

Modern Applied Statistics(Chap 11)

```
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

1) Principal Component analysis

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

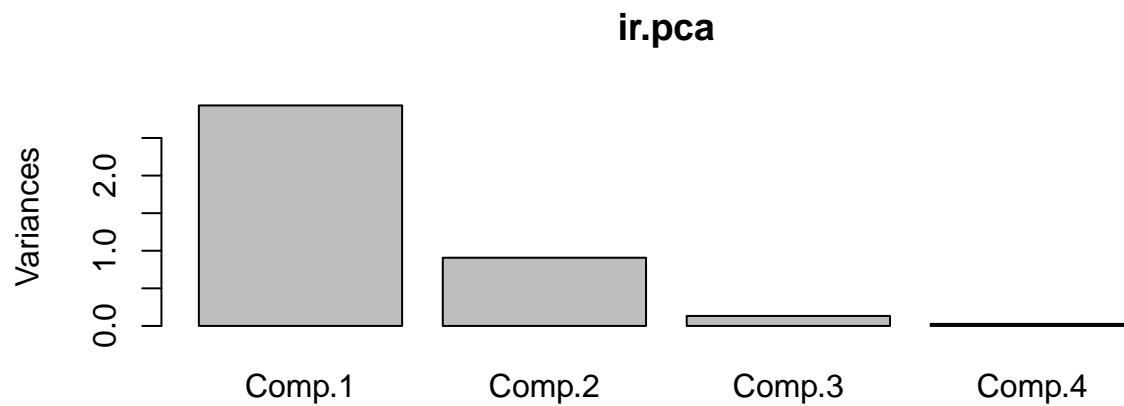
# Principal Component for the log-transformed iris data.
(ir.pca <- princomp(log(ir), cor = TRUE))
```

```
## 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.
```

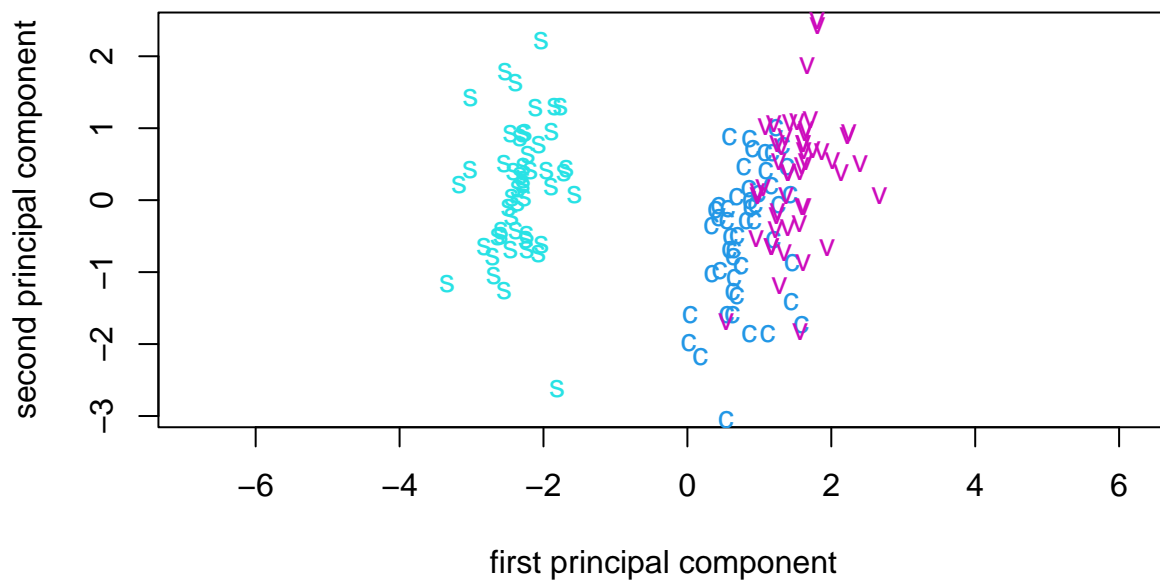
```
summary(ir.pca)
```

```
## 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-transformed 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))
```



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

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

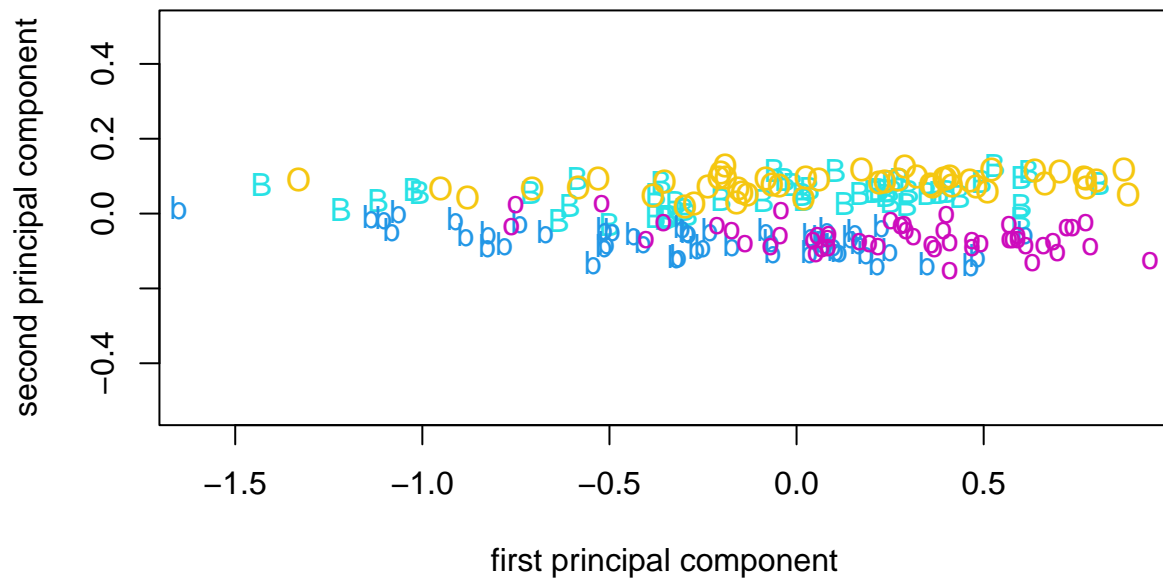
```
## Call:
## princomp(x = lcrabs)
##
## Standard deviations:
##      Comp.1      Comp.2      Comp.3      Comp.4      Comp.5
## 0.5166405 0.0746536 0.0479144 0.0248040 0.0090522
##
## 5 variables and 200 observations.
```

```
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
  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", "0", "o")[rep(1:4, each = 50)])
  glyph_colour(g$lcrabs) <- crabs.grp
  colorscheme(g) <- "Paired 4"
}
```

