Modern Applyed Statistics(Chap 11)

```r
library(MASS)
library(class)
library(fastICA)
library(cluster)
options(width=65, digits=5)

# install.packages("../package/xgobi_1.2-15.tar.gz", repos = NULL, type = "source")
# install.packages("../package/RGtk2_2.20.36.tar.gz", repos = NULL, type = "source")
# install.packages("../package/rggobi_2.1.22.tar.gz", repos = NULL, type = "source")
```

11.1 Visualization methods

```r
# data load

## Iris data
ir <- rbind(iris3[,,1], iris3[,,2], iris3[,,3])
ir.species <- factor(c(rep("s", 50), rep("c", 50), rep("v", 50)))

## Crabs data
lcrabs <- log(crabs[, 4:8])
crabs.grp <- factor(c("B", "b", "O", "o")[rep(1:4, each = 50)])
```

1) Principal Component analysis

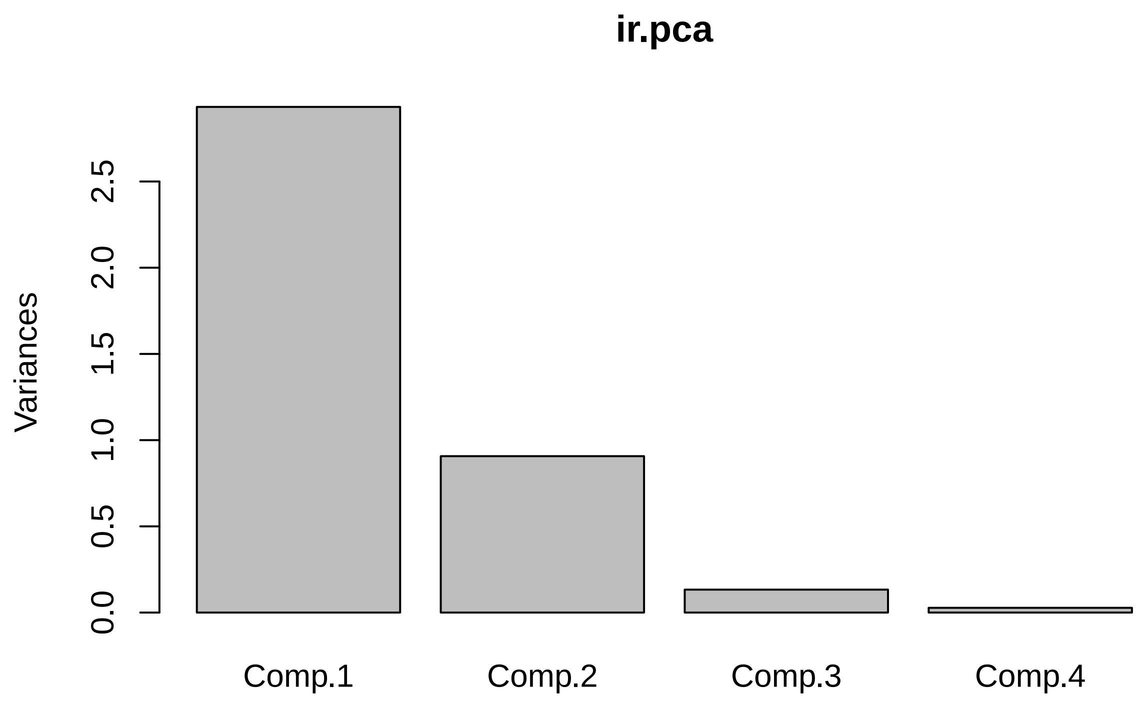
```r
# Principal Component for the log-transforrned iris data.
(ir.pca <- princomp(log(ir), cor = TRUE))
```

```
## Call:
## princomp(x = log(ir), cor = TRUE)
##
## Standard deviations:
##  Comp.1  Comp.2  Comp.3  Comp.4
## 1.71246 0.95238 0.36470 0.16568
##
##  4  variables and  150 observations.
```

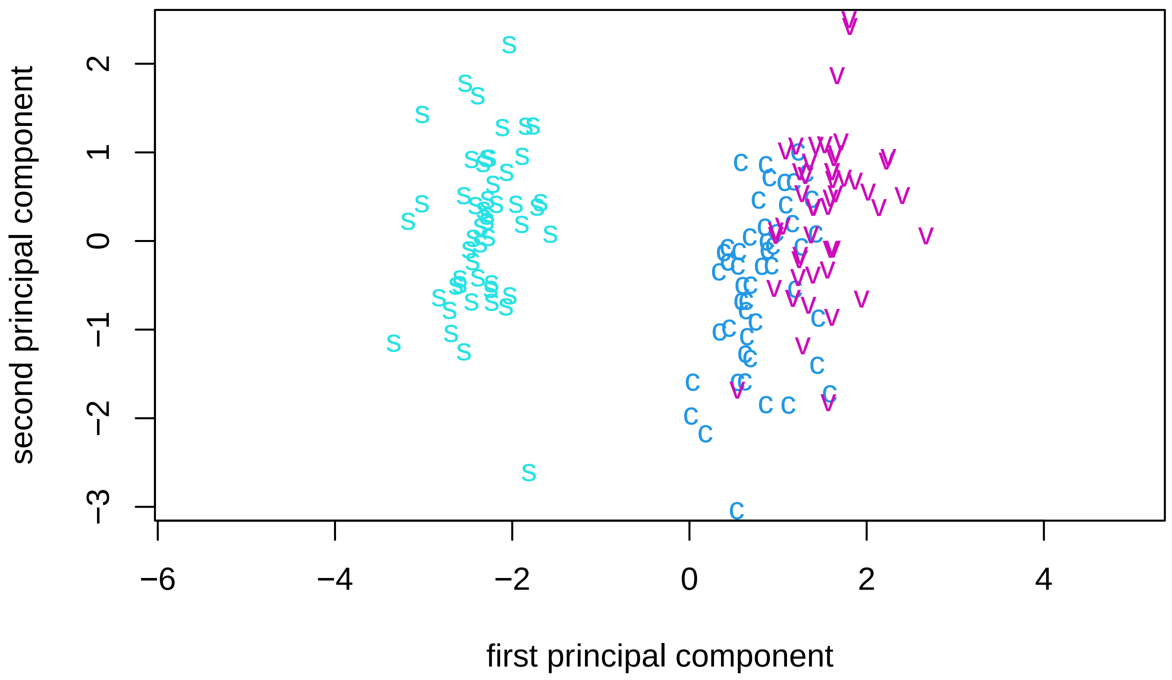
```r
summary(ir.pca)
```

```
## Importance of components:
##                           Comp.1  Comp.2   Comp.3    Comp.4
## Standard deviation     1.71246 0.95238 0.364703 0.1656840
## Proportion of Variance 0.73313 0.22676 0.033252 0.0068628
## Cumulative Proportion  0.73313 0.95989 0.993137 1.0000000
```

```
plot(ir.pca)
```



```
# First two principal components for the log-transforrned iris data.
ir.pc <- predict(ir.pca)
eqscplot(ir.pc[, 1:2], type = "n",
         xlab = "first principal component",
         ylab = "second principal component")
text(ir.pc[, 1:2], labels = as.character(ir.species),
     col = 3 + unclass(ir.species))
```



```r
# Principal Component for the crabs data.
(lcrabs.pca <- princomp(lcrabs))
```

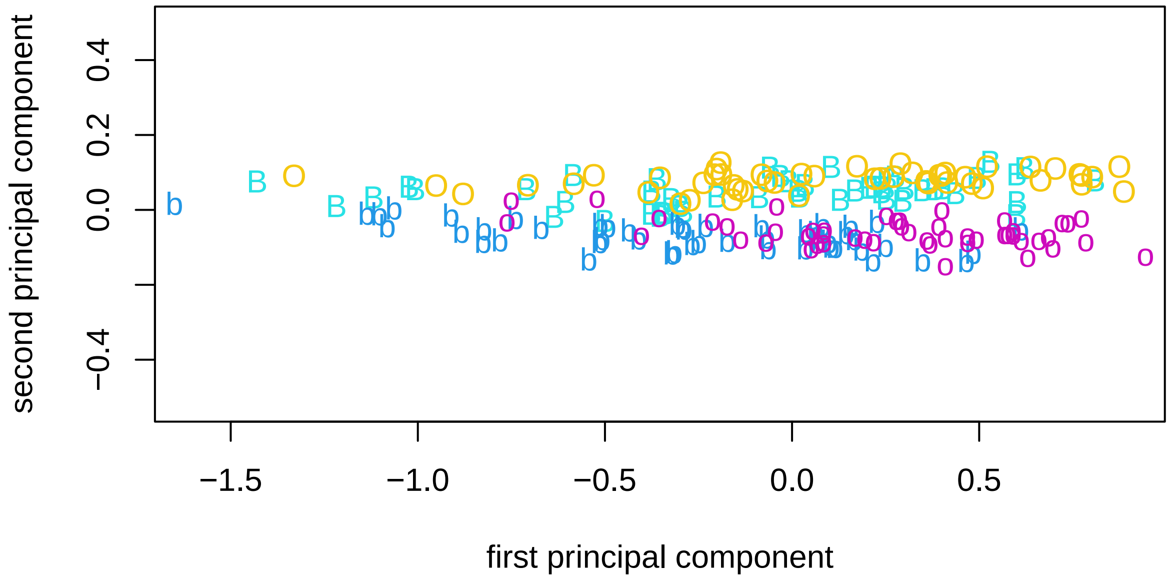
```
## Call:
## princomp(x = lcrabs)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
## 0.5166405 0.0746536 0.0479144 0.0248040 0.0090522
##
##  5  variables and  200 observations.
```

```r
loadings(lcrabs.pca)
```

```
##
## Loadings:
##    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## FL  0.452  0.157  0.438  0.752  0.114
## RW  0.387 -0.911
## CL  0.453  0.204 -0.371        -0.784
## CW  0.440        -0.672         0.591
## BD  0.497  0.315  0.458 -0.652  0.136
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## SS loadings       1.0    1.0    1.0    1.0    1.0
## Proportion Var    0.2    0.2    0.2    0.2    0.2
## Cumulative Var    0.2    0.4    0.6    0.8    1.0
```

```
lcrabs.pc <- predict(lcrabs.pca)
dimnames(lcrabs.pc) <- list(NULL, paste("PC", 1:5, sep = ""))
```

```
# First two principal components for the crabs data.
eqscplot(lcrabs.pc[, 1:2], type = "n",
         xlab = "first principal component",
         ylab = "second principal component")
text(lcrabs.pc[, 1:2], labels = as.character(crabs.grp),
     col = 3 + as.integer(crabs.grp))
```



