dimensional reduction

dimensional reduction

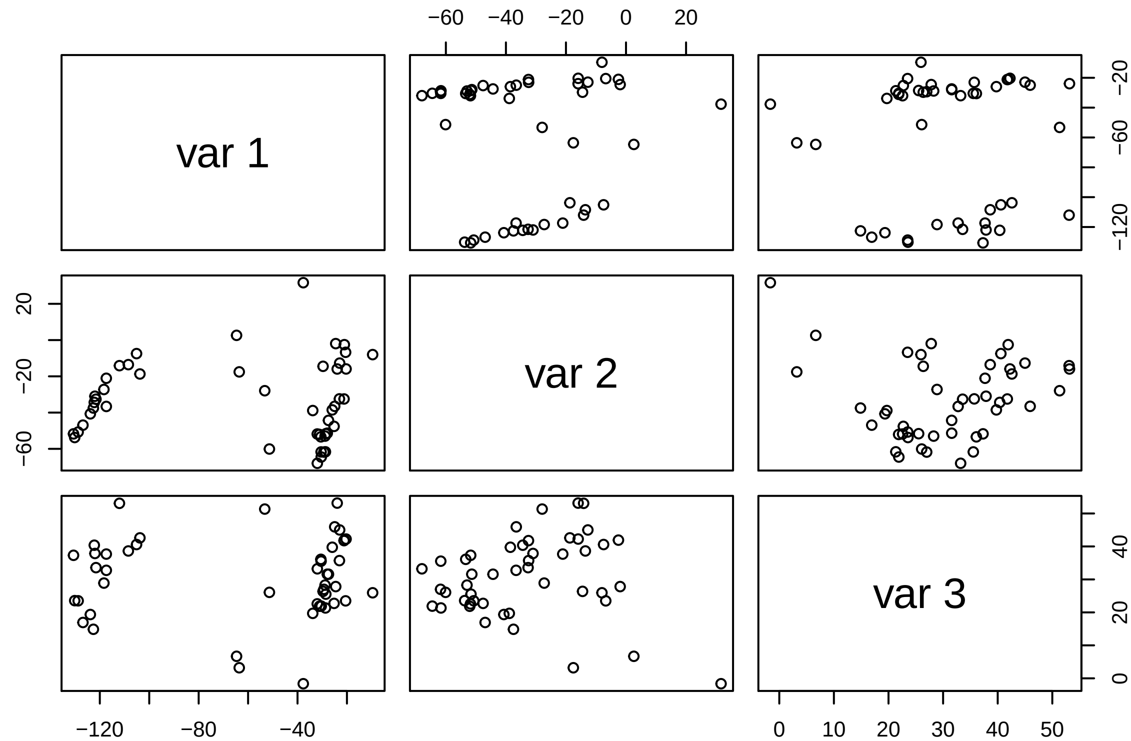
principle componenet analysis (PCA)

let's first try to code PCA:

```
myPCA <- function(x, d=2){
  #step 1: covariance estimation
  #scale function to center the data: scale=False, not normalize, only centered
  xbar=scale(x,center=TRUE,scale = FALSE)
  sigma=1/nrow(xbar)*t(xbar)%*%xbar

  #step 2: eigen-decomposition
  out=eigen(sigma)

  #step 3: return the PC axes
  return(out$vector[,1:d])
  #return(out)

}
```

now let's try it on some data:

```
data("swiss")
u=myPCA(swiss,d=3)

#now we have to project the data
xproj=as.matrix(swiss) %*% u
#xproj
#plot(xproj, type='p', pch=19) # for 1-2 dimension
pairs(xproj)#for more then 2 dimension
```



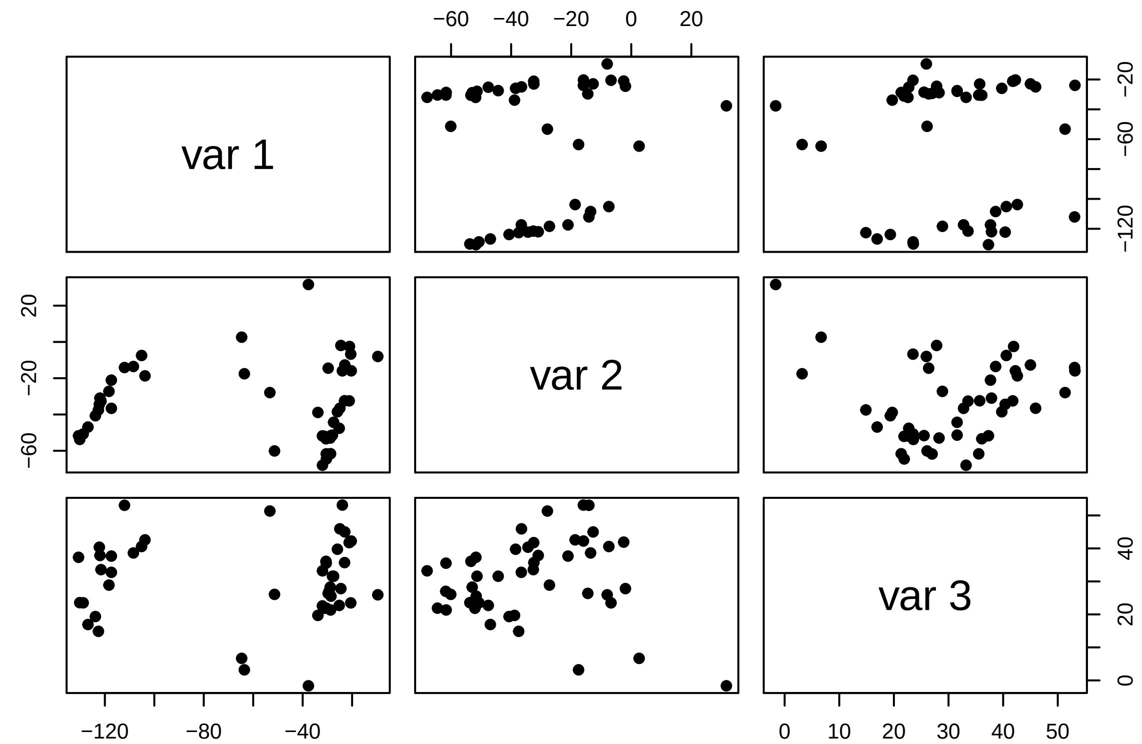

```
#so we move from 6 dimensional spaces to other number of dimensional spaces
```

