patientCluster Examples

Jenna Reps

February 17, 2016

Contents

1 Introduction

This document details the various functionalities of the patientCluster package. The first section gives a step by step example of clustering patients given a previously created cohort (cluster patients) and the second section details how to use the package to aid the creation of health topics (cluster concept ids).

The patientCluster packagecomes with a range of default clustering functions, namely kmeans, generalized low rank models and a type of consensus clustering but also enables users to use their own clustering function.

The cluster process follows the general flow:

1. extract data - extractData()
2. find clusters in the data - clusterPatients()
3. evaluate and compare clusters - evaluateClusters()
4. export the clusters to json for interactive visulisation - exportJson()

2 Default Model Example

The first step is extracting the patientCluster packagedata. This requires setting up a database connection and using an existing cohort or creating a new cohort table. You also need to initiate the h2o cluster (this requires downloading and installing h2o from: http://www.h2o.ai/) and I recommend setting the fftempdir to a directory on your computer with plenty of space.

2.1 Loading Libraries

Load the following libraries:

```
library(DatabaseConnector)
library(SqlRender)
library(ffbase)
library(patientCluster)
library(h2o)

# general initial settings '50g' means 50GB - you may need to reduce
options(fftempdir = "drive:/FFtemp")
h2o.init(nthreads = -1, max_mem_size = '50g')
```

2.2 Extracting Data

To set up a database connection for the user 'JoeDoe' with a password of 'password1' on the server 'server name' with a work schema (must have read/write access) of 'Work.dbo' using parallel data warehouse:

```
dbms <- "pdw"
user <- JoeDoe
pw <- password1
server <- "server name"
port <- 1
oracleTempSchema <- NULL
cdmVersion <- 5
# add your CDm schema below:
cdmDatabaseSchema <- 'CDM.dbo'


connectionDetails <- DatabaseConnector::createConnectionDetails(dbms = dbms,
                                                                server = server,
                                                                user = user,
                                                                password = pw,
                                                                port = port)
```

The following SQL code will create a cohort table for smokers (people with concept id

2101895 recorded) with the cohort start date being the first recording date. Smoking has conceptId 1; the data are stored in table 'smokeall_yourDataBase' in the schema 'scratch.dbo'.

```
# all people with smoking recorded (concept: 2101895)
sql <- "select * into scratch.dbo.smokeall_yourDataBase
       from (select person_id subject_id,
         OBSERVATION_DATE as cohort_start_date,
          OBSERVATION_DATE as cohort_end_date,
          1 cohort_concept_id,
row_number() over (partition by person_id order by OBSERVATION_DATE asc) rn
          from CDM.dbo.observation
          where observation_concept_id in (2101895)) temp
          where temp.rn=1;"
sql <- translateSql(sql, 'sql server','pdw')\$sql
executeSql(conn, sql)
```

With our connection and cohort tables created, we now need to run dataExtract() with the inputs:

- connectionDetails - the connection details
- cdmDatabaseSchema - the schema of the CDM where the people in the cluster come from
- workDatabaseSchema - the schema where you will write any tables required during data extraction
- cohortid - the id of the cohort you wish to extract
- agegroup - if you want to select an age subset use this, otherwise set it to NULL
- gender - if you want to select only males or females use this, otherwise set it to NULL
- type - 'condition' or 'group' - condition means the clustering will be done using all condition concept ids as features, group means the clustering will be done using sets of concepts as features defined using the groupDef input (see examples below).
- groupDef - either 'default' which uses predefined concept sets or a table with two columns: 'covariate' and 'concept_id', where the concept_ids corresponding to the same covariate are a concept set.
- historyStart and historyEnd define the time period relative to the cohort start date to search for the concept_id recordings.
- loc - the location to save any results

For example, to extract the default concept set features for all smokers (all ages and genders) where the presence of each concept id is searched from 365 days prior to cohort start until 1 day prior to cohort start run:

```
d1.groupdefault <- dataExtract(connectionDetails,
                  cdmDatabaseSchema= 'CDM.dbo',
                 cohortDatabaseSchema='scratch.dbo',
                 workDatabaseSchema='scratch.dbo',
                 cohortid=1, agegroup=NULL, gender=NULL,
                 type='group', groupDef = 'default',
                 historyStart=1,historyEnd=365, loc=getwd())
```

To only extract the data for people aged 40-60 run:

```
d1.groupdefault4060 <- dataExtract(connectionDetails,
                  cdmDatabaseSchema= 'CDM.dbo',
                 cohortDatabaseSchema='scratch.dbo',
                 workDatabaseSchema='scratch.dbo',
                 cohortid=1, agegroup=3, gender=NULL,
                 type='group', groupDef = 'default',
                 historyStart=1,historyEnd=365, loc=getwd())
```