3) Distance methods

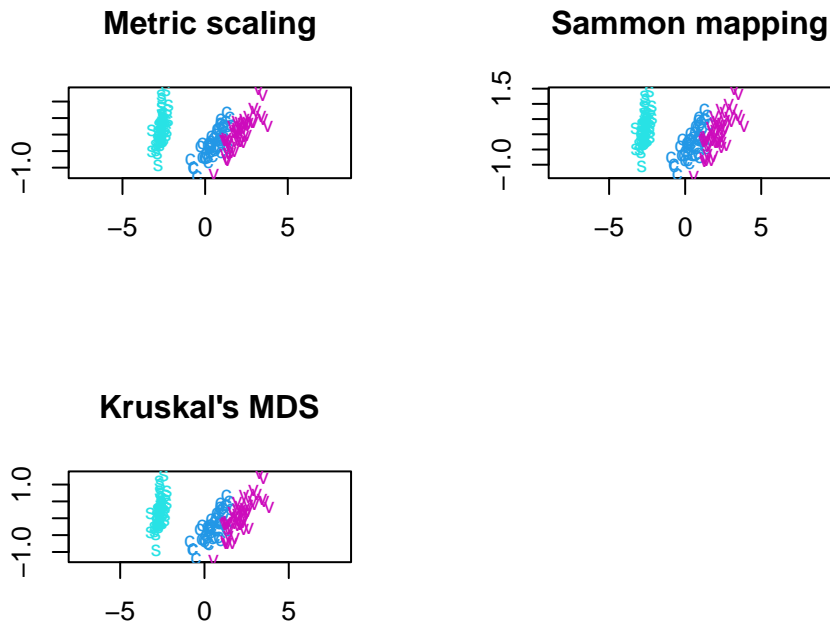
```
# Distance-based representations of the iris data
ir.scal <- cmdscale(dist(ir) , k = 2, eig = T)

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

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

```
par(mfrow = c(2,2))

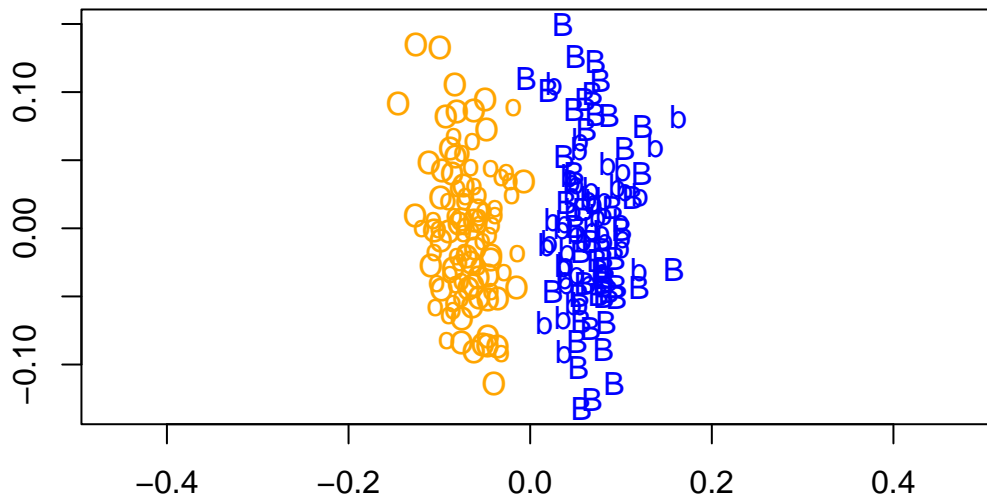
eqscplot(ir.scal$points, type = "n", main = "Metric scaling", ratio = 2)
text(ir.scal$points, labels = as.character(ir.species),
     col = 3 + as.integer(ir.species), cex = 0.8)
ir.sam <- sammon(dist(ir[-143,]))
eqscplot(ir.sam$points, type = "n", main = "Sammon mapping", ratio = 2)
text(ir.sam$points, labels = as.character(ir.species[-143]),
     col = 3 + as.integer(ir.species), cex = 0.8)
ir.iso <- isoMDS(dist(ir[-143,]))
eqscplot(ir.iso$points, type = "n", main = "Kruskal's MDS", ratio = 2)
text(ir.iso$points, labels = as.character(ir.species[-143]),
     col = 3 + as.integer(ir.species), cex = 0.8)
```



```

# 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$sex),]
lcrabs.sam <- sammon(dist(dslcrabs))
eqscplot(-lcrabs.sam$points, type = "n", xlab = "", ylab = "", tol = 0.08, ratio = 1.5)
text(-lcrabs.sam$points , labels = as.character(crabs.grp),
     col = rep(c("blue", "orange"), each = 100))