now we can combine the ploting into the function

```
myPCAPlot <- function(x, d=2){
  #step 1: covariance estimation
  #scale function to center the data: scale=False, not normalize, only centered
  xbar=scale(x,center=TRUE,scale = FALSE)
  sigma=1/nrow(xbar)*t(xbar)%*%xbar

  #step 2: eigen-decomposition
  out=eigen(sigma)

  #step 3: return the PC axes
  #return(out$vector[,1:d])
  #return(out)
  u=out$vector[,1:d]
  xproj=as.matrix(x) %*% u
  if(d<=2)
    plot (xproj, type = 'p', pch = 19)
  else
    pairs(xproj,pch=19)
  return(list(u=u,lambda=out$values))
}
```

try with swiss

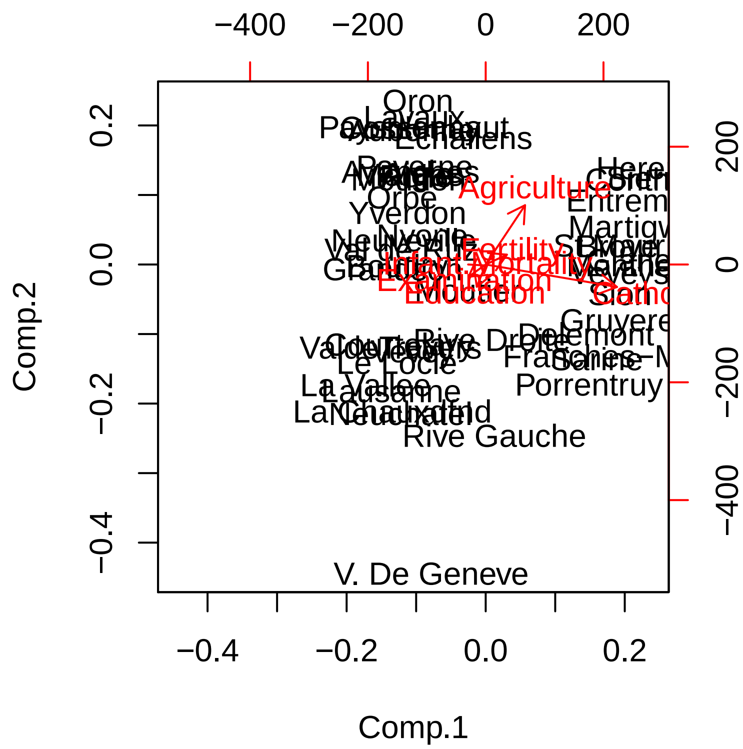
```r
myPCAPlot(swiss,d=3)
```



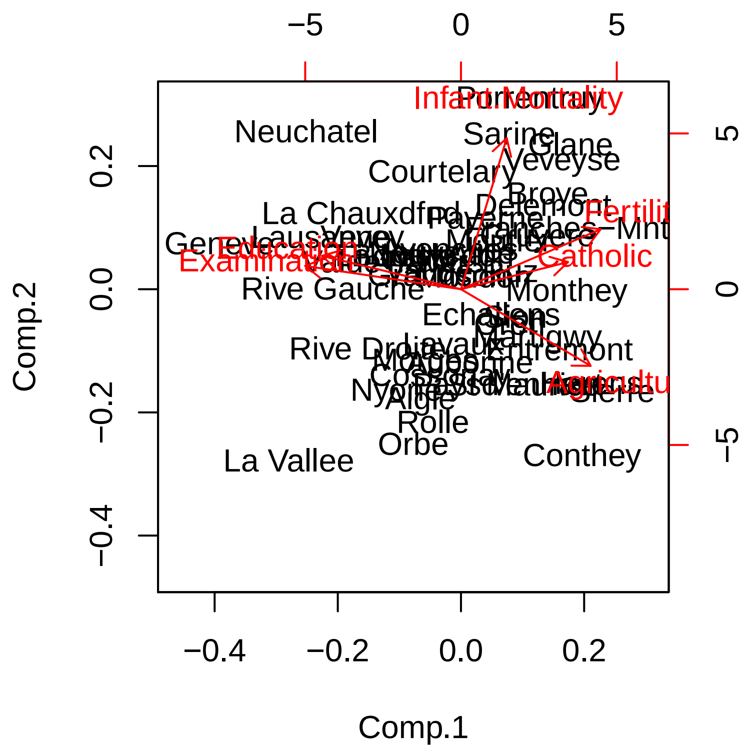

```
## $u
##              [,1]        [,2]        [,3]
## [1,] -0.15163143 -0.14270789  0.81454413
## [2,] -0.28121756 -0.85914886 -0.35256541
## [3,]  0.12207834  0.17688621 -0.18767793
## [4,]  0.06329733  0.32260928 -0.40096045
## [5,] -0.93748965  0.32543441 -0.07870742
## [6,] -0.01131739  0.01498883  0.10014161
##
## $lambda
## [1] 1880.67818  456.72826  142.19366   22.16770   13.09300    6.05952
```

if we use the R function 'princomp':