To only extract the data for people aged 40-60 but extract the features from 720 days prior to index until 365 days prior to index run:

```
d1.groupdefault4060_2 <- dataExtract(connectionDetails,
                  cdmDatabaseSchema= 'CDM.dbo',
                 cohortDatabaseSchema='scratch.dbo',
                 workDatabaseSchema='scratch.dbo',
                 cohortid=1, agegroup=3, gender=NULL,
                 type='group', groupDef = 'default',
                 historyStart=365,historyEnd=720, loc=getwd())
```

To extract your own topic definitions first create the topic table:

```
topics <- data.frame(covariate=c(1,1,1,2,2,3),
   concept_id=c(23,650,26,43,44,10002))
```

where topic 1 consists of concept_ids 23, 650 and 26, topic 2 consists of concept_ids 43 and 44 and topic 3 consists of concept_id 10002.

Then we run:

```
d1.groupown <- dataExtract(connectionDetails,
                  cdmDatabaseSchema= 'CDM.dbo',
                 cohortDatabaseSchema='scratch.dbo',
                 workDatabaseSchema='scratch.dbo',
                 cohortid=1, agegroup=NULL, gender=NULL,
                 type='group', groupDef = topics,
                 historyStart=1,historyEnd=365, loc=getwd())
```

to extract the whether each person in the smoking cohort has the three topics recorded 356 days prior to cohort start until 1 day prior to cohort start.

The final option is to not define topics/groups of concept ids but to extract all condition concept ids. This is accomplished using:

```
d2.raw <-  dataExtract(connectionDetails,
                       cdmDatabaseSchema= 'CDM.dbo',
                      cohortDatabaseSchema='scratch.dbo',
                      workDatabaseSchema='scratch.dbo',
                      cohortid=1, agegroup=NULL, gender=NULL,
                      type='condition', groupDef = NULL,
                      historyStart=1,historyEnd=180, loc=getwd())
```

where this will extract all condition concepts that occurred between 180 days prior to cohort start to 1 day prior to cohort start for all the people in the smoking cohort.

2.3 Saving/loading cluster data

To save the cluster data run:

```
saveClusterData(d1.groupdefault, file=file.path(getwd(), 'cluster_data') )
```

the data can then be loaded using: To save the cluster data run:

```
d1.groupdefault <- loadClusterData(file=file.path(getwd(), 'cluster_data') )
```

2.4 Clustering

With our data extracted and the features created using topic concept sets or corresponding to all condition concept ids, we can now cluster the cohort data.

The clustering function requires the previously extracted data, a vector specifying the min and max age to include, the gender to do the clustering on, the clustering method and size as well as how to process the data prior to clustering. To generate clusters on the default group smoking data for all people aged between 0 and 110 years using kmeans to find 100 clusters and normalising the data prior to clustering:

```
c1.groupdefault <- clusterPeople(d1.groupdefault, ageSpan=c(0,110),
    gender=NULL,
                     method='kmeans', clusterSize=100, normalise=T, fraction=F,
                         binary=F,
                     covariatesToInclude=NULL,covariatesToExclude=NULL,covariatesGroups=NULL)
```

In the previous code the pre-processing was set to normalise (for each topic/group feature value subtract the mean number of times the topic is recorded for the cohort patients and divide by the standard deviation of the number of times the topic is recorded for the cohort patients). This aims to scale the topics/groups fairly, even when one is recorded more often than others, and makes calculating the distances in kmeans not put more weight on a specific topic.

If you wanted to cluster all the smoking cohort (no age/gender restrictions) and use the fraction of the persons total records that correspond to each group/topic of the newly defined concept groups as features for the clustering (to extract 100 clusters), run:

```
c1.groupown1 <- clusterPeople(d1.groupown , ageSpan=NULL, gender=NULL,
                    method='kmeans', clusterSize=100, normalise=F, fraction=T,
                        binary=F,
                    covariatesToInclude=NULL,covariatesToExclude=NULL,covariatesGroups=NULL)
```

Where if a person has concept ids 23, 26 and 44 recorded during 365-1 days prior to index then their group/topic features when fraction pre-processing is implemented would be: 0.67 for topic 1 (two of three concepts correspond to topic 1) , 0.33 for topic 3 (one of the three concepts corresponds to topic 3) and 0 for topic 2 (as they had no topic 2 concept ids). Because the fraction pre-processing is scaling the features between 0 and 1, normalizing the data is not necessary (although it can still be done in addition).