2) Exploratory projection pursuit

```
if(FALSE) { ## needs interaction with XGobi, or, better, rggobi
  ## Both have been withdrawn for R.
  library(xgobi)
  xgobi(lcrabs, colors = c("SkyBlue", "SlateBlue", "Orange",
                         "Red")[rep(1:4, each = 50)])
  xgobi(lcrabs, glyphs = 12 + 5*rep(0:3, each = 50, 4))

  library(rggobi)
  g <- ggobi(lcrabs)
  d <- displays(g)[[1]]
  pmode(d) <- "2D Tour"
  crabs.grp <- factor(c("B", "b", "O", "o")[rep(1:4, each = 50)])
  glyph_colour(g$lcrabs) <- crabs.grp
  colorscheme(g) <- "Paired 4"
}
```

3) Distance methods

```
# Distance-based representations of the iris data
par(mfrow = c(2,2))

ir.scal <- cmdscale(dist(ir) , k = 2, eig = T)
eqscplot(ir.scal$points, type = "n", main = "Metric scaling")
text(ir.scal$points, labels = as.character(ir.species), col = 3 + as.integer(ir.species), cex = 0.8)

distp <- dist(ir)
dist2 <- dist(ir.scal$points)
sum((distp - dist2)^2)/sum(distp^2) # calculating a measure of 'stress'
```

```
## [1] 0.0017469
```

```
ir.sam <- sammon(dist(ir[-143,]))
```

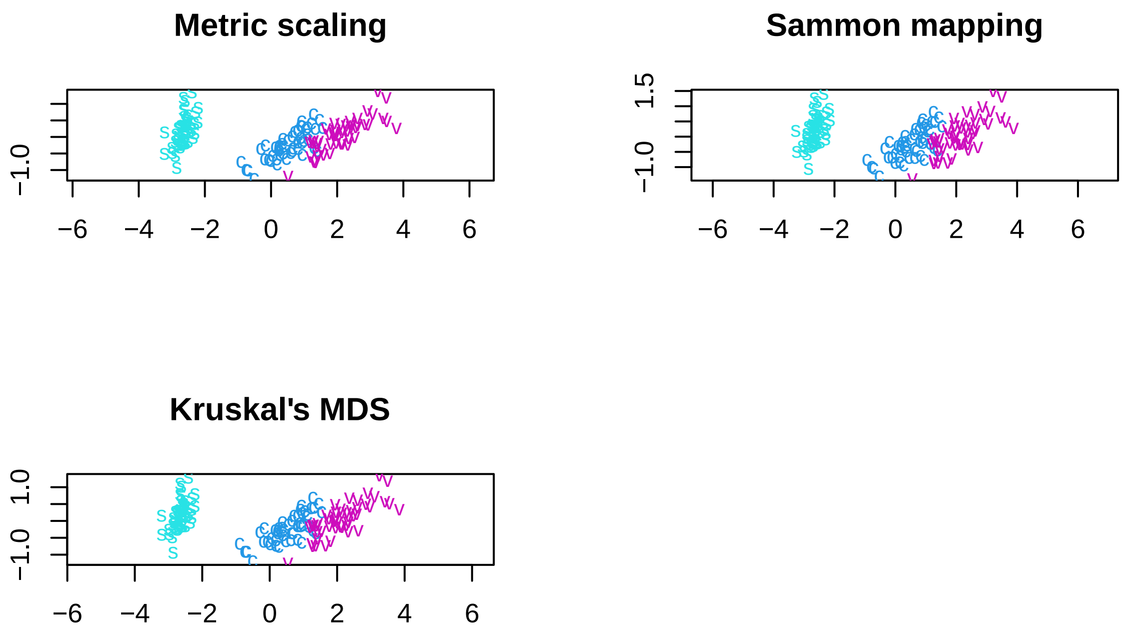
```
## Initial stress        : 0.00678
## stress after  10 iters: 0.00404, magic = 0.500
## stress after  12 iters: 0.00402
```

```
eqscplot(ir.sam$points, type = "n", main = "Sammon mapping")
text(ir.sam$points, labels = as.character(ir.species[-143]), col = 3 + as.integer(ir.species), cex = 0.

ir.iso <- isoMDS(dist(ir[-143,]))
```

```
## initial  value 3.024856
## iter   5 value 2.638471
## final  value 2.579979
## converged
```

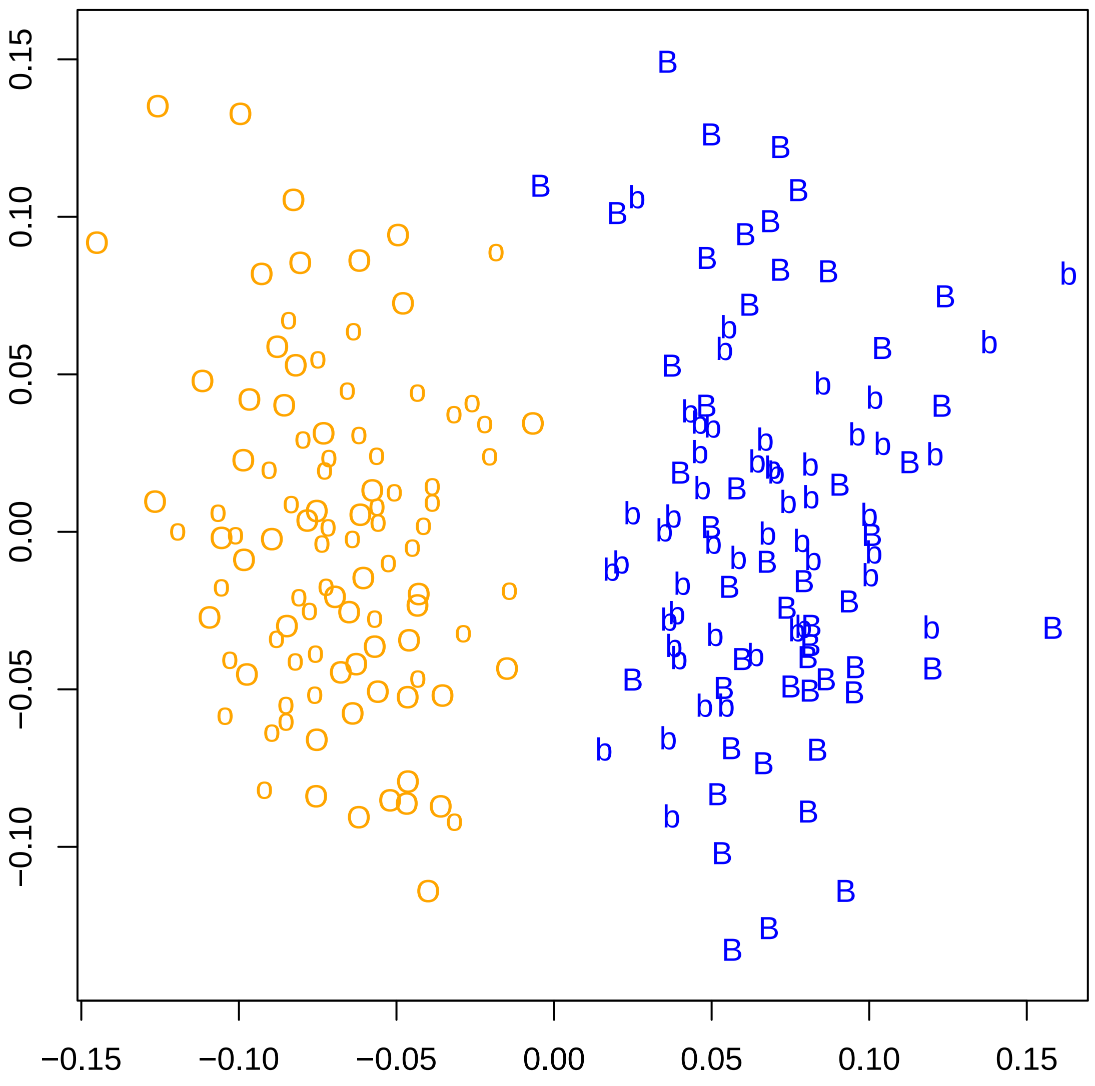
```
eqscplot(ir.iso$points, type = "n", main = "Kruskal's MDS")
text(ir.iso$points, labels = as.character(ir.species[-143]), col = 3 + as.integer(ir.species), cex = 0.
```



```r
# Sammon mapping of crabs data
cr.scale <- 0.5 * log(crabs$CL * crabs$CW)
slcrabs <- lcrabs - cr.scale
cr.means <- matrix(0, 2, 5)
cr.means[1,] <- colMeans(slcrabs[crabs$sex == "F", ])
cr.means[2,] <- colMeans(slcrabs [crabs$sex == "M", ])
dslcrabs <- slcrabs - cr.means[as.numeric(crabs$sex),]
lcrabs.sam <- sammon(dist(dslcrabs))
```

```
## Initial stress        : 0.01902
## stress after  10 iters: 0.01321, magic = 0.500
## stress after  20 iters: 0.01318, magic = 0.500
```

