```
load("diabetes.RData")

# 0) Preparing Datasets ======================================
diabetes1 <- as.data.frame(scale(diabetes))
diabetes2 <- diabetes
diabetes2[1,3] <- 8.1
diabetes2 <- as.data.frame(scale(diabetes2))

X <- diabetes1[,1:2]
Y <- diabetes1[,3:5]

X2 <- diabetes2[,1:2]
Y2 <- diabetes2[,3:5]

# a) Using function cancor ===================================
my.cancor <- cancor(X, Y)


# b) plotting canonical variables ============================
plot((as.matrix(X) %*% my.cancor$xcoef)[,1], (as.matrix(Y) %*% my.cancor$ycoef)[,1])
plot((as.matrix(X) %*% my.cancor$xcoef)[,2], (as.matrix(Y) %*% my.cancor$ycoef)[,2])


# c) Using function CCAgrid ==================================

# methodological difference: uses projection pursuit to calculate cca, method gives the
method to calculate the correlation

#install.packages("ccaPP")
require(ccaPP)

cancor.pearson  <- CCAgrid(X,Y, k=2, method="pearson")
cancor.spearman <- CCAgrid(X,Y, k=2, method="spearman")

par(mfrow=c(3,2))
plot((as.matrix(X) %*% my.cancor$xcoef)[,1], (as.matrix(Y) %*% my.cancor$ycoef)[,1],
xlab=expression(phi[1]), ylab=expression(eta[1]))
title("cancor 1")
plot((as.matrix(X) %*% my.cancor$xcoef)[,2], (as.matrix(Y) %*% my.cancor$ycoef)[,2],
xlab=expression(phi[2]), ylab=expression(eta[2]))
title("cancor 2")

plot((as.matrix(X) %*% cancor.pearson$A)[,1], (as.matrix(Y) %*% cancor.pearson$B)[,1],
xlab=expression(phi[1]), ylab=expression(eta[1]))
title("pearson 1")
plot((as.matrix(X) %*% cancor.pearson$A)[,2], (as.matrix(Y) %*% cancor.pearson$B)[,2],
xlab=expression(phi[2]), ylab=expression(eta[2]))
title("pearson 2")

plot((as.matrix(X) %*% cancor.spearman$A)[,1], (as.matrix(Y) %*% cancor.spearman$B)[,1],
xlab=expression(phi[1]), ylab=expression(eta[1]))
title("spearman 1")
plot((as.matrix(X) %*% cancor.spearman$A)[,2], (as.matrix(Y) %*% cancor.spearman$B)[,2],
xlab=expression(phi[2]), ylab=expression(eta[2]))
title("spearman 2")

# d) replacing first value with 8.1 ==========================
dev.off()
```

```
my.cancor2 <- cancor(X2, Y2)

cancor.pearson2  <- CCAgrid(X2,Y2, k=2, method="pearson")
cancor.spearman2 <- CCAgrid(X2,Y2, k=2, method="spearman")

par(mfrow=c(3,2))
plot((as.matrix(X2) %*% my.cancor2$xcoef)[,1], (as.matrix(Y2) %*% my.cancor2$ycoef)[,1],
xlab=expression(phi[1]), ylab=expression(eta[1]))
title("cancor 1")
plot((as.matrix(X2) %*% my.cancor2$xcoef)[,2], (as.matrix(Y2) %*% my.cancor2$ycoef)[,2],
xlab=expression(phi[2]), ylab=expression(eta[2]))
title("cancor 2")

plot((as.matrix(X2) %*% cancor.pearson2$A)[,1], (as.matrix(Y2) %*%
cancor.pearson2$B)[,1], xlab=expression(phi[1]), ylab=expression(eta[1]))
title("pearson 1")
plot((as.matrix(X2) %*% cancor.pearson2$A)[,2], (as.matrix(Y2) %*%
cancor.pearson2$B)[,2], xlab=expression(phi[2]), ylab=expression(eta[2]))
title("pearson 2")

plot((as.matrix(X2) %*% cancor.spearman2$A)[,1], (as.matrix(Y2) %*%
cancor.spearman2$B)[,1], xlab=expression(phi[1]), ylab=expression(eta[1]))
title("spearman 1")
plot((as.matrix(X2) %*% cancor.spearman2$A)[,2], (as.matrix(Y2) %*%
cancor.spearman2$B)[,2], xlab=expression(phi[2]), ylab=expression(eta[2]))
title("spearman 2")

# e) permutation test ========================================

perm.test <- function(X,Y,cancor.fun, R){
  # calculating "real" canonical correlation coefficient
  my.coef <- cancor.fun(X,Y)$cor[1]
  # initialising counter variable
  counter <- c()
  for(i in 1:R){
    # permutating sample
    Y.temp <- Y[sample(dim(Y)[1]),]
    # adding to counter the boolean value if the canonical correlation coefficient of the
permuted sample is greater that the original one
    counter <- c(counter, (cancor.fun(X,Y.temp)$cor[1] > my.coef))
  }
  # the p value is the proportion of true values
  p.value <- as.numeric(prop.table(table(counter))["TRUE"])
  return(p.value)
}

# running this with 1000 samples, 100 gave almost no results
perm.test(X,Y, function(...){CCAgrid(..., k=2, method="pearson")}, 1000) # 0.001
perm.test(X,Y, function(...){CCAgrid(..., k=2, method="spearman")}, 1000) # 0
perm.test(X2,Y2, function(...){CCAgrid(..., k=2, method="pearson")}, 1000) # 0.34
perm.test(X2,Y2, function(...){CCAgrid(..., k=2, method="spearman")}, 1000) # 0.001
# spearman performs better with outliers
```