MRU: Analyses of Diabetes data

true

May 19, 2022

Diabetes data

See Andrews and Herzberg (1985), Data. Springer. Page 215. Original data from Reaven, G.M. and Miller, R.G. (1979). An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia*, **16**, 17 - 24.

Data are present for 145 subjects. For each participant we have measures on three predictor variables: - Relative weight (RW) - Imsulin Response (IR) - Steady state plasma glucose (SSPG)

The response variable is a clinical diagnosis with three categories: 1 = Overt diabetes, 2 = Chemical diabetes, 3 = Non-diabetic. There are 33 participants in category 1, 36 participants in category 2, and 76 participants in class 3.

See also: https://www.rpubs.com/beane/n4_2

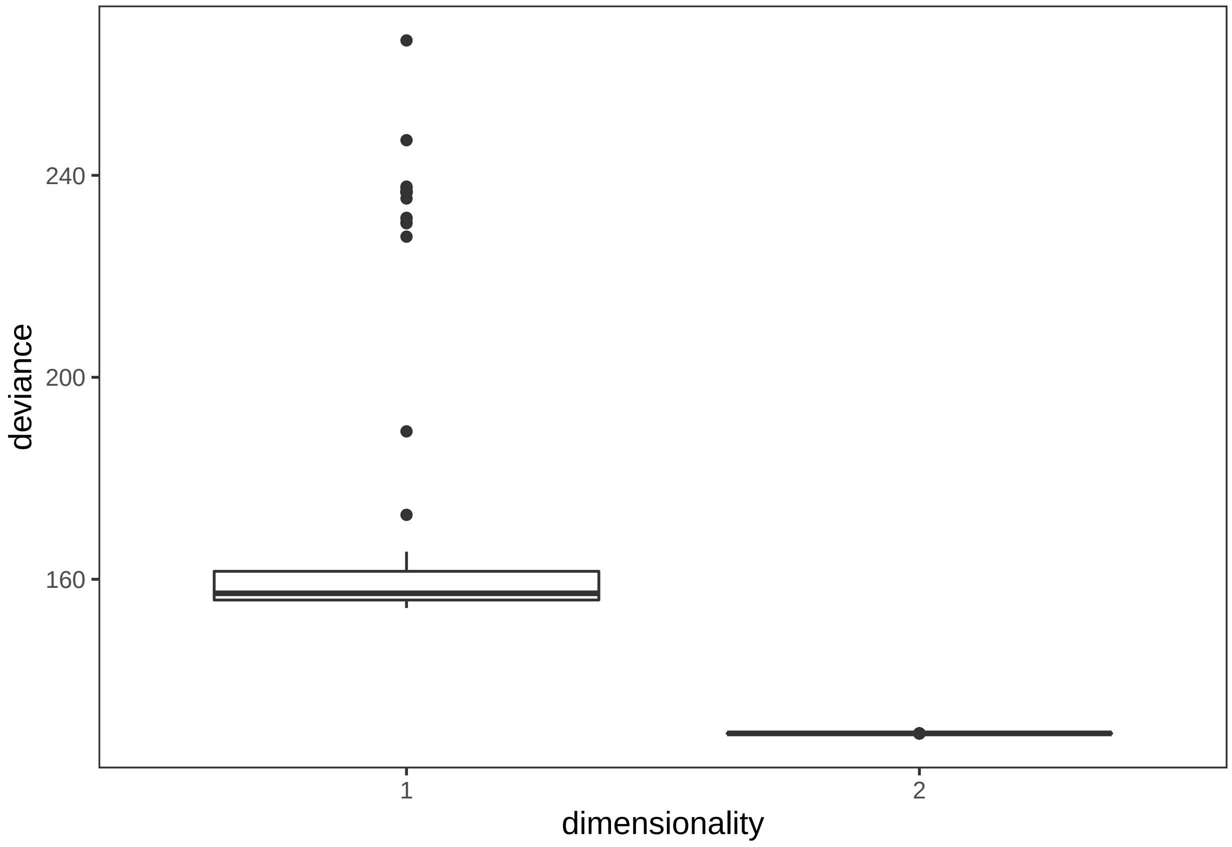
```
library(nnet)
library(ggplot2, quietly = TRUE, warn.conflicts = FALSE)
library(GGally, quietly = TRUE, warn.conflicts = FALSE)

dat <- read.table("~/surfdrive/MultinomialBook/Datasets/diabetesdata/diabetes2.txt", sep=",", he
dat = dat[, c(5, 8, 9, 10)]
X = as.matrix(dat[, c(1,2,3)])
X = scale(X, center = FALSE, scale = FALSE)
colnames(X) = c("RW", "IR", "SSPG")
X.df = data.frame(X)
y = dat[, 4]
G = class.ind(y)
y = factor(y, levels = c("1","2" ,"3"), labels = c("Overt", "Chemical", "Non"))
colnames(G) = c("OD", "CD", "ND")
#ggpairs(X.df, aes(color=factor(y)))
```