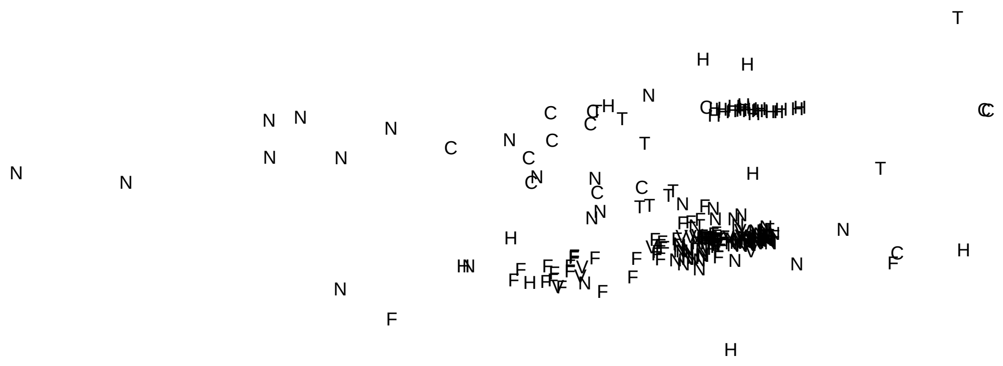
```r
eqscplot(-lcrabs.sam$points, type = "n", xlab = "", ylab = "")
text(-lcrabs.sam$points , labels = as.character(crabs.grp), col = rep(c("blue", "orange"), each = 100))
```



```r
# Isotonic multidimensional scaling representation of the fgl data.
fgl.iso <- isoMDS(dist(as.matrix(fgl[-40, -10])))
```

```
## initial  value 11.518169
## iter   5 value 6.353547
## iter  10 value 5.993823
## iter  15 value 5.913937
## final  value 5.888284
## converged
```

```
eqscplot(fgl.iso$points, type = "n", xlab = "", ylab = "", axes = FALSE)
# either
# for(i in seq(along = levels(fgl$type))) {
#   set <- fgl$type[-40] == levels(fgl$type)[i]
#   points(fgl.iso$points[set,], pch = 18, cex = 0.6, col = 2 + i)}
# key(text = list(levels(fgl$type), col = 3:8))
# or
text(fgl.iso$points, labels = c("F", "N", "V", "C", "T", "H")[fgl$type[-40]], cex = 0.6)
```



```
fgl.iso3 <- isoMDS(dist(as.matrix(fgl[-40, -10])), k = 3)
```

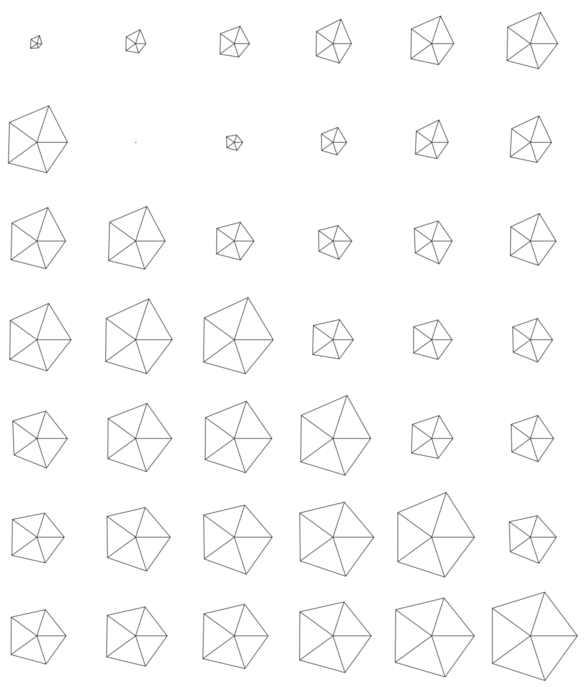
```
## initial  value 7.487849
## iter   5 value 3.178295
## iter  10 value 2.807260
## iter  15 value 2.590478
## iter  20 value 2.539430
## final  value 2.533004
## converged
```

```
# S: brush(fgl.iso3$points)
fgl.col <- c("SkyBlue", "SlateBlue", "Orange", "Orchid", "Green", "HotPink")[fgl$type]
# xgobi(fgl.iso3$points, colors = fgl.col)
```

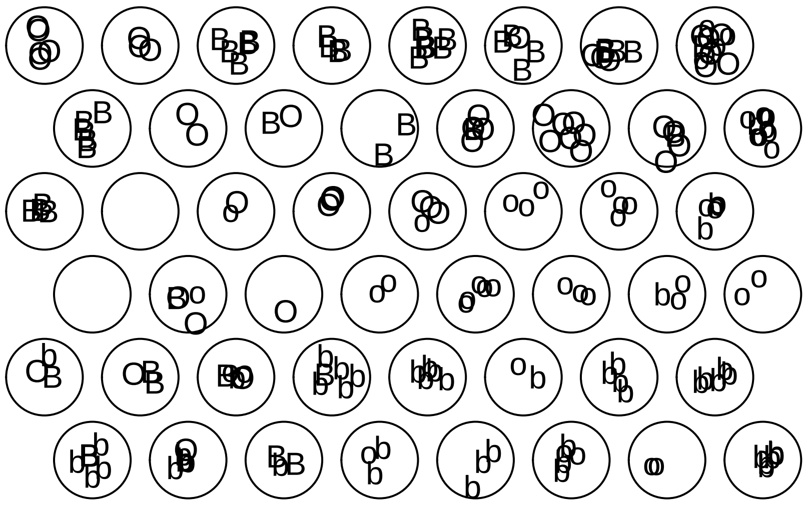
4) Self-organizing maps

```r
# Batch SOM applied to the crabs dataset.
gr <- somgrid(topo = "hexagonal")
crabs.som <- batchSOM(lcrabs, gr, c(4, 4, 2, 2, 1, 1, 1, 0, 0))
```

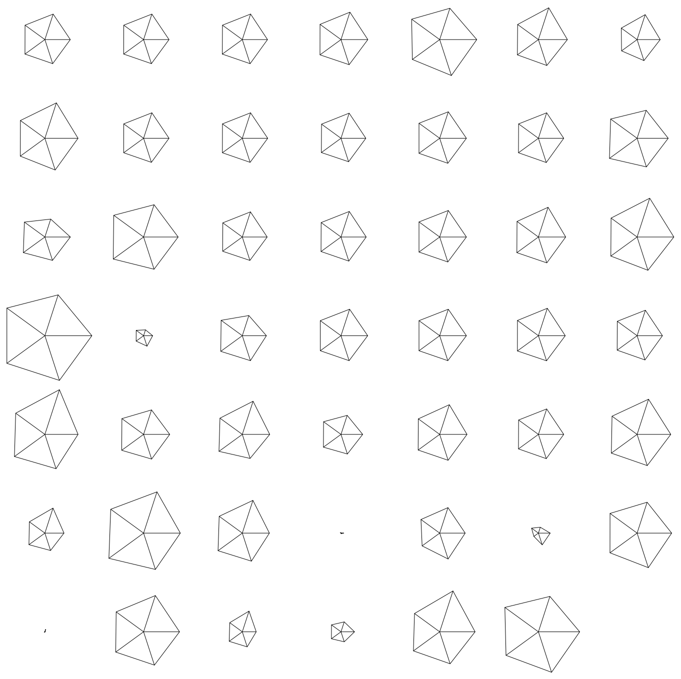
```r
# stars plot of the representatives
stars(crabs.som$codes, labels = NULL)
```



```r
# Plot that shows the assignments of the original points
bins <- as.numeric(knn1(crabs.som$code, lcrabs, 0:47))
plot(crabs.som$grid, type = "n",
     xlim = c(min(crabs.som$grid$pts[,1])-0.4, max(crabs.som$grid$pts[,1])+0.4),
     ylim = c(min(crabs.som$grid$pts[,2])-0.4, max(crabs.som$grid$pts[,2])+0.4))
symbols(crabs.som$grid$pts[, 1], crabs.som$grid$pts[, 2], circles = rep(0.4, 48), inches = FALSE, add =
text(crabs.som$grid$pts[bins, ] + rnorm(400, 0, 0.1), as.character(crabs.grp))
```