Alternatively, you could pre-process by only counting a topics concepts ids once by setting binary=T and then calculate the fraction by setting fraction=T. This would mean the group/topic features for a person who has concept ids 23, 26 and 44 recorded during 365-1 days prior to index, their features would be 0.5 for topics 1 and 2 as they had concepts corresponding to each topic. This pre-processing is accomplished by running:

```
c1.groupown2 <- clusterPeople(d1.groupown , ageSpan=NULL, gender=NULL,
                    method='kmeans', clusterSize=100, normalise=F, fraction=T,
                        binary=T,
                    covariatesToInclude=NULL,covariatesToExclude=NULL,covariatesGroups=NULL)
```

To implement consensus clustering which will run kmeans 10 (repeats) times and then implement generalised low rank modelling on the clusters returned by kmeans for the smoking data to return 100 clusters using the default topics run:

```
c2.groupdefault_concensus <- clusterPeople(d1.groupdefault, ageSpan=c(0,110),
    gender=NULL,
                            method='concensus', clusterSize=100, glrmFeat=NULL,
                            normalise=T, fraction=F, binary=F,
                            covariatesToInclude=NULL,covariatesToExclude=NULL,
                                                    covariatesGroups=NULL,
                            extraparameters = list(csample=0.8, rsample=0.9,
                                repeats=10)
```

To cluster the patients using the data with features corresponding to conditions rather than groups, it is recommended to use generalised low rank models rather than kmeans. The reason kmeans is less suitable is due to there likely being a large number of features (there are thousands of condition concept ids), and calculating distances is problematic over highly dimensional space. The generalised low rank model method firstly runs matrix factorization to create data driven topics (the number of topics is controlled by glrmFeat) and then implements kmeans on each persons topic features. To run this method for people between 20 and 50 years old to firstly find 50 topics and then return 50 clusters:

```
c2.raw <- clusterPeople(d2.raw, ageSpan=c(20,50), gender=NULL,
                    method='glrm', glrmFeat=50, clusterSize=50, normalise=F,
                        fraction=T,
                    covariatesToInclude=NULL,covariatesToExclude=NULL,covariatesGroups=NULL)
```

The options covariatesToInclude and covariatesToExclude enable you to pick out specific concept ids or topics/groups to apply the clustering to.

2.5 Evaluating clusters

After running the clustering you can then extract summary statistics about each cluster and find how diverse the topics/groups were across the clusters. This can give you feedback and help redo and improve the clustering. The output of the cluster evaluation can also be exported into json format and viewed using an interactive javascript interface we have developed.

To evaluate any cluster result simply run (with any cluster result as the input):

```
cluster.eval <- clusterEval(c1.groupdefault)
```

2.6 Exporting clusters

To view the clusters you can export them into a suitable JSON format. To extract all the clusters run:

```
jsonFormat(cluster.eval)
```

alternatively, you may find you only want to view clusters that satifsty a minimum size, this can be accomplished by finding which of the clusters you want to view and inputting a vector of cluster indexes that you with to view interactively:

```
# find the cluster indexes of the clusters of interest
# in this case the clusters with more than 280 people in
coi <- (1:100)[table(c1$clusters$predict)>280]

# run the jsonFormat but add the clusters of interest vector
# the second input
jsonFormat(cluster.eval, coi)
```

3 Creating data-driven Topics

In addition to clustering patients, this package also provides a way to cluster concepts and create concept topics/groups. The idea is to use a data driven approach to aid the discovery of topics, where the output of the clustering will be inspected by and expert and modified, then the clustering reapplied until the expert is happy with the results.

To cluster the concepts you need to input the connection details and schema of the CDM in addition to the parameters used to determine the clustering method. The concept clustering extracts features based on ingredients that occur before the concepts (indications=F) or after the concepts (indications=T) and searches dayStart to dayEnd relative to the first time the concept is recorded for each patient in the data. min_obs is a constraint on the minimum number of people in the database who must have the ingredient recorded for it to be used as a feature in the clustering.

To obtain 150 topics/groups of condition concept ids via a generalise low rank model that finds 100 topics followed by kmeans, run:

```
datadriven.topics <-
   clusterConcepts(connectionDetails,cdmDatabaseSchema='CDM.dbo',
              method='glrm', clusterSize=150,topicSize=100, scale=T,
              covariatesToInclude=NULL,
              indications=T, dayStart=1,dayEnd=30,
              use_min_obs=TRUE, min_obs=365, extraparameters=NULL)
```

You can then extract summar details about the topics using:

```
datadriven.defs <- topicEval(datadriven.topics)
```

and then export the topics/groups to the file topic_location.JSON in the working directory by running:

```
topicJsonFormat(datadriven.defs, loc=file.path(getwd(), 'topic_location'))
```