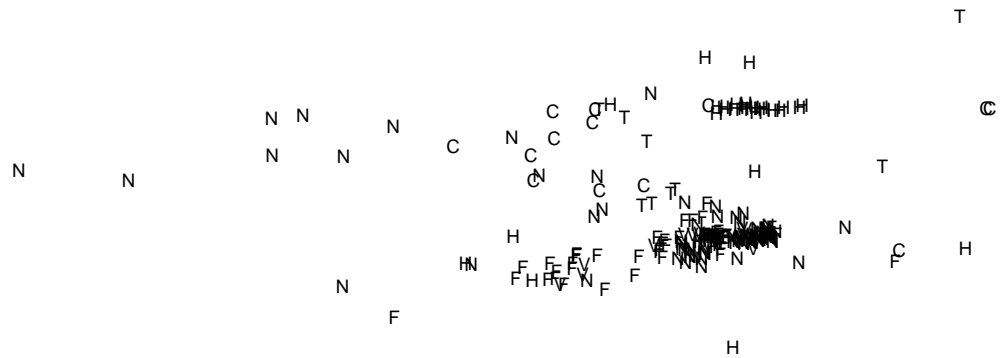
```



```

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

```

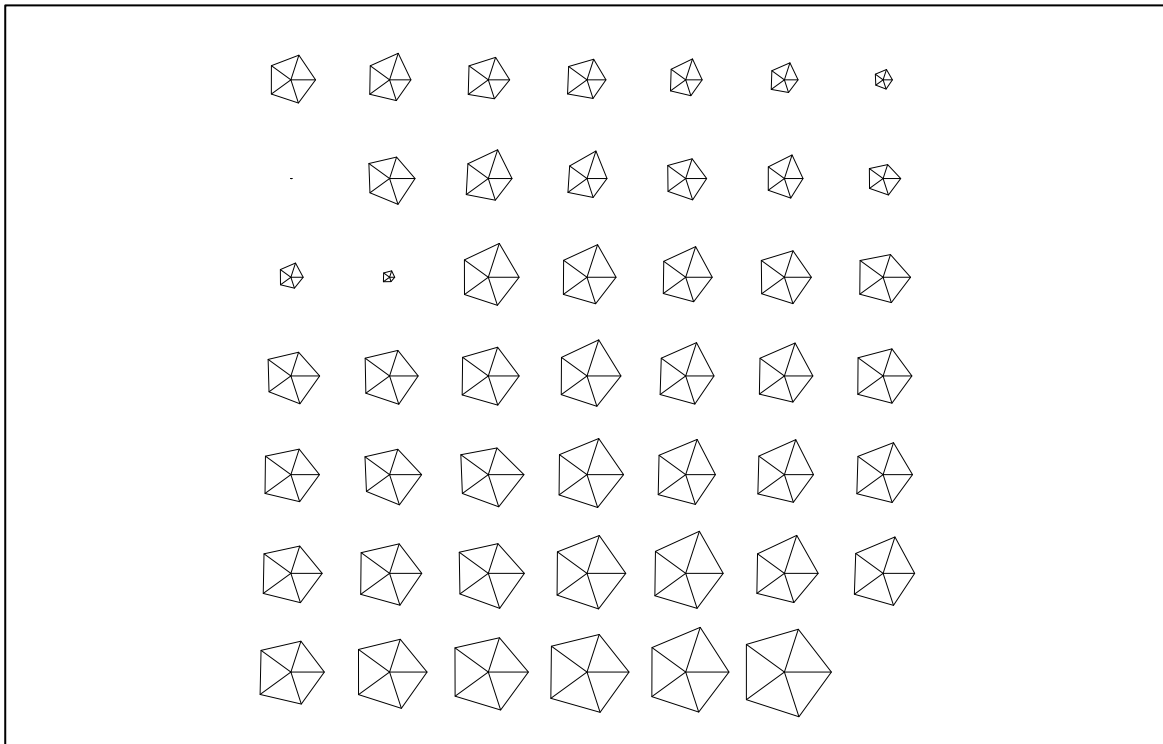


```
fgl.iso3 <- isoMDS(dist(as.matrix(fgl[-40, -10])), k = 3)
# S: brush(fgl.iso3$points)
fgl.col <- c("SkyBlue", "SlateBlue", "Orange", "Orchid", "Green", "HotPink")[fgl$type]
# xgobi(fgl.iso3$points, colors = fgl.col)
```

4) Self-organizing maps

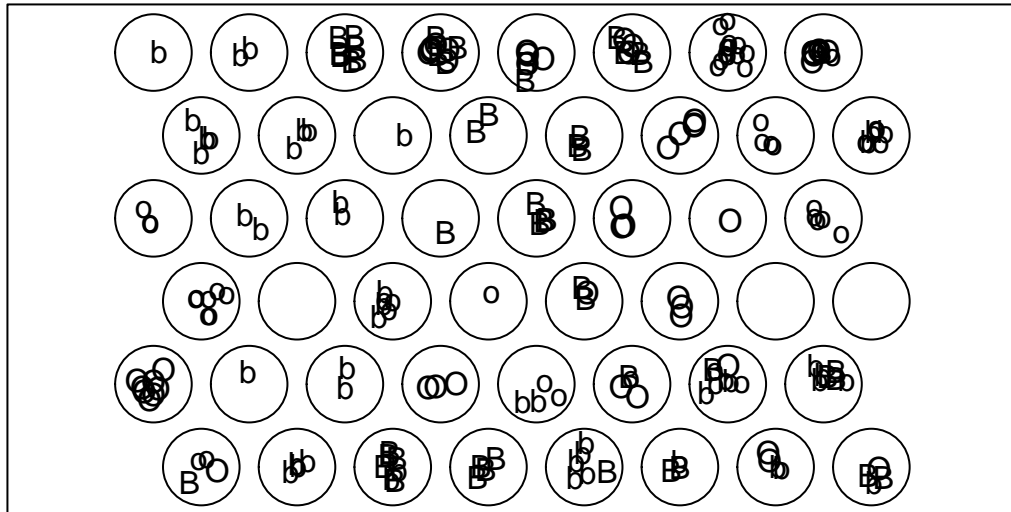
```
# Batch SOM applied to the crabs dataset.
set.seed(0)
gr <- somgrid(topo = "hexagonal")
crabs.som <- batchSOM(lcrabs, gr, c(4, 4, 2, 2, 1, 1, 1, 0, 0))

