Profile Regression vs other clustering methods on synthetic data

Bemsibom Toh

2020-12-16T21:13:14-05:00

# Introduction

Profile regression is a clustering technique which links the clusters to an outcome of interest via a regression model. It differs from traditional distance-based clustering techniques in that it uses a stochastic model-based approach.

In general, models are fitted using Markov Chain Monte Carlo sampling methods, which outputs a different partition of the data at each iteration of the sampler.

The aim of this article is to compare the performance of profile regression to other clustering methods, such as Hierarchical clustering, K-Means, K-mode and Latent Class Analysis. We will do this by applying profile regression to a set of 1000 simulated datasets, generated from code written by [Linda Nichols](https://warwick.ac.uk/fac/sci/statistics/staff/academic-research/lnichols/) and [Tom Taverner](https://research.birmingham.ac.uk/portal/en/persons/thomas-taverner(68156dc8-5825-4005-8f6c-559fcbf6717f).html). The mathematical model used to generate these datasets will be the subject of another article.

We begin by loading the datasets and libraries we need as follows:

library(data.table)  
library(ggplot2)  
library(data.table)  
library(ggplot2)  
library(lpSolve)  
library(viridisLite)  
library(viridis)  
library(hrbrthemes)  
hrbrthemes::import\_roboto\_condensed()

## You will likely need to install these fonts on your system as well.  
##   
## You can find them in [/Library/Frameworks/R.framework/Versions/4.0/Resources/library/hrbrthemes/fonts/roboto-condensed]

library(mclust)

## Package 'mclust' version 5.4.6  
## Type 'citation("mclust")' for citing this R package in publications.

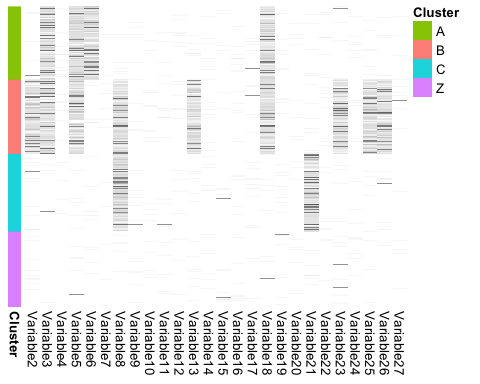
library(mcclust)  
library(coda)  
library(PReMiuM)  
source('/Users/bemsi/Documents/BSU/code/transform\_bham\_data.R')

Next we load the different datasets we will use for this project:

setwd('/Users/bemsi/Documents/BSU/code/')  
birmingham.aris <- get(load('/Users/bemsi/Documents/BSU/code/sim\_20201009\_ri.RData')) #ARIs from the other methods  
  
#profile regression with no outcome  
filelist = list.files("/Users/bemsi/Documents/BSU/code/SimulationOutputNoOutcome", pattern = "\\pear\_aris.txt$", full.names=TRUE)   
premium.aris.no.outcome <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
#profile regression with random outcome  
filelist = list.files("/Users/bemsi/Documents/BSU/code/SimulationOutputRandomOutcome", pattern = "\\pear\_aris.txt$", full.names=TRUE)  
premium.aris.random.outcome <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
#profile regression with deterministic outcome  
filelist = list.files("/Users/bemsi/Documents/BSU/code/SimulationOutputDetOutcome", pattern = "\\pear\_aris.txt$", full.names=TRUE)  
premium.aris.det.outcome <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
#profile regression with semi-deterministic outcome 1  
filelist = list.files("/Users/bemsi/Documents/BSU/code/DetOutcomeOne", pattern = "\\pear\_aris.txt$", full.names=TRUE)  
premium.aris.det.outcome.1 <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
#profile regression with semi-deterministic outcome 2  
filelist = list.files("/Users/bemsi/Documents/BSU/code/DetOutcomeThree", pattern = "\\pear\_aris.txt$", full.names=TRUE)  
premium.aris.det.outcome.2 <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
# Load and transform the first dataset in the bank of simulated datasets  
birmingham.dfs <- get(load('/Users/bemsi/Documents/BSU/code/sim\_20201009\_df.rData'))  
sample.dataframe <- birmingham.dfs[[1]]  
inputs <- transform\_bham\_data.one(sample.dataframe)  
  
