Paper Abstract 11SSS

Joao Pinelo Silva

September 12, 2016

(MODEL: 1\_PLSDA)

# Loading and Preparing Data

input\_med <- read.csv(file = "med86220\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_sol <- read.csv(file = "sol69102\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_hair <- read.csv(file = "hair96020\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_pedu <- read.csv(file = "pedu85200\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_pubs <- read.csv(file = "pubs56302\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_trav <- read.csv(file = "trav79110\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_hosp <- read.csv(file = "hosp86101\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_cafes <- read.csv(file = "cafes56102\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_starbucks <- read.csv(file = "starbucks\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
input\_waterstones <- read.csv(file = "waterstones\_output\_table.txt",  
 header = TRUE, sep = ",", dec = ".")  
  
# Table to select segments without any events to add to each of the landuse coded as 0  
sg <- read.table(file = "m25seg\_40stbs\_800\_1000\_1600\_3200\_n.txt",  
 header = TRUE, sep = "\t")  
used\_seg <- unique(c(input\_cafes$Depthmap\_R, input\_hair$Depthmap\_R,  
 input\_hosp$Depthmap\_R, input\_med$Depthmap\_R,  
 input\_pedu$Depthmap\_R, input\_pubs$Depthmap\_R,  
 input\_sol$Depthmap\_R, input\_starbucks$Depthmap\_R,  
 input\_trav$Depthmap\_R, input\_waterstones$Depthmap\_R))  
seg\_no\_land\_use <- sg[!(sg$Ref %in% used\_seg), ]  
seg\_no\_land\_use <- seg\_no\_land\_use[ , -c(1:8,18:27)]  
names(seg\_no\_land\_use) <- c("ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
seg\_no\_land\_use$activity <- 0  
rc <- seg\_no\_land\_use

# I/O Ratio

# Number of segments without activity to add to each group this is to be  
# multiplied by the number of segment with activity  
zeros <- 1

# Individual Activity Map Segments

cafes <- input\_cafes[input\_cafes$Depthmap\_R != 0, ] # remove events with no segment  
cafes <- cafes[,-c(1:4,6:10,20:30)]  
names(cafes) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
cafes <- rbind(cafes, rc[sample(nrow(rc), nrow(cafes)/zeros), ] )  
cafes$activity <- as.factor(cafes$activity)  
levels(cafes$activity) <- c("N","Y")  
cafes <- cafes[ ,-7]  
str(cafes)

## 'data.frame': 3338 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 335752 15741061 1752635 9445724 335752 ...  
## $ ch1600 : num 3553 30968 7897 19105 3553 ...  
## $ ch3200 : num 12970 123991 21925 90253 12970 ...  
## $ ch800 : num 517 8444 1069 2558 517 ...  
## $ int\_n : num 36466 27254 34831 33369 36466 ...  
## $ int1600 : num 363 331 394 318 363 ...  
## $ int3200 : num 1055 922 1008 939 1055 ...  
## $ int800 : num 117 141 151 105 117 ...

hair <- input\_hair[input\_hair$Depthmap\_R != 0, ] # remove events with no segment  
hair <- hair[,-c(1:4,6:10,20:30)]  
names(hair) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
hair <- rbind(hair, rc[sample(nrow(rc), nrow(hair)/zeros), ] )  
hair$activity <- as.factor(hair$activity)  
levels(hair$activity) <- c("N","Y")  
hair <- hair[ ,-7]  
str(hair)

## 'data.frame': 4066 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 955546 24571548 986487 27760156 1845094 ...  
## $ ch1600 : num 2077 15760 35430 108933 15588 ...  
## $ ch3200 : num 5476 66982 83555 888752 77723 ...  
## $ ch800 : num 508 2439 9210 8830 4191 ...  
## $ int\_n : num 29041 34218 42944 41848 43409 ...  
## $ int1600 : num 301 377 898 1000 1105 ...  
## $ int3200 : num 699 1020 2955 2971 4348 ...  
## $ int800 : num 110 140 277 328 357 ...

hosp <- input\_hosp[input\_hosp$Depthmap\_R != 0, ] # remove events with no segment  
hosp <- hosp[,-c(1:4,6:10,20:30)]  
names(hosp) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
hosp <- rbind(hosp, rc[sample(nrow(rc), nrow(hosp)/zeros), ] )  
hosp$activity <- as.factor(hosp$activity)  
levels(hosp$activity) <- c("N","Y")  
hosp <- hosp[ ,-7]  
str(hosp)

## 'data.frame': 2416 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 5233330 24571548 2078347 6061303 45947880 ...  
## $ ch1600 : num 30442 15760 7480 67397 116494 ...  
## $ ch3200 : num 257905 66982 34167 417995 677763 ...  
## $ ch800 : num 3656 2439 2049 4840 10491 ...  
## $ int\_n : num 43471 34218 34981 49667 49667 ...  
## $ int1600 : num 1053 377 362 1293 1352 ...  
## $ int3200 : num 3260 1020 1118 4042 4153 ...  
## $ int800 : num 277 140 152 309 367 ...

