# MRU: Analyses of Liver Enzyme data

true

May 25, 2022

## Data

Plomteux showed that good separation between acute viral hepatitis (AVH, 57 patients), persistent chronic hepatitis (PCH, 44 patients), aggressive chronic hepatitis (ACH, 40 patients) and post-necrotic cirrhosis (PNN, 77 patients) could be achieved with only three liver function test.

The three analytes, aspartate aminotransferase (AST), alanine aminotransferase (ALT) and glutamate dehydrogenase (GIDH), were used to construct a four-group logistic model.

```
load("~/surfdrive/MultinomialBook/Datasets/liverdata/liverdata.RData")
colnames(liverdat) = c("Disease", "AS", "AL", "GD", "X?")
X = liverdat[ ,-c(1,5)]
xnames= colnames(X)
X.df = data.frame(X)
y = liverdat[ , 1]
y = factor(y, levels = c("1","2" ,"3", "4"), labels = c("AVH","PCH","ACH","PNC"))
G = class.ind(y)
colnames(G) = c("AVH","PCH","ACH","PNC")
G = G[, c(4,1,2,3)]
#ggpairs(X.df, aes(color=factor(y)))
```

The data seems to be very skew. Let us try a transformation down the ladder of powers:

```
XX = log(X)
X.df = data.frame(XX)
#ggpairs(X.df, aes(color=factor(y)))
X = scale(XX, center = TRUE, scale = FALSE)
```

This looks much better. Therefore, in the remainder we will work with these log-transformed predictor variables.

Lesaffre and Albert: "In a full parametric approach such as normal discriminant analysis, the first action would be to transform the data to remove skewness. In a semiparametric model, such as the logistic model, it is debatable whether to apply the transformation on the marginal distributions"

```
rm(X.df, XX, liverdat)
```

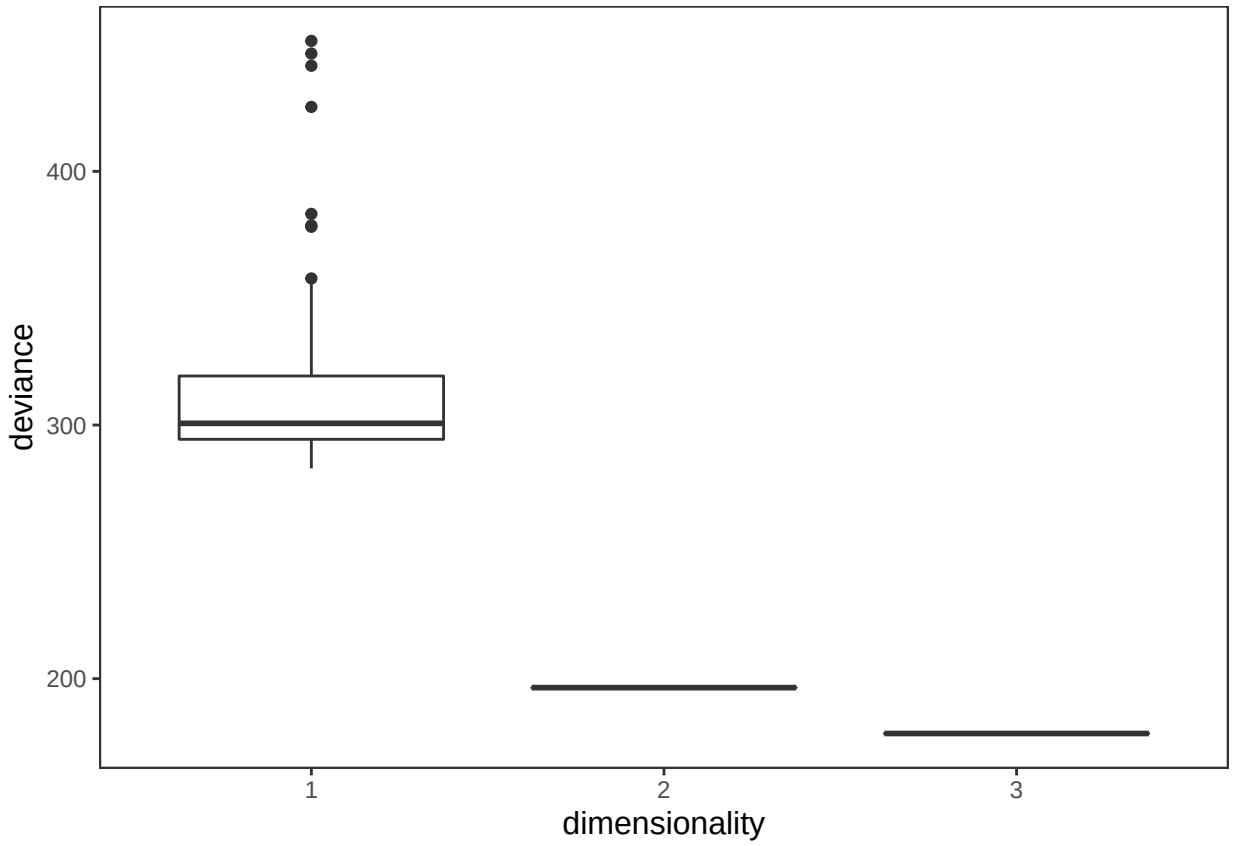
## Investigating Local optima

```
set.seed(1234)

M1results = vector(mode = "list", length = 100)
M2results = vector(mode = "list", length = 100)
M3results = vector(mode = "list", length = 100)

M1results[[1]] = mru.da(X, G, m = 1)
M2results[[1]] = mru.da(X, G, m = 2)
M3results[[1]] = mru.da(X, G, m = 3)

for(r in 2:100){
  M1results[[r]] = mru.random( X, G, m = 1 )
  M2results[[r]] = mru.random( X, G, m = 2 )
  M3results[[r]] = mru.random( X, G, m = 3 )
}
save(M1results, M2results, M3results, file = "liverresults.Rdata")
```

```
library(ggplot2)
dev1 = rep(NA, 100)
dev2 = rep(NA, 100)
dev3 = rep(NA, 100)
for(r in 1:100){
  dev1[r] = M1results[[r]]$deviance
  dev2[r] = M2results[[r]]$deviance
  dev3[r] = M3results[[r]]$deviance
}
df = data.frame(repl = rep(1:100, 3), dimensionality = as.factor(rep(1:3, each = 100)), deviance

p = ggplot(df, aes(dimensionality, deviance)) + geom_boxplot() + theme_apa()
ggsave("~/surfdrive/multldm/mrmdu/paper/figures/liverresults.pdf", plot = p)
```

```
## Saving 6.5 x 4.5 in image
```

```
p
```