#modifying original dataframe  
birmingham.aris$PRegr <- premium.aris.no.outcome$V1  
birmingham.aris.one <- reshape2::melt(birmingham.aris , measure.vars=c("LCA", "kmode", "HCA", "MCAkm", "kmean", "PRegr"))  
names(birmingham.aris.one) <- c("Method", "ARI")

We now visualize the data, to see whether there is any apparent clustering structure in it.

clusters <- data.frame(inputs$inputData$group)  
colnames(clusters) <- "Cluster"  
pheatmap::pheatmap(inputs$inputData[2:27],   
 cluster\_rows = F,  
 cluster\_cols = F,  
 show\_rownames = FALSE,  
 legend = FALSE,  
 cutree\_rows = 4,   
 col = c("white", "black"),  
 annotation\_row = clusters)



We can see that there is some structure in the dataset, and there are three clearly identifiable clusters in the dataset, as well as one ‘noisy’ cluster.

# No outcome

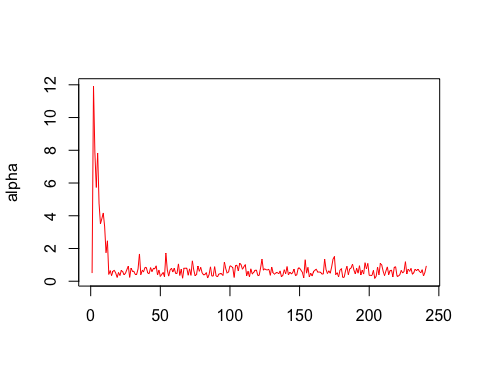
We begin by looking at the output of profile regression when there is no supervision - i.e., we carry out profile regression with no outcome to guide the clustering algorithm.

## Convergence analysis

We will use three methods to assess convergence of the profile regression algorithm: a visual method using traceplots of certain parameters, the [Gelman Diagnostic](https://www.rdocumentation.org/packages/coda/versions/0.19-4/topics/gelman.diag), and the [Geweke Diagnostic](https://www.rdocumentation.org/packages/coda/versions/0.19-4/topics/geweke.diag).

We begin by plotting the values of alpha through the different steps of the MCMC.

birmingham.dataframes <- get(load('~/Documents/BSU/code/sim\_20201009\_df.RData'))  
sample.dataframe <- birmingham.dataframes[[1]] #We are taking the first dataframe in our series  
group.labels <- sample.dataframe$group  
alphas <- fread("~/Documents/BSU/code/MCMCDiagnostics/myOutput\_1\_alpha.txt")  
plot(alphas$V1, type='l', col='red', ylab='alpha', xlab='')



We can that the algorithm converges after several iterations.

We now run the Geweke diagnostic test on the alphas to see if there is convergence.

mcmc\_alpha <- mcmc(data=alphas$V1)  
geweke.diag(mcmc\_alpha)

##   
## Fraction in 1st window = 0.1  
## Fraction in 2nd window = 0.5   
##   
## var1   
## 1.573

We obtain a z-value of about 1.57, which is within the acceptance region for a significance level of 0.05.

To run the Gelman diagnostic, we need two chains. So we run Premium twice on the same dataset, using different starting points (seeds), and then proceed to carry out this test.