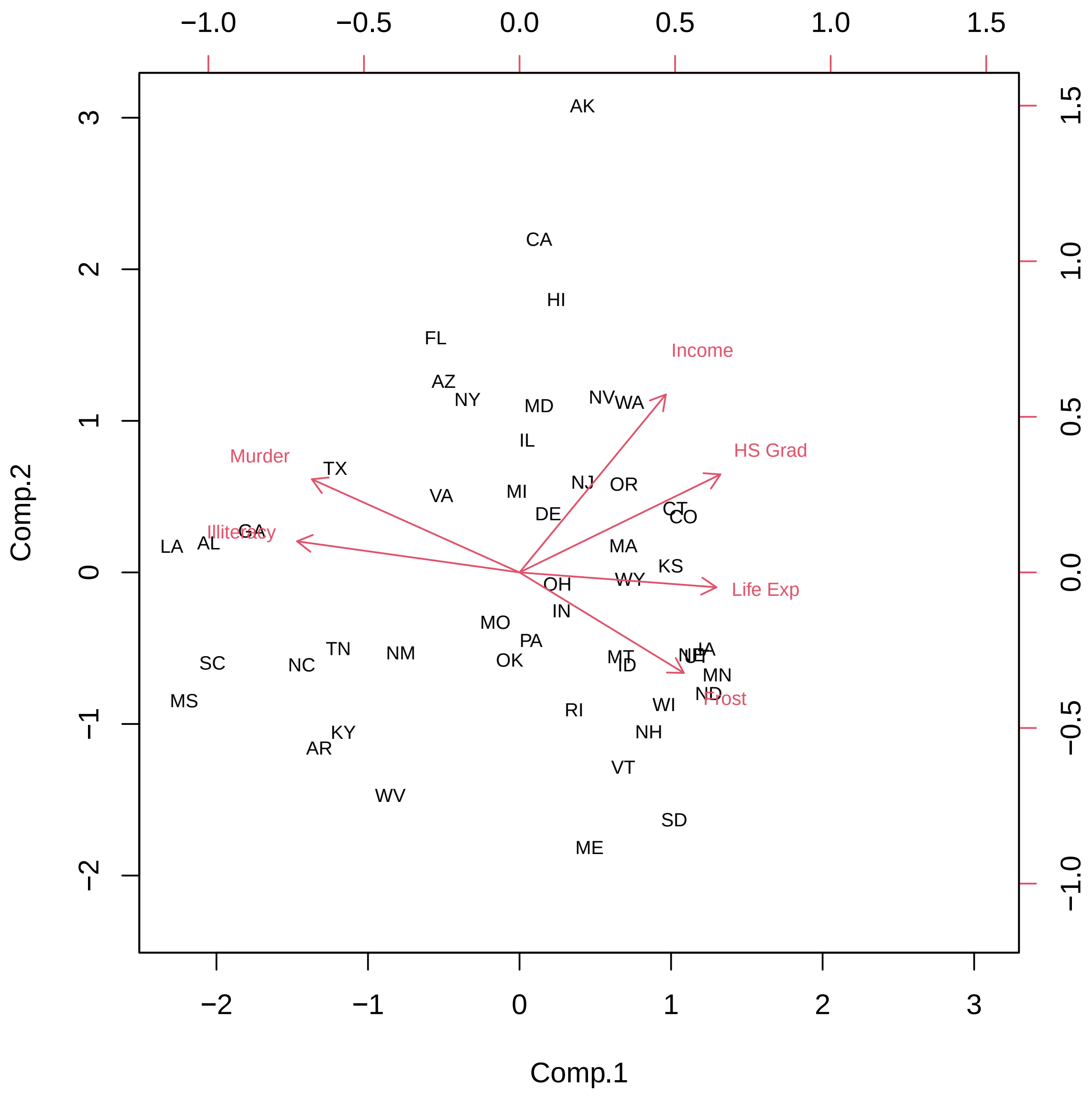


```
# Traditional SOM applied to the crabs dataset.
crabs.som2 <- SOM(lcrabs, gr); stars(crabs.som2$codes)
```



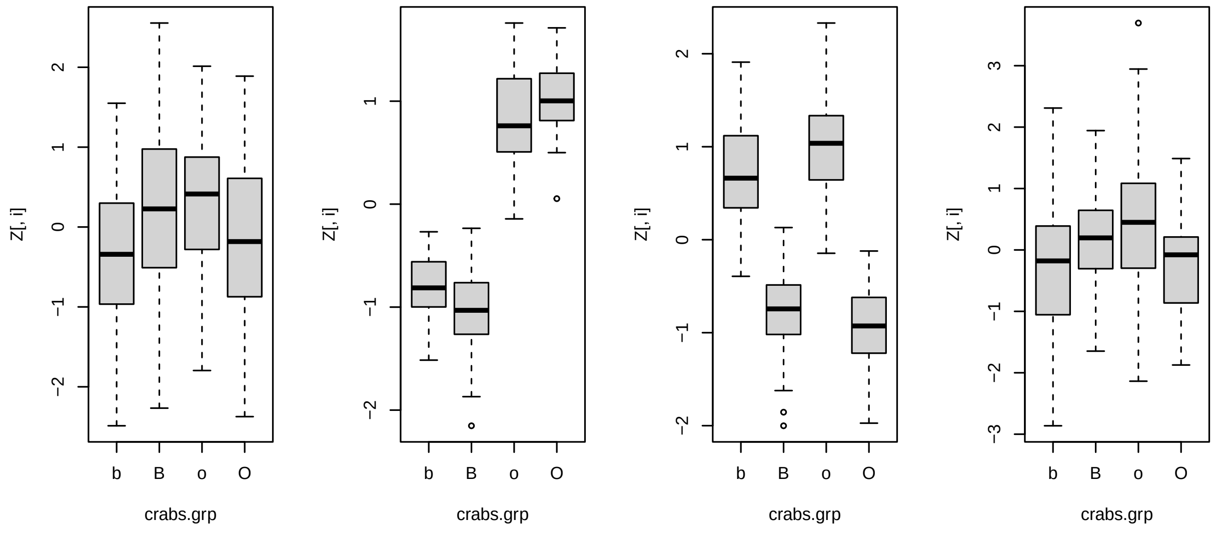
5) Biplots

```
# Principal component biplot of the part of the state.x77 data.
state <- state.x77[, 2:7]; row.names(state) <- state.abb
biplot(princomp(state, cor = TRUE), pc.biplot = TRUE, cex = 0.7, expand = 0.8)
```



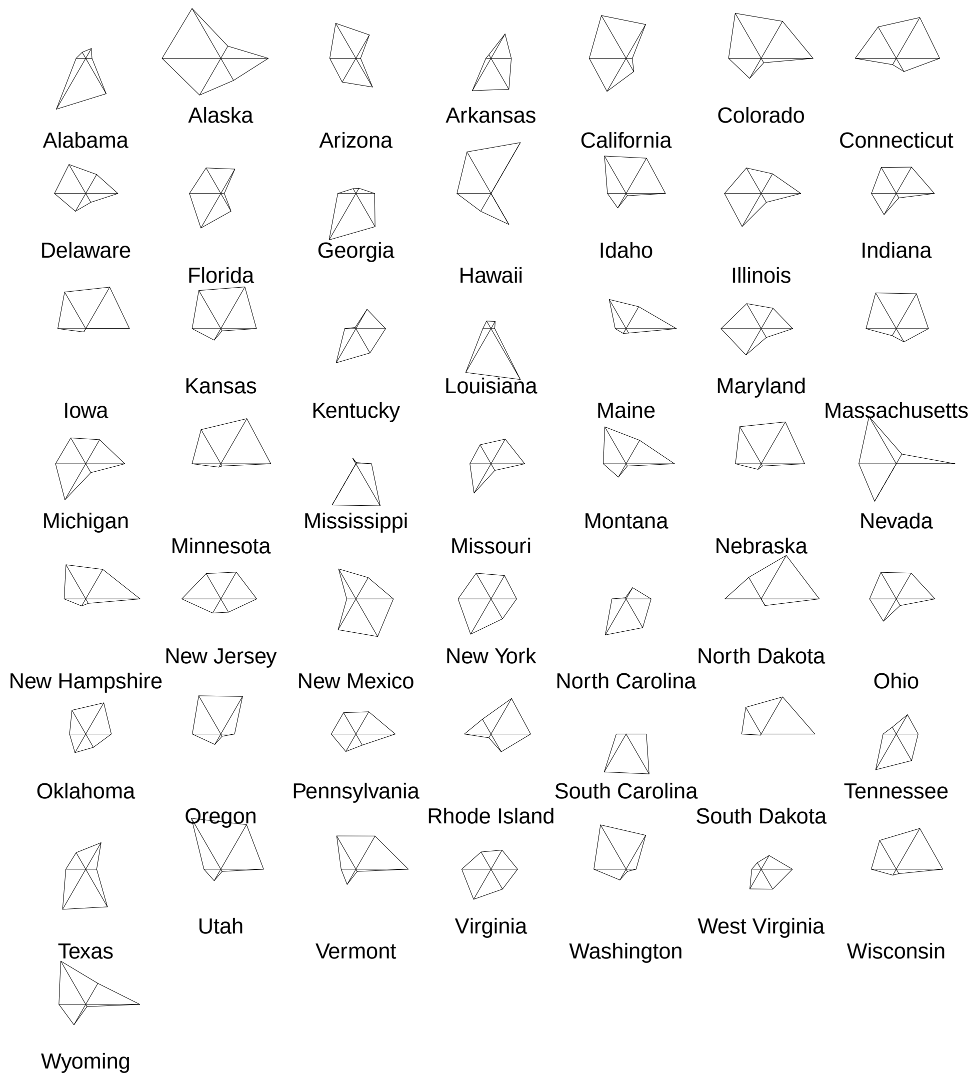
6) Independent component analysis

```
nICA <- 4
crabs.ica <- fastICA(crabs[, 4:8], nICA)
Z <- crabs.ica$S
par(mfrow = c(1, nICA))
for(i in 1:nICA) boxplot(Z[, i] ~ crabs.grp)
```



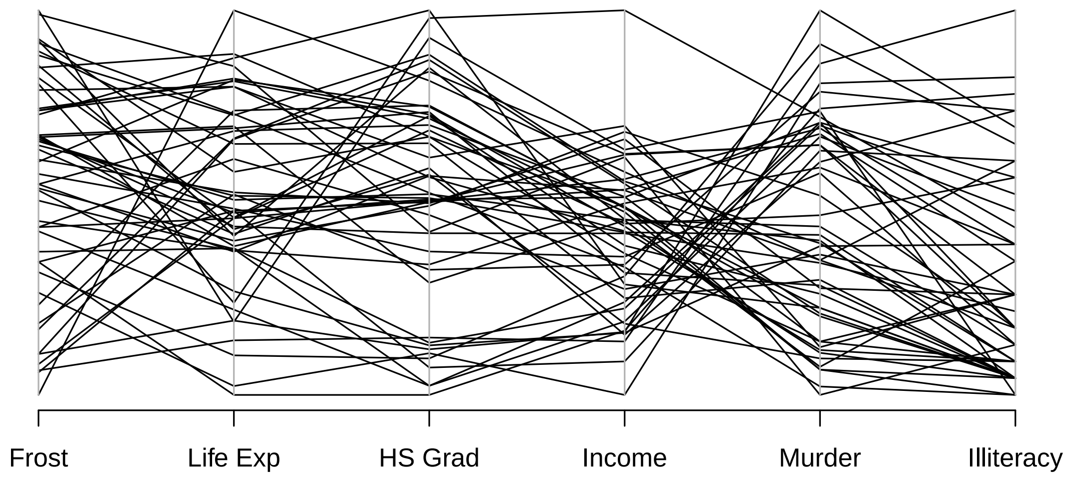
7) Glyph representations

```r
# stars plot of the state.x77 dataset.
stars(state.x77[, c(7, 4, 6, 2, 5, 3)])
```