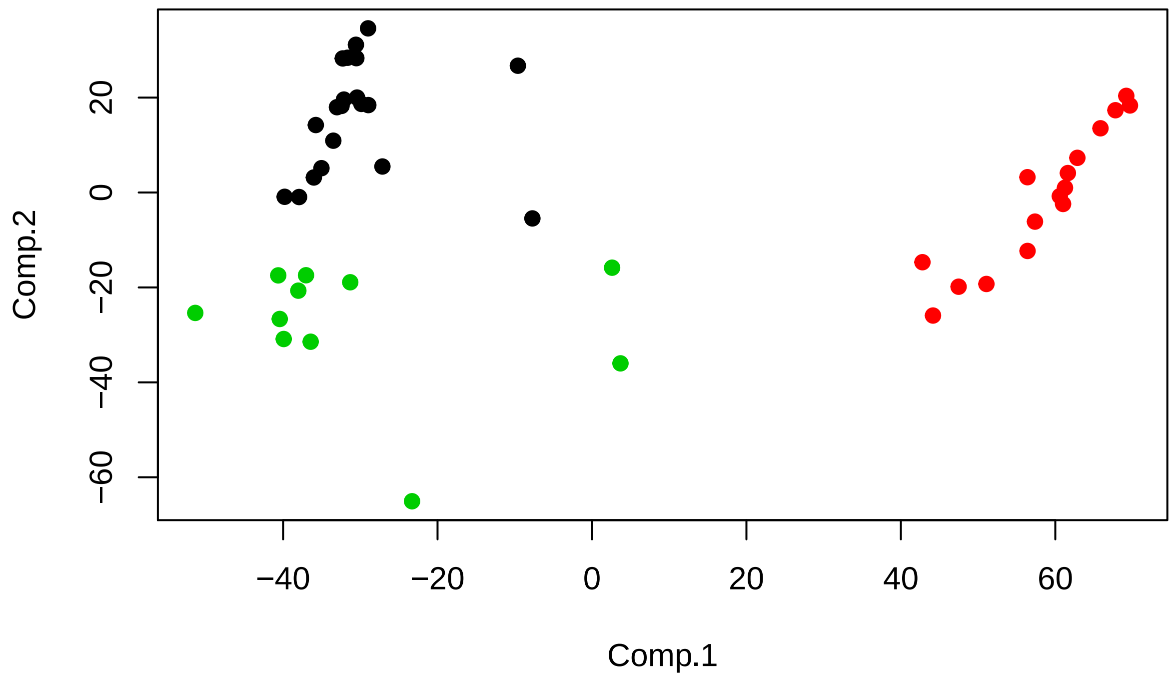
```r
#comparision between PCA and scaled PCA:
out=princomp(swiss)
out1=princomp(swiss,cor = TRUE)#scaleD ?????????
xproj=predict(out,swiss)
#par(mfrow=c(1,2))
biplot(out)
```



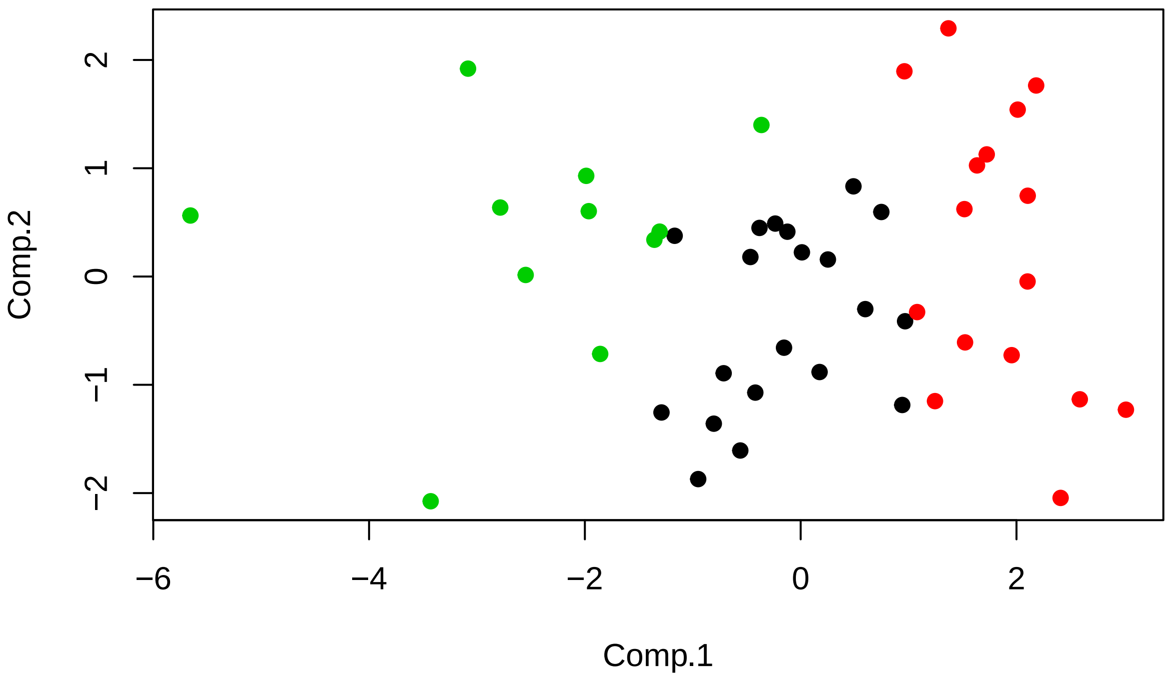

```r
biplot(out1)
```




```
clus=kmeans(swiss,3)
plot(predict(out,swiss),col=clus$cluster,pch=19)
```