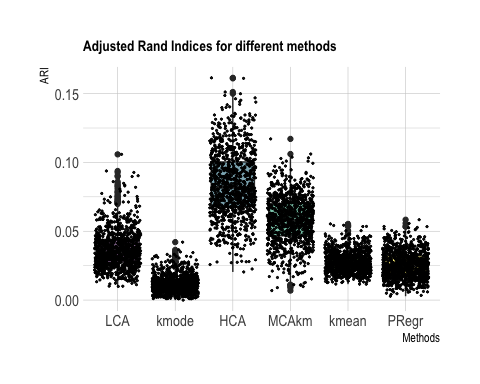
# runInfoObj <- profRegr(yModel = inputs$yModel,  
# xModel = inputs$xModel,  
# nSweeps = 30000,  
# nBurn = 2000,  
# data = inputs$inputData[,2:27],  
# output="myOutput\_unsupervised\_gelman\_1",  
# covNames = names(inputs$inputData[,2:27]),  
# reportBurnIn = TRUE,  
# seed = 12,  
# excludeY = TRUE)  
#   
# runInfoObj <- profRegr(yModel = inputs$yModel,  
# xModel = inputs$xModel,  
# nSweeps = 30000,  
# nBurn = 2000,  
# data = inputs$inputData[,2:27],  
# output="myOutput\_unsupervised\_gelman\_2",  
# covNames = names(inputs$inputData[,2:27]),  
# reportBurnIn = TRUE,  
# seed = 13,  
# excludeY = TRUE)  
  
alpha\_1 <- fread("~/Documents/BSU/code/myOutput\_unsupervised\_gelman\_1\_alpha.txt")  
alpha\_2 <- fread("~/Documents/BSU/code/myOutput\_unsupervised\_gelman\_2\_alpha.txt")  
mcmc\_alpha\_1 <- mcmc(data=alpha\_1$V1)  
mcmc\_alpha\_2 <- mcmc(data=alpha\_2$V1)  
combined.chains <- mcmc.list(mcmc\_alpha\_1,mcmc\_alpha\_2)  
gelman.diag(combined.chains)

## Potential scale reduction factors:  
##   
## Point est. Upper C.I.  
## [1,] 1 1

The Gelman diagnostic here is 1. A factor of 1 means that between variance and within chain variance are equal, larger values mean that there is still a notable difference between chains.

Now we visualize the distribution of ARIs for the different methods when applied on the dataset.

ggplot(birmingham.aris.one, aes(x=Method, y=ARI, fill=Method)) +  
 geom\_boxplot() +  
 scale\_fill\_viridis(discrete = TRUE, alpha=0.6) +  
 geom\_jitter(color="black", size=0.4, alpha=0.9) +  
 theme\_ipsum() +  
 theme(  
 legend.position="none",  
 plot.title = element\_text(size=11)  
 ) +  
 ggtitle("Adjusted Rand Indices for different methods") +  
 xlab("Methods")



The ARIs are quite low in general. A possible reason for this is that there may be empty rows in the dataset which need to be eliminated.

Unsupervised profile regression does… okay, as you can see from the boxplots. Given that the number of clusters present in the dataset is known to the other clustering algorithms, but not to profile regression, this is not entirely a bad situation.

## Outcome - guided profile regression

Now, we want to investigate the effect of incorporating an outcome we incorporate an outcome to the dataset randomly. We now want to improve on the clarity of the outcome used in the analysis. ‘Clarity’ in this scenario is a measure of correlation between the binary data and the outcome. We will experiment with discrete, binary outcomes in the first instance, and consider different levels of clarity. To do this, we will control the probability of the outcome being equal to 1 for a given patient - the closer this probability is to 1, the clearer the outcome is. This is done as follows:

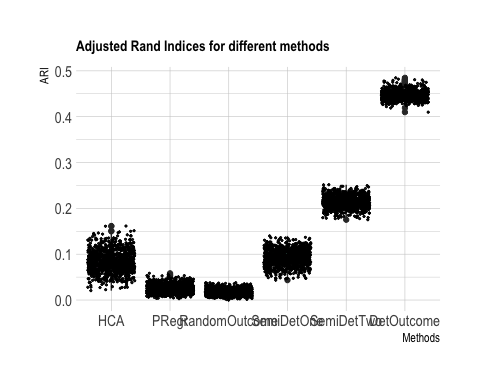
And then the outcome vector is added on to the dataset, and then analysed accordingly.