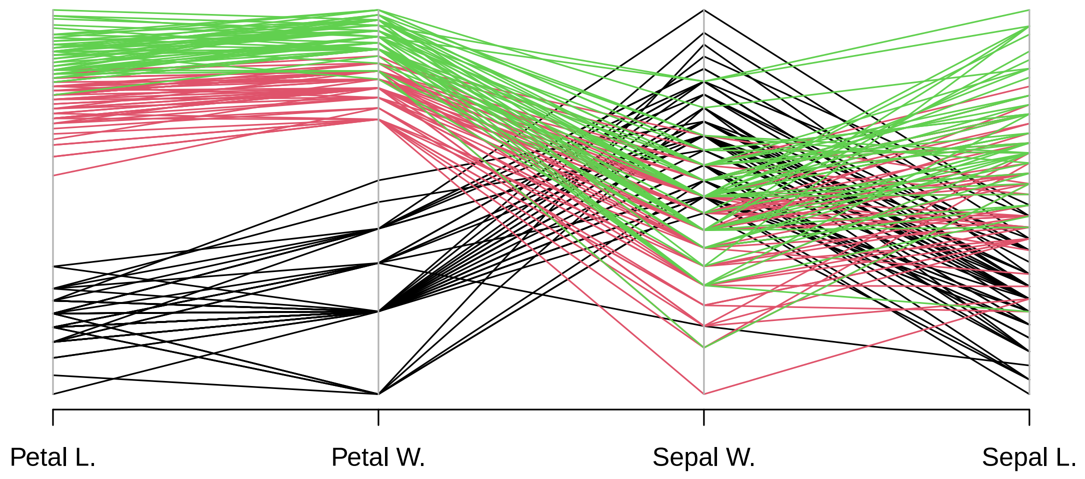


8) Parallel coordinate plots

```
# Parallel coordinates plots of the state.x77 dataset.
parcoord(state.x77[, c(7, 4, 6, 2, 5, 3)])
```

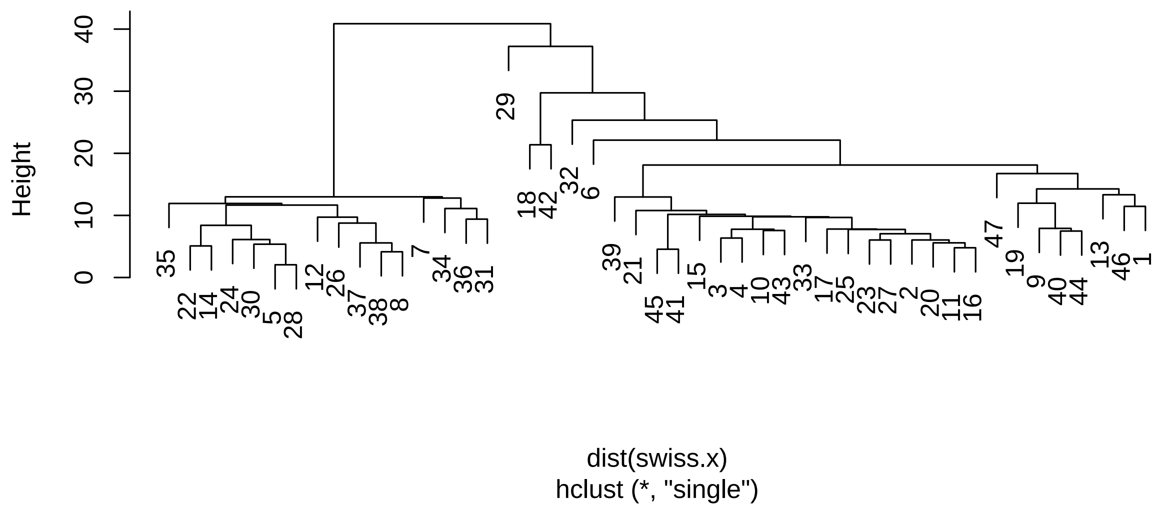


```
# Parallel coordinates plots of the log-transforrned iris data
parcoord(log(ir)[, c(3, 4, 2, 1)], col = 1 + (0:149)%/%50)
```



11.2 Cluster Analysis

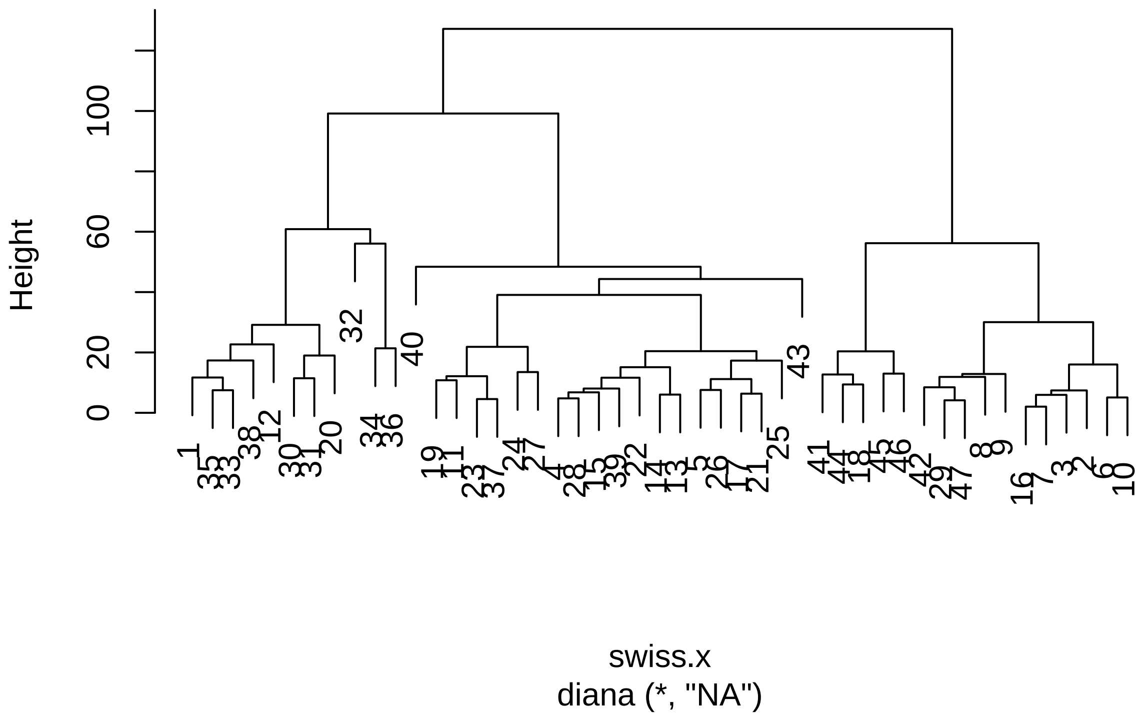
```r
# Dendograms for the socio-economic data on Swiss provinces computed by single-link clustering
swiss.x <- as.matrix(swiss[,-1])
h <- hclust(dist(swiss.x), method = "single")
plot(h, labels = h$order, main = "")
```



```r
cutree(h, 3)
```

```
##   Courtelary     Delemont Franches-Mnt      Moutier   Neuveville
##            1            2            2            1            1
##   Porrentruy        Broye        Glane      Gruyere       Sarine
##            2            2            2            2            2
##      Veveyse        Aigle      Aubonne     Avenches     Cossonay
##            2            1            1            1            1
##    Echallens     Grandson     Lausanne    La Vallee       Lavaux
##            1            1            1            1            1
##       Morges       Moudon        Nyone         Orbe         Oron
##            1            1            1            1            1
##      Payerne Paysd'enhaut        Rolle        Vevey      Yverdon
##            1            1            1            1            1
##      Conthey    Entremont       Herens     Martigwy      Monthey
##            2            2            2            2            2
##   St Maurice       Sierre         Sion       Boudry La Chauxdfnd
##            2            2            2            1            1
##     Le Locle    Neuchatel   Val de Ruz ValdeTravers V. De Geneve
##            1            1            1            1            3
##  Rive Droite  Rive Gauche
##            1            1
```

