Package ‘patientCluster’

May 2, 2016

Type Package

Title Performs patient clustering for patient cohorts using OHDSI CDM v5

Version 0.1

Date 2015-10-10

Author Jenna Reps

Maintainer Jenna Reps <reps12@hotmail.com>

Description More about what it does (maybe more than one line)

License What license is it under?

LazyData TRUE

Depends R (>= 3.2.2),
DatabaseConnector (>= 1.3.0),

Imports ggplot2,
gridExtra,
ff,
ffbase (>= 0.12.1),
plyr,
Rcpp (>= 0.11.2),
RJDBC,
SqlRender (>= 1.1.3),
reshape2,
dplyr,
h2o (>= 3.6.0.3)

Suggests OhdsiRTools

RoxygenNote 5.0.0

R topics documented:

clusterConcepts *clusterConcepts*

Description

Create topics by clusting condition_concept_ids based on ingredience counts

Usage

```
clusterConcepts(dbconnection, cdmDatabaseSchema = NULL, method = "kmeans",
  clusterSize = 10, topicSize = NULL, scale = T,
  covariatesToInclude = NULL, indications = T, dayStart = 1,
  dayEnd = 30, use_min_obs = TRUE, min_obs = 100,
  extraparameters = NULL, updateProgress = NULL, ...)
```

Arguments

`dbconnection` using DatabaseConnector - connect to cdm database

`cdmDatabaseSchema` - cdm schema used to extract data from

`method` class:character - method used to do clustering (currently only supports kmeans)

`clusterSize` class:numeric - number of clusters returned,

`topicSize` class:numeric - number of topics in glrm

`scale` class:boolean - whether to use ingredience percentage scale for clustering

`covariatesToInclude` class:character vector - features to include: default NULL

`indications` class:boolean extract drug indicator features;Default TRUE

`dayStart` class:integer number of days relative to condition_concept_code to start looking for drugs

`dayEnd` class:integer number of days relative to condition_concept_code to stop looking for drugs

`use_min_obs` class:boolean whether to remove ingredient features that are rare

`min_obs` clkass:integer threshold used when use_min_obs is TRUE to determine what is rare

Value

list contining definition data.frame containing columes for concept_id, covariate (cluster id)

Examples

```
clusterConcepts()
```

clusterEval *Summarise the differences between the clusters*

Description

This function simply calculates summaries of each cluster and returns these as a list.

Usage

```
clusterEval(clusterResult)
```

Arguments

`clusterResult`	A list of class 'clusterResult' return by running clusterPeople()

Details

This function only has one input, the clusterResults obtained by applying clusterPeople

Value

A list containing:

`clusterMeans`	A data frame containing the mean of each feature per cluster
`clusterSds`	A data frame containing the standard deviation of each feature value per cluster
`clusterFrac`	A data frame containing the fraction of each cluster with non-zero values for the feature

Examples

```
# set database connection
dbconnection <- DatabaseConnector::createConnectionDetails(dbms = dbms,server = server,
user = user,password = pw,port = port,schema = cdmDatabaseSchema)

# then extract the data - in thie example using default groups
clusterData <- dataExtract(dbconnection, cdmDatabaseSchema,
cohortDatabaseSchema=cdmDatabaseSchema,
workDatabaseSchema='scratch.dbo',
cohortid=2000006292, agegroup=NULL, gender=NULL,
type='group', groupDef = 'default',
historyStart=1,historyEnd=365,  loc=getwd())

# initialise the h2o cluster
h2o.init(nthreads=-1, max_mem_size = '50g')

# cluster the males aged between 30 and 50 into 15 clusters
clusterPeople <- clusterRun(clusterData, ageSpan=c(30,50), gender=8507,
                         method='kmeans', clusterSize=15,
                         normalise=F, binary=F,fraction=T)

# get the summary details of each cluster:
clusterSum <- clusterEval(clusterResult)
```

clusterPeople *Runs kmeans or generalised low rank models on the cluster data*

Description

This function clusters patients into subgroups based on covariates corresponding to sets of concept_ids or concept_ids. It is recommended to use generalised low rank models to preprocess the data when clustering patients using individual concept_ids and reduce the dimensionality before applying k-means. When the data extraction used covariate groups kmeans can be run directly.

Usage

```
clusterPeople()

## Default S3 method:
clusterPeople(clusterData, ageSpan=c(0,100), gender=8507, method='kmeans',
              clusterSize=10, glrmFeat=NULL,normalise=T, binary=T,
              fraction=F, covariatesToInclude=NULL,covariatesToExclude=NULL,
              covariatesGroups=NULL, loc=loc)
```

Arguments

`minAge` class:numeric default(NULL)- the minimum age a person in the cohort must be to be included in the data

`maxAge` class:numeric default(NULL)- the maximum age a person in the cohort must be to be included in the data

`gender` class:numeric - gender concept_id (8507- male; 8532-female)

`method` class:character - method used to do clustering (currently only supports kmeans)

`clusterSize` class:numeric - number of clusters returned,

`glrmFeat` class:numeric - number of features engineered by generalised low rank model

