Non-parametric density analysis

Andreas Kryger Jensen

14 February, 2018

Define functions

```
rm(list=ls())
library(parallel)
library(pROC)
library(vioplot)
library(fields)

trapz <- function (x, y) {
  n = length(y)
  sum(diff(x) * (y[-n] + y[-1]) / 2)
}

invExpMap <- function(time, mu, s, eps = 1e-10) {
  #Projects the point s to the tangent space at mu
  theta <- acos(trapz(time, mu * s))

  if (theta < eps)
    z <- rep(0, length(s))
  else
    z <- (theta / sin(theta)) * (s - mu * cos(theta))

  z
}

expMap <- function(time, mu, z) {
  #Projects the point z on the tangent space at mu to the sphere
  norm <- sqrt(trapz(time, z * z))
  s <- cos(norm) * mu + sin(norm) * z/norm
  s
}

getRiemannianCenter <- function(time, s, stp = 0.3, iterMax=5000) {
  n = ncol(s)

  #Initialize
  iter <- 1
  vec = matrix(0, nrow(s), n)
  normVec = rep(0, iterMax)

  #First iteration
  mu = rowMeans(s)
  vec <- sapply(1:ncol(s), function(k) invExpMap(time, mu, s[,k]))
  vbar <- rowMeans(vec)
  normVec[iter] <- sqrt(trapz(time, vbar * vbar))

  #Iterate until convergence
  while(iter <= iterMax & normVec[iter] > 1e-08) {
    iter <- iter + 1
    mu <- expMap(time, mu, stp * vbar)
    vec <- sapply(1:ncol(s), function(k) invExpMap(time, mu, s[,k]))
    vbar <- rowMeans(vec)
```

```
    normVec[iter] <- sqrt(trapz(time, vbar * vbar))
  }

  mu
}

normalize <- function(t, d) {
  dF <- approxfun(t, d)
  int <- integrate(dF, min(t), max(t))$value
  d / int
}

getPrincipalDecomposition <- function(time, z, K) {
  decomp <- prcomp(z, center=FALSE)
  psi <- decomp$x
  zeta <- decomp$rotation
  lambda <- decomp$sdev^2
  cumVar <- cumsum(lambda) / sum(lambda)

  list(zeta = zeta[,1:K], lambda = lambda[1:K], psi = psi[,1:K], cumVar = cumVar[1:K])
}

predictDensity <- function(time, zeta, psi, mu) {
  z <- rep(0, length(time))
  for(k in 1:length(zeta)) {
    z <- z + zeta[k] * psi[,k]
  }
  expMap(time, mu, z)^2
}

pmv <- function(time, alpha, mu, lambda, psi) {
  if(alpha == 0) {
    mu^2
  } else {
    expMap(time, mu, alpha * sqrt(lambda) * psi)^2
  }
}
```

Load data

```
load(file="pigData.RData")

timeOriginal <- dat$time
d <- t(dat$d)
meas <- dat$meas
id <- dat$pig
rm(dat1, dat2, dat)

#Get individual modes
modes <- sapply(1:ncol(d), function(q) {
  opt <- optimize(splinefun(timeOriginal, d[,q]), interval=c(-1000, -200), maximum=TRUE)$maximum
})

#Center w.r.t. modes
time <- seq(-200, 600, length.out=500)
dCenter <- sapply(1:ncol(d), function(k) {
  f <- approxfun(timeOriginal - modes[k], d[,k], rule=2) #extrapolate a bit in the tails
```

```
  f(time)
})

time <- seq(0, 1, length.out = length(time))
dCenterNorm <- sapply(1:ncol(dCenter), function(k) normalize(time, dCenter[,k]))
```

Riemannian analysis

```
s <- sqrt(dCenterNorm)

mu <- getRiemannianCenter(time, s)
z <- sapply(1:ncol(s), function(k) invExpMap(time, mu, s[,k]))
mean(rowMeans(z)^2) #check zero mean
```

