Similarity

Shantam Gupta

May 10, 2018

Using similarity for Statistical Process Control & Monitoring

Train data represented as S0(reference data) with N0 as the number of train data points available(entire historical record). Test data represented as Sw(Data)

```
Train <- read.csv('lumos_training_set.csv')
Test <- read.csv('lumos_all_set.csv')

#remove repeated measurements and reshape the dataset
ind <- which(with( Train, (Train$PepSeq=="EYEATLEEC(Carbamidomethyl)C(Carbamidomethyl)AK" | Train$PepSe
S0<-Train[-ind,]
S0<-S0[,-2]
Train<-S0
S0$PepSeq<- gsub("\\(Carbamidomethyl\\)","",S0$PepSeq)
S0 <- reshape(S0, idvar = "idfile", timevar = "PepSeq", direction = "wide")
RESPONSE<-c("GO")
S0 <- cbind(S0,RESPONSE)

ind <- which(with( Test, (Test$PepSeq=="EYEATLEEC(Carbamidomethyl)C(Carbamidomethyl)AK" | Test$PepSeq==
Data0<-Test[-ind,]
Data0<-Data0[,-2]
Data0$PepSeq<- gsub("\\(Carbamidomethyl\\)","",Data0$PepSeq)
Data1 <- Data0[1:8 + rep(seq(0, nrow(Data0), by=100), each=8),]
Data1 <- reshape(Data1, idvar = "idfile", timevar = "PepSeq", direction = "wide")
RESPONSE<-c("NOGO")
Data <- cbind(Data1,RESPONSE)
```

Installing the Package

```
#install.packages("lsa")
library(lsa) # cosine
```

```
## Loading required package: SnowballC
```

```
#?cosine
```

Filtering numeric features

```
#makes sure the data is numeric
sw <- sapply(Data[,c(-1,-50)],as.numeric)
s0 <- sapply(S0[,c(-1,-50)],as.numeric)
```

Taking colum means to get the average representative point for train data(s0)

```
avg_s0 <- colMeans(s0)
```

Finding similarity between train and test data point(Exploration)

Cosine Similarity

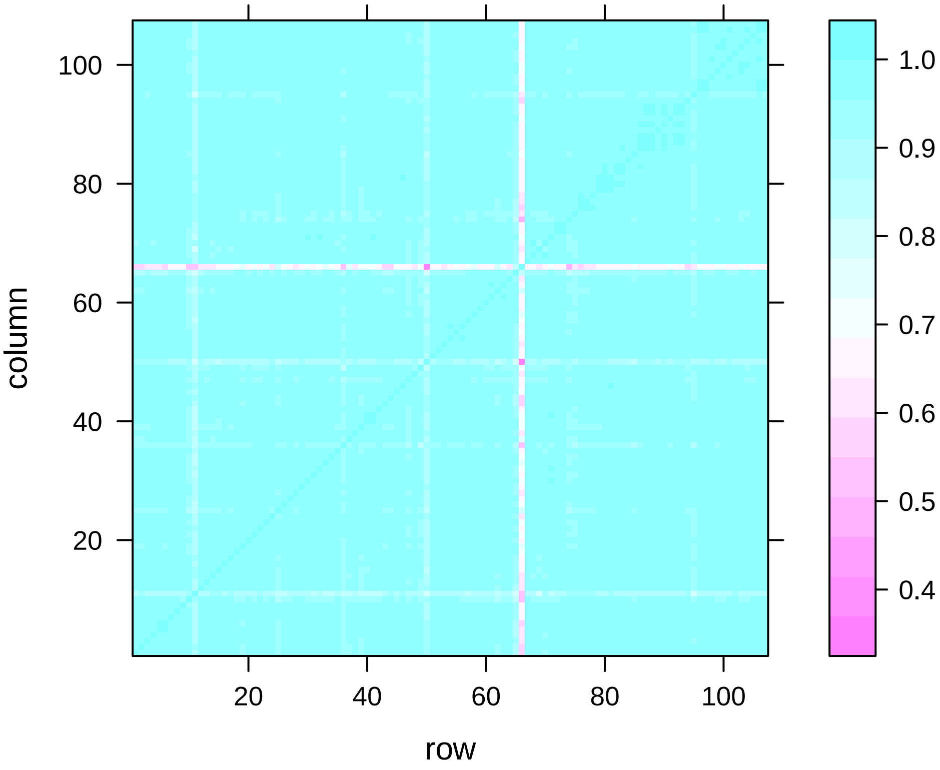
Train data

```
train_cs <- cosine(t(as.matrix(s0)))

#finding the least similar data points in the matrix
which(train_cs == min(train_cs), arr.ind = TRUE)
```

```
##      row col
## [1,]  66  50
## [2,]  50  66
```