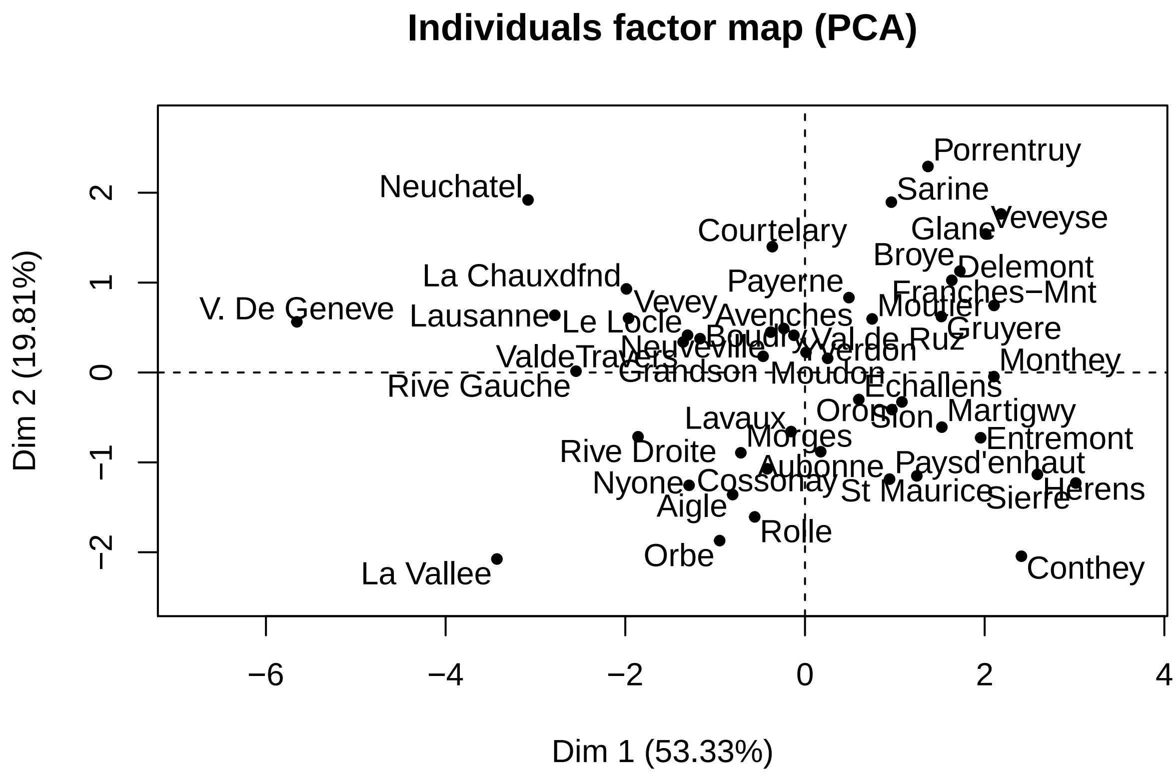

```
plot(predict(out1,swiss),col=clus$cluster,pch=19) #once it's scaled, the cluster is not that obvious
```

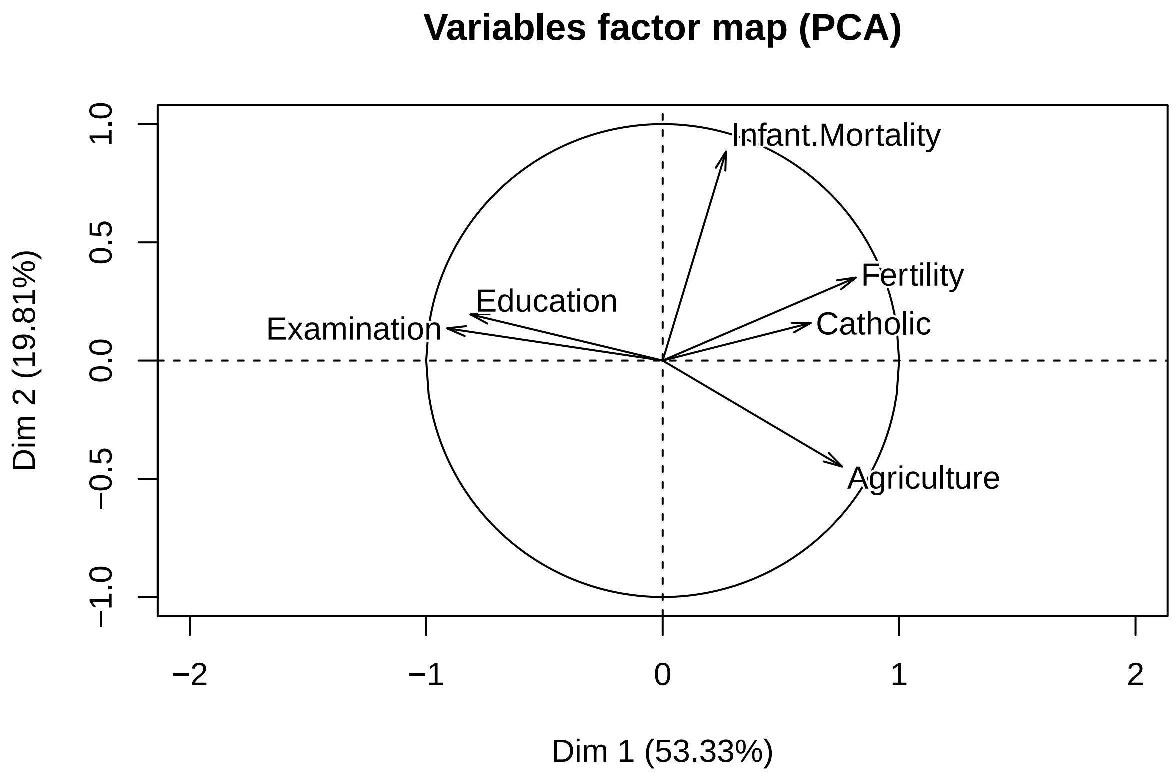



```
#it is not clear since it's combining the 2 plot
```