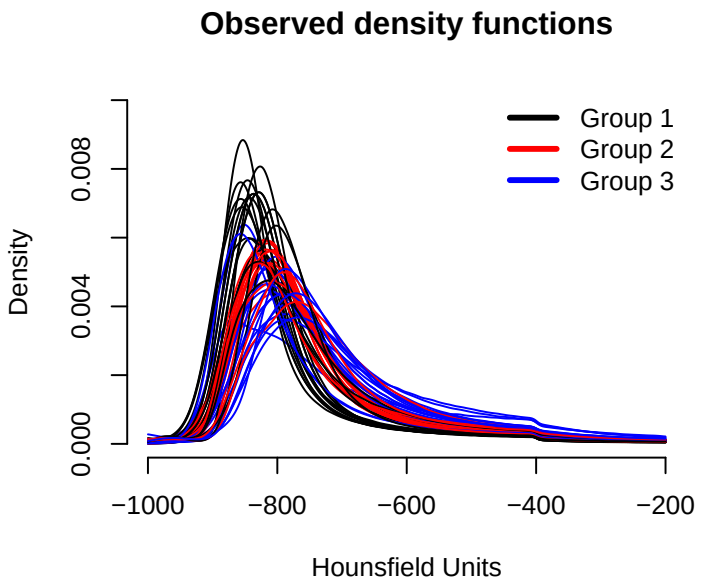
```
## [1] 6.326378e-17
```

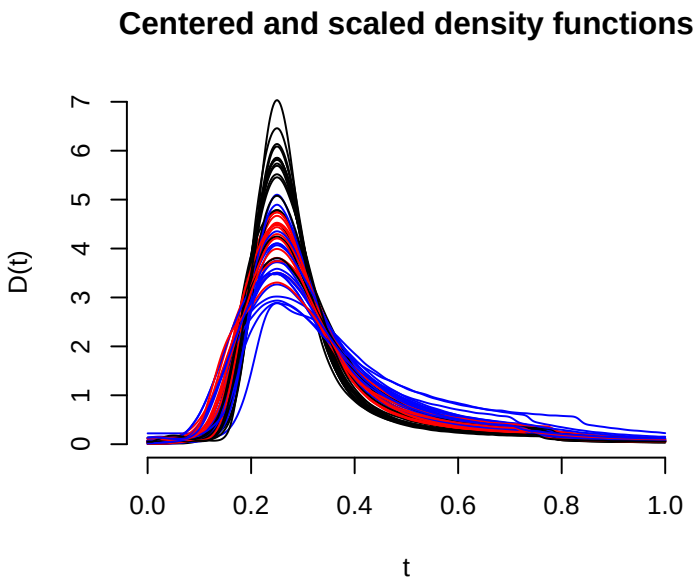
```
K <- 10
pc <- getPrincipalDecomposition(time, z, K)

round(pc$cumVar * 100, 2)
```

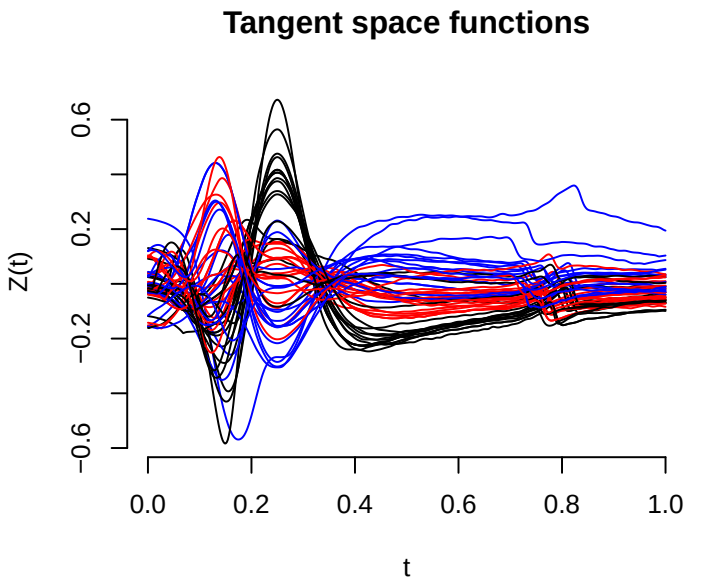
```
##  [1] 63.64 87.50 93.58 96.06 98.01 98.84 99.23 99.41 99.55 99.68
```