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



```
set.seed(0)

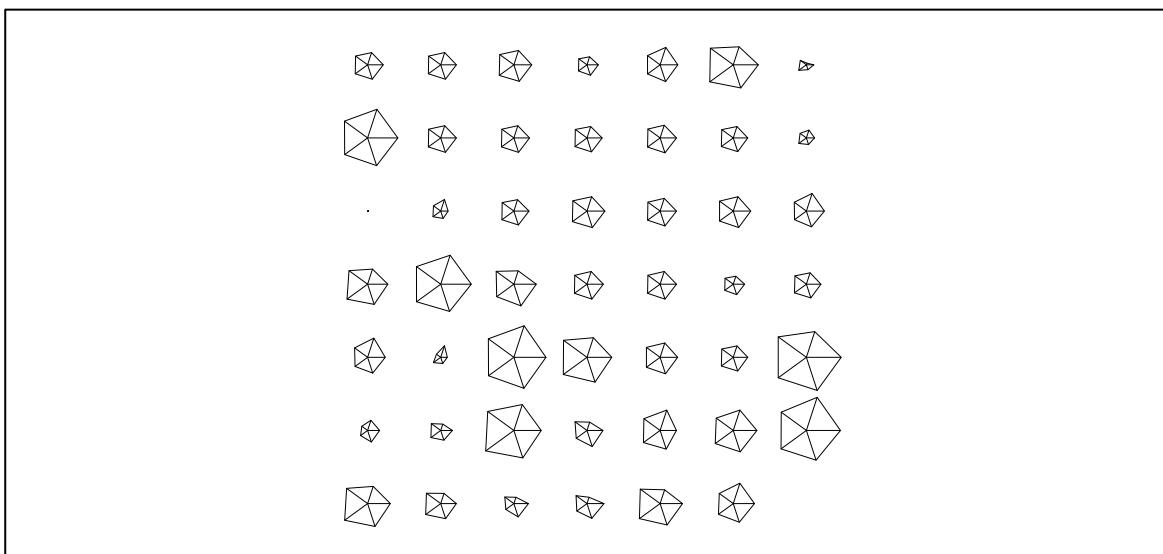
# Plot that shows the assignments of the original points
bins <- as.numeric(knn1(crabs.som$code, lcrabs, 0:47))
plot(crabs.som$grid, type = "n", frame.plot = T,
     xlim = c(min(crabs.som$grid$pts[,1])-0.4, max(crabs.som$grid$pts[,1])+0.4),
     ylim = c(min(crabs.som$grid$pts[,2])-0.4, max(crabs.som$grid$pts[,2])+0.4))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

```
set.seed(0)
```

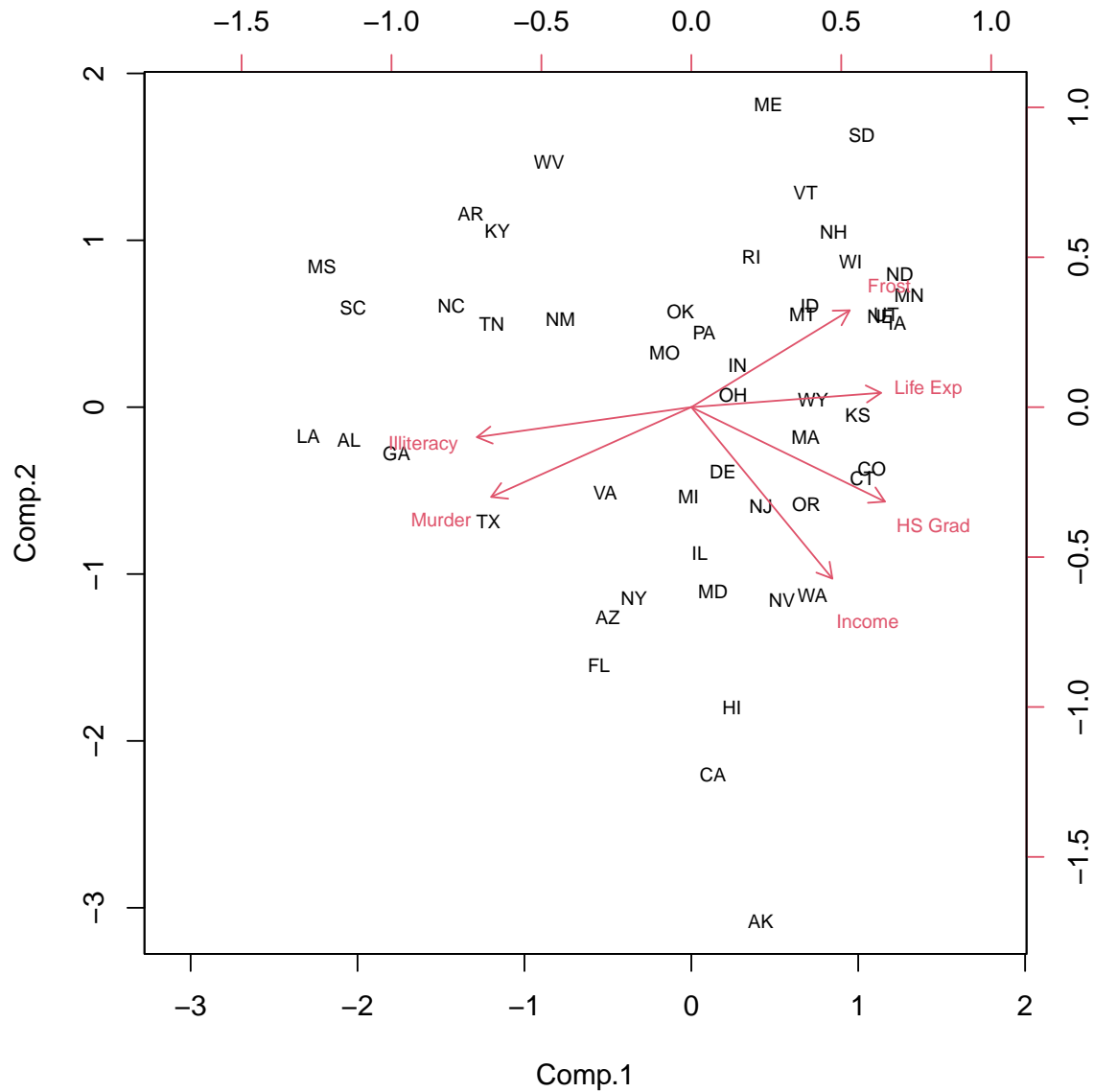
```
# Traditional SOM applied to the crabs dataset.
```

