.643865158 0.308503205 0.166364113 0.018374149
## [7] 0.003085252
##
## Rotation (n x k) = (7 x 7):
##                PC1        PC2         PC3        PC4          PC5           PC6
## mois    0.06470937 -0.6282759  0.42166894  0.2207216  0.006470293 -0.4464499018
## prot    0.37876090 -0.2697067 -0.74602744  0.0105932  0.387982788  0.0001715203
## fat     0.44666592  0.2343791  0.19930871  0.5070422 -0.173367634  0.5254028685
## ash     0.47188953 -0.1109904 -0.05627269 -0.5523985 -0.670885701 -0.0588609281
## sodium  0.43570289  0.2016617  0.45516887 -0.4462769  0.602614079 -0.0031309852
## carb   -0.42491371  0.3203121 -0.05223651 -0.3343395 -0.007436899  0.0005088535
## cal     0.24448730  0.5674576 -0.11331559  0.2792632 -0.078003175 -0.7219138527
##                 PC7
## mois   -0.4185690354
## prot   -0.2767646428
## fat    -0.3776715255
## ash    -0.0560214003
## sodium  0.0005243238
## carb   -0.7760679112
## cal    -0.0120598098
```

```
plot(p)
```

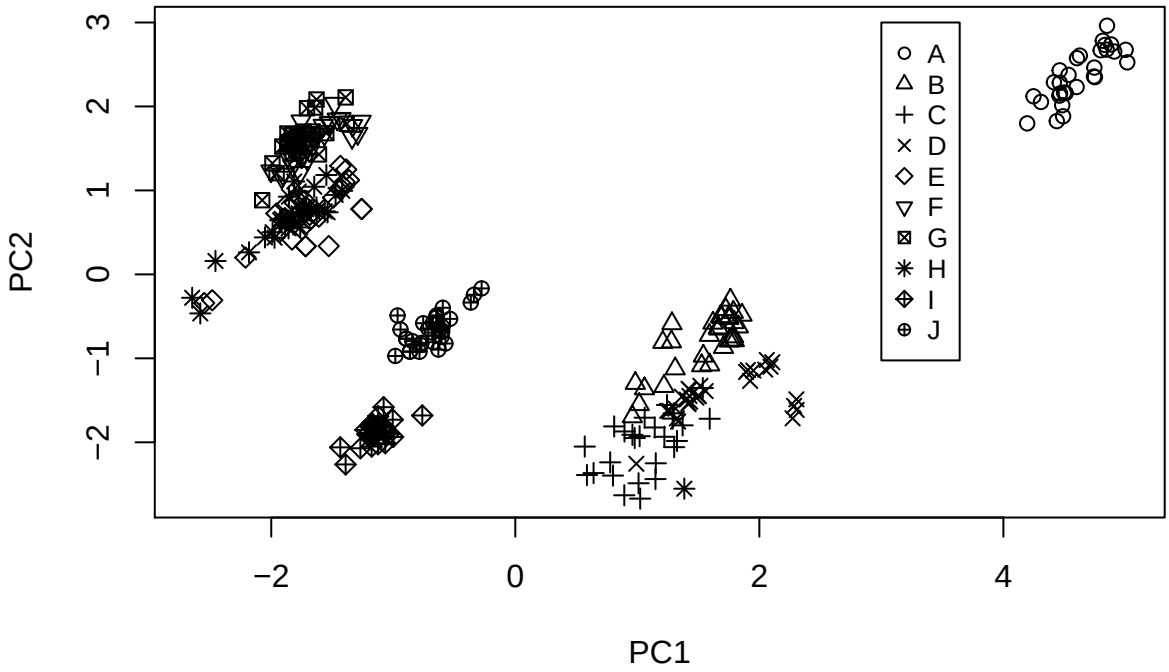


Az első két főkomponens a variancia 92 százalékáért felel, ez fontos információ! Ábrázoljuk most a két első főkomponens síkjában az adathalmazt, jelölve az egyes márkákat!