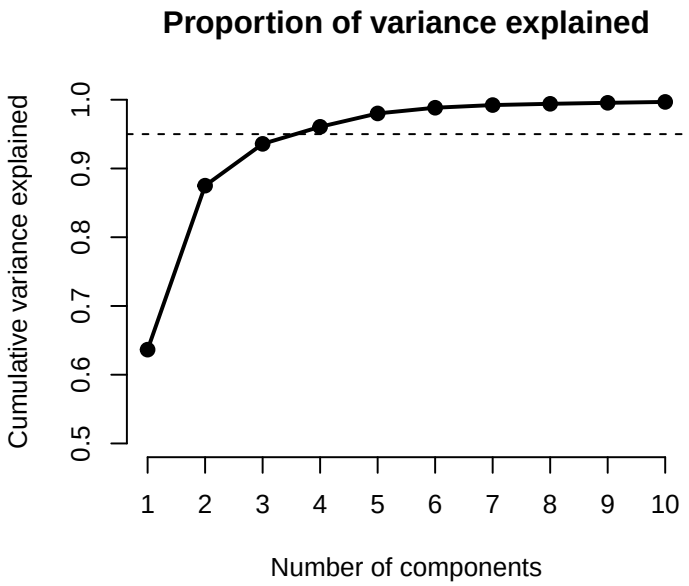
Data plot

```
par(mfrow=c(2,2), bty="n")
matplot(timeOriginal, d[,meas == 1 | meas == 2 | meas == 4], type="l", lty=1, ylim=c(0,0.01),
        xlab="Hounsfield Units", ylab="Density", col=meas[meas == 1 | meas == 2 | meas == 4])
title("Observed density functions")
legend("topright", c("Group 1", "Group 2", "Group 3"), col=c(1,2,4), bty="n", lwd=3)

matplot(time, dCenterNorm[,meas == 1 | meas == 2 | meas == 4], type="l", lty=1, xlab="t",
        ylab="D(t)", col=meas[meas == 1 | meas == 2 | meas == 4])
title("Centered and scaled density functions")

matplot(time, z[,meas == 1 | meas == 2 | meas == 4], type="l", lty=1, xlab="t", ylab="Z(t)",
        col=meas[meas == 1 | meas == 2 | meas == 4])
title("Tangent space functions")

plot(1:10, pc$cumVar, type="o", pch=19, lwd=2, xlab="Number of components",
     ylab="Cumulative variance explained", xaxt="n", ylim=c(0.5, 1))
abline(h=0.95, lty=2)
axis(1, 1:10)
title("Proportion of variance explained")
```