med <- input\_med[input\_med$Depthmap\_R != 0, ] # remove events with no segment  
med <- med[,-c(1:4,6:10,20:30)]  
names(med) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
med <- rbind(med, rc[sample(nrow(rc), nrow(med)/zeros), ] )  
med$activity <- as.factor(med$activity)  
levels(med$activity) <- c("N","Y")  
med <- med[ ,-7]  
str(med)

## 'data.frame': 1906 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 0 3133269 0 0 0 ...  
## $ ch1600 : num 0 37835 0 0 0 ...  
## $ ch3200 : num 0 159030 0 0 0 ...  
## $ ch800 : num 0 7386 0 0 0 ...  
## $ int\_n : num 34758 46636 36726 33124 49194 ...  
## $ int1600 : num 324 1598 488 331 952 ...  
## $ int3200 : num 926 5185 1191 1037 3463 ...  
## $ int800 : num 117 458 202 103 282 ...

pedu <- input\_pedu[input\_pedu$Depthmap\_R != 0, ] # remove events with no segment  
pedu <- pedu[,-c(1:4,6:10,20:30)]  
names(pedu) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
pedu <- rbind(pedu, rc[sample(nrow(rc), nrow(pedu)/zeros), ] )  
pedu$activity <- as.factor(pedu$activity)  
levels(pedu$activity) <- c("N","Y")  
pedu <- pedu[ ,-7]  
str(pedu)

## 'data.frame': 784 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 0 42977328 16626824 4651797 31690400 ...  
## $ ch1600 : num 0 34659 22755 9695 108276 ...  
## $ ch3200 : num 0 251701 135351 36207 730172 ...  
## $ ch800 : num 0 4288 2689 2030 6481 ...  
## $ int\_n : num 30000 33351 34789 37150 43682 ...  
## $ int1600 : num 349 350 338 367 532 ...  
## $ int3200 : num 678 1081 1092 1110 1927 ...  
## $ int800 : num 163 119 113 102 241 ...

pubs <- input\_pubs[input\_pubs$Depthmap\_R != 0, ] # remove events with no segment  
pubs <- pubs[,-c(1:4,6:10,20:30)]  
names(pubs) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
pubs <- rbind(pubs, rc[sample(nrow(rc), nrow(pubs)/zeros), ] )  
pubs$activity <- as.factor(pubs$activity)  
levels(pubs$activity) <- c("N","Y")  
pubs <- pubs[ ,-7]  
str(pubs)

## 'data.frame': 1946 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 949987 0 9937249 926069 11714423 ...  
## $ ch1600 : num 2976 0 46698 6050 126552 ...  
## $ ch3200 : num 11257 0 253588 33240 542202 ...  
## $ ch800 : num 356 0 6529 1207 25905 ...  
## $ int\_n : num 38572 45359 32796 45276 43613 ...  
## $ int1600 : num 413 1226 456 1233 1089 ...  
## $ int3200 : num 1168 4366 1152 4832 3652 ...  
## $ int800 : num 91.6 415.5 167.4 342.1 380.9 ...

sol <- input\_sol[input\_sol$Depthmap\_R != 0, ] # remove events with no segment  
sol <- sol[,-c(1:4,6:10,20:30)]  
names(sol) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
sol <- rbind(sol, rc[sample(nrow(rc), nrow(sol)/zeros), ] )  
sol$activity <- as.factor(sol$activity)  
levels(sol$activity) <- c("N","Y")  
sol <- sol[ ,-7]  
str(sol)

## 'data.frame': 2178 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 2.08e+08 0.00 0.00 9.59e+05 0.00 ...  
## $ ch1600 : num 101710 0 0 19845 0 ...  
## $ ch3200 : num 490869 0 0 38418 0 ...  
## $ ch800 : num 15040 0 0 5497 0 ...  
## $ int\_n : num 36850 32711 41778 45400 35420 ...  
## $ int1600 : num 505 205 931 1177 311 ...  
## $ int3200 : num 968 718 3400 3227 1126 ...  
## $ int800 : num 220.8 76.1 293.3 328.5 92.7 ...

starbucks <- input\_starbucks[input\_starbucks$Depthmap\_R != 0, ] # remove events with no segment  
starbucks <- starbucks[,-c(1:4,6:10,20:30)]  
names(starbucks) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
starbucks <- rbind(starbucks, rc[sample(nrow(rc), nrow(starbucks)/zeros), ] )  
starbucks$activity <- as.factor(starbucks$activity)  
levels(starbucks$activity) <- c("N","Y")  
starbucks <- starbucks[ ,-7]  
str(starbucks)

