

Implementing Existing Prediction Models using the OHDSI PatientLevelPrediction Framework

Jenna Reys, Martijn J. Schuemie, Patrick B. Ryan, Peter R. Rijnbeek

2018-10-07

Contents

1	Introduction	1
2	Model implementation