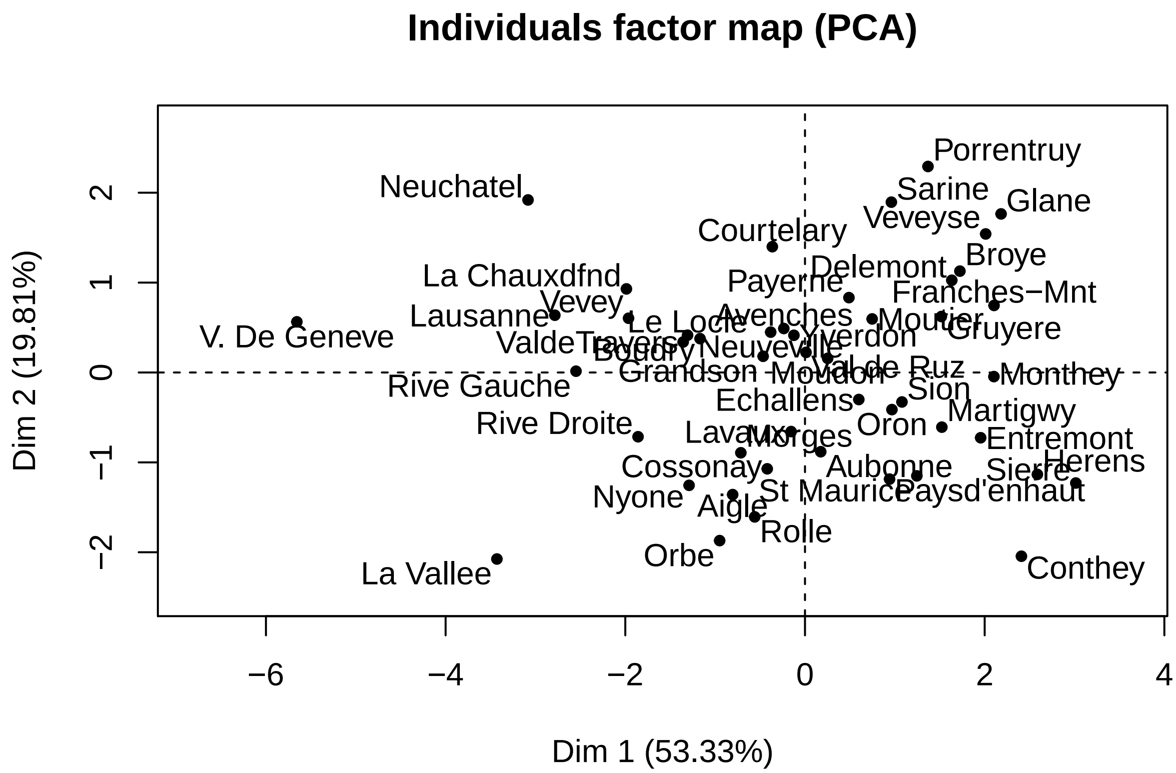
in the ‘FactoMineR’ package several additional visualizations are possible:

```
#install.packages('FactoMineR')
#here unlike biplot it seperate 2 graph
library(FactoMineR)
out=PCA(swiss)
```


Individuals factor map (PCA)
Dim 1 (53.33%)
Dim 2 (19.81%)
Porrentruy
Neuchatel
Sarine
Glane
Veveyse
Courtelary
Broye
Delemont
La Chauxdfnd
Payerne
Franches−Mnt
V. De Geneve
Vevey
Moutier
Lausanne
Le Locle
Avenches
Gruyere
Boudry
Val de Ruz
Neuveville
Yverdon
ValdeTravers
Monthey
Grandson
Moudon
Rive Gauche
Echallens
Oron
Sion
Martigwy
Lavaux
Morges
Entremont
Rive Droite
Aubonne
Paysd'enhaut
Nyone
Cossonay
St Maurice
Sierre
Herens
Aigle
Rolle
Orbe
La Vallee
Conthey


Variables factor map (PCA)
Infant.Mortality
Fertility
Catholic
Education
Examination
Agriculture
Dim 2 (19.81%)
Dim 1 (53.33%)
−2
−1
0
1
2
−1.0
−0.5
0.0
0.5
1.0

```
plot(out)
```