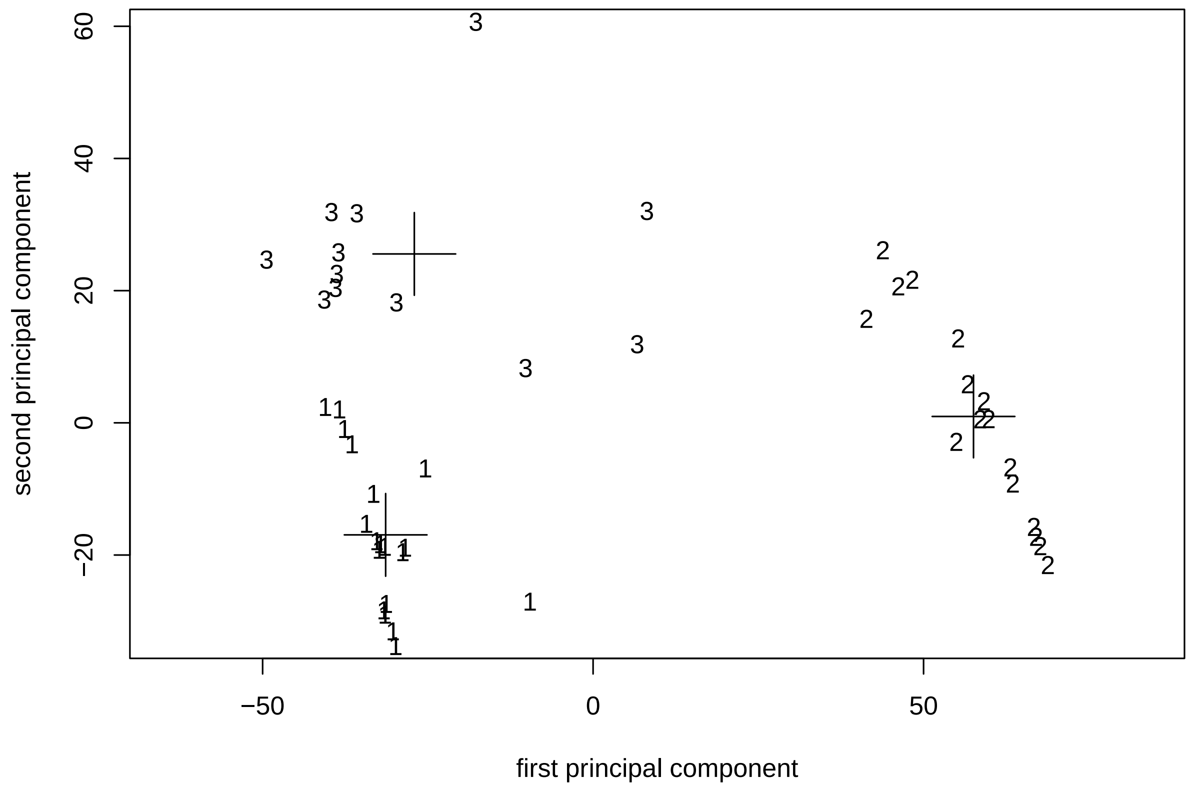
```r
# Dendograms for the socio-economic data on Swiss provinces computed by divisive clustering
d <- diana(swiss.x)
pltree(d, labels = d$order, main = "")
```



```r
# First two principal components for the swiss data and labeling by the groups assigned by K-means
h <- hclust(dist(swiss.x), method = "average")
initial <- tapply(swiss.x, list(rep(cutree (h, 3), ncol(swiss.x)), col(swiss.x)), mean)
dimnames(initial) <- list(NULL, dimnames(swiss.x)[[2]])
km <- kmeans(swiss.x, initial)
(swiss.pca <- princomp(swiss.x))
```

```
## Call:
## princomp(x = swiss.x)
##
## Standard deviations:
##  Comp.1  Comp.2  Comp.3  Comp.4  Comp.5
## 42.8963 21.2019  7.5880  3.6879  2.7211
##
##  5  variables and  47 observations.
```

```r
swiss.px <- predict(swiss.pca); swiss.px[,2] <- -swiss.px[,2]
dimnames(km$centers)[[2]] <- dimnames(swiss.x)[[2]]
swiss.centers <- predict(swiss.pca, km$centers); swiss.centers[,2] <- -swiss.centers[,2]
eqscplot(swiss.px[, 1:2], type = "n",
         xlab = "first principal component" , ylab = "second principal component")
text(swiss.px[, 1:2], labels = km$cluster)
points(swiss.centers[,1:2], pch = 3, cex = 5)
identify(swiss.px[, 1:2], cex = 0.5)
```

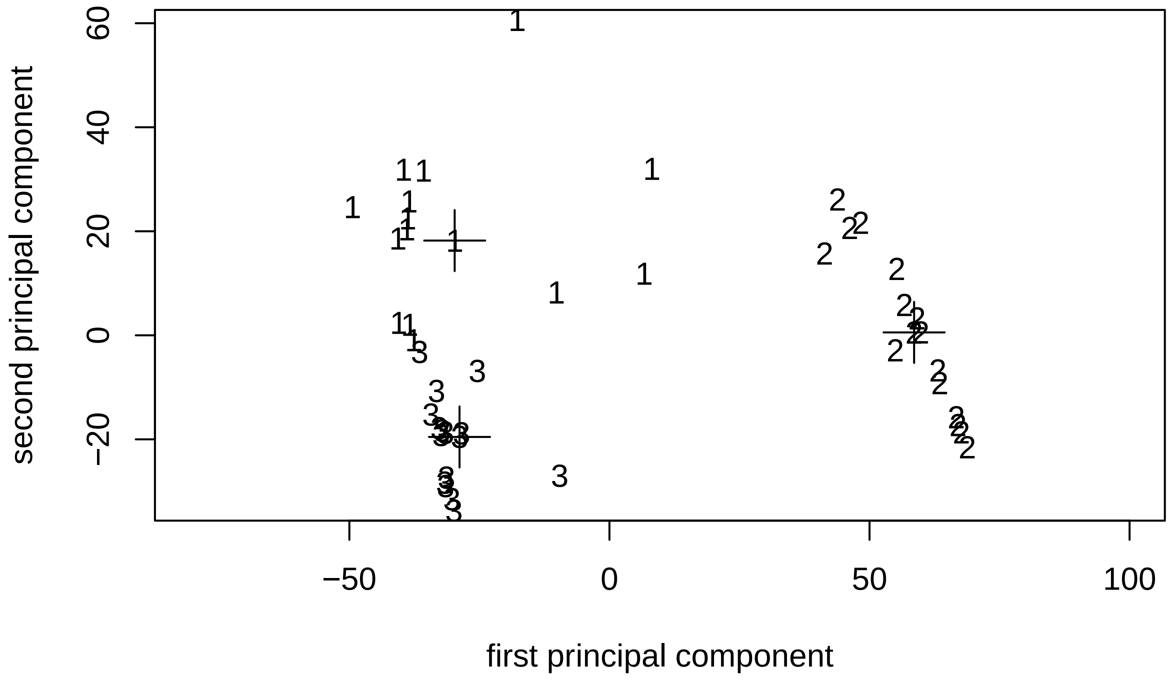


```
## integer(0)
```

```
swiss.pam <- pam(swiss.px, 3)
summary(swiss.pam)
```

```
## Medoids:
##        ID  Comp.1    Comp.2 Comp.3  Comp.4  Comp.5
## Vevey 29 -29.754  18.20822 1.4268  1.3173  0.9530
## Glane  8  58.572   0.55358 2.2304  4.1756  4.2287
## Rolle 28 -28.823 -19.54413 3.1523 -2.3862 -2.4685
## Clustering vector:
##   Courtelary     Delemont Franches-Mnt      Moutier   Neuveville
##            1            2            2            1            3
##   Porrentruy        Broye        Glane      Gruyere       Sarine
##            2            2            2            2            2
##      Veveyse        Aigle      Aubonne    Avenches     Cossonay
##            2            3            3            3            3
##    Echallens     Grandson     Lausanne    La Vallee       Lavaux
##            3            1            1            1            3
##       Morges       Moudon        Nyone         Orbe         Oron
##            3            3            3            3            3
##      Payerne Paysd'enhaut        Rolle        Vevey      Yverdon
##            3            3            3            1            3
##      Conthey    Entremont       Herens     Martigwy      Monthey
##            2            2            2            2            2
##   St Maurice       Sierre         Sion       Boudry La Chauxdfnd
##            2            2            2            1            1
##     Le Locle    Neuchatel   Val de Ruz ValdeTravers V. De Geneve
##            1            1            1            1            1
##  Rive Droite  Rive Gauche
##            1            1
## Objective function:
##  build   swap
## 18.866 17.190
##
## Numerical information per cluster:
##      size max_diss av_diss diameter separation
## [1,]   15   50.339  23.160   72.976     10.159
## [2,]   16   33.594  17.240   56.198     40.865
## [3,]   16   22.424  11.541   37.144     10.159
##
## Isolated clusters:
##  L-clusters: character(0)
##  L*-clusters: character(0)
##
## Silhouette plot information:
##              cluster neighbor sil_width
## La Chauxdfnd       1        3  0.450749
## Le Locle           1        3  0.443123
## Lausanne           1        3  0.432000
## Neuchatel          1        3  0.421224
## Courtelary         1        3  0.383092
## ValdeTravers       1        3  0.372514
## Vevey              1        3  0.371576
## La Vallee          1        3  0.370931
## V. De Geneve       1        3  0.345291
## Rive Gauche        1        2  0.202987
## Rive Droite        1        3  0.069220
## Moutier            1        3  0.051850
## Grandson           1        3 -0.087632
## Boudry             1        3 -0.159338
## Val de Ruz         1        3 -0.238440
## Veveyse            2        1  0.800007
## Glane              2        3  0.799660
## Monthey            2        3  0.799606
## St Maurice         2        3  0.793382
## Martigwy           2        3  0.786611
## Sion               2        1  0.781538
## Broye              2        3  0.777086
## Entremont          2        3  0.755605
## Gruyere            2        1  0.754639
## Sierre             2        3  0.746634
## Conthey            2        3  0.735009
## Herens             2        3  0.717159
## Sarine             2        1  0.642198
## Franches-Mnt       2        1  0.625301
## Delemont           2        1  0.614282
## Porrentruy         2        1  0.555966
## Aubonne            3        1  0.738386
## Rolle              3        1  0.730207
## Avenches           3        1  0.717628
## Morges             3        1  0.715969
## Cossonay           3        1  0.712481
## Payerne            3        1  0.712087
## Aigle              3        1  0.706067
## Lavaux             3        1  0.701155
## Oron               3        1  0.692961
## Moudon             3        1  0.673348
## Paysd'enhaut       3        1  0.671930
## Orbe               3        1  0.653260
## Yverdon            3        1  0.572295
## Echallens          3        1  0.544783
## Nyone              3        1  0.471723
## Neuveville         3        1  0.328130
## Average silhouette width per cluster:
## [1] 0.22861 0.73029 0.64640
## Average silhouette width of total data set:
## [1] 0.54162
##
## 1081 dissimilarities, summarized :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.05   27.30   58.10   60.70   93.90  127.00
## Metric :  euclidean
## Number of objects : 47
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"
##  [5] "isolation"  "clusinfo"   "silinfo"    "diss"
##  [9] "call"       "data"
```

```
eqscplot(swiss.px[, 1:2], type = "n",
         xlab = "first principal component", ylab = "second principal component")
text(swiss.px[,1:2], labels = swiss.pam$clustering)
points(swiss.pam$medoid[,1:2], pch = 3, cex = 3)
```



```
fanny(swiss.px, 3)
```

```
## Fuzzy Clustering object of class 'fanny' :
## m.ship.expon.      2
## objective     354.02
## tolerance      1e-15
## iterations        17
## converged          1
## maxit            500
## n                 47
## Membership coefficients (in %, rounded):
##              [,1] [,2] [,3]
## Courtelary     72    8   20
## Delemont       19   64   17
## Franches-Mnt   19   64   17
## Moutier        49   16   35
## Neuveville     41    7   52
## Porrentruy     22   59   19
## Broye           7   85    7
## Glane           6   88    6
## Gruyere        11   79   10
## Sarine         18   67   15
## Veveyse         7   87    6
## Aigle          15    5   79
## Aubonne        15    6   79
## Avenches       15   5  81
## Cossonay       17   7  75
## Echallens      26  16  58
## Grandson       56   8  36
## Lausanne       73   8  19
## La Vallee      68   9  23
## Lavaux         18   8  73
## Morges         14   5  81
## Moudon         18   6  76
## Nyone          31   8  61
## Orbe           19   5  75
## Oron           19   9  72
## Payerne        15   5  80
## Paysd'enhaut   20   8  72
## Rolle          13   5  82
## Vevey          74   7  19
## Yverdon        25   6  69
## Conthey        11  78  11
## Entremont       9  81  10
## Herens         12  76  12
## Martigwy        7  86   7
## Monthey         7  86   7
## St Maurice      7  87   7
## Sierre         10  79  11
## Sion            8  84   8
## Boudry         54   7  39
## La Chauxdfnd   70   9  20
## Le Locle       78   6  16
## Neuchatel      66  11  23
## Val de Ruz     49   8  43
## ValdeTravers   73   7  20
## V. De Geneve   49  23  28
## Rive Droite    39  29  32
## Rive Gauche    43  29  27
## Fuzzyness coefficients:
## dunn_coeff normalized
##    0.57628    0.36442
## Closest hard clustering:
##   Courtelary     Delemont Franches-Mnt      Moutier   Neuveville
##            1            2            2            1            3
##   Porrentruy        Broye        Glane      Gruyere       Sarine
##            2            2            2            2            2
##      Veveyse        Aigle      Aubonne     Avenches     Cossonay
##            2            3            3            3            3
##    Echallens     Grandson     Lausanne    La Vallee       Lavaux
##            3            1            1            1            3
##       Morges       Moudon        Nyone         Orbe         Oron
##            3            3            3            3            3
##      Payerne Paysd'enhaut        Rolle        Vevey      Yverdon
##            3            3            3            1            3
##      Conthey    Entremont       Herens     Martigwy      Monthey
##            2            2            2            2            2
##   St Maurice       Sierre         Sion       Boudry La Chauxdfnd
##            2            2            2            1            1
##     Le Locle    Neuchatel   Val de Ruz ValdeTravers V. De Geneve
##            1            1            1            1            1
##  Rive Droite  Rive Gauche
##            1            1
##
## Available components:
##  [1] "membership"  "coeff"       "memb.exp"    "clustering"
##  [5] "k.crisp"     "objective"   "convergence" "diss"
##  [9] "call"        "silinfo"     "data"
```

```
## From the on-line Errata:
##
##   `The authors of mclust have chosen to re-use the name for a
##   completely incompatible package. We can no longer recommend its
##   use, and the code given in the first printing does not work in R's
##   mclust-2.x.'
##

## And later mclust was given a restrictive licence, so this example
## has been removed.  Finally in 2012 it was given an OpenSource licence.
```