Using this code, we can derive four different outcomes:

* *Random outcome*: The probabilities are c(0.5, 0.5)
* *Deterministic outcome*: The probabilities are c(1,0)
* *Semi-deterministic outcome 1*: he probabilities are c(0.6,0.4)
* *Semi-deterministic outcome 2*: he probabilities are c(0.8,0.2)

We now run PReMiuM on datasets with these outcomes incorporated, and we get the following results:

birmingham.aris$NoOutcome <- premium.aris.no.outcome$V1  
birmingham.aris$RandomOutcome <- premium.aris.random.outcome$V1  
birmingham.aris$DetOutcome <- premium.aris.det.outcome$V1  
birmingham.aris$SemiDetOne <- premium.aris.det.outcome.1$V1  
birmingham.aris$SemiDetTwo <- premium.aris.det.outcome.2$V1  
  
interesting.columns <- c('HCA', 'PRegr', 'RandomOutcome', 'SemiDetOne', 'SemiDetTwo', 'DetOutcome')  
   
birmingham.aris.two <- reshape2::melt(birmingham.aris[interesting.columns], measure.vars=interesting.columns)  
names(birmingham.aris.two) <- c("Method", "ARI")  
  
ggplot(birmingham.aris.two, aes(x=Method, y=ARI, fill=Method)) +  
 geom\_boxplot() +  
 scale\_fill\_viridis(discrete = TRUE, alpha=0.6) +  
 geom\_jitter(color="black", size=0.4, alpha=0.9) +  
 theme\_ipsum() +  
 theme(  
 legend.position="none",  
 plot.title = element\_text(size=11)  
 ) +  
 ggtitle("Adjusted Rand Indices for different methods") +  
 xlab("Methods")



We can see from there that a random outcome (which has no discernible relationship with the rest of the dataset) performs worse than no outcome at all. But once there is some structure in the outcome, the performance of Profile Regression improves quite drastically.

## Same analysis, this time with a cleaner dataset

This time, we

#profile regression with no outcome  
filelist = list.files("/Users/bemsi/Documents/BSU/code/NewDetOutcome1/SimulationOutput", pattern = "\\pear\_aris.txt$", full.names=TRUE)   
clean.premium.no.outcome <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
#profile regression with random outcome  
filelist = list.files("/Users/bemsi/Documents/BSU/code/NewDetOutcome2/SimulationOutput", pattern = "\\pear\_aris.txt$", full.names=TRUE)  
clean.premium.det.outcome.two <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
#profile regression with deterministic outcome  
filelist = list.files("/Users/bemsi/Documents/BSU/code/NewDetOutcome3/SimulationOutput", pattern = "\\pear\_aris.txt$", full.names=TRUE)  
clean.premium.det.outcome.three <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
#profile regression with semi-deterministic outcome 1  
filelist = list.files("/Users/bemsi/Documents/BSU/code/NewDetOutcome4/SimulationOutput", pattern = "\\pear\_aris.txt$", full.names=TRUE)  
clean.premium.det.outcome <- rbindlist(sapply(filelist, fread, simplify=FALSE, USE.NAMES = TRUE))  
  
inputs.cleaned <- transform\_bham\_data(sample.dataframe)

We visualize again:

clusters <- data.frame(inputs.cleaned$inputData$group)  
colnames(clusters) <- "Cluster"  
pheatmap::pheatmap(inputs$inputData[2:27],   
 cluster\_rows = F,  
 cluster\_cols = F,  
 show\_rownames = FALSE,  
 legend = FALSE,  
 cutree\_rows = 4,   
 col = c("white", "black"),  
 annotation\_row = clusters)