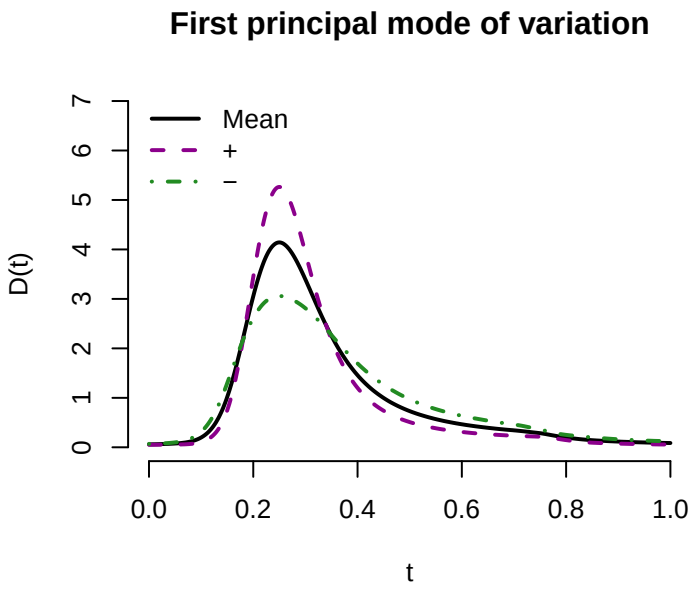


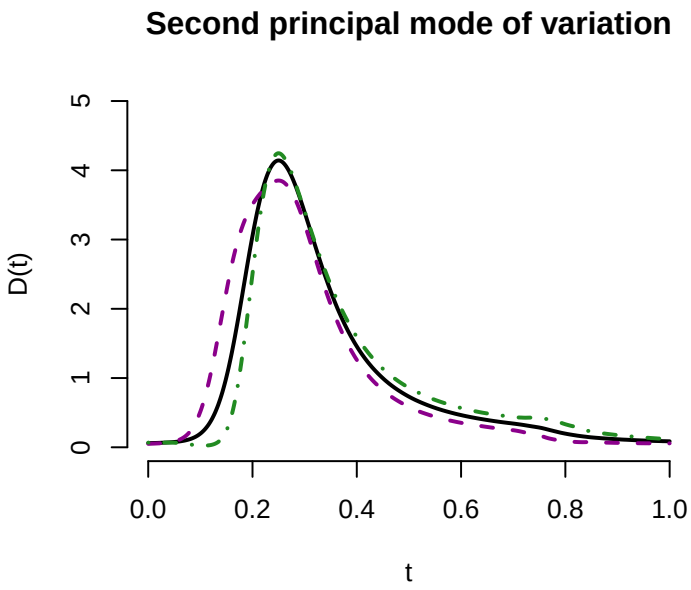


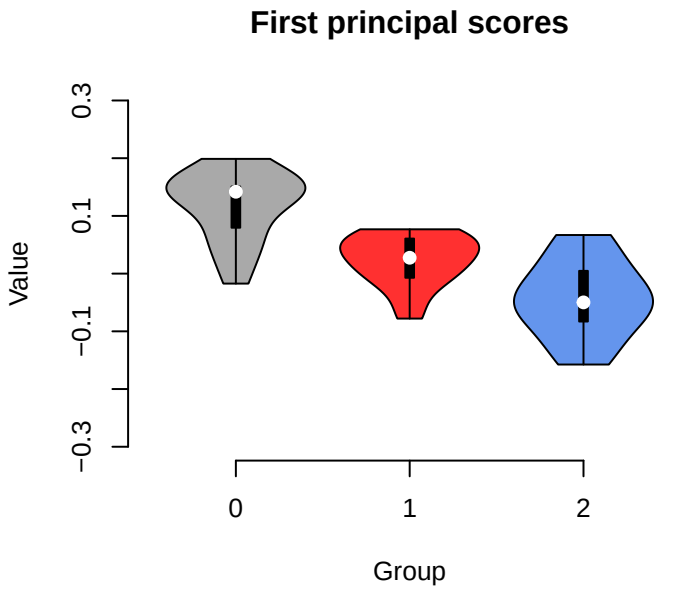
Plot of principal modes

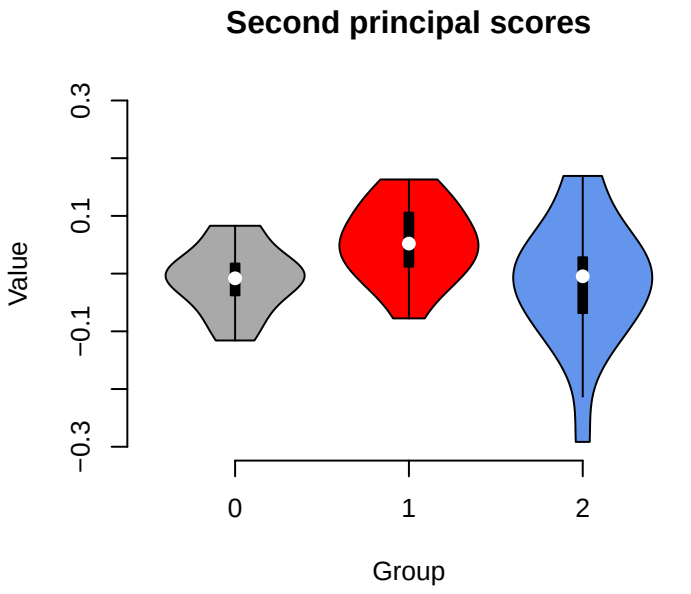
```
par(mfrow=c(2,2), bty="n")
plot(time, pmv(time, 0, mu, pc$lambda[1], pc$psi[,1]), type="l", ylim=c(0,7), lwd=2, xlab="t", ylab="D(t)")
lines(time, pmv(time, 0.1, mu, pc$lambda[1], pc$psi[,1]), type="l", col="darkmagenta", lwd=2, lty=2)
lines(time, pmv(time, -0.1, mu, pc$lambda[1], pc$psi[,1]), type="l", col="forestgreen", lwd=2, lty=4)
title("First principal mode of variation")
legend("topleft", c("Mean", "+", "-"), lwd=2, col=c("black", "darkmagenta", "forestgreen"),
       lty=c(1,2,4), bty="n", seg.len=2)

plot(time, pmv(time, 0, mu, pc$lambda[2], pc$psi[,2]), type="l", ylim=c(0,5), lwd=2, xlab="t", ylab="D(t)")
lines(time, pmv(time, 0.35, mu, pc$lambda[2], pc$psi[,2]), type="l", col="darkmagenta", lwd=2, lty=2)
lines(time, pmv(time, -0.35, mu, pc$lambda[2], pc$psi[,2]), type="l", col="forestgreen", lwd=2, lty=4)
title("Second principal mode of variation")

plot(0, 0, type="n", xlim=c(-0.5,2.5), ylim=c(-0.3,0.3), bty="n", xaxt="n",
     xlab="Group", ylab="Value")
axis(1, at=0:2)
```

```
vioplot(pc$zeta[meas == 1, 1], col="darkgray", at = 0, add=TRUE)
vioplot(pc$zeta[meas == 2, 1], col="firebrick1", at = 1, add=TRUE)
vioplot(pc$zeta[meas == 4, 1], col="cornflowerblue", at = 2, add=TRUE)
title("First principal scores")

plot(0, 0, type="n", xlim=c(-0.5,2.5), ylim=c(-0.3,0.3), bty="n", xaxt="n",
     xlab="Group", ylab="Value")
axis(1, at=0:2)
vioplot(pc$zeta[meas == 1, 2], col="darkgray", at = 0, add=TRUE)
vioplot(pc$zeta[meas == 2, 2], col="red", at = 1, add=TRUE)
vioplot(pc$zeta[meas == 4, 2], col="cornflowerblue", at = 2, add=TRUE)
title("Second principal scores")
```