## Analysis with Distance Model

```
out.liver1 = M1results[[which.min(dev1)]]
out.liver2 = M2results[[which.min(dev2)]]
out.liver3 = M3results[[which.min(dev3)]]

fit = matrix(NA, 3, 3)
fit[ , 1] = c(1,2,3)
fit[1, 2] = out.liver1$deviance
fit[2, 2] = out.liver2$deviance
fit[3, 2] = out.liver3$deviance
fit[1, 3] = ncol(X) * 1 + ncol(G) * 1
fit[2, 3] = ncol(X) * 2 + ncol(G) * 2 - 2*1/2
fit[3, 3] = ncol(X) * 3 + ncol(G) * 3 - 3*2/2
colnames(fit) = c("Dimensionality", "Deviance", "#params")
fit
```

```
##      Dimensionality Deviance #params
## [1,]              1 282.8796       7
## [2,]              2 196.4582      13
## [3,]              3 178.3907      18
```

Let us have a closer look at the two dimensional solution.

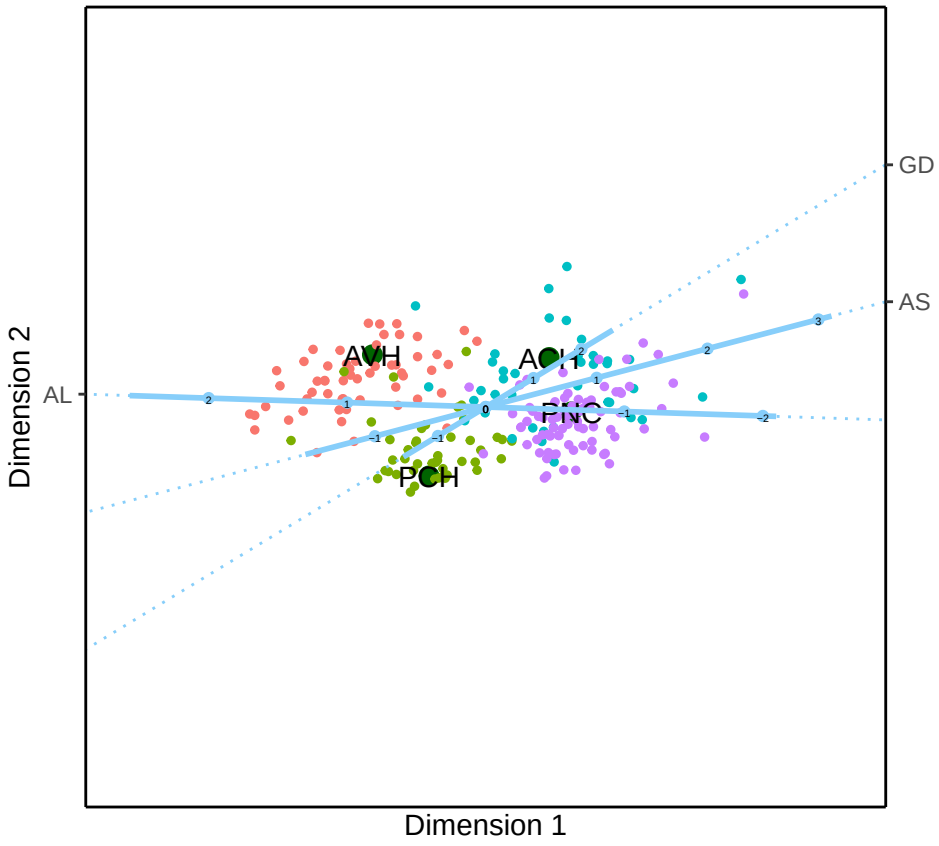
```
##           [,1]      [,2]
## [1,]  4.879450 1.2840477
## [2,] -6.096060 0.1961097
## [3,]  2.102674 1.2740412
```

```
##           [,1]       [,2]
## [1,]  3.767210 -0.2132406
## [2,] -4.987692  2.2998865
## [3,] -2.496821 -3.0312745
## [4,]  2.789280  2.1811121
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the exist
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in max(x): no non-missing arguments to max; returning -Inf
```




```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in min(x): no non-missing arguments to max; returning -Inf
```