`normalise` class:boolean - whether to center the data prior to clustering

`binary` class:boolean - whether to treat features as binary

`fraction` class:boolean - whether to treat features as fraction of total records

`covariatesToInclude`
class:character vector - features to include: default NULL

`covariatesToExclude`
class:character vector - features to exclude;Default NULL

`covariatesGroups`
class:covariatecluster result of clusterCovariate();Default NULL

`extraparameters`
- a list of parameters that can be used when adding a non default cluster method

`cohortid` class:numeric - id of cohort in cohort table

Details

This function performs kmeans clustering or general low rank model clustering on clusterData extraced from the CDM using dataExtract(). The user can specify a subset of the data based on ageSpan=c(lowerAgeLimit, upperAgeLimit) and gender=gender_concept_id and then the clustering method 'kmeans' or 'glrm' and the required cluster size: clusterSize=10.

When method 'kmeans' is chosen, the people are clustered using kmeans from the h2o package into clusterSize number of groups. When method 'glrm' is chosen, a glrm is run on the data to reduce the dimensionality to glrmFeat number of features and then kmeans is run on the reduced dimensionality data to cluster the people into clusterSize number of groups.

The data can be pre-processed using the normalise, binary and fraction variables. When normalise is TRUE then the data have the feature means subtracted and the result is divided by the feature standard deviation. When binary is TRUE, each feature for a person is set to 1 if the patient has the feature in the covariate list and 0 otherwise. When binary is set to FALSE the feature value is set to the number of concepts in the feature set that the patient has in the covariates list (e.g. if feature 1 consists of three concept_ids, 12, 1 and 304 and patient 1 has none of these concept_ids in the covariate list, he will have 0 in the feature 1 column, whereas if patient 2 has concept_id 12 and 304, she will have 2 in the feature 1 column). When fraction is TRUE then the features for each patient are scaled by dividing by the total sum of the patient's feature values (e.g. if patient 1 has value 3 for feature 5, value 1 for feature 10 and 0 for all other features then if fraction =TRUE this will be scale to 3/4 for feature 5 and 1/4 for feature 10).

The user can also specify covariates to include/exclude from the clustering by specifying the covariate_ids in a vector, for example setthing covariatesToInclude=c(1,3,10,45) will cluster the data using only the four specified covariates whereas setting covariatesToExclude=c(1,3,10,45) will exclude the specified covariates from the clustering.

Value

A list is returned of class 'clusterResult' containing:

`strata`	An ffdf containing the row_id (unique reference of the person), their age and gender
`covariates`	An ffdf containing the covariates each person has in sparse format
`covariateRef`	An ffdf containing the description of each covariate
`clusters`	A data frame containing the cluster allocated for each row_id
`centers`	A data frame containing the cluster centers returned by the kmeans algorithm
`metadata`	A list containing the information about the paramaters set to extract the data and do the clustering
`newData`	An ffdf containing the reduced dimensionality data returned when glrm preprocessing is done
`features`	An ffdf containing the clustering of the original covariates by glrm

Author(s)

Jenna Reps

References

todo...

Examples

```
# set database connection
dbconnection <- DatabaseConnector::createConnectionDetails(dbms = dbms,server = server,
user = user,password = pw,port = port,schema = cdmDatabaseSchema)

# then extract the data - in thie example using default groups
clusterData <- dataExtract(dbconnection, cdmDatabaseSchema,
cohortDatabaseSchema=cdmDatabaseSchema,
workDatabaseSchema='scratch.dbo',
cohortid=2000006292, agegroup=NULL, gender=NULL,
type='group', groupDef = 'default',
historyStart=1,historyEnd=365,  loc=getwd())

# initialise the h2o cluster
h2o.init(nthreads=-1, max_mem_size = '50g')

# cluster the males aged between 30 and 50 into 15 clusters
clusterPeople <- clusterRun(clusterData, minAge=30, maxAge=50, gender=8507,
                         method='kmeans', clusterSize=15,
                         normalise=F, binary=F,fraction=T)
```

clusterVisual *Plots the different cluster visulisations*

Description

Plots barcharts of each cluster's center and saves to the directory specified by the user

Usage

```
clusterVisual(clusterResult, saveLoc = getwd())
```

Arguments

`saveLoc`	class:character - directory where the results of the clustering are saved
`clusterresult`	output from applying clusterPeople()

Details

This function only has two inputs, the clusterResults obtained by applying clusterPeople and the location to save the plots.

Examples

```
clusterVisual(clusterResult, saveLoc='C:/Documents')
```

dataExtract	*This extracts the history features for each person in the cohort with the specific age/gender*

Description

This function connects to the CDM and constructs the data used to do the clustering - this is either condition_concept_ids that are recorded during the defined time period relative to the cohort start date for each person in the cohort or covariate concept_sets that are specified by using the 'default' grouping or inputing a dataframe with columns: definition and concept_id specifiying the concept_ids that make up each covariate definition, see examples below.