The second predictor is very much skewed, therefore, we do a log-transform

```
X[, 2] = log(X[, 2])
X = scale(X)
```

Investigating Local optima

```
set.seed(12345)
M1results = vector(mode = "list", length = 100)
M2results = vector(mode = "list", length = 100)

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

for(r in 2:100){
  M1results[[r]] = mru.random( X, G, m = 1 )
  M2results[[r]] = mru.random( X, G, m = 2 )
}

save(M1results, M2results, file = "diabetesresults.Rdata")

library(ggplot2)
dev1 = rep(NA, 100)
dev2 = rep(NA, 100)
for(r in 1:100){
  dev1[r] = M1results[[r]]$deviance
  dev2[r] = M2results[[r]]$deviance
}

df = data.frame(repl = rep(1:100, 2), dimensionality = as.factor(rep(1:2, each = 100)), deviance

p = ggplot(df, aes(dimensionality, deviance)) + geom_boxplot() + theme_apa()
ggsave("~/surfdrive/multldm/mrmdu/paper/figures/diabetesresults.pdf", plot = p, width = 11.7, he
p
```



Analysis with Distance Model

```
out.diab1 = M1results[[which.min(dev1)]]
out.diab2 = M2results[[which.min(dev2)]]
```

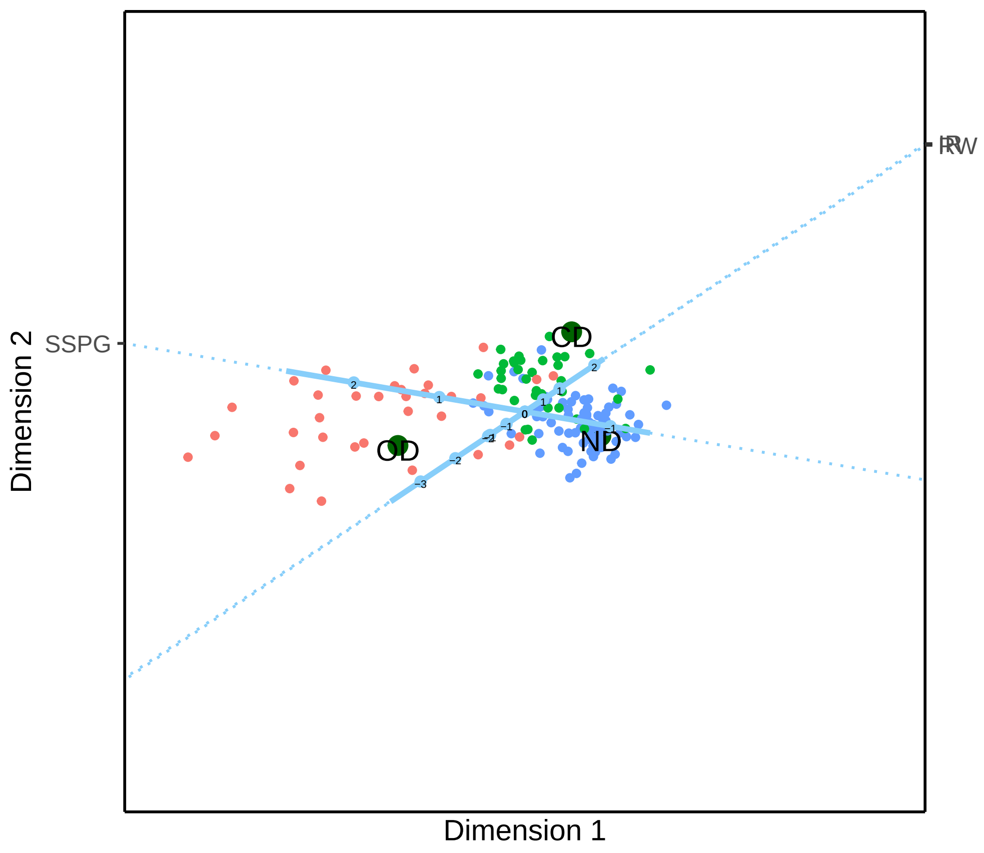
```
NN = as.data.frame(X %*% out.diab2$B)
colnames(NN) = c("Dim1", "Dim2")

VV = as.data.frame(out.diab2$V)
colnames(VV) = c("Dim1", "Dim2")

Xo = X
B = out.diab2$B
P = nrow(B)

# 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


p = 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") +
    coord_fixed()

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

######################################################
# variable axes with ticks and markers for predictors
######################################################
xcol = "lightskyblue"
p = p + 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(p)$layout$panel_scales_x[[1]]$range$range, ggplot_build(p)$la

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)

xnames = colnames(X)
p = p + 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
p = p + 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
p = p + coord_fixed() + theme_bw()
```