Covariate model

```
zeta <- pc$zeta[meas == 1 | meas == 2,]
y <- meas[meas == 1 | meas == 2] - 1
```

```r
pigID <- id[meas == 1 | meas == 2]

logitDat <- data.frame(id = pigID, y = y, mode = modes[meas == 1 | meas == 2],
                       z1 = zeta[,1], z2 = zeta[,2], z3 = zeta[,3])

doLOO <- function(modelFormula) {
  looOut <- unlist(lapply(1:9, function(k) {
    dTrain <- subset(logitDat, id != k)
    dValid <- subset(logitDat, id == k)

    m <- glm(modelFormula, family=binomial(), data=dTrain)
    predict(m, newdata=dValid, type="response")
  }))
  auc <- auc(logitDat$y, looOut)
  predClass <- as.numeric(looOut > 0.5)
  confusion <- table(true = logitDat$y, pred = predClass)
  acc <- mean(predClass == logitDat$y)
  brier <- mean((predClass - logitDat$y)^2)
  list(confusion=confusion, acc=acc, auc = as.numeric(auc), brier = brier)
}

doLOO(y ~ mode)
```

```
## $confusion
##     pred
## true  0  1
##    0 15  3
##    1  2 16
##
## $acc
## [1] 0.8611111
##
## $auc
## [1] 0.845679
##
## $brier
## [1] 0.1388889
```

```r
doLOO(y ~ z1)
```

```
## $confusion
##     pred
## true  0  1
##    0 14  4
##    1  2 16
##
## $acc
## [1] 0.8333333
##
## $auc
## [1] 0.8395062
##
## $brier
## [1] 0.1666667
```

```r
doLOO(y ~ z2)
```

```
## $confusion
##     pred
## true  0  1
##    0 13  5
```

```
##    1  5 13
##
## $acc
## [1] 0.7222222
##
## $auc
## [1] 0.7438272
##
## $brier
## [1] 0.2777778
```

```
doLOO(y ~ z3)
```

```
## $confusion
##     pred
## true  0  1
##    0 13  5
##    1  5 13
##
## $acc
## [1] 0.7222222
##
## $auc
## [1] 0.7407407
##
## $brier
## [1] 0.2777778
```

```
doLOO(y ~ mode + z1)
```

```
## $confusion
##     pred
## true  0  1
##    0 13  5
##    1  2 16
##
## $acc
## [1] 0.8055556
##
## $auc
## [1] 0.8734568
##
## $brier
## [1] 0.1944444
```

```
doLOO(y ~ mode + z2)
```

```
## $confusion
##     pred
## true  0  1
##    0 13  5
##    1  2 16
##
## $acc
## [1] 0.8055556
##
## $auc
## [1] 0.8271605
##
## $brier
## [1] 0.1944444
```

```
doLOO(y ~ z1 + z2)
```

```
## $confusion
##     pred
## true  0  1
##    0 15  3
##    1  0 18
##
## $acc
## [1] 0.9166667
##
## $auc
## [1] 0.9197531
##
## $brier
## [1] 0.08333333
```