```
crabs.som2 <- SOM(lcrabs, gr); stars(crabs.som2$codes, frame.plot = T)
```



5) Biplots

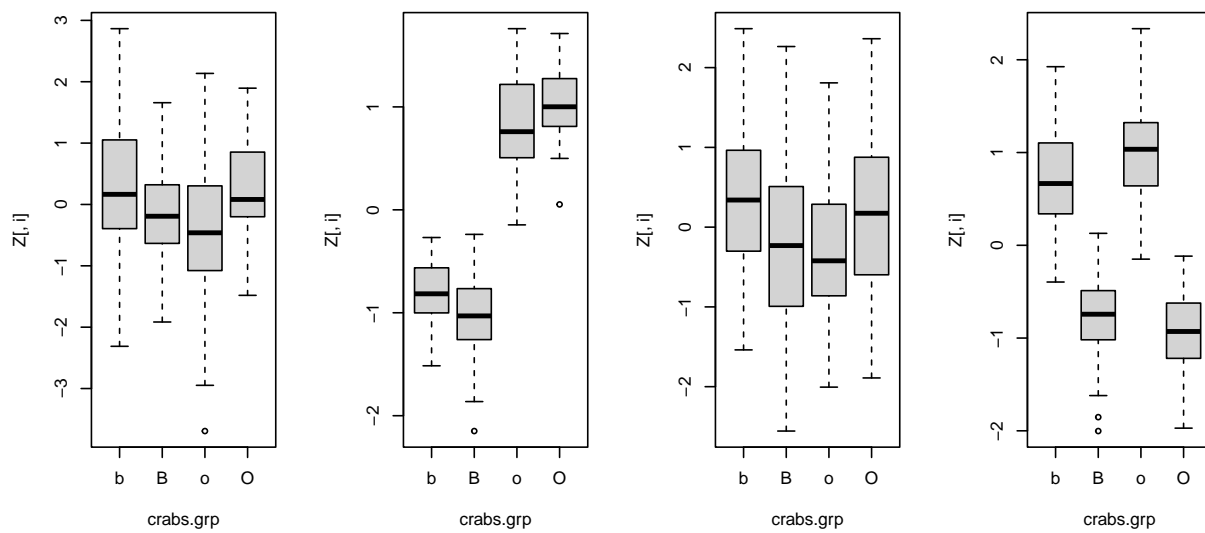
```
# Principal component biplot of the part of the state.x77 data.
state <- state.x77[, 2:7]; row.names(state) <- state.abb
state.pca <- princomp(state, cor = TRUE)
state.pca$loadings[,2] <- -state.pca$loadings[,2]
state.pca$scores[,2] <- -state.pca$scores[,2]
biplot(state.pca, pc.biplot = TRUE, cex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



6) Independent component analysis

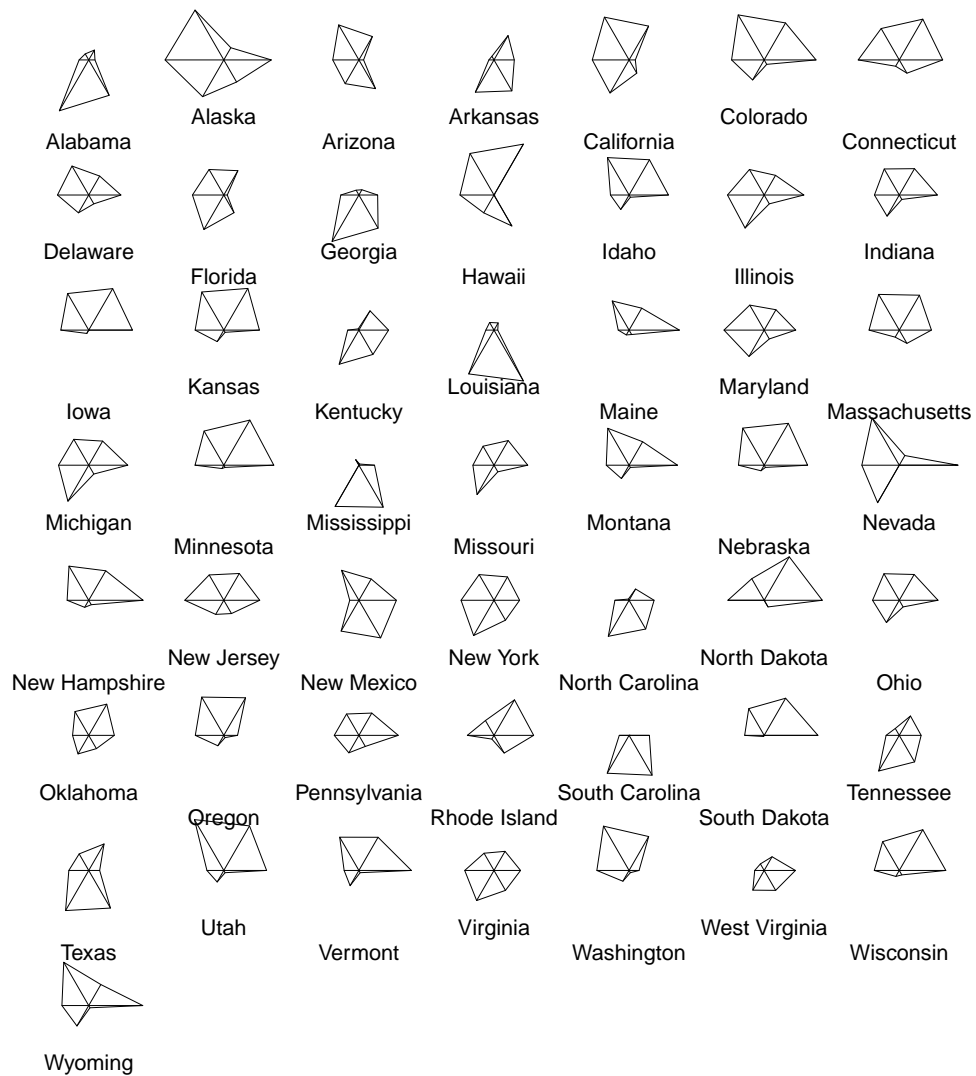
```
set.seed(0)

# Boxplots of four 'signals' recovered by ICA from the crabs data.
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)
```