Let us have a further look at the classification performance

```
D = outer(diag(U %*% t(U)), rep(1, 4)) + outer(rep(1,218), diag(V %*% t(V))) - 2* U %*% t(V)
D = sqrt(D)
PR = exp(-D)/rowSums(exp(-D))
yhat = apply(PR, 1, which.max)
```

```r
yy = apply(G, 1, which.max)
tab1 = table(yy, yhat)
tab1
```

```
##    yhat
## yy   1  2  3  4
##   1 69  0  1  7
##   2  0 53  2  2
##   3  3  4 36  1
##   4 14  2  2 22
```

In total, 82.5688073 percent is correctly classified

## Simulation

First develop a function that generates data based on population class points, regression weights and some covariance matrix:

```r
gendata = function(N, covX, B, V){
  library(mvtnorm)
  library(poLCA)
  P = nrow(B)
  C = nrow(V)
  X = rmvnorm(N, mean = rep(0, P), sigma = covX)
  X = scale(X, center = TRUE, scale = FALSE)
  U = X %*% B

  D = sqrt(outer(diag(U %*% t(U)), rep(1, C)) + outer(rep(1, N), diag(V %*% t(V))) - 2 * U %*% t
  Pr = exp(-D)/rowSums(exp(-D))
  G = class.ind(rmulti(Pr))
  output = list(X = X, G = G)
}

mse = function(A, B){(A-B)^2}
```