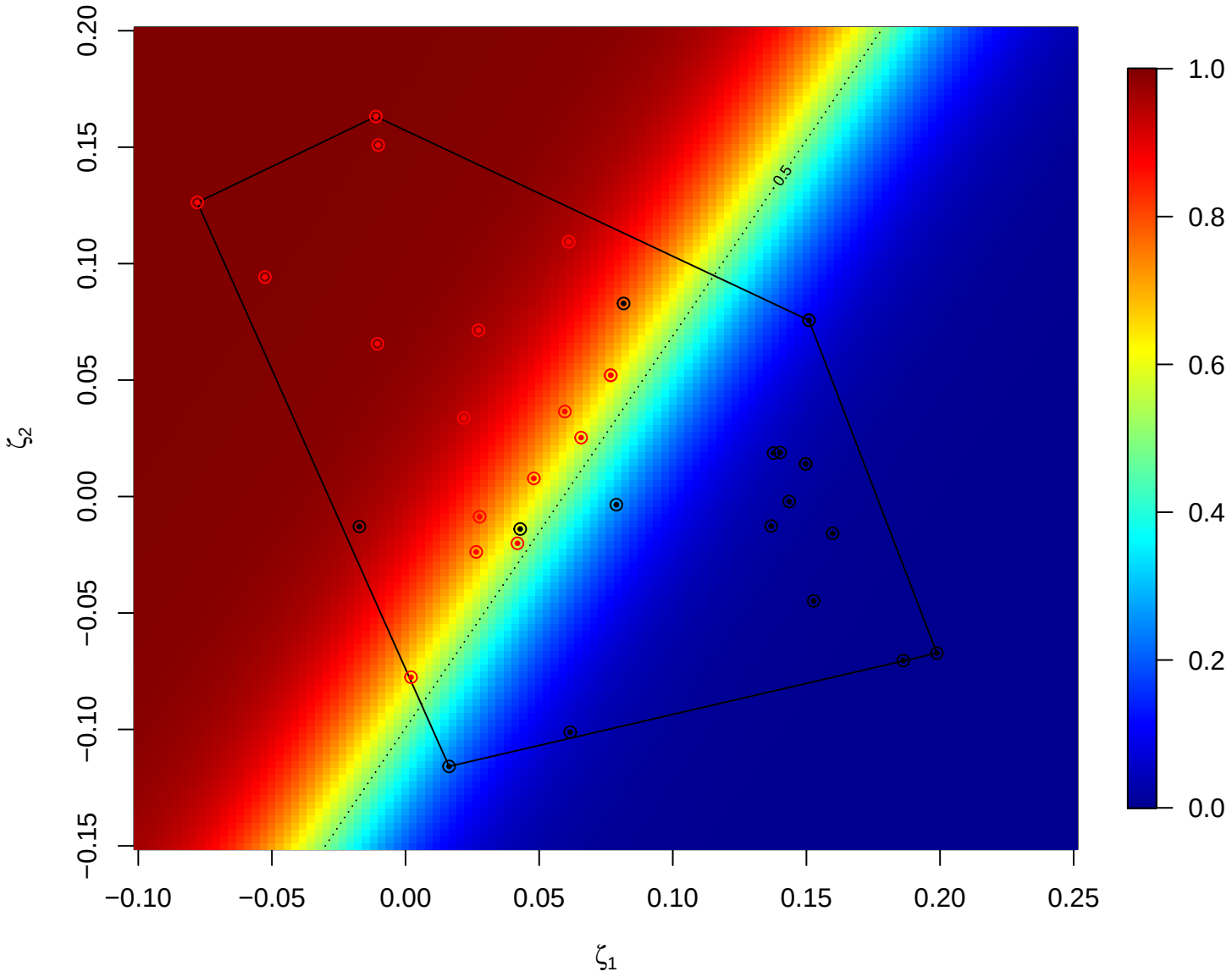
```
doLOO(y ~ mode + z1 + z2)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## $confusion
##     pred
## true  0  1
##    0 15  3
##    1  2 16
##
## $acc
## [1] 0.8611111
##
## $auc
## [1] 0.8518519
##
## $brier
## [1] 0.1388889
```

```
m <- glm(y ~ z1 + z2, family=binomial(), data=logitDat)
zSeq1 <- seq(-0.1, 0.25, length.out=120)
zSeq2 <- seq(-0.15, 0.2, length.out=120)
probMat <- outer(zSeq1, zSeq2, function(x, y) {
  predict(m, data.frame(z1 = x, z2 = y), type="response")
})

image.plot(zSeq1, zSeq2, probMat, nlevel=1024,
           xlab=expression(zeta[1]), ylab=expression(zeta[2]),
           useRaster = TRUE)
conhul <- chull(zeta[,1:2])
lines(zeta[c(conhul, conhul[1]), 1:2], lwd=1)
points(zeta[,1], zeta[,2], cex=0.5, col=y+1, pch=20)
points(zeta[,1], zeta[,2], cex=1, col=y+1)
contour(zSeq1, zSeq2, probMat, levels=0.5, add=TRUE, lty=3)
```