## 'data.frame': 314 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 3.21e+06 4.60e+06 2.25e+07 7.28e+08 0.00 ...  
## $ ch1600 : num 24357 26236 30536 47731 0 ...  
## $ ch3200 : num 115013 122098 225239 378149 0 ...  
## $ ch800 : num 3399 5626 1776 7463 0 ...  
## $ int\_n : num 45402 35272 32800 31231 45695 ...  
## $ int1600 : num 1324 655 387 300 1623 ...  
## $ int3200 : num 4546 1879 938 682 5250 ...  
## $ int800 : num 286.3 246 92.7 163.3 437.1 ...

trav <- input\_trav[input\_trav$Depthmap\_R != 0, ] # remove events with no segment  
trav <- trav[,-c(1:4,6:10,20:30)]  
names(trav) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
trav <- rbind(trav, rc[sample(nrow(rc), nrow(trav)/zeros), ] )  
trav$activity <- as.factor(trav$activity)  
levels(trav$activity) <- c("N","Y")  
trav <- trav[ ,-7]  
str(trav)

## 'data.frame': 2118 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 18318526 11416348 3325820 4669597 0 ...  
## $ ch1600 : num 104478 46830 3183 17609 0 ...  
## $ ch3200 : num 295447 100673 22235 76336 0 ...  
## $ ch800 : num 31111 7161 504 1707 0 ...  
## $ int\_n : num 34320 32434 35115 35005 26642 ...  
## $ int1600 : num 498 265 240 282 159 ...  
## $ int3200 : num 1320 655 809 876 473 ...  
## $ int800 : num 231.2 133.3 71.4 93 60.7 ...

waterstones <- input\_waterstones[input\_waterstones$Depthmap\_R != 0, ] # remove events with no segment  
waterstones <- waterstones[,-c(1:4,6:10,20:30)]  
names(waterstones) <- c("activity","ch\_n","ch1600","ch3200","ch800","int\_n",  
 "int1000","int1600","int3200","int800")   
waterstones <- rbind(waterstones, rc[sample(nrow(rc), nrow(waterstones)/zeros), ] )  
waterstones$activity <- as.factor(waterstones$activity)  
levels(waterstones$activity) <- c("N","Y")  
waterstones <- waterstones[ ,-7]  
str(waterstones)

## 'data.frame': 52 obs. of 9 variables:  
## $ activity: Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ch\_n : num 2.51e+10 8.26e+04 3.97e+06 0.00 0.00 ...  
## $ ch1600 : num 1249359 3704 40362 0 0 ...  
## $ ch3200 : num 10321945 5924 149898 0 0 ...  
## $ ch800 : num 154329 2458 7402 0 0 ...  
## $ int\_n : num 54694 34151 45619 44391 33719 ...  
## $ int1600 : num 3069 539 1577 1456 497 ...  
## $ int3200 : num 7609 1128 4565 4097 980 ...  
## $ int800 : num 1218 269 385 375 239 ...

all\_activities <- rbind(cafes, hair, med, pedu, pubs, sol, starbucks, trav, waterstones)

# Model Cafes

df <- cafes  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 2504  
##   
## > nrow(testing)  
## [1] 834  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))

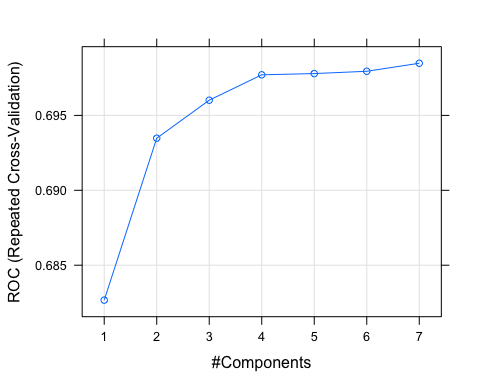
## Loading required package: pls

##   
## Attaching package: 'pls'

## The following object is masked from 'package:caret':  
##   
## R2

## The following object is masked from 'package:stats':  
##   
## loadings

##   
## > plsFit  
## Partial Least Squares   
##   
## 2504 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 2254, 2254, 2254, 2253, 2254, 2252, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.6826693 0.7252741 0.4459175  
## 2 0.6934763 0.6773143 0.6174222  
## 3 0.6960137 0.6395005 0.6573778  
## 4 0.6977091 0.6371132 0.6624317  
## 5 0.6977916 0.6365735 0.6605630  
## 6 0.6979447 0.6411111 0.6576402  
## 7 0.6984801 0.6405799 0.6640296  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 7.   
##   
## > plot(plsFit)