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

```
p = p + 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())

p
```

```
## 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/diabetesplot1.pdf", plot = p, width = 11.7, heig
```

```
## 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

```
U = X %*% out.diab2$B
V = out.diab2$V
D = outer(diag(U %*% t(U)), rep(1, 3)) + outer(rep(1,145), diag(V %*% t(V))) - 2* U %*% t(V)
D = sqrt(D)
PR = exp(-D)/rowSums(exp(-D))
yhat = apply(PR, 1, which.max)
yy = apply(G, 1, which.max)
tab1 = table(yy, yhat)
tab1
```

```
##     yhat
## yy    1  2  3
##    1 28  3  2
##    2  0 25 11
##    3  3  4 69
```

In total, 84.137931 percent is correctly classified

Simulation

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

```
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.

```
B = out.diab2$B
V = out.diab2$V
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))
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)
```

```r
    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))

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

  Vslong = data.frame(cbind(rep(1:3, 100), Vslong))
  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)) +
    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)]) +
    scale_fill_manual(values = brewer.pal(8,"Paired")[c(2,4,6)]) +
    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)

  plt = plt + geom_point(data = Vdf, aes(x = x, y = y), colour = brewer.pal(8,"Paired")[c(2,4,6)
    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")

  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,]  0.3856581 0.84368576
## [2,]  0.7775494 1.55098966
## [3,] -2.2398346 0.09986247
##
## [[2]]
##              [,1]      [,2]
## [1,]  0.1776748 0.1928095
## [2,]  0.3358643 0.3842803
## [3,] -1.0605932 0.4553338
##
## [[3]]
##               [,1]       [,2]
## [1,]  0.04841232 0.01796654
## [2,]  0.06345937 0.10010858
## [3,] -0.18950044 0.02702219
##
## [[4]]
##               [,1]       [,2]
## [1,]  0.01377544 0.03675064
## [2,]  0.05996587 0.07336381
## [3,] -0.12396502 0.01431713
```

```
Brmse
```

```
## [[1]]
##          [,1]     [,2]
## [1,] 0.914138 2.110204
## [2,] 1.323116 3.334598
## [3,] 3.204671 4.410441
##
## [[2]]
##           [,1]      [,2]
## [1,] 0.4021271 0.5571703
## [2,] 0.6052836 0.9419243
## [3,] 1.5533346 1.8215015
##
## [[3]]
##           [,1]      [,2]
## [1,] 0.1574449 0.1567797
## [2,] 0.2243578 0.2698306
## [3,] 0.4101972 0.4864332
##
## [[4]]
##           [,1]      [,2]
## [1,] 0.1107905 0.1841503
## [2,] 0.1594131 0.2525501
## [3,] 0.3077955 0.3991201
```

```
Vbias
```

```
## [[1]]
##           [,1]        [,2]
## [1,] -2.948902 -0.06717924
## [2,]  1.092606  1.66043261
## [3,]  1.456375  0.86095827
```

```
##
## [[2]]
##            [,1]       [,2]
## [1,] -1.2018806 -0.3076604
## [2,]  0.5735573  0.5307577
## [3,]  0.7229168  0.0658683
##
## [[3]]
##             [,1]        [,2]
## [1,] -0.35040302 -0.05988244
## [2,]  0.08051139  0.22817991
## [3,]  0.20502234 -0.06546809
##
## [[4]]
##             [,1]        [,2]
## [1,] -0.26593371 -0.03416892
## [2,]  0.06556974  0.18911498
## [3,]  0.17355190 -0.04170152
```