Response model

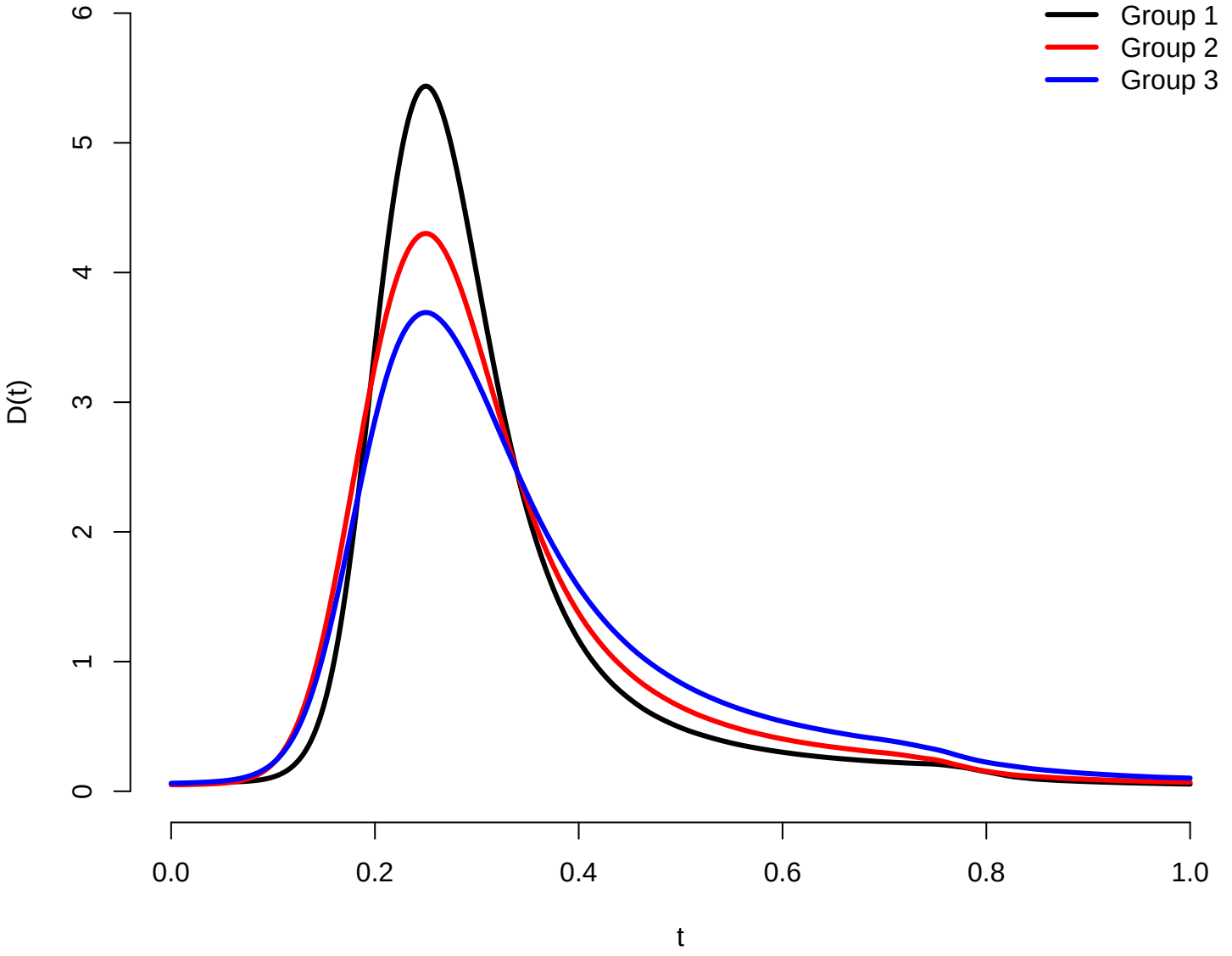
```
zeta <- cbind(modes[meas == 1 | meas == 2 | meas == 4], pc$zeta[meas == 1 | meas == 2 | meas == 4,])
zeta <- zeta[,1:4]
x <- meas[meas == 1 | meas == 2 | meas == 4]


m <- lm(zeta[,-1] ~ factor(x) - 1)
m0 <- lm(zeta[,-1] ~ 1)
anova(m, m0)
```

```
## Analysis of Variance Table
##
## Model 1: zeta[, -1] ~ factor(x) - 1
## Model 2: zeta[, -1] ~ 1
##   Res.Df Df  Gen.var.  Pillai approx F num Df den Df   Pr(>F)
## 1     51    0.0045648
## 2     53  2 0.0062805 0.74849   9.9679      6    100 1.31e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
coef(m)
```

```
##                     PC1          PC2         PC3
## factor(x)1  0.11785416 -0.008929312  0.03634587
## factor(x)2  0.02410281  0.053752786 -0.02462702
## factor(x)4 -0.04349340 -0.013190114 -0.01101901
```

```r
m <- lm(zeta[,-1] ~ factor(x))
plot(time, predictDensity(time, predict(m, data.frame(x = "1")), pc$psi, mu),
     type="l", lwd=3, bty="n", xlab="t", ylab="D(t)", ylim=c(0,6))
lines(time, predictDensity(time, predict(m, data.frame(x = "2")), pc$psi, mu),
      type="l", col=2, lwd=3)
lines(time, predictDensity(time, predict(m, data.frame(x = "4")), pc$psi, mu),
      type="l", col=4, lwd=3)
legend("topright", c("Group 1", "Group 2", "Group 3"), col=c(1,2,4), bty="n", lwd=3)
```




```r
summary(lm(zeta[,-1] ~ factor(x)))
```

```
## Response PC1 :
##
## Call:
## lm(formula = PC1 ~ factor(x))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.135119 -0.035069  0.009359  0.035442  0.110374
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.11785    0.01345   8.761 9.60e-12 ***
## factor(x)2  -0.09375    0.01902  -4.928 9.18e-06 ***
## factor(x)4  -0.16135    0.01902  -8.481 2.59e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05707 on 51 degrees of freedom
## Multiple R-squared:  0.5872, Adjusted R-squared:  0.571
## F-statistic: 36.28 on 2 and 51 DF,  p-value: 1.587e-10
##
##
## Response PC2 :
##
## Call:
## lm(formula = PC2 ~ factor(x))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27838 -0.04709  0.00136  0.03992  0.18247
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.008929   0.018755  -0.476    0.636
## factor(x)2   0.062682   0.026524   2.363    0.022 *
## factor(x)4  -0.004261   0.026524  -0.161    0.873
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07957 on 51 degrees of freedom