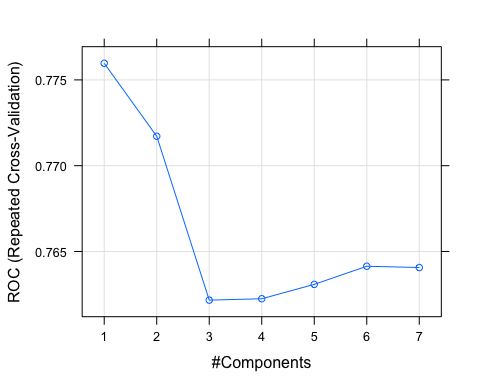


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 1 2 2 1 2 2 1 2 1 2 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 4 0.5217787 0.4782213  
## 14 0.4708889 0.5291111  
## 15 0.4028860 0.5971140  
## 18 0.6000724 0.3999276  
## 23 0.4396844 0.5603156  
## 35 0.4112498 0.5887502  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 289 134  
## Y 128 283  
##   
## Accuracy : 0.6859   
## 95% CI : (0.6531, 0.7172)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.3717   
## Mcnemar's Test P-Value : 0.7574   
##   
## Sensitivity : 0.6930   
## Specificity : 0.6787   
## Pos Pred Value : 0.6832   
## Neg Pred Value : 0.6886   
## Prevalence : 0.5000   
## Detection Rate : 0.3465   
## Detection Prevalence : 0.5072   
## Balanced Accuracy : 0.6859   
##   
## 'Positive' Class : N   
##

# Model Solicitors

df <- sol  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 1634  
##   
## > nrow(testing)  
## [1] 544  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 1634 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1471, 1471, 1470, 1471, 1471, 1471, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.7759699 0.8348138 0.5156981  
## 2 0.7717137 0.8037840 0.5373482  
## 3 0.7621637 0.7544214 0.5789070  
## 4 0.7622439 0.7540199 0.5793235  
## 5 0.7630822 0.7605691 0.5780939  
## 6 0.7641382 0.7728144 0.5727893  
## 7 0.7640640 0.7683278 0.5740139  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 1.   
##   
## > plot(plsFit)

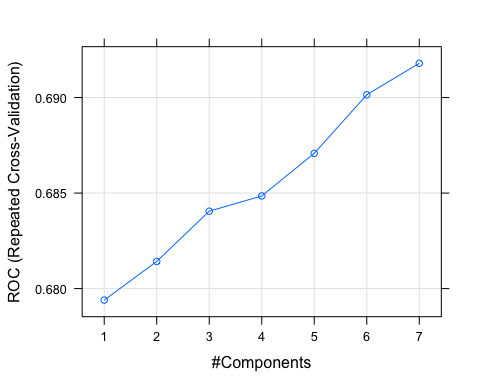


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 1 1 2 2 2 2 2 2 1 2 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 18 0.5656341 0.4343659  
## 19 0.5419879 0.4580121  
## 26 0.4742303 0.5257697  
## 29 0.4887952 0.5112048  
## 31 0.4348771 0.5651229  
## 39 0.4911497 0.5088503  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 223 136  
## Y 49 136  
##   
## Accuracy : 0.6599   
## 95% CI : (0.6184, 0.6997)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 3.687e-14   
##   
## Kappa : 0.3199   
## Mcnemar's Test P-Value : 2.568e-10   
##   
## Sensitivity : 0.8199   
## Specificity : 0.5000   
## Pos Pred Value : 0.6212   
## Neg Pred Value : 0.7351   
## Prevalence : 0.5000   
## Detection Rate : 0.4099   
## Detection Prevalence : 0.6599   
## Balanced Accuracy : 0.6599   
##   
## 'Positive' Class : N   
##

# Model Medical

df <- med  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 1430  
##   
## > nrow(testing)  
## [1] 476  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 1430 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1287, 1287, 1287, 1286, 1287, 1288, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.6793960 0.7635759 0.4499609  
## 2 0.6814208 0.7095396 0.5064032  
## 3 0.6840517 0.6713289 0.5548383  
## 4 0.6848505 0.6741132 0.5571596  
## 5 0.6870807 0.6759846 0.5473852  
## 6 0.6901462 0.7025561 0.5273474  
## 7 0.6917966 0.6932055 0.5478547  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 7.   
##   
## > plot(plsFit)

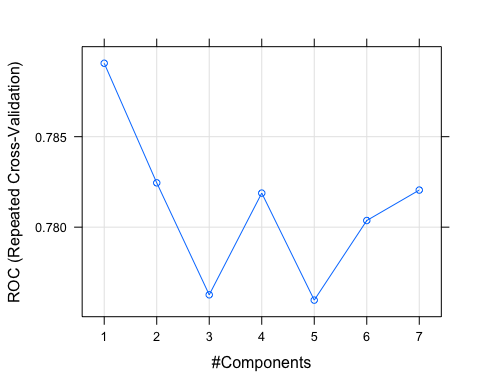


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 1 2 1 2 2 1 2 1 2 1 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 2 0.5127999 0.4872001  
## 3 0.3728239 0.6271761  
## 5 0.5220177 0.4779823  
## 6 0.3576680 0.6423320  
## 10 0.4665988 0.5334012  
## 11 0.5420507 0.4579493  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 165 119  
## Y 73 119  
##   
## Accuracy : 0.5966   
## 95% CI : (0.551, 0.641)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 1.438e-05   
##   
## Kappa : 0.1933   
## Mcnemar's Test P-Value : 0.001164   
##   
## Sensitivity : 0.6933   
## Specificity : 0.5000   
## Pos Pred Value : 0.5810   
## Neg Pred Value : 0.6198   
## Prevalence : 0.5000   
## Detection Rate : 0.3466   
## Detection Prevalence : 0.5966   
## Balanced Accuracy : 0.5966   
##   
## 'Positive' Class : N   
##