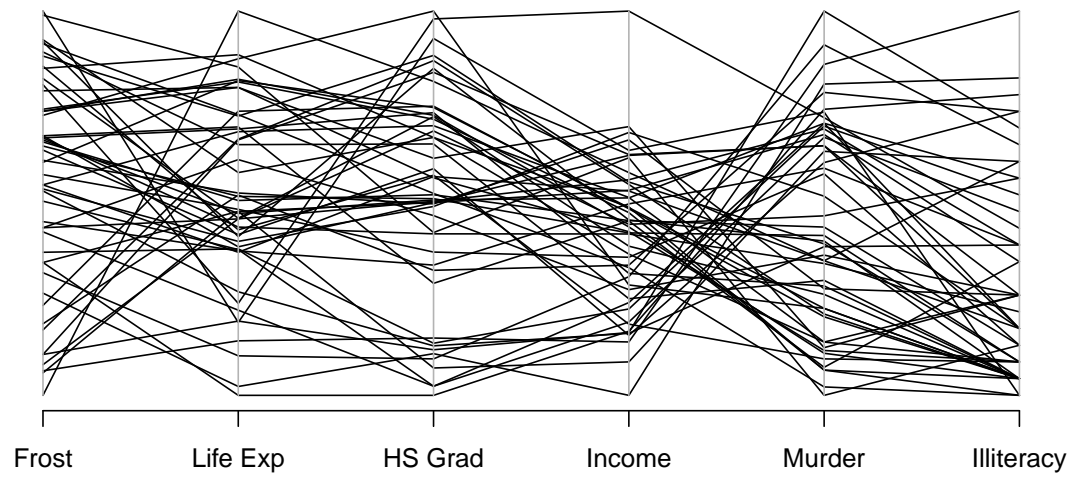
7) Glyph representations

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

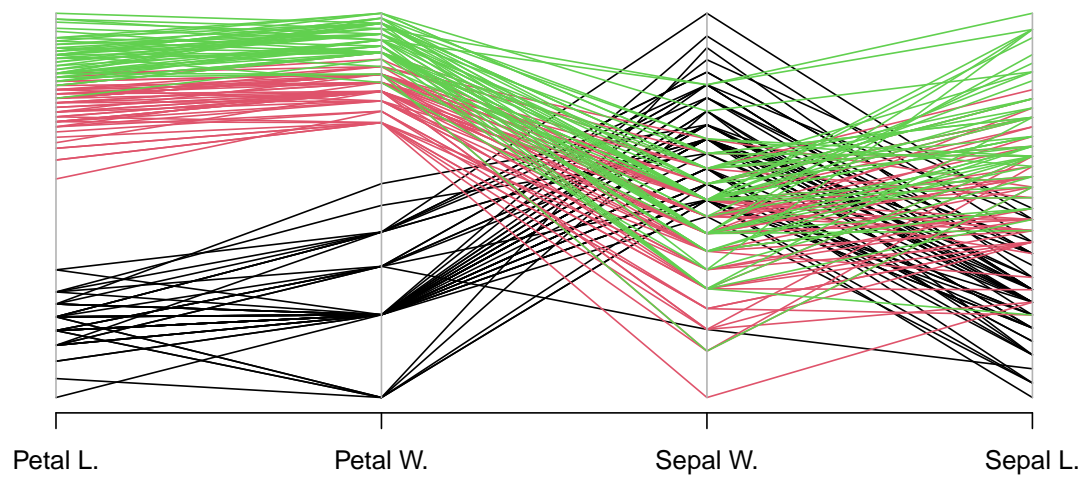


8) Parallel coordinate plots

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

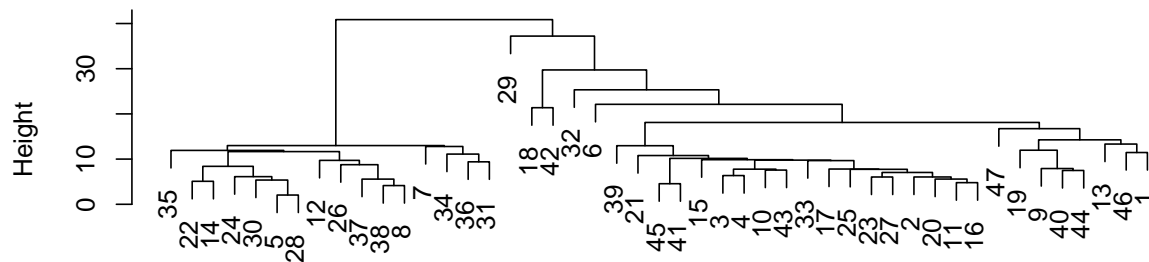


```
# Parallel coordinates plots of the log-transformed iris data  
parcoord(log(ir)[, c(3, 4, 2, 1)], col = 1 + (0:149)%/%50)
```



11.2 Cluster Analysis

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



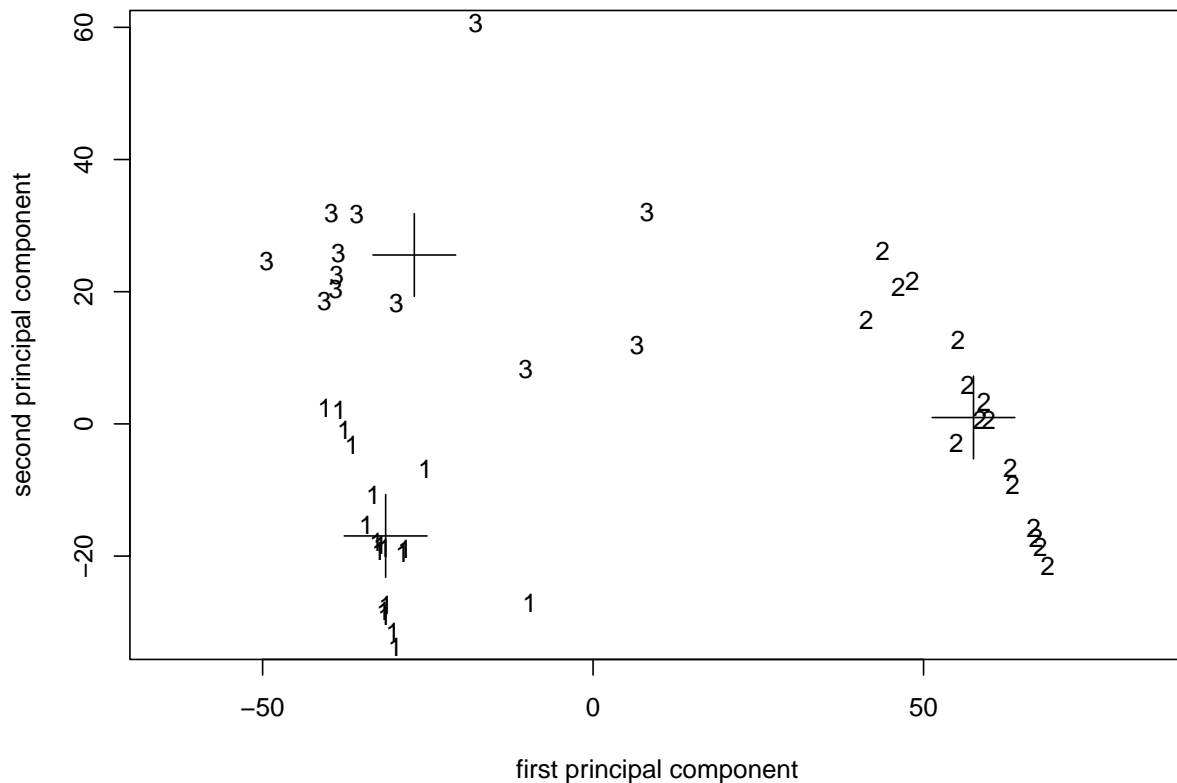
```
# Dendograms for the socio-economic data on Swiss provinces by divisive clustering
d <- diana(swiss.x, )
pltree(d, labels = d$order, main = "", xlab = "", sub = "")
```



```

# 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))
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)