```
plot(p$x,type='n')
text(p$x, labels=df$brand, col=as.numeric(df$brand))
```

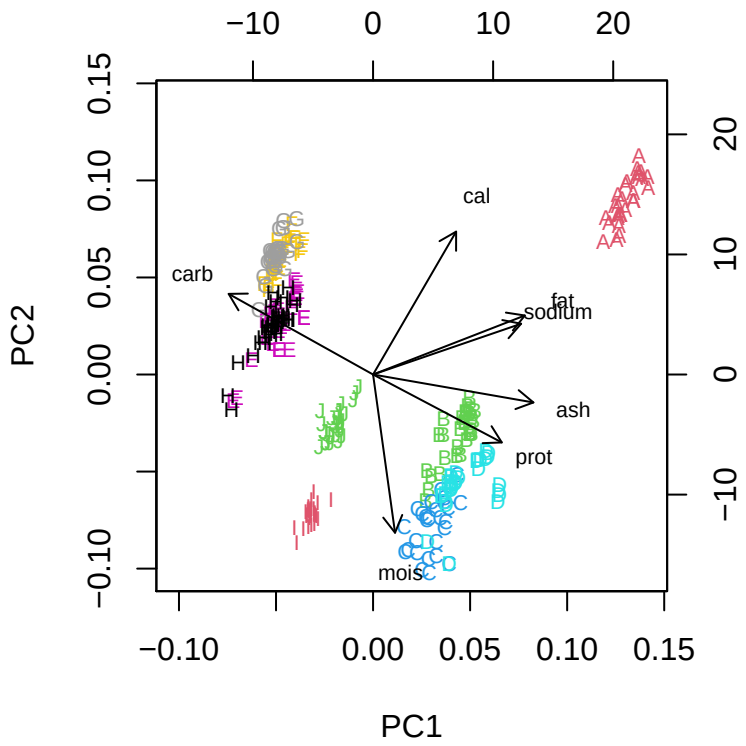



```
plot(p$x,pch=as.numeric(df$brand))
legend(3,3,legend=levels(df$brand),pch=1:length(levels(df$brand)),cex=0.8)
```



Vizsgáljuk meg a biplot ábrát, amely az egyes változók korrelációinak a vizsgálatát is segíti. Láthatjuk, hogy elsősorban a zsír és a nátrium között van erős kapcsolat.

```
palette("default")
coloredBiplot(p,pch=as.numeric(df$brand),col=1,
              xlabs=df$brand,cex=0.7,xlabs.col=as.numeric(df$brand)+1)
```

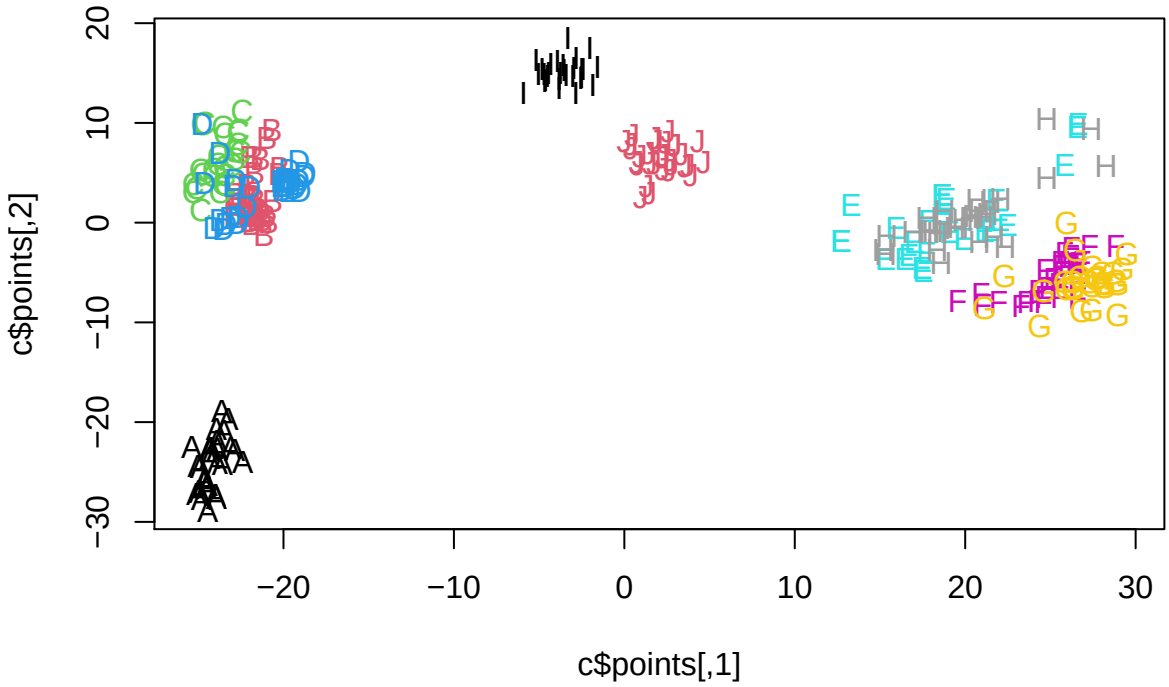


Az adathalmazra alkalmazhatunk két dimenziós MDS algoritmust is, ennek ábráján is jól elkülöníthetők az egyes pizza csoportok:

```
d<-dist(df[,-(1:2)])
c<-cmdscale(d,2,eig=T)
print(sprintf("GOF: %g",c$GOF))
```

```
## [1] "GOF: 0.968508" "GOF: 0.968508"
```

```
plot(c$points,type='n')
text(c$points,label=df$brand,col=as.numeric(df$brand))
```




```
dret <- dist(c$points)
error<- dret - d
print(sqrt(sum(error^2))/(nrow(df)^2))
```

```
## [1] 0.004921815
```

```
summary(dret)
```

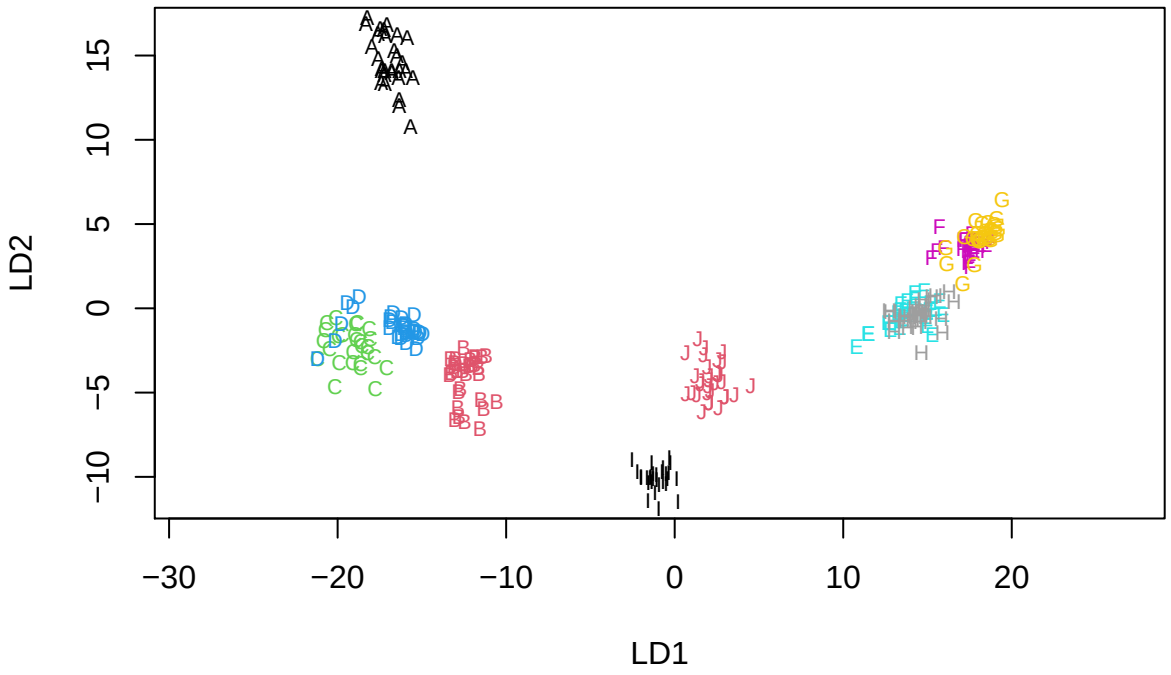
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.39   26.63   27.47   43.27   64.40
```

```
df.scaled<-scale(df[,-(1:2)])
p<-lda(df.scaled,df$brand)
print(p)
```

```
## Call:
## lda(df.scaled, grouping = df$brand)
##
## Prior probabilities of groups:
##         A         B         C         D         E         F         G
## 0.09666667 0.10333333 0.09000000 0.10666667 0.09333333 0.10000000 0.09666667
##         H         I         J
## 0.11000000 0.09666667 0.10666667
##
## Group means:
##         mois        prot        fat        ash     sodium       carb
## A -1.1448629  1.04651292  2.5867031  1.8754078  2.6644695 -1.1857016
## B  1.0891542  0.04120716  0.8234259  0.6539337  0.8517136 -1.0479967
## C  0.8975560  1.96630671 -0.1179214  0.5120012 -0.5523988 -1.1546750
## D  0.7084887  1.37661536  0.1577354  1.3252524  0.1231242 -1.0387579
## E -0.5045388 -0.87664992 -0.5697431 -0.9113495 -0.5943288  0.9277667
## F -1.2036794 -0.85098427 -0.4239095 -0.9132434 -0.5599991  1.2159126
## G -1.3254527 -0.79836831 -0.5109085 -0.9343268 -0.6091594  1.3071170
## H -0.5315526 -0.85152116 -0.6615691 -0.9664881 -0.6840397  0.9827757
```

```
## I  1.4330274 -0.46476234 -0.7986984 -0.4213181 -0.4918450 -0.1663503
## J  0.5372072 -0.43629865 -0.4351180 -0.2114995 -0.1485726  0.1037826
##           cal
## A  2.42372594
## B -0.12907716
## C -0.68078676
## D -0.43152858
## E -0.02753304
## F  0.52416459
## G  0.52282985
## H -0.07492255
## I -1.43034367
## J -0.63313034
##
## Coefficients of linear discriminants:
##              