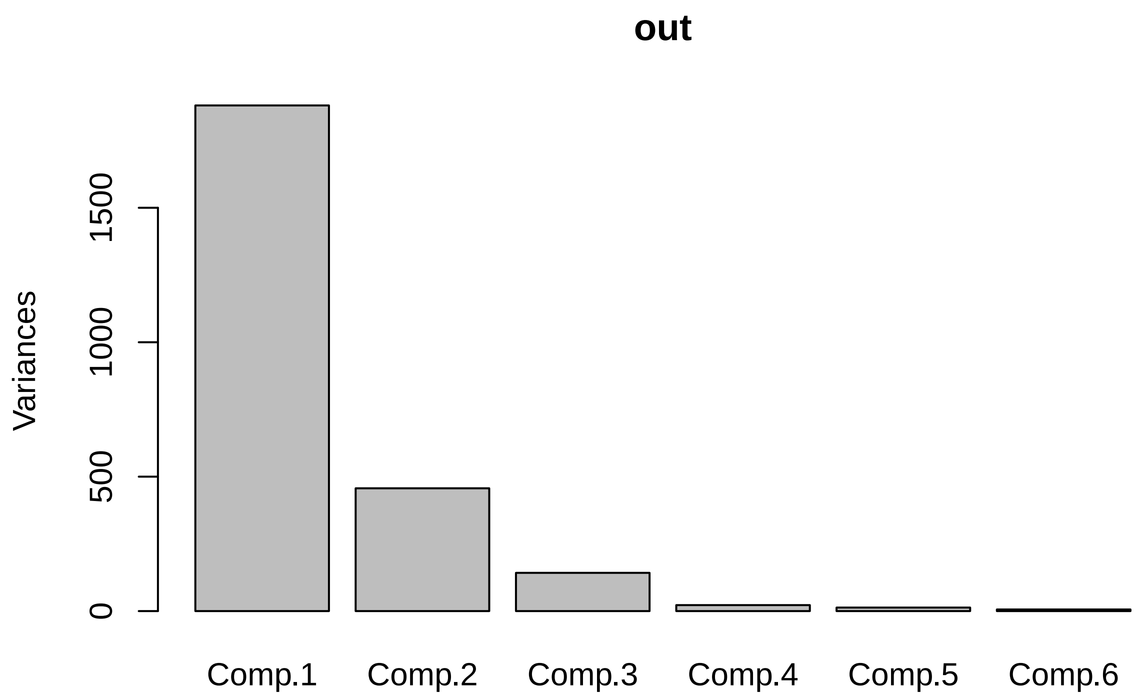
exercise: use the three methods we saw in class for chossig the right number of components to retain

90% rule,eigenvalue scree:

```
out=princomp(swiss)
out$sdev
```

```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
## 43.366787 21.371202 11.924498  4.708259  3.618425  2.461609
```

```
plot(out) #variance of the variable
```



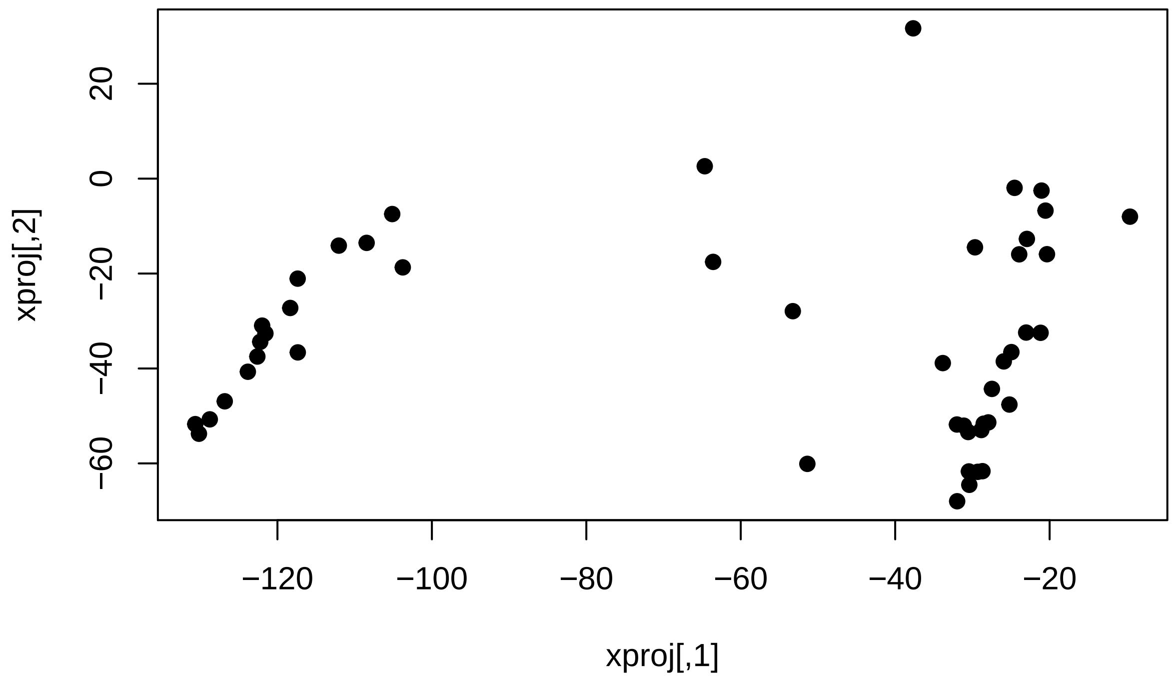

```r
summary(out)
```

```
## Importance of components:
##                           Comp.1     Comp.2      Comp.3      Comp.4
## Standard deviation     43.3667866 21.3712016 11.92449843 4.708258751
## Proportion of Variance  0.7460284  0.1811752  0.05640546 0.008793495
## Cumulative Proportion   0.7460284  0.9272036  0.98360907 0.992402568
##                             Comp.5      Comp.6
## Standard deviation     3.618425207 2.461609258
## Proportion of Variance 0.005193739 0.002403694
## Cumulative Proportion  0.997596306 1.000000000
```