## Multiple R-squared:  0.1354, Adjusted R-squared:  0.1015
## F-statistic: 3.994 on 2 and 51 DF,  p-value: 0.02448
##
##
## Response PC3 :
##
## Call:
## lm(formula = PC3 ~ factor(x))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.149375 -0.053935 -0.007099  0.032054  0.233456
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03635    0.01637   2.220   0.0309 *
## factor(x)2  -0.06097    0.02315  -2.633   0.0112 *
## factor(x)4  -0.04736    0.02315  -2.046   0.0460 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06946 on 51 degrees of freedom
## Multiple R-squared:  0.1303, Adjusted R-squared:  0.09624
## F-statistic: 3.822 on 2 and 51 DF,  p-value: 0.0284
```

```
summary(lm(zeta[,-1] ~ relevel(factor(x), "2")))
```

```
## Response PC1 :
```

```
##
## Call:
## lm(formula = PC1 ~ relevel(factor(x), "2"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.135119 -0.035069  0.009359  0.035442  0.110374
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                0.02410    0.01345   1.792 0.079115 .
## relevel(factor(x), "2")1   0.09375    0.01902   4.928 9.18e-06 ***
## relevel(factor(x), "2")4  -0.06760    0.01902  -3.553 0.000831 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05707 on 51 degrees of freedom
## Multiple R-squared:  0.5872, Adjusted R-squared:  0.571
## F-statistic: 36.28 on 2 and 51 DF,  p-value: 1.587e-10
##
##
## Response PC2 :
##
## Call:
## lm(formula = PC2 ~ relevel(factor(x), "2"))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27838 -0.04709  0.00136  0.03992  0.18247
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                0.05375    0.01876   2.866  0.00603 **
## relevel(factor(x), "2")1  -0.06268    0.02652  -2.363  0.02197 *
## relevel(factor(x), "2")4  -0.06694    0.02652  -2.524  0.01476 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07957 on 51 degrees of freedom
## Multiple R-squared:  0.1354, Adjusted R-squared:  0.1015
## F-statistic: 3.994 on 2 and 51 DF,  p-value: 0.02448
##
##
## Response PC3 :
##
## Call:
## lm(formula = PC3 ~ relevel(factor(x), "2"))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.149375 -0.053935 -0.007099  0.032054  0.233456
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -0.02463    0.01637  -1.504   0.1387
## relevel(factor(x), "2")1   0.06097    0.02315   2.633   0.0112 *
## relevel(factor(x), "2")4   0.01361    0.02315   0.588   0.5593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.06946 on 51 degrees of freedom
## Multiple R-squared:  0.1303, Adjusted R-squared:  0.09624
## F-statistic: 3.822 on 2 and 51 DF,  p-value: 0.0284
```

```
save.image("nonParametricAnalysis.RData")
```