```



```

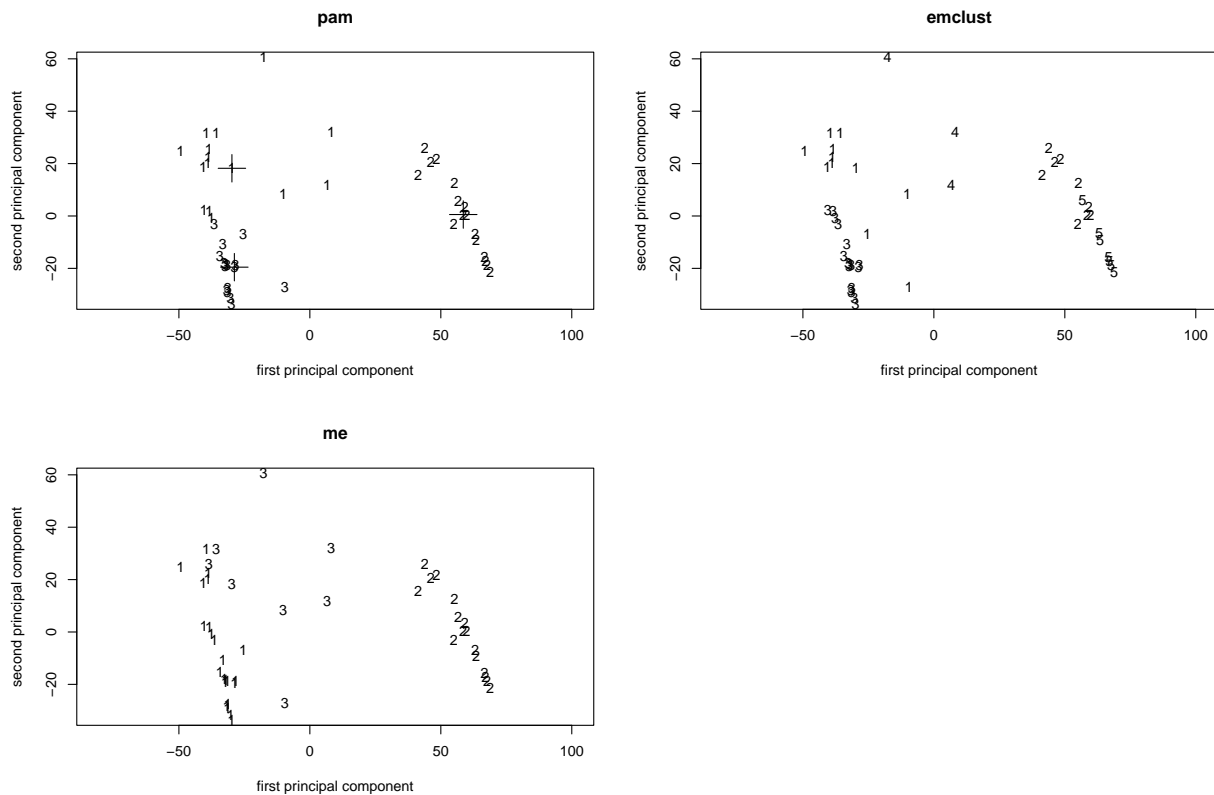
# Clusterings of the Swiss provinces data by pam, me, emclust
par(mfrow = c(2,2))

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

library(mclust)
vals <- mclustBIC(swiss.x)
sm <- summary(vals, swiss.x)
eqscplot (swiss.px [, 1: 2], type = "n",
          xlab = "first principal component" , ylab = "second principal component",
          main = "emclust")
text(swiss.px[, 1:2], labels = sm$classification)

h <- hc(modelName = "VWV", swiss.x)
mh <- as.vector(hclass(h, 3))
z <- me(modelName = "VWV", swiss.x, z = 0.5*(unmap(mh)+1/3))
eqscplot(swiss.px[, 1:2], type = "n",
          xlab = "first principal component", ylab = "second principal component",
          main = "me")
text(swiss.px[, 1:2], labels = max.col(z$z))