LD1         LD2          LD3        LD4        LD5        LD6
## mois   -6.335858 -11.0453512  3.299633555 -2.2240480  30.059038  -6.352910
## prot   -6.593992  -0.9358961 -3.809027134  1.7454941  25.975118  10.368211
## fat    -1.251370  -4.1936959  1.485057905 -0.8922595  39.673960  42.241417
## ash    -1.010374  -1.2197786 -2.977076473 -7.1166665   3.171028  -0.461449
## sodium -1.716567   2.9287232  5.629814731  3.0215234   3.055311  -1.200271
## carb    2.244512  -9.1741730  0.009423866 -4.9456335  69.914579  30.101694
## cal    -2.060415  -1.0093191 -0.102393138 -0.9300474 -10.849910 -36.169588
##               LD7
## mois   132.9218497
## prot    84.5099925
## fat    110.6785976
## ash     17.9193099
## sodium  -0.7950004
## carb   237.5302253
## cal     11.3732787
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5    LD6    LD7
## 0.7689 0.1396 0.0853 0.0056 0.0004 0.0001 0.0000
```

```
plot(p,dimen=2,col=as.numeric(df$brand))
```



Gyakorló feladatok

1. Végezzük el az `iris` adathalmaz PCA felbontását normalizált változókra is. Mit tapasztal?
2. A fehérje adatbázisra (protein.txt) végezze el az adatok PCA felbontását, és elemezze az adathalmazt! Szükséges az adathalmazt normalizálni? Az első két főkomponens az információk hány százalékát tartalmazza? Az első két komponens alapján tudnánk csoportosítani az egyes országokat? Az első főkomponensben melyik fehérjeforrás szerepel a legnagyobb súllyal (határozza meg a főkomponens vektorok alapján és biplot alapján is)?
3. A fehérje adathalmazra futtassunk MDS algoritmust! Hasonlítsuk össze a PCA esetében az első két főkomponenst tekintve! Milyen különbséget látunk? Gondoljuk át, van-e értelme MDS algoritmust használni erre az adathalmazra (van az egyes sorok távolságának jelentése?)!

Irodalom

Fisher, R. A. 1936. “The Use of Multiple Measurements in Taxonomic Problems.” https://digital.library.adelaide.edu.au/dspace/bitstream/2440/15227/1/138.pdf.