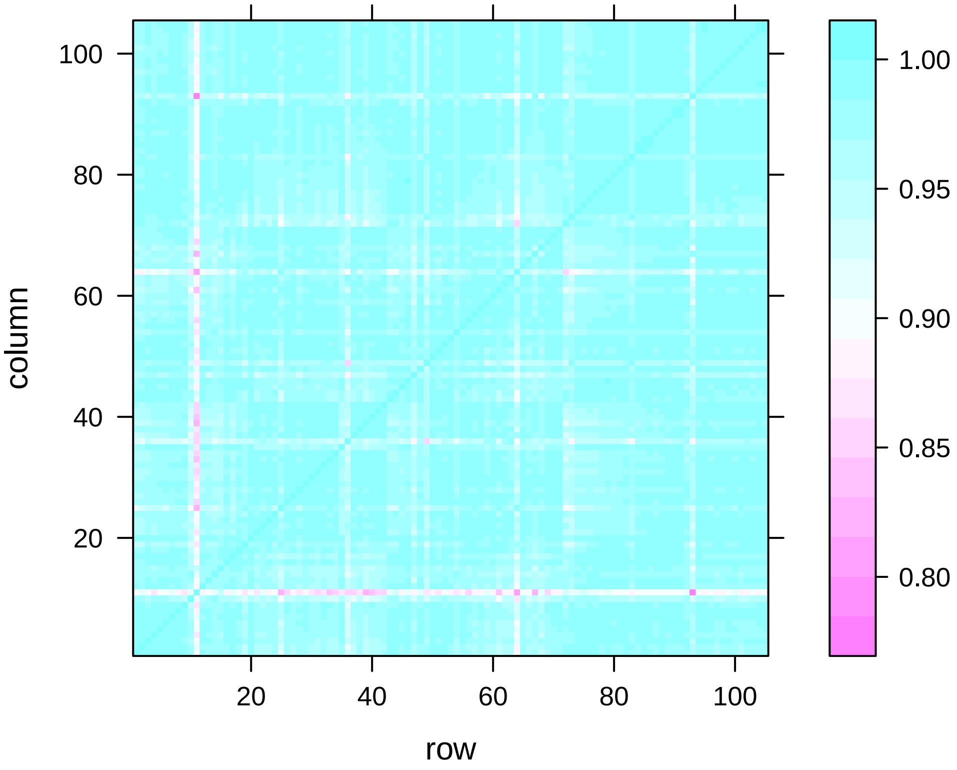
The success for this method is determined by the fact how similarity affects the distribution of the data. Let's look at the distribution of similarity in train data

```
library(lattice)

levelplot(train_cs)
```



let us remove the above rows which have low similarity

```
levelplot(train_cs[-c(66,50),-c(66,50)])
```

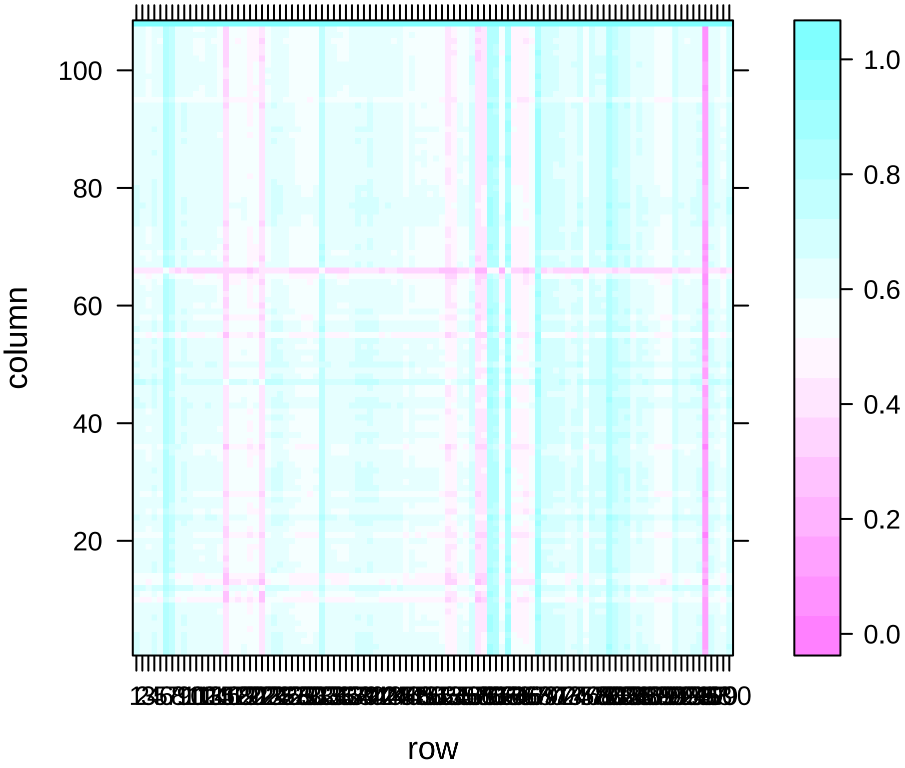


Comparing the test data point with all reference data and finding the least similarity

Plotting some of the values

```
min_similarity <- c()
test_similarity <- data.frame()
#finding cosine similarity between test data point(sw) and train data(s0)
for(i in 1:nrow(sw)){
  matrix <- as.matrix(rbind(s0,sw[i,]))
  train_test_i_cs <- cosine(t(as.matrix(rbind(s0,sw[i,]))))

  #taking the last element from the matrix and finding the lowest similarity
  min_similarity[i] <- min(train_test_i_cs[108,])

  #storing the similarity of test data points
  test_similarity <- rbind(test_similarity,train_test_i_cs[108,])
}
```

```
names(test_similarity) <- NULL
row.names(test_similarity) <- NULL
levelplot(as.matrix(test_similarity[1:100,]))
```



The 107 columns represent the similarity with train data(s0). The rows are first 100 test data points. For great results the above matrix should show colors with low similarity value