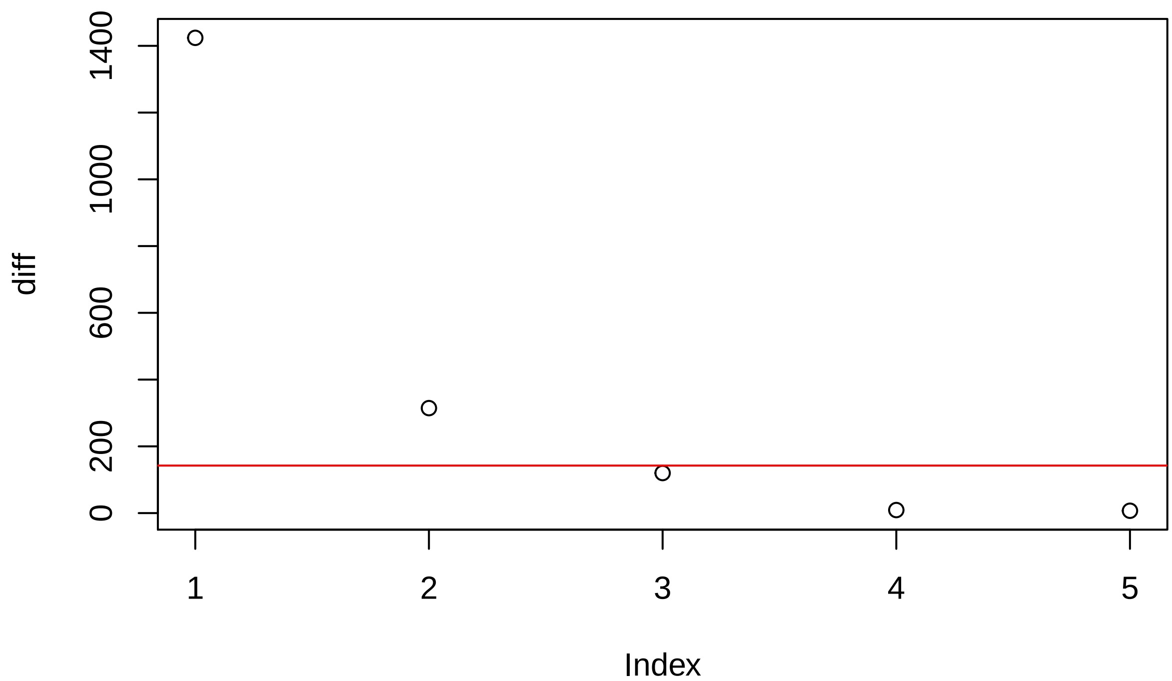
```r
# with the 90% rule we can take already first 2 variable
#eigenvalue scree: 3 is still important so choose between 2-3
```

The Cattell's test:

```r
out=myPCAPlot(swiss)
```




```
diff=abs(diff(out$lambda))
plot(diff)
abline(h=0.1*max(diff),col='red')
```




```
#cattell say we should retain 2+1=3 dimension
```

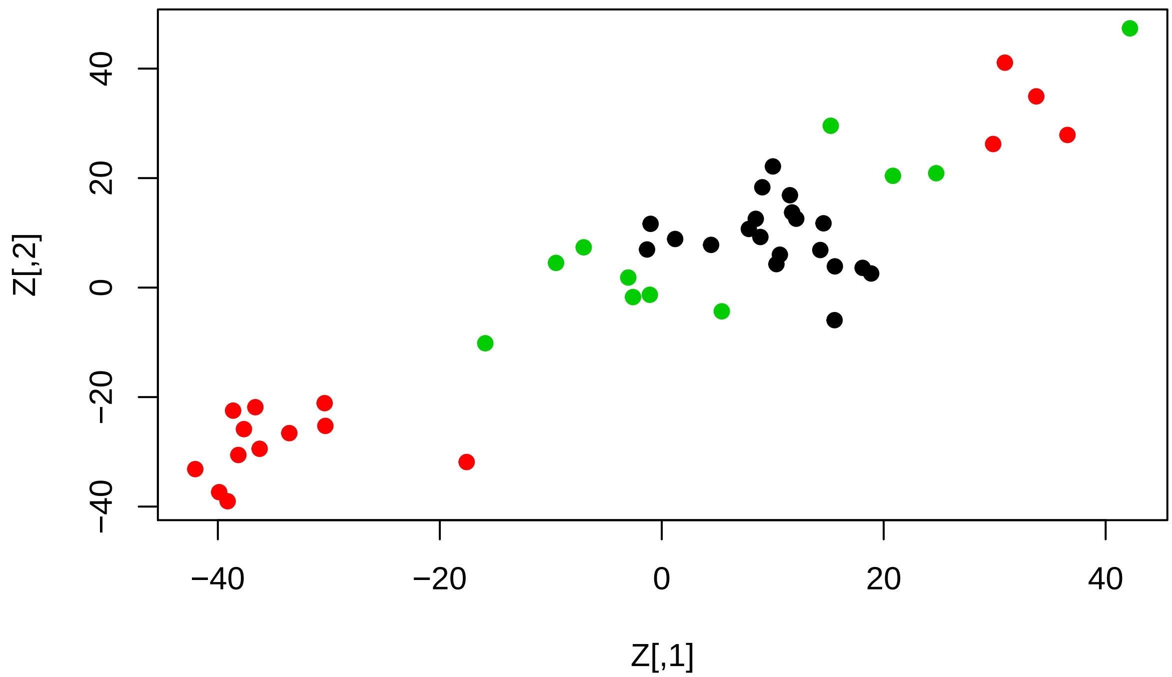
MDS: Multi dimensional scaling

```
myMDS<- function(x,d=2){
  n=nrow(x);
  D =as.matrix(dist(x)) #dist() for continuous data, if it's other type, we need to find the apporpriat
  fun<-function(par,D){
    Z=matrix(par,ncol=d)
    s=0
    for(i in 1:n)
      for(j in 1:n)
        s=s+(D[i,j]-sqrt(sum(Z[i,]-Z[j,])^2))^2
    return(s) #return s
  }
  zstart=matrix(runif(n*d),ncol=d)
  out=optim(zstart,fun,D=D,method='SANN')
  Z=out$par
}

Z=myMDS(swiss,d=2)
Z
```

```
##              [,1]        [,2]
```

```
##  [1,]  -1.072692  -1.311884
##  [2,]  29.862481  26.223676
##  [3,] -30.311328 -25.262600
##  [4,]  10.016776  22.131668
##  [5,]   4.446070   7.800568
##  [6,] -17.579059 -31.863214
##  [7,] -38.623252 -22.475850
##  [8,] -36.234708 -29.448289
##  [9,] -36.619207 -21.847400
## [10,] -30.376528 -21.108971
## [11,]  36.559829  27.880655
## [12,]  18.871348   2.573557
## [13,]  12.115318  12.580564
## [14,]   8.471652  12.559274
## [15,]  14.576332  11.739828
## [16,]  -1.326661   6.960145
## [17,]  15.573996  -5.946418
## [18,]  -2.588884  -1.732843
## [19,]  -9.525880   4.508308
## [20,]   9.061513  18.322361
## [21,]  14.286759   6.860202
## [22,]   7.848856  10.709643
## [23,]  10.334114   4.290972
## [24,]  15.599655   3.875896
## [25,]  11.551308  16.871476
## [26,]   8.889130   9.240755
## [27,]  11.747680  13.722272
## [28,]  18.103042   3.605089
## [29,]   5.411410  -4.336935
## [30,]  10.643379   6.008368
## [31,] -42.035979 -33.150837
## [32,]  30.921202  41.082104
## [33,] -39.880415 -37.368111
## [34,] -38.150340 -30.575192
## [35,] -37.650347 -25.851444
## [36,]  33.750006  34.926526
## [37,] -39.117722 -39.025319
## [38,] -33.560205 -26.582765
## [39,]  -1.006708  11.645277
## [40,]  15.230159  29.560467
## [41,]  -3.015084   1.832691
## [42,]  24.733879  20.891635
## [43,]   1.207141   8.879745
## [44,]  -7.030778   7.351308
## [45,]  42.191971  47.351505
## [46,]  20.834148  20.418903
## [47,] -15.900820 -10.168220
```

```r
plot(Z,type='p',pch=19,col=clus$cluster)
```




```
#MDS is very time consuming
```