And now we simulate data with these parameters and varying sample sizes (100, 200, 500, 1000). On the generated data sets we fit the multinomial restricted unfolding and we compare the obtained results with the population parameters.

```r
V = out.liver2$V
B = out.liver2$B
covX = cov(X)

# small simulation
Ns = c(100, 200, 500, 1000)
plts = vector(mode = "list", length = length(Ns))
Bbias = vector(mode = "list", length = length(Ns))
Brmse = vector(mode = "list", length = length(Ns))
Vbias = vector(mode = "list", length = length(Ns))
```

```r
Vrmse = vector(mode = "list", length = length(Ns))

source("ggbagplot.R")
set.seed(1234)
for(n in 1:length(Ns)){
  N = Ns[n]
  Bs = vector(mode = "list", length = 100)
  Vs = vector(mode = "list", length = 100)


  for(rep in 1:100){
    #cat("This is repetition:", rep, "\n")
    mydat = gendata(N, covX, B, V)
    out <- mru.user(mydat$X, mydat$G, m = 2, B.start = B, V.start = V)
    # orthogonal procrustes analysis on U
    pq = svd(t(V) %*% out$V)
    TT = pq$v %*% t(pq$u)
    Bs[[rep]] = out$B %*% TT
    Vs[[rep]] = out$V %*% TT
  }

  # # bias
  Bbias[[n]] = Reduce("+", Bs) / length(Bs) - B
  Vbias[[n]] = Reduce("+", Vs) / length(Vs) - V
  #
  # rmse
  Brmse[[n]] = sqrt(Reduce("+" ,lapply(Bs, mse, B))/length(Bs))
  Vrmse[[n]] = sqrt(Reduce("+" ,lapply(Vs, mse, V))/length(Vs))

  ######################################################
  ######################################################
  # A visualization of the results
  ######################################################
  ######################################################

  Bdf = data.frame(B); colnames(Bdf) = c("x", "y")
  Vdf = data.frame(V); colnames(Vdf) = c("x", "y")
  Vdf$class = as.factor(c(1,2,3,4))

  # class points
  Vslong = matrix(NA, 400, 2); Bslong = matrix(NA, 300, 2)
  for(r in 1:100){
    Vslong[((r-1)*4 + 1):(r*4), ] = Vs[[r]]
    Bslong[((r-1)*3 + 1):(r*3), ] = Bs[[r]]
  }

  Vslong = data.frame(cbind(rep(1:4, 100), Vslong))
```

```r
colnames(Vslong) = c("class", "x", "y")
Vslong$class = as.factor(Vslong$class)

# hull_data <-
#   Vslong %>%
#   group_by(class) %>%
#   slice(chull(x, y))

# plt = ggplot(Vslong, aes(x = x, y = y, col = class)) +
#   # xlim(-15,15) +
#   # ylim(-15,15) +
#   geom_point(alpha = 0.5) +
#   geom_polygon(data = hull_data, aes(fill = class, colour = class), alpha = 0.3, show.legend
#   scale_color_manual(values = brewer.pal(9,"Greens")[6:9]) +
#   scale_fill_manual(values = brewer.pal(9,"Greens")[6:9])

plt = ggplot(Vslong, aes(x = x, y = y, col = class)) +
  geom_point(alpha = 0.5) +
  geom_bag(prop = 0.9, aes(fill = class, colour = class), alpha = 0.3, show.legend = FALSE) +
  scale_color_manual(values = brewer.pal(8,"Paired")[c(2,4,6,8)]) +
  scale_fill_manual(values = brewer.pal(8,"Paired")[c(2,4,6,8)]) +
  theme(legend.position = "none")

# predictor variables
Bslong = data.frame(cbind(rep(1:3, 100), Bslong))
colnames(Bslong) = c("pred", "x", "y")
Bslong$pred = as.factor(Bslong$pred)

plt = plt +
  # geom_abline(intercept = 0, slope = Bslong$y/Bslong$x, col = "lightskyblue", alpha = 0.3) +
  geom_point(data = Bslong, aes(x = x, y = y), colour = "darkblue", alpha = 0.3)
  # geom_bag(data = Bslong, prop = 0.9, aes(fill = pred, colour = pred), alpha = 0.3, show.leg
  # scale_color_manual(values = brewer.pal(9,"Blues")[c(5,6,7)]) +
  # scale_fill_manual(values = brewer.pal(9,"Blues")[c(5,6,7)]) +
  # theme(legend.position = "none")

plt = plt +
  geom_point(data = Vdf, aes(x = x, y = y), colour = brewer.pal(8,"Paired")[c(2,4,6,8)], shape
  geom_abline(intercept = 0, slope = Bdf[ ,2]/Bdf[ ,1], colour = "darkblue", size = 1) +
  geom_point(data = Bdf, aes(x =x, y = y), col = "darkblue", size = 5)

plt = plt +
  labs(
    x = "Dimension 1",
    y = "Dimension 2"
    ) +
  xlim(-15,15) +
```

```
    ylim(-15,15) +
    coord_fixed() +
    theme_apa() +
    theme(legend.position = "none")

  # for(r in 1:100){
  #   Bdfr = data.frame(Bs[[r]]); colnames(Bdfr) = c("x", "y")
  #   Vdfr = data.frame(Vs[[r]]); colnames(Vdfr) = c("x", "y")
  #   plt = plt + geom_point(data = Bdfr, aes(x = x, y = y), col = "darkblue", size = 0.2, alpha
  #     geom_point(data = Vdfr, aes(x = x,y = y), col = "darkred", size = 0.2, alpha = 0.1)
  # }

  plts[[n]] = plt + labs(title = paste("Estimates for N = ", N))
}
```

```
## Loading required package: scatterplot3d
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
Bbias
```

```
## [[1]]
##           [,1]      [,2]
## [1,]  3.620008 1.1025808
## [2,] -4.469864 0.4394513
## [3,]  1.391912 1.2633305
##
## [[2]]
##            [,1]       [,2]
## [1,]  0.9455979  0.4421151
## [2,] -1.1745763 -0.1351301
## [3,]  0.4333506  0.2146858
##
## [[3]]
##            [,1]        [,2]
## [1,]  0.2839432  0.17984227
## [2,] -0.3534823 -0.08487556
## [3,]  0.1248168  0.08515563
##
## [[4]]
##             [,1]        [,2]
## [1,]  0.07374764 -0.01333318
## [2,] -0.10173632  0.04872432
```

```
## [3,]  0.03233577  0.05406315
```

```
Brmse
```

```
## [[1]]
##          [,1]     [,2]
## [1,] 5.993275 2.530939
## [2,] 7.008845 2.905615
## [3,] 2.387652 2.653818
##
## [[2]]
##           [,1]      [,2]
## [1,] 1.7027575 0.8319590
## [2,] 1.9860571 1.0120295
## [3,] 0.8215892 0.5785447
##
## [[3]]
##           [,1]      [,2]
## [1,] 0.6868572 0.3842227
## [2,] 0.7818931 0.4274692
## [3,] 0.3801673 0.2692266
##
## [[4]]
##           [,1]      [,2]
## [1,] 0.3606358 0.1965473
## [2,] 0.4193411 0.2589059
## [3,] 0.2340592 0.1830463
```

```
Vbias
```

```
## [[1]]
##           [,1]       [,2]
## [1,]  2.418296  0.5160625
## [2,] -2.964994  1.3581654
## [3,] -1.555151 -1.3236074
## [4,]  2.113751  1.2601008
##
## [[2]]
##            [,1]        [,2]
## [1,]  0.6450590  0.07049692
## [2,] -0.5263180  0.72260950
## [3,]  0.1555752 -0.51817903
## [4,]  0.4420542  0.42639355
##
## [[3]]
##              [,1]         [,2]
## [1,]  0.225676025 -0.008791923
## [2,] -0.174931835  0.168637316
## [3,] -0.005796341 -0.208832872
## [4,]  0.162211077  0.098139309
```

```
##
## [[4]]
##              [,1]        [,2]
## [1,]  0.07467040  0.01877506
## [2,] -0.07291315  0.04433833
## [3,] -0.02524971 -0.01352306
## [4,]  0.05759520  0.04847037
```

```
Vrmse
```

```
## [[1]]
##          [,1]     [,2]
## [1,] 4.078486 1.712234
## [2,] 4.993449 3.591645
## [3,] 4.045315 3.594587
## [4,] 3.828595 2.219028
##
## [[2]]
##          [,1]      [,2]
## [1,] 1.429504 0.6227668
## [2,] 1.103787 1.9071151
## [3,] 1.572870 1.6629270
## [4,] 1.282721 0.8997762
##
## [[3]]
##           [,1]      [,2]
## [1,] 0.5218706 0.3082293
## [2,] 0.5703439 0.7450966
## [3,] 0.7011275 0.7648110
## [4,] 0.5009869 0.3146613
##
## [[4]]
##           [,1]      [,2]
## [1,] 0.3742923 0.1681263
## [2,] 0.3816971 0.4371841
## [3,] 0.3687681 0.4770918
## [4,] 0.3470131 0.2837593
```

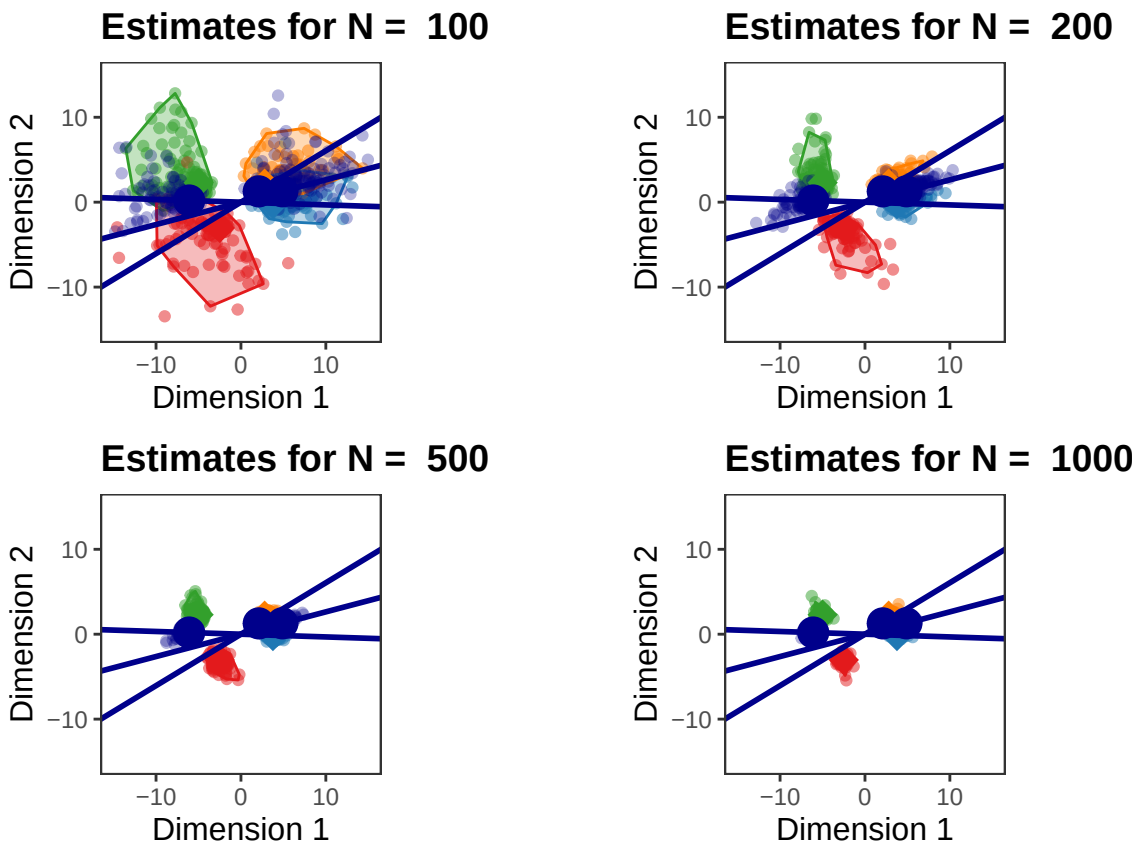
```
save(Bbias, Brmse, Vbias, Vrmse, file = "simliverresults.Rdata")
plt = ggarrange(plts[[1]], plts[[2]], plts[[3]], plts[[4]], nrow = 2, ncol = 2)
```

```
## Warning: Removed 14 rows containing non-finite values (statbag).
```

```
## Warning: Removed 14 rows containing missing values (geom_point).
```

```
## Warning: Removed 24 rows containing missing values (geom_point).
```

```
plt
```




```
ggsave("~/surfdrive/multldm/mrmdu/paper/figures/simliver.pdf", plot = plt, width = 11.7, height
```

## Analysis with Squared Distance Model

Identification restrictions

- Translation: set the coordinates for first class to zero
- Rotation: set the upper triangle of **V to zero**
- **Scaling: set some elements in V to one**

The following function minimizes the deviance for the IPC model.

With the following code we first create our design matrix and initiate parameters for minimization of the deviance function. Then we call the `optim`-function to find a set of parameters that maximizes the likelihood or minimizes the deviance.

```
X = cbind(1,X)
pars0 = c(rep(0,8), rnorm(3))
out = optim(pars0,ipc.deviance, G = G, X = X, method = "BFGS", control = list(trace = 2, maxit =
```

```
## initial  value 636.628961
## iter  10 value 249.638502
## iter  20 value 213.349672
```

```
## iter  30 value 211.224901
## iter  40 value 210.792655
## final  value 210.792241
## converged
```

The value of the deviance is 210.7922407 for which the estimated parameters are

```
P = ncol(X)
C = ncol(G)

B = matrix(out$par[1:(2*P)], P, 2)
B
```

```
##           [,1]       [,2]
## [1,] -0.434482  0.1222094
## [2,] -4.438044 -3.2698188
## [3,]  6.005340  2.9487862
## [4,] -1.847469 -1.8763967
```

```
U = X %*% B

V = matrix(0, C, 2)
V[2,1] = V[3,2] = 1
V[3:C,1] = out$par[(2*P+1):(2*P+2)]
V[4:C,2] = out$par[(2*P+3)]
V
```

```
##           [,1]       [,2]
## [1,] 0.0000000  0.0000000
## [2,] 1.0000000  0.0000000
## [3,] 0.2681650  1.0000000
## [4,] 0.2683579 -0.3846213
```

Let us have a further look at the classification performance

```
D2 = outer(diag(U %*% t(U)), rep(1, 4)) + outer(rep(1,218), diag(V %*% t(V))) - 2* U %*% t(V)
PR2 = exp(-D2)/rowSums(exp(-D2))
yhat2 = apply(PR2, 1, which.max)
tab2 = table(yy, yhat2)
tab2
```

```
##    yhat2
## yy   1  2  3  4
##   1 72  0  1  4
##   2  0 52  3  2
##   3  3  4 36  1
##   4 23  2  1 14
```

In total, 79.8165138 percent is correctly classified

### Graphical representation

Let us have a further look at the classification performance

```
D2 = outer(diag(UU %*% t(UU)), rep(1, 4)) + outer(rep(1,218), diag(VV %*% t(VV))) - 2* UU %*% t(
PR2 = exp(-D2)/rowSums(exp(-D2))
yhat2 = apply(PR2, 1, which.max)
tab2 = table(yy, yhat2)
tab2
```

```
##    yhat2
## yy   1  2  3  4
##   1 72  0  1  4
##   2  0 52  3  2
##   3  3  4 36  1
##   4 23  2  1 14
```

In total, 79.8165138 percent is correctly classified

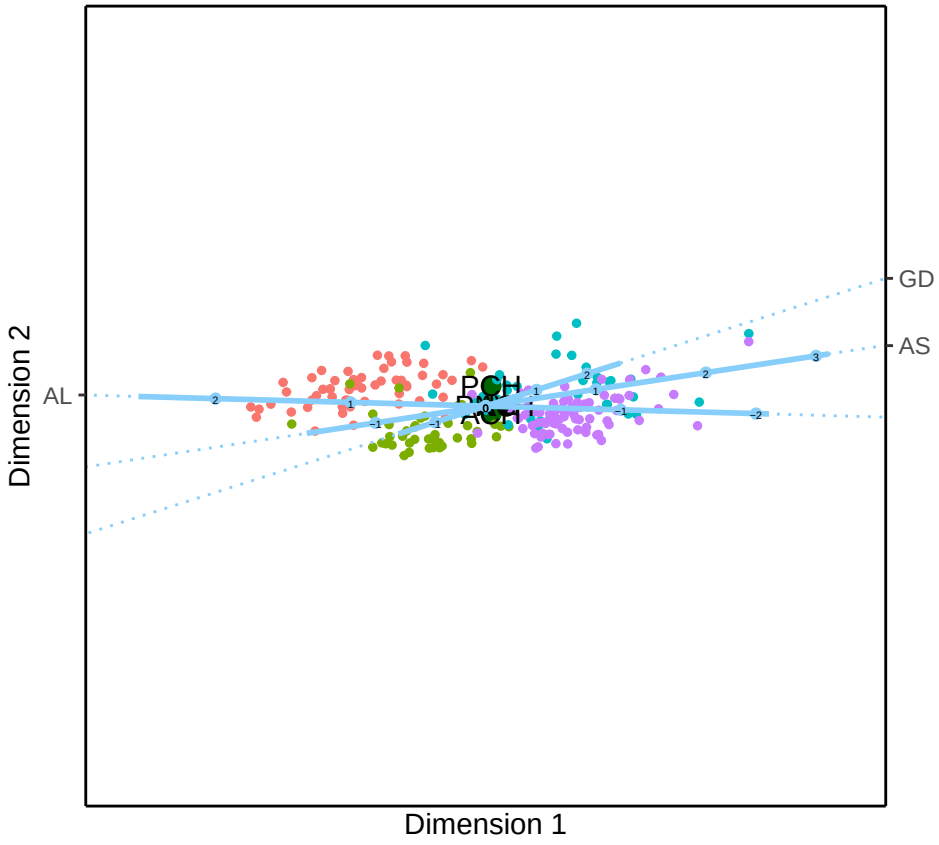
```
NN = as.data.frame(UU)
colnames(NN) = c("Dim1", "Dim2")

VV = as.data.frame(V)
colnames(VV) = c("Dim1", "Dim2")

P = nrow(B)
Xo = X[ , -1]

# for solid line
MCx1 <- data.frame(labs=character(),
                   varx = integer(),
                   Dim1 = double(),
                   Dim2 = double(), stringsAsFactors=FALSE)
# for markers
MCx2 <- data.frame(labs=character(),
                   varx = integer(),
                   Dim1 = double(),
                   Dim2 = double(), stringsAsFactors=FALSE)

ll = 0
lll = 0
for(pp in 1:P){
  b = matrix(B[pp , ], 2, 1)
  # solid line
  minx = min(Xo[, pp])
  maxx = max(Xo[, pp])
  m.x1 = c(minx,maxx)
  markers1 = matrix(m.x1, 2, 1)
  markerscoord1 = outer(markers1, b) # markers1 %*% t(b %*% solve(t(b) %*% b))
  MCx1[(ll + 1): (ll + 2), 1] = paste0(c("min", "max"), pp)
```

```
  MCx1[(ll + 1): (ll + 2), 2] = pp
  MCx1[(ll + 1): (ll + 2), 3:4] = markerscoord1
  ll = ll + 2
  # markers
  m.x2 = pretty(Xo[, pp])
  m.x2 = m.x2[which(m.x2 > minx & m.x2 < maxx)]
  l.m = length(m.x2)
  markers2 = matrix(m.x2, l.m, 1)
  markerscoord2 = outer(markers2, b) # markers2 %*% t(b %*% solve(t(b) %*% b))
  MCx2[(lll + 1): (lll + l.m), 1] = paste(m.x2)
  MCx2[(lll + 1): (lll + l.m), 2] = pp
  MCx2[(lll + 1): (lll + l.m), 3:4] = markerscoord2
  lll = lll + l.m
} # loop p

p2 = ggplot() +
    geom_point(data = VV, aes(x = Dim1, y = Dim2), colour = "darkgreen", size = 3) +
    geom_point(data = NN, aes(x = Dim1, y = Dim2, color = y), size = 1, show.legend = FALSE) +
    xlab("Dimension 1") +
    ylab("Dimension 2")

p2 = p2 + geom_text(data = VV, aes(x = Dim1, y = Dim2),
                    label = colnames(G),
                    vjust = 0, nudge_y = -0.5)


xcol = "lightskyblue"
p2 = p2 + geom_abline(intercept = 0, slope = B[,2]/B[,1], colour = xcol, linetype = 3) +
    geom_line(data = MCx1, aes(x = Dim1, y = Dim2, group = varx), col = xcol, size = 1) +
    geom_point(data = MCx2, aes(x = Dim1, y = Dim2), col = xcol) +
    geom_text(data = MCx2, aes(x = Dim1, y = Dim2, label = labs), nudge_y = -0.08, size = 1.5)

a = ceiling(max(abs(c(ggplot_build(p2)$layout$panel_scales_x[[1]]$range$range, ggplot_build(p2)$

idx1 = apply(abs(B), 1, which.max)
t = s = rep(NA,(P))
for(pp in 1:(P)){
    t[(pp)] = (a * 1.1)/(abs(B[pp,idx1[(pp)]])) * B[pp,-idx1[(pp)]]
    s[(pp)] = sign(B[pp,idx1[(pp)]])
}
CC = cbind(idx1, t, s)
bottom = which(CC[, "idx1"] == 2 & CC[, "s"] == -1)
top =  which(CC[, "idx1"] == 2 & CC[, "s"] == 1)
right = which(CC[, "idx1"] == 1 & CC[, "s"] == 1)
left = which(CC[, "idx1"] == 1 & CC[, "s"] == -1)
```

```
p2 = p2 + scale_x_continuous(limits = c(-a,a), breaks = CC[bottom, "t"], labels = xnames[bottom]
                             sec.axis = sec_axis(trans ~ ., breaks = CC[top, "t"], labels = xnam
p2 = p2 + scale_y_continuous(limits = c(-a,a), breaks = CC[left, "t"], labels = xnames[left],
                             sec.axis = sec_axis(trans ~ ., breaks = CC[right, "t"], labels = xn
p2 = p2 + coord_fixed()

p2 = p2 + theme_bw()
p2 = p2 + theme(axis.line = element_line(colour = "black"),
                    panel.grid.major = element_blank(),
                    panel.grid.minor = element_blank(),
                    panel.border = element_blank(),
                    panel.background = element_blank())

p2
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in max(x): no non-missing arguments to max; returning -Inf
```




```
ggsave("~/surfdrive/multldm/mrmdu/paper/figures/liverplot2.pdf", plot = p2, width = 11.7, height
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in min(x): no non-missing arguments to max; returning -Inf
```

## Multinomial Logistic Regression

As a comparison we also fit a standard multinomial logistic regression on this data set. Here is the R-function for minimizing the multinomial deviance again.

We use the first catgeory (PNN) as baseline and fit the model with the following code:

```
pars0 = matrix(0,12,1)
out.mlr <- optim(pars0, multinomial.deviance, NULL, Y = G, X = X, method="BFGS",control = list(t
```

```
## initial  value 604.424341
## iter  10 value 222.513627
## iter  20 value 192.815978
## iter  30 value 192.635822
## final  value 192.635374
## converged
```

The deviance is 192.6353737 which is lower, but not substantially than the deviance of the ideal point model 210.7922407. In this case the number of parameters of the multinomial logistic regression model is 12, while for the ideal point model it is 11.

```
B = matrix(out.mlr$par,ncol(X),ncol(G)-1)
colnames(B) = c('AVH/PNN', 'PCH/PNN', 'ACH/PNN')
rownames(B) = c('Intercept', colnames(X)[-1])
knitr::kable(B, digits = 2, caption = 'Estimated parameter values')
```

Table 1: Estimated parameter values

| | AVH/PNN | PCH/PNN | ACH/PNN |
|---|---|---|---|
| Intercept | -1.86 | -0.16 | 0.38 |
| AS | -9.50 | -9.68 | -1.99 |
| AL | 13.49 | 9.95 | 2.72 |
| GD | -4.47 | -3.82 | 1.14 |

Using the same procedure as before we can obtain standard errors, $z$-statistics and $p$-values for each of the parameter estimates.

```
SEs = sqrt(diag(solve(out.mlr$hessian)))
zstats = out.mlr$par/SEs
pvals = 1 - pnorm(abs(zstats))
TAB = cbind(out.mlr$par, SEs, zstats, pvals)
colnames(TAB) = c("estimates", "stand.error", "z", "p")
knitr::kable(TAB, digits = 2, caption = 'Estimates and Standard Errors')
```

Table 2: Estimates and Standard Errors

| estimates | stand.error | z | p |
|---|---|---|---|
| -1.86 | 0.58 | -3.22 | 0.00 |
| -9.50 | 1.34 | -7.08 | 0.00 |

| estimates | stand.error | z | p |
|---|---|---|---|
| 13.49 | 1.48 | 9.11 | 0.00 |
| -4.47 | 0.95 | -4.69 | 0.00 |
| -0.16 | 0.40 | -0.39 | 0.35 |
| -9.68 | 1.29 | -7.48 | 0.00 |
| 9.95 | 1.31 | 7.59 | 0.00 |
| -3.82 | 0.80 | -4.74 | 0.00 |
| 0.38 | 0.31 | 1.23 | 0.11 |
| -1.99 | 0.51 | -3.91 | 0.00 |
| 2.72 | 0.50 | 5.43 | 0.00 |
| 1.14 | 0.30 | 3.85 | 0.00 |