11.3 Factor analysis

```
ability.FA <- factanal(covmat = ability.cov, factors = 1)
ability.FA
```

```
##
## Call:
## factanal(factors = 1, covmat = ability.cov)
##
## Uniquenesses:
## general picture  blocks    maze reading   vocab
##   0.535   0.853   0.748   0.910   0.232   0.280
##
## Loadings:
##         Factor1
## general 0.682
## picture 0.384
## blocks  0.502
## maze    0.300
## reading 0.877
## vocab   0.849
##
##                Factor1
## SS loadings      2.443
## Proportion Var   0.407
##
## Test of the hypothesis that 1 factor is sufficient.
## The chi square statistic is 75.18 on 9 degrees of freedom.
## The p-value is 1.46e-12
```

```
(ability.FA <- update(ability.FA, factors = 2))
```

```
##
## Call:
## factanal(factors = 2, covmat = ability.cov)
##
## Uniquenesses:
## general picture  blocks    maze reading   vocab
##   0.455   0.589   0.218   0.769   0.052   0.334
##
## Loadings:
##         Factor1 Factor2
## general 0.499   0.543
## picture 0.156   0.622
## blocks  0.206   0.860
## maze    0.109   0.468
## reading 0.956   0.182
## vocab   0.785   0.225
##
##                Factor1 Factor2
## SS loadings      1.858   1.724
## Proportion Var   0.310   0.287
## Cumulative Var   0.310   0.597
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 6.11 on 4 degrees of freedom.
## The p-value is 0.191
```

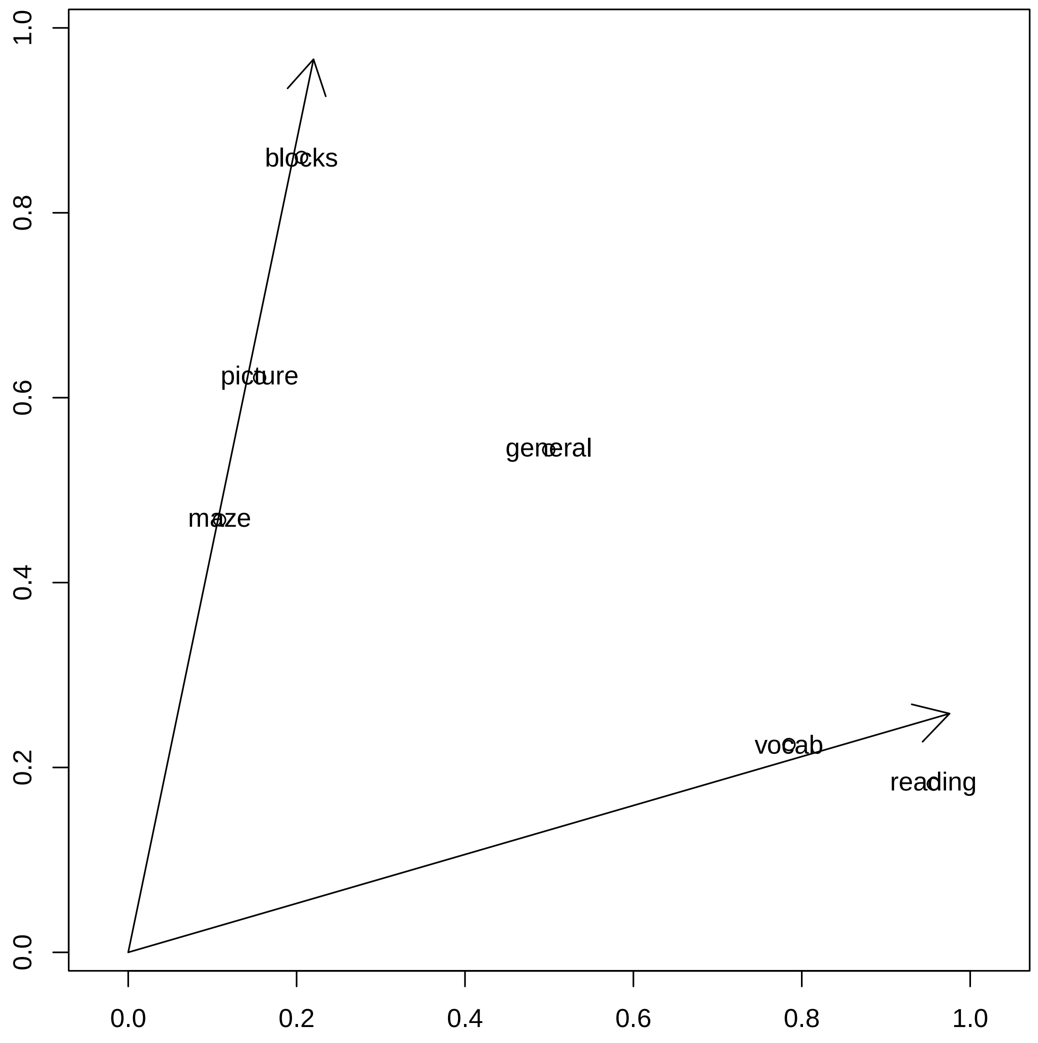
```
#summary(ability.FA)
round(loadings(ability.FA) %*% t(loadings(ability.FA)) +
        diag(ability.FA$uniq), 3)
```

```
##         general picture blocks  maze reading vocab
## general   1.000   0.416  0.570 0.308   0.577 0.514
## picture   0.416   1.000  0.567 0.308   0.262 0.262
## blocks    0.570   0.567  1.000 0.425   0.353 0.355
## maze      0.308   0.308  0.425 1.000   0.189 0.190
## reading   0.577   0.262  0.353 0.189   1.000 0.791
## vocab     0.514   0.262  0.355 0.190   0.791 1.000
```

```
# Factor rotations
library(GPArotation)
L <- loadings(ability.FA)
print(oblirot <- oblimin(L))
```