Usage

```
dataExtract(dbconnection = NULL, cdmDatabaseSchema = NULL,
  cohortDatabaseSchema = NULL, cohortid = 100, ageMin = NULL,
  ageMax = NULL, gender = NULL, type = "group", groupDef = "default",
  historyStart = 1, historyEnd = 180, ffloc = NULL, debug = NULL, ...)
```

Arguments

`dbconnection`	class:connectionDetails - the database connection details requires Library(DatabaseConnector)
`cdmDatabaseSchema`	class:character - database schema containing cdm tables
`cohortDatabaseSchema`	class:character - database schema containing cohort
`cohortid`	class:numeric - id of cohort in cohort table
`gender`	class:numeric - gender concept_id (8507- male; 8532-female)
`type`	class:character - features used by clustering (condition i.e. all condition_concept_ids or group i.e. concept sets),
`groupDef`	class:dataframe - a dataframe containing covariate concept_sets - must have the columns definition and concept_id
`historyStart`	class:numeric days prior to index to start searching person records for features
`historyEnd`	class:numeric days prior to index to stop searching person records for features
`ffloc`	class:character - specifies the directory where the ff files are stored
`debug`	class:character - default(NULL) otherise specifies the directory where the main SQL for extraction is written to for debugging
`minAge`	class:numeric default(NULL)- the minimum age a person in the cohort must be to be included in the data
`maxAge`	class:numeric default(NULL)- the maximum age a person in the cohort must be to be included in the data

Value

clusterData class:clusterData - a list containing:

`strata`	an ffdf containing the age/gender/row_id for each person in the cohort
`covariates`	an ffdf containing the covariates for each person in the cohort
`covariateRef`	an ffdf containing details about the covariates
`metadata`	a list containing details about the data extraction

See Also

DatabaseConnector, OhdsiRTools, SqlRender, ggplot2, reshape2, dplyr, plyr

Examples

```
# to extract the males ages between 30 and 45 in cdm_test.dbo.cohort with id 21
# and find whether they have the default concept definitions 1 to 60 days prior
# to cohort start:
dbconnection <- DatabaseConnector::createConnectionDetails(dbms = dbms,server = server,
user = user,password = pw,port = port,schema = cdmDatabaseSchema)

data <- dataExtract(dbconnection, cdmDatabaseSchema='cdm_test.dbo',
                    cohortDatabaseSchema='cdm_test.dbo', cohort_id=21,
                    minAge = 30, maxAge=45, gender=8507,
                    type='group', groupDef='default',
                    historyStart=1,historyEnd=60,
                    ffloc='C:fftemps')

# to extract the cluster data using user specified concept sets:
# where definition 1 contains concept_ids: 101,32011,1 and 63434
#       definition 2 contains concept_ids: 12,13
#       definition 3 contains concept_ids: 450453,21435324,232,3424,4534435 and 3453
groupDef <- data.frame(covariate=c(1,1,1,1,2,2,3,3,3,3,3,3),
                       concept_id =c(c(101,32011,1,63434), c(12,13),
                                     c(450453,21435324,232,3424,4534435,3453))
data <- dataExtract(dbconnection, cdmDatabaseSchema='cdm_test.dbo',
                    cohortDatabaseSchema='cdm_test.dbo', cohort_id=21,
                    type='group', groupDef=groupDef,
                    historyStart=1,historyEnd=180,
                    ffloc='C:fftemps')
```

developPredictionModels

Develop prediction model to predict cluster probabilities

Description

This function creates a binary logistic regression model for each cluster

Usage

```
developPredictionModels(covariates, cluster)
```

Arguments

covariates	An ffdf containing the covariates and values
cluster	A dataframe containing row_ids and cluster columns

Details

This function trains a predictive model using the input covariates to predict cluster membership

Value

A list containing:

`predictionModels`	A named list containing the trained models for each cluster (the names are the clusters predicted by the models)

`loadClusterData` *Load the cluster data from a folder*

Description

`loadClusterData` loads an object of type `clusterData` from a folder in the file system.

Usage

```
loadClusterData(file, readOnly = FALSE)
```

Arguments

`file`	The name of the folder containing the data.
`readOnly`	If true, the data is opened read only.

Details

The data will be written to a set of files in the folder specified by the user.

Value

An object of class clusterData

Examples

```
# todo
```

`saveClusterData` *Save the clustering data to folder*

Description

`saveClusterData` saves an object of type clusterData to folder.

Usage

```
saveClusterData(cData, file, overwrite = F)
```

Arguments

`cData` An object of type `clusterData` as generated using `dataExtract`.

`file` The name of the folder where the data will be written. The folder should not yet exist.

Details

The data will be written to a set of files in the folder specified by the user.

Examples

```
# todo
```

`topicEval` *topicEval*

Description

This function simply calculates summaries of each topic and returns these as a list.

Usage

```
topicEval(clust.res, threshold = NULL)
```

Arguments

`clust.res` A list return by running clusterConcepts()

Details

This function only has one input, the clusterResults obtained by applying clusterConcepts

Value

A list containing:

`topicsOrdered` A data frame containing the mean of each feature per cluster

`topicsMax` A data frame containing the standard deviation of each feature value per cluster

`topicsKmeans` A data frame containing the fraction of each cluster with non-zero values for the feature

Index