# Model Starbucks

df <- starbucks  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 236  
##   
## > nrow(testing)  
## [1] 78  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 236 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 212, 212, 213, 212, 213, 212, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.7890572 0.8621212 0.5876263  
## 2 0.7824495 0.8224747 0.6351010  
## 3 0.7762626 0.7797980 0.6414141  
## 4 0.7818813 0.7598485 0.6636364  
## 5 0.7759680 0.7489899 0.6608586  
## 6 0.7803662 0.7515152 0.6666667  
## 7 0.7820497 0.7545455 0.6666667  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 1.   
##   
## > plot(plsFit)

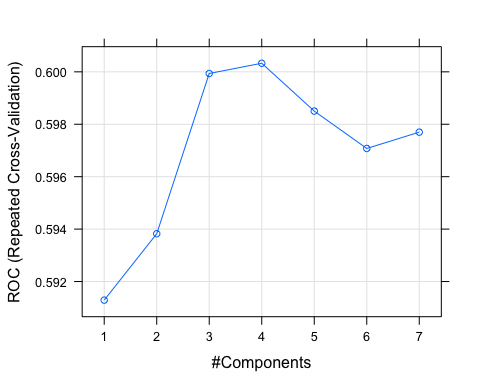


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 2 2 1 1 1 2 2 1 2 2 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 12 0.1846079 0.8153921  
## 18 0.3643009 0.6356991  
## 22 0.5729788 0.4270212  
## 23 0.5155152 0.4844848  
## 26 0.5769512 0.4230488  
## 28 0.4702924 0.5297076  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 34 17  
## Y 5 22  
##   
## Accuracy : 0.7179   
## 95% CI : (0.6047, 0.8141)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 7.474e-05   
##   
## Kappa : 0.4359   
## Mcnemar's Test P-Value : 0.01902   
##   
## Sensitivity : 0.8718   
## Specificity : 0.5641   
## Pos Pred Value : 0.6667   
## Neg Pred Value : 0.8148   
## Prevalence : 0.5000   
## Detection Rate : 0.4359   
## Detection Prevalence : 0.6538   
## Balanced Accuracy : 0.7179   
##   
## 'Positive' Class : N   
##

# Model Hospitals

df <- hosp  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 1812  
##   
## > nrow(testing)  
## [1] 604  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 1812 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1631, 1631, 1631, 1630, 1631, 1631, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.5912906 0.6766097 0.4400488  
## 2 0.5938236 0.6059829 0.5305535  
## 3 0.5999377 0.5949532 0.5651241  
## 4 0.6003273 0.5879528 0.5758282  
## 5 0.5985025 0.5956858 0.5684575  
## 6 0.5970766 0.5864917 0.5669760  
## 7 0.5976986 0.5868702 0.5732316  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 4.   
##   
## > plot(plsFit)

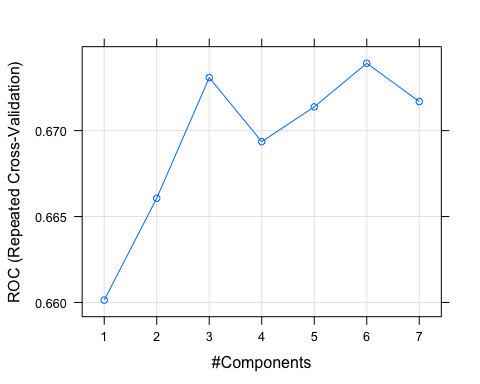


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 2 1 1 1 1 2 2 1 1 2 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 15 0.4826097 0.5173903  
## 20 0.5172275 0.4827725  
## 26 0.5589767 0.4410233  
## 27 0.5161887 0.4838113  
## 28 0.5006440 0.4993560  
## 39 0.4927298 0.5072702  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 179 145  
## Y 123 157  
##   
## Accuracy : 0.5563   
## 95% CI : (0.5156, 0.5964)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.00318   
##   
## Kappa : 0.1126   
## Mcnemar's Test P-Value : 0.19957   
##   
## Sensitivity : 0.5927   
## Specificity : 0.5199   
## Pos Pred Value : 0.5525   
## Neg Pred Value : 0.5607   
## Prevalence : 0.5000   
## Detection Rate : 0.2964   
## Detection Prevalence : 0.5364   
## Balanced Accuracy : 0.5563   
##   
## 'Positive' Class : N   
##