```



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
##
## 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
##
## 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
```

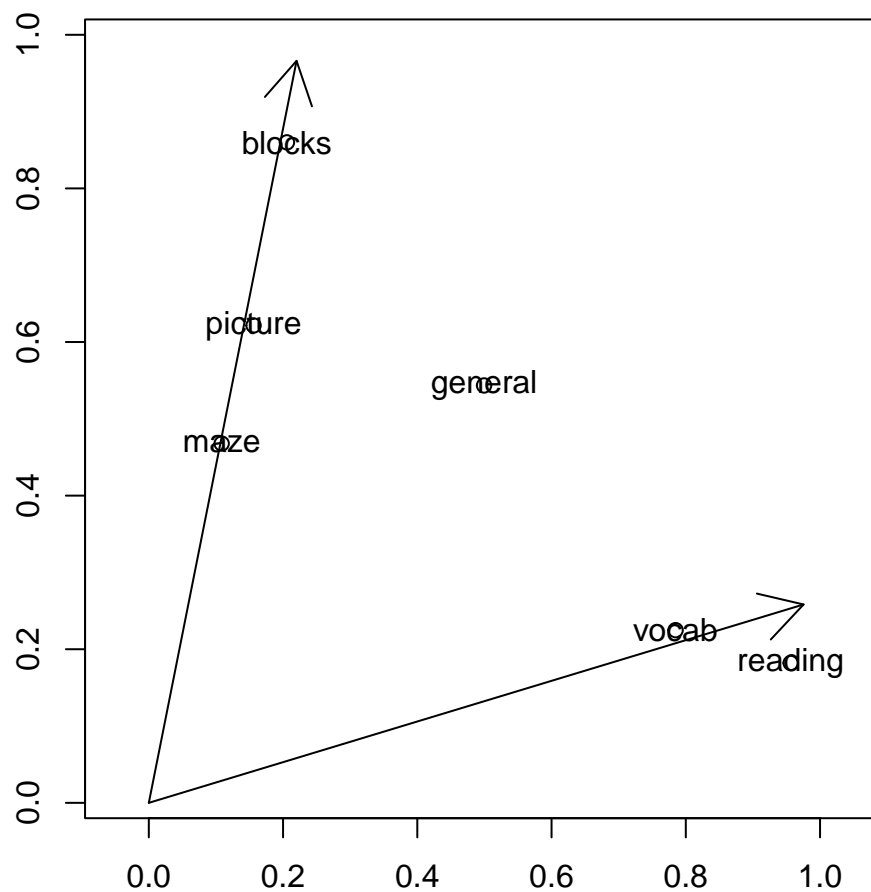
```
#summary(ability.FA)
round(loadings(ability.FA) %*% t(loadings(ability.FA)) +
      diag(ability.FA$uniq), 3)
```

```
##          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))
```

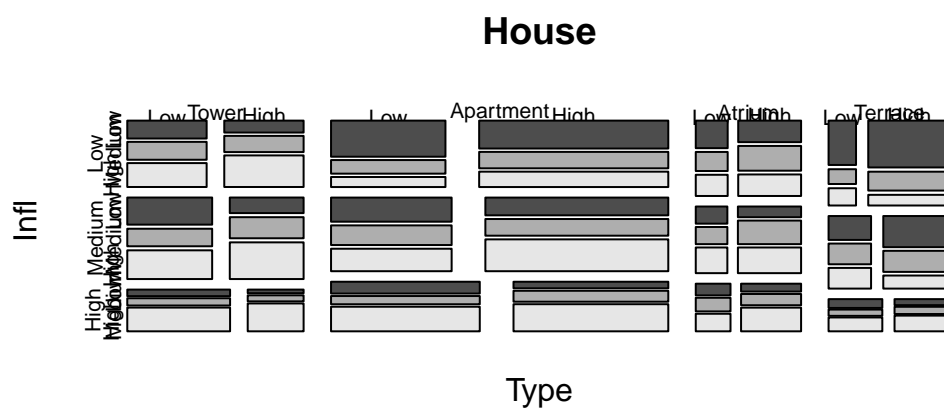
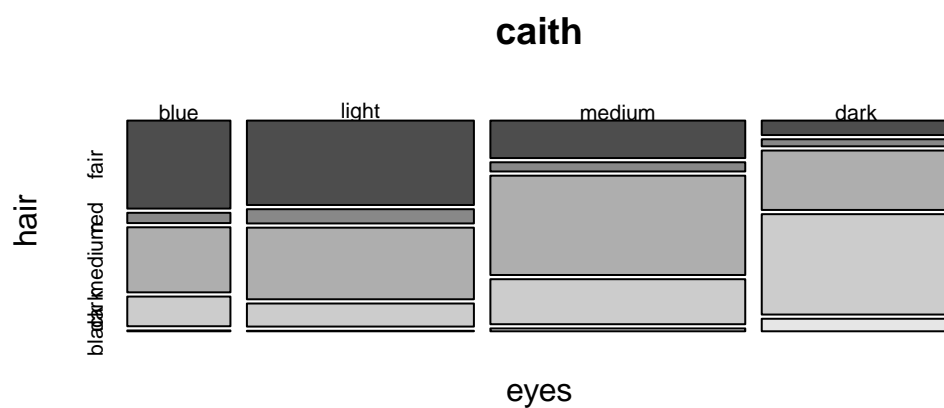
```
## Oblique rotation method Oblimin Quartimin converged.
## Loadings:
##          Factor1 Factor2
## general    0.3864  0.4745
## picture   -0.0110  0.6459
## blocks    -0.0263  0.8961
## maze      -0.0180  0.4883
## reading    0.9901 -0.0371
## vocab      0.7906  0.0526
##
## Rotating matrix:
##          [,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")
eqsplot(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)
```



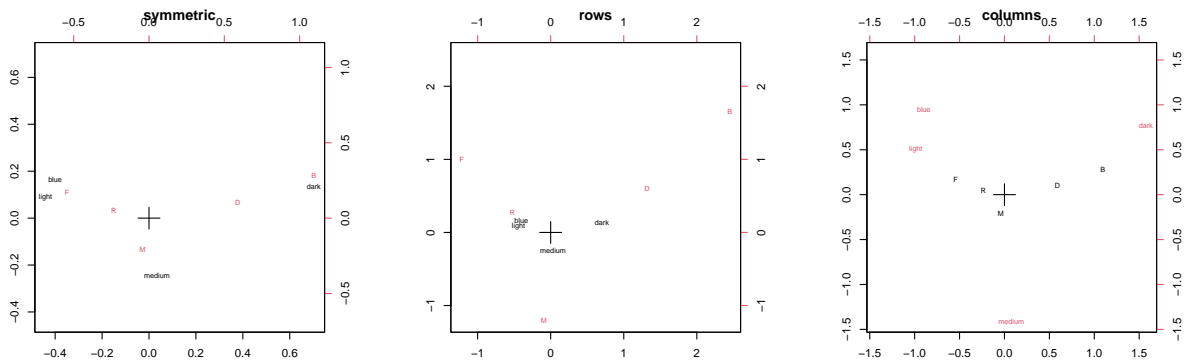
```
corresp(caith)
```

```
## 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))

```