- MDS in R 'cmdscale' Much better in performance

```
d=dist(swiss)
res=cmdscale(d,k=2,eig=TRUE)
res
```

```
## $points
##                    [,1]        [,2]
## Courtelary    37.032433 -17.4348788
## Delemont     -42.797334 -14.6876683
## Franches-Mnt -51.081639 -19.2740356
## Moutier        7.716707  -5.4587215
## Neuveville    35.032658   5.1260970
## Porrentruy   -44.161953 -25.9224124
## Broye        -56.392984   3.2255060
## Glane        -61.258244   0.9998919
## Gruyere      -56.405711 -12.3159788
## Sarine       -47.477237 -19.8509107
## Veveyse      -61.008008  -2.4123170
## Aigle         28.965873  18.4219674
## Aubonne       31.653458  28.3931122
## Avenches      32.123633  19.5819402
## Cossonay      32.274690  28.2546346
```

```
## Echallens      9.595877  26.7118107
## Grandson      39.800148  -0.9053319
## Lausanne      40.435132 -26.6464679
## La Vallee     51.376323 -25.3760191
## Lavaux        30.572130  31.1329195
## Morges        32.435116  18.2481593
## Moudon        33.028767  17.9772680
## Nyone         27.138257   5.4808462
## Orbe          35.765096  14.2149538
## Oron          28.996500  34.5897700
## Payerne       30.425130  20.0025379
## Paysd'enhaut  30.515769  28.3119655
## Rolle         29.864476  18.6702353
## Vevey         31.307144 -18.9143568
## Yverdon       33.499133  10.9174996
## Conthey      -67.788045  17.3371596
## Entremont    -65.850277  13.5345554
## Herens       -69.193539  20.3626868
## Martigwy     -62.858166   7.3053848
## Monthey      -60.583087  -0.7701055
## St Maurice   -61.629377   4.0987185
## Sierre       -69.670618  18.3442505
## Sion         -57.369412  -6.1344239
## Boudry        37.933435  -0.9525147
## La Chauxdfnd  39.920865 -30.8689579
## Le Locle      38.027990 -20.6843295
## Neuchatel     36.433011 -31.4410984
## Val de Ruz    36.020078   3.1624867
## ValdeTravers  40.627884 -17.4614718
## V. De Geneve  23.301944 -65.0529176
## Rive Droite   -2.605495 -15.8418221
## Rive Gauche   -3.688530 -35.9996173
##
## $eig
##  [1]  8.839187e+04  2.146623e+04  6.683102e+03  1.041882e+03  6.153710e+02
##  [6]  2.847974e+02  1.718857e-11  1.055757e-11  6.353581e-12  4.313025e-12
## [11]  3.096859e-12  2.205915e-12  2.081644e-12  2.027854e-12  1.995170e-12
## [16]  1.768706e-12  1.710028e-12  1.689229e-12  1.368958e-12  1.292917e-12
## [21]  1.011941e-12  6.688735e-13  6.069093e-13  5.290955e-13  3.429863e-13
## [26]  1.641147e-13  7.494036e-14  3.532351e-14 -8.270456e-14 -3.319275e-13
## [31] -4.524524e-13 -5.877947e-13 -9.702069e-13 -1.068181e-12 -1.106508e-12
## [36] -1.174723e-12 -1.205118e-12 -1.214137e-12 -1.271281e-12 -1.335532e-12
## [41] -1.514998e-12 -1.558048e-12 -1.789720e-12 -1.978327e-12 -2.077321e-12
## [46] -2.263560e-12 -4.255388e-12
##
## $x
## NULL
##
## $ac
## [1] 0
##
## $GOF
## [1] 0.9272036 0.9272036
```

```r
plot(res$points,type='p',pch=19,col=clus$cluster)
```