# Model Primary Education

df <- pedu  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 588  
##   
## > nrow(testing)  
## [1] 196  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 588 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 528, 529, 530, 530, 529, 530, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.6601357 0.7211494 0.4919923  
## 2 0.6660653 0.6590421 0.5341379  
## 3 0.6730865 0.6385441 0.5728352  
## 4 0.6693590 0.6295402 0.5762835  
## 5 0.6713835 0.6486973 0.5739847  
## 6 0.6739220 0.6556322 0.5693870  
## 7 0.6716916 0.6488123 0.5693870  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 6.   
##   
## > plot(plsFit)

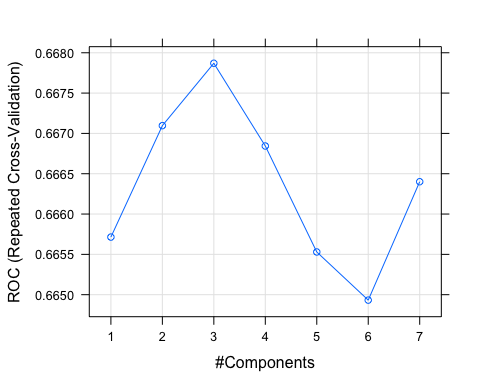


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 1 2 2 1 1 2 2 2 2 1 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 2 0.5199432 0.4800568  
## 5 0.4644706 0.5355294  
## 19 0.4463052 0.5536948  
## 44 0.5357074 0.4642926  
## 45 0.5154864 0.4845136  
## 47 0.4535277 0.5464723  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 65 39  
## Y 33 59  
##   
## Accuracy : 0.6327   
## 95% CI : (0.561, 0.7002)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.000125   
##   
## Kappa : 0.2653   
## Mcnemar's Test P-Value : 0.555690   
##   
## Sensitivity : 0.6633   
## Specificity : 0.6020   
## Pos Pred Value : 0.6250   
## Neg Pred Value : 0.6413   
## Prevalence : 0.5000   
## Detection Rate : 0.3316   
## Detection Prevalence : 0.5306   
## Balanced Accuracy : 0.6327   
##   
## 'Positive' Class : N   
##

# Model Hairdresser

df <- hair  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 3050  
##   
## > nrow(testing)  
## [1] 1016  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 3050 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 2745, 2746, 2746, 2745, 2746, 2744, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.6657143 0.7206556 0.4458935  
## 2 0.6670962 0.6697110 0.5628354  
## 3 0.6678699 0.6301542 0.6129128  
## 4 0.6668437 0.6227210 0.6203431  
## 5 0.6655296 0.6163857 0.6260205  
## 6 0.6649325 0.6109162 0.6242790  
## 7 0.6664010 0.6179036 0.6218639  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 3.   
##   
## > plot(plsFit)

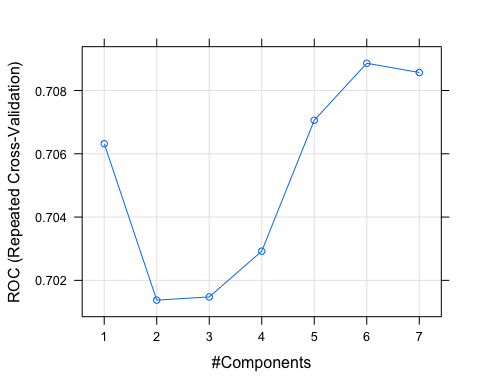


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 2 2 2 2 1 2 2 2 2 1 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 9 0.4888012 0.5111988  
## 19 0.4873679 0.5126321  
## 23 0.4774761 0.5225239  
## 30 0.4680328 0.5319672  
## 37 0.5984437 0.4015563  
## 39 0.4059132 0.5940868  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 327 181  
## Y 181 327  
##   
## Accuracy : 0.6437   
## 95% CI : (0.6134, 0.6732)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.2874   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.6437   
## Specificity : 0.6437   
## Pos Pred Value : 0.6437   
## Neg Pred Value : 0.6437   
## Prevalence : 0.5000   
## Detection Rate : 0.3219   
## Detection Prevalence : 0.5000   
## Balanced Accuracy : 0.6437   
##   
## 'Positive' Class : N   
##

# Model Travel Agency

df <- trav  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 1590  
##   
## > nrow(testing)  
## [1] 528  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 1590 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1432, 1430, 1432, 1430, 1432, 1430, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.7063179 0.7731646 0.4564241  
## 2 0.7013752 0.7186920 0.5060338  
## 3 0.7014767 0.6658544 0.5768724  
## 4 0.7029222 0.6625053 0.5881962  
## 5 0.7070594 0.6759072 0.5814399  
## 6 0.7088610 0.6855591 0.5655327  
## 7 0.7085688 0.6780116 0.5755802  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 6.   
##   
## > plot(plsFit)