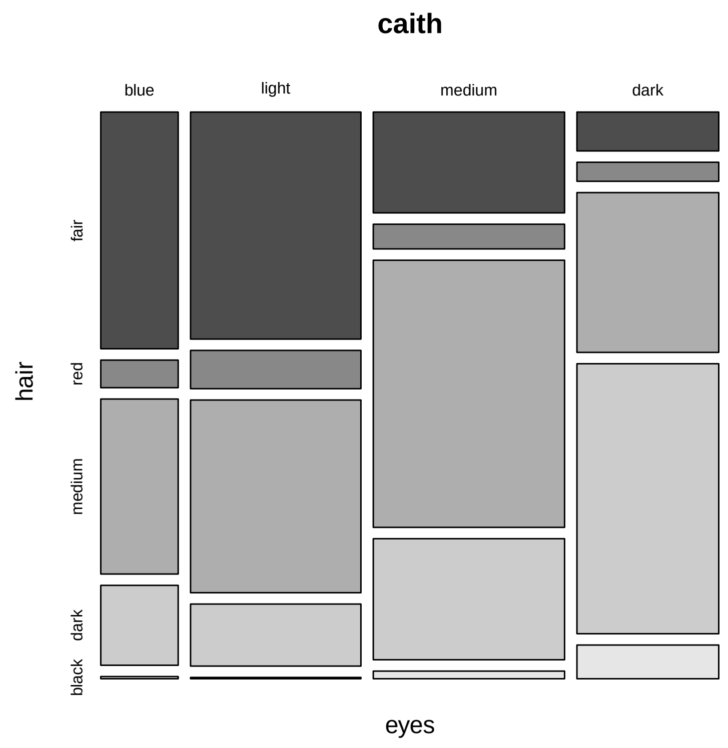
```
## Oblique rotation method Oblimin Quartimin converged.
## Loadings:
##         Factor1 Factor2
## general  0.3864  0.4745
## picture -0.0110  0.6459
## blocks  -0.0263  0.8961
## maze    -0.0180  0.4883
## reading  0.9901 -0.0371
## vocab    0.7906  0.0526
##
## Rotating matrix:
##         [,1]   [,2]
## [1,]  1.091 -0.249
## [2,] -0.292  1.102
##
## Phi:
##       [,1]  [,2]
## [1,] 1.000 0.465
## [2,] 0.465 1.000
```

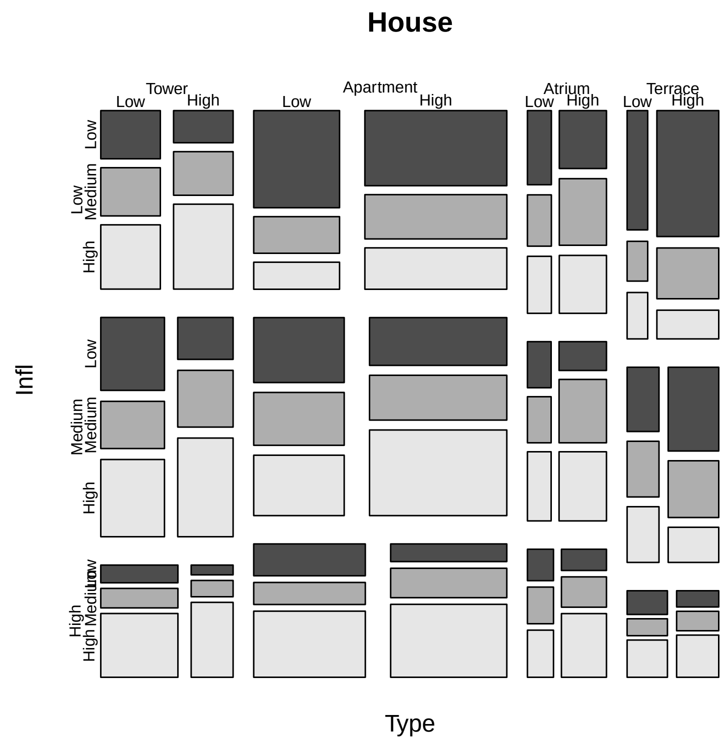
```
par(pty = "s")
eqscplot(L, xlim = c(0,1), ylim = c(0,1))
if(interactive()) identify(L[1:6,1], dimnames(L)[[1]])
naxes <- oblirot$Th
arrows(rep(0, 2), rep(0, 2), naxes[,1], naxes[,2])
text(L[1:6,1:2], dimnames(L)[[1]])
```



11.4 Discrete multivariate analysis

```
par(mfrow = c(2,1))
# Mosaic plots for Fisher's data on people from Caithness
caith <- as.matrix(caith)
names(dimnames(caith)) <- c("eyes", "hair")
mosaicplot(caith, color = TRUE)
# Mosaic plots for Copenhagen housing satisfaction data
House <- xtabs(Freq ~ Type + Infl + Cont + Sat, housing)
mosaicplot(House, color = TRUE)
```

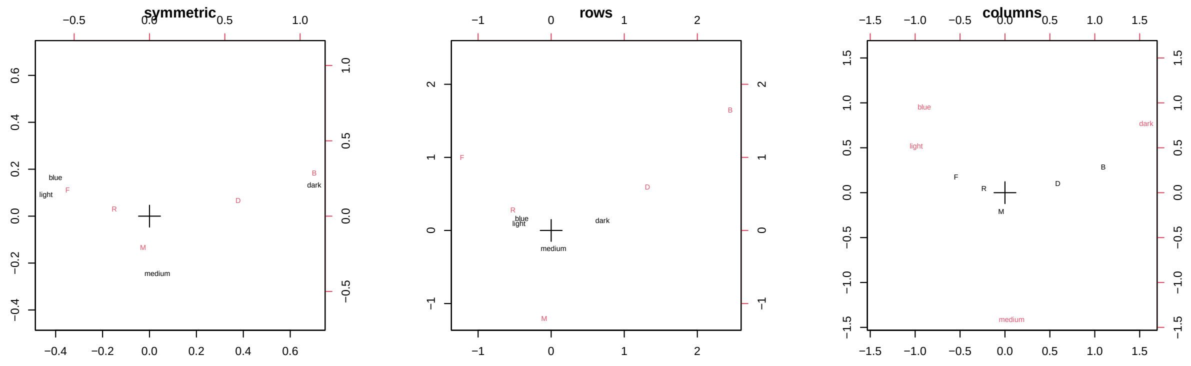

caith
blue
light
medium
dark
hair
fair
red
medium
dark
black
eyes


House
Tower
Apartment
Atrium
Terrace
Low
High
Infl
Low
Medium
High
Type

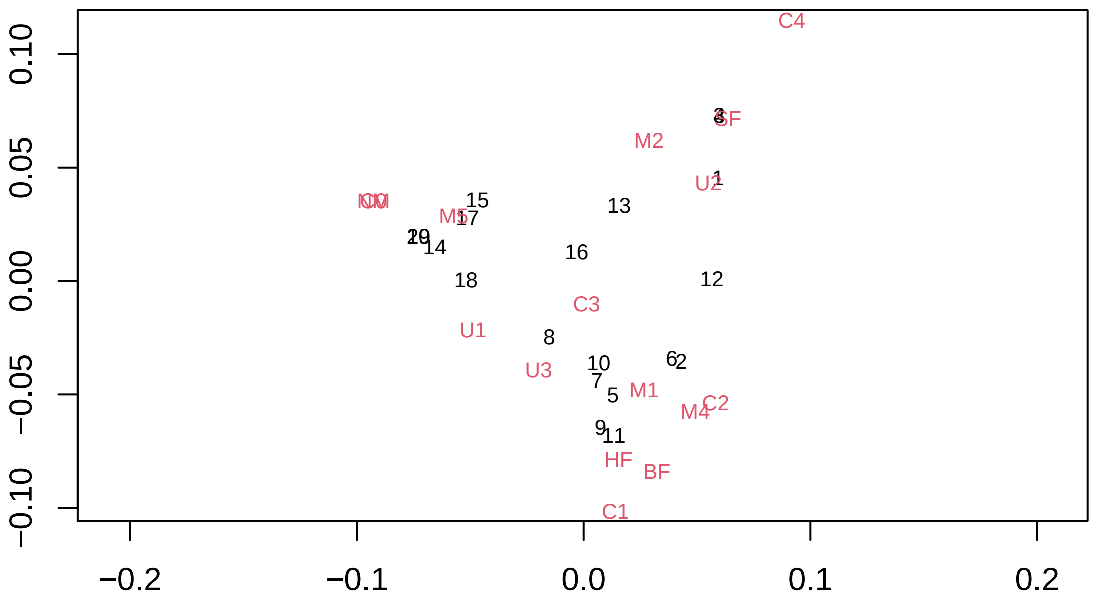
```
corresp(caith)
```

```
## First canonical correlation(s): 0.44637
##
##  eyes scores:
##      blue     light    medium      dark
## -0.896793 -0.987318  0.075306  1.574347
##
##  hair scores:
##      fair       red    medium      dark     black
## -1.218714 -0.522575 -0.094147  1.318885  2.451760
```

```
# Three variants of correspondence analysis plots from Fisher's data
caith2 <- caith
dimnames(caith2)[[2]] <- c("F", "R", "M", "D", "B")
par(mfcol = c(1, 3))
plot(corresp(caith2, nf = 2)); title("symmetric")
plot(corresp(caith2, nf = 2), type = "rows"); title("rows")
plot(corresp(caith2, nf = 2), type = "col"); title("columns")
```



```
# Multiple correspondence analysis plot of dataset farms
farms.mca <- mca(farms, abbrev = TRUE)  # Use levels as names
plot(farms.mca, cex = rep(0.7, 2))
```


C4
3
4
SF
M2
1
U2
CU
MM
15
M5
17
13
20
19
14
16
18
12
C3
U1
8
10
6
2
U3
7
M1
5
M4
C2
9
11
HF
BF
C1
0.10
0.05
0.00
−0.05
−0.10
−0.2
−0.1
0.0
0.1
0.2