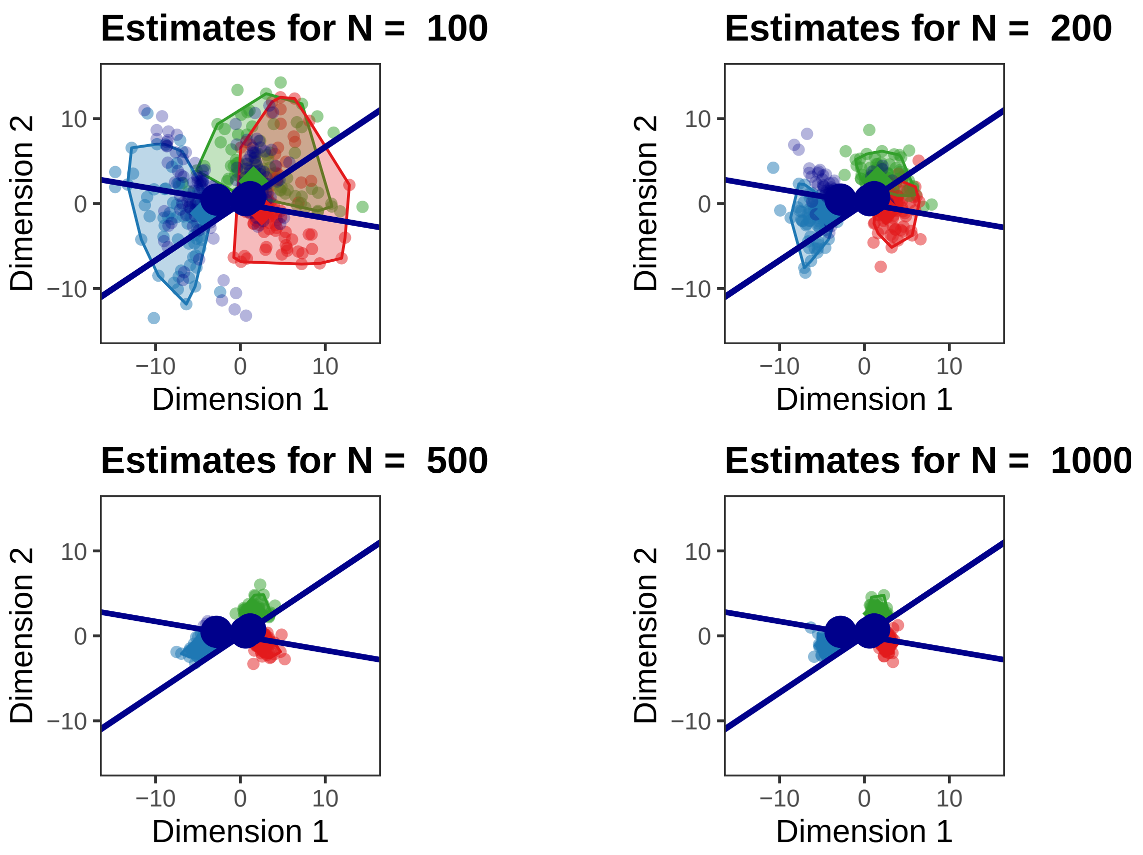
```
Vrmse
```

```
## [[1]]
##          [,1]     [,2]
## [1,] 4.494359 4.542346
## [2,] 3.947443 3.954655
## [3,] 2.924217 4.935194
##
## [[2]]
##          [,1]     [,2]
## [1,] 1.874786 1.987102
## [2,] 1.964721 1.484755
## [3,] 1.477667 1.867667
##
## [[3]]
##           [,1]      [,2]
## [1,] 0.7754410 0.6327525
## [2,] 0.7388117 0.6704182
## [3,] 0.6486471 0.7102151
##
## [[4]]
##           [,1]      [,2]
## [1,] 0.5177829 0.5214450
## [2,] 0.4687015 0.4956927
## [3,] 0.3855481 0.6282798
```

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

```
## Warning: Removed 7 rows containing non-finite values (statbag).
## Warning: Removed 7 rows containing missing values (geom_point).
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
plt
```



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

Squared distance model

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

```
## initial  value 383.918967
## iter  10 value 139.548042
## iter  20 value 132.576683
## final  value 132.576650
## converged
```

```
B = matrix(out$par, 4, 2)
V = matrix(0, 3, 2); V[2, 1] = 1; V[3, 2] = 1
V = V - matrix(1,3,1) %*% B[1, , drop = FALSE]
# rotation to principal axes
U = X[, -1] %*% B[-1, ]
```

```
R = eigen(t(U)%*%U)$vectors
B = B[-1, ] %*% R
V = V %*% R
X = X[, -1]
U = X %*% B
```

The estimated deviance is 132.58.