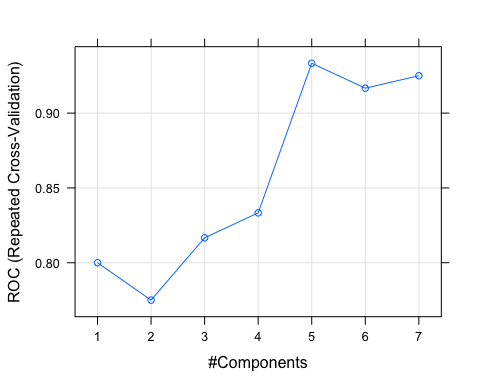


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 2 2 1 2 2 1 2 1 1 2 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 11 0.4246628 0.5753372  
## 19 0.4892100 0.5107900  
## 20 0.5685353 0.4314647  
## 25 0.4101104 0.5898896  
## 26 0.4746010 0.5253990  
## 30 0.5423945 0.4576055  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 185 104  
## Y 79 160  
##   
## Accuracy : 0.6534   
## 95% CI : (0.6111, 0.694)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 8.225e-13   
##   
## Kappa : 0.3068   
## Mcnemar's Test P-Value : 0.07604   
##   
## Sensitivity : 0.7008   
## Specificity : 0.6061   
## Pos Pred Value : 0.6401   
## Neg Pred Value : 0.6695   
## Prevalence : 0.5000   
## Detection Rate : 0.3504   
## Detection Prevalence : 0.5473   
## Balanced Accuracy : 0.6534   
##   
## 'Positive' Class : N   
##

# Model Waterstones

df <- waterstones  
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 40  
##   
## > nrow(testing)  
## [1] 12  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 40 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 36, 36, 36, 36, 36, 36, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.8000000 0.8500000 0.4333333  
## 2 0.7750000 0.8166667 0.6166667  
## 3 0.8166667 0.8500000 0.7000000  
## 4 0.8333333 0.8666667 0.6166667  
## 5 0.9333333 0.8833333 0.6333333  
## 6 0.9166667 0.8833333 0.6333333  
## 7 0.9250000 0.8500000 0.6500000  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 5.   
##   
## > plot(plsFit)

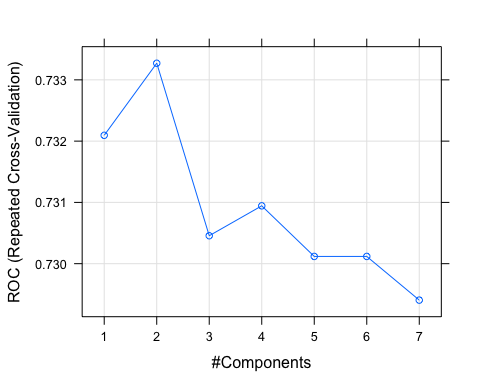


##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 2 2 2 2 2 1 1 1 2 2 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 6 0.05035891 0.9496411  
## 10 0.34272833 0.6572717  
## 15 0.46756727 0.5324327  
## 17 0.37396675 0.6260332  
## 18 0.36565687 0.6343431  
## 27 0.56633609 0.4336639  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 4 1  
## Y 2 5  
##   
## Accuracy : 0.75   
## 95% CI : (0.4281, 0.9451)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.073   
##   
## Kappa : 0.5   
## Mcnemar's Test P-Value : 1.000   
##   
## Sensitivity : 0.6667   
## Specificity : 0.8333   
## Pos Pred Value : 0.8000   
## Neg Pred Value : 0.7143   
## Prevalence : 0.5000   
## Detection Rate : 0.3333   
## Detection Prevalence : 0.4167   
## Balanced Accuracy : 0.7500   
##   
## 'Positive' Class : N   
##

# Model Public Houses

df <- pubs   
source("1\_PLSDA.R", echo = TRUE, print.eval = TRUE, max.deparse.length = 1000000)

##   
## > library(caret)  
##   
## > inTrain <- createDataPartition(y = df$activity, p = 0.75,   
## + list = FALSE)  
##   
## > training <- df[inTrain, ]  
##   
## > testing <- df[-inTrain, ]  
##   
## > nrow(training)  
## [1] 1460  
##   
## > nrow(testing)  
## [1] 486  
##   
## > ctrl <- trainControl(method = "repeatedcv", repeats = 3,   
## + classProbs = TRUE, summaryFunction = twoClassSummary)  
##   
## > plsFit <- train(activity ~ ., data = training, method = "pls",   
## + tuneLength = 15, trControl = ctrl, metric = "ROC", preProc = c("center",   
## + "scale"))  
##   
## > plsFit  
## Partial Least Squares   
##   
## 1460 samples  
## 8 predictor  
## 2 classes: 'N', 'Y'   
##   
## Pre-processing: centered (8), scaled (8)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1314, 1314, 1314, 1314, 1314, 1314, ...   
## Resampling results across tuning parameters:  
##   
## ncomp ROC Sens Spec   
## 1 0.7320948 0.7945205 0.5178082  
## 2 0.7332708 0.7484018 0.5511416  
## 3 0.7304560 0.6963470 0.6009132  
## 4 0.7309439 0.6890411 0.6200913  
## 5 0.7301182 0.6890411 0.6205479  
## 6 0.7301182 0.6977169 0.6141553  
## 7 0.7294051 0.6940639 0.6191781  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was ncomp = 2.   
##   
## > plot(plsFit)



##   
## > plsActivities <- predict(plsFit, newdata = testing)  
##   
## > str(plsActivities)  
## Factor w/ 2 levels "N","Y": 1 2 2 1 2 2 1 2 1 1 ...  
##   
## > plsProbs <- predict(plsFit, newdata = testing, type = "prob")  
##   
## > head(plsProbs)  
## N Y  
## 3 0.5414196 0.4585804  
## 5 0.4000833 0.5999167  
## 8 0.4267100 0.5732900  
## 12 0.5413036 0.4586964  
## 14 0.4872897 0.5127103  
## 15 0.4784543 0.5215457  
##   
## > confusionMatrix(data = plsActivities, testing$activity)  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 195 115  
## Y 48 128  
##   
## Accuracy : 0.6646   
## 95% CI : (0.6207, 0.7065)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 1.693e-13   
##   
## Kappa : 0.3292   
## Mcnemar's Test P-Value : 2.347e-07   
##   
## Sensitivity : 0.8025   
## Specificity : 0.5267   
## Pos Pred Value : 0.6290   
## Neg Pred Value : 0.7273   
## Prevalence : 0.5000   
## Detection Rate : 0.4012   
## Detection Prevalence : 0.6379   
## Balanced Accuracy : 0.6646   
##   
## 'Positive' Class : N   
##

### Notes

**Sensitivity** (also called the true positive rate, the recall, or probability of detection[1] in some fields) measures the proportion of positives that are correctly identified as such (e.g., the percentage of sick people who are correctly identified as having the condition).

**Specificity** (also called the true negative rate) measures the proportion of negatives that are correctly identified as such (e.g., the percentage of healthy people who are correctly identified as not having the condition).

title <- "The Syntactic Signature of Starbucks' Location: Towards a machine-learning approach to location decision-making"

alt title: Looking for Starbucks? Follow these steps!

# Abstract

### The Syntactic Signature of Starbucks' Location: Towards a machine-learning approach to location decision-making

### Land use studies and urban economies???

Key words: space syntax, land use, machine learning

This paper describes an experimental method that has been developed to investigate the relationship between space syntax variables, and the location of specific activities. The method aims at identifying the signature patterns of several land use categories based on their location, regarding syntactic variables. Syntactic variables are known to correlate with urban phenomena such as pedestrian and vehicular traffic flow (Penn, 1998; Lerman, Rofè et al. 2014). The natural movement theory (Hillier, 1993) suggests that such phenomena are the effect of the asymmetry created by the spatial configuration. The theory proposes that location is determined based on configurational properties, to attain particular exposure to users. The natural movement paradigm suggests that land use categories require a set of characteristics of accessibility measured by space syntax variables, which would lead to the desired exposure. Based on this assumption, we use machine learning techniques to find the underlying location requirements of 16702 events, over ten categories, in London's metropolitan region. Both supervised and unsupervised methods were applied. Through partial least squares discriminant analysis (PLSDA), a supervised method, it was possible to learn the pattern of occupation (model) of several categories, based, exclusively, on several measurements of the variables integration and choice. When applied to testing datasets, the models showed encouraging results. For example, the model used in category Starbucks reached an accuracy of 0.65 (95% CI: 0.61, 0.70), at P-Value [Acc > NIR] < 0.001, Mcnemar's Test P-Value < 0.001, Sensitivity: 0.8162, Specificity: 0.4853, on the testing set. Despite the moderate but statistically significant results, the experimental method shows promising for the identification of the syntactic signature of certain categories of land use. Identifying such signatures could lead to the incorporation of more fact-based data into location decision making, increasing its intelligence. Extensive use of the methodology could perhaps reveal a new classification of land use based on the clarity of their syntactic signature. The results also serve as a partial endorsement of the natural movement theory.

### References

Hillier, B., A. Penn, J. Hanson, T. Grajewski and J. Xu (1993). "Natural Movement - Or, Configuration and Attraction in Urban Pedestrian Movement." Environment and Planning B: Planning and Design - Pion Ltd 20(1): 29-66.

Lerman, Y., Y. Rofè and I. Omer (2014). "Using Space Syntax to Model Pedestrian Movement in Urban Transportation Planning." Geographical Analysis 46(4): 392-410.

Penn, A., Hillier, B., Banister, D., Xu, J. (1998). "Configurational modelling of urban movement networks." Environment and Planning B-Planning & Design 25: 59.