Let us have a further look at the classification performance

```
D2 = outer(diag(U %*% t(U)), rep(1, 3)) + outer(rep(1,145), diag(V %*% t(V))) - 2* U %*% t(V)
PR2 = exp(-D2)/rowSums(exp(-D2))
yhat2 = apply(PR2, 1, which.max)
yy = apply(G, 1, which.max)
tab2 = table(yy, yhat2)
tab2
```

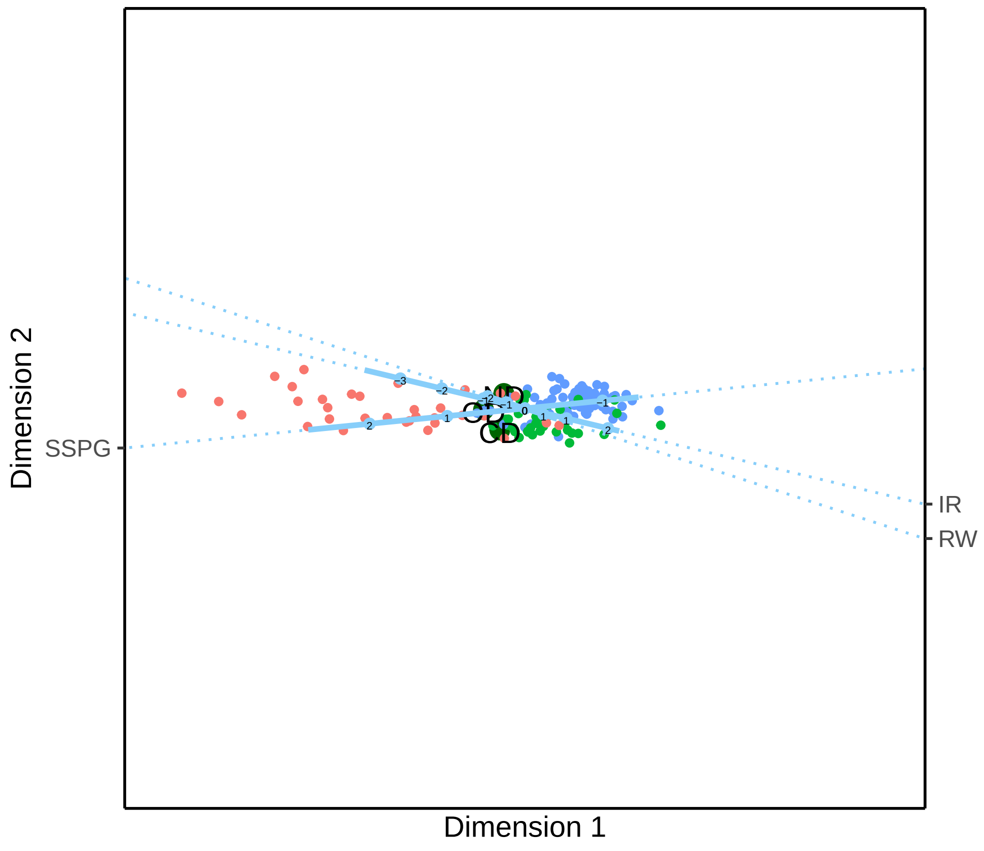
```
##     yhat2
## yy    1  2  3
##    1 28  3  2
##    2  0 24 12
##    3  3  4 69
```

In total, 83.4482759 percent is correctly classified

```
NN = as.data.frame(X %*% B)
colnames(NN) = c("Dim1", "Dim2")

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

P = nrow(B)
Xo = X

# 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)$


P = ncol(X)
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)

xnames = colnames(X)
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() + 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/diabetesplot2.pdf", plot = p2, width = 11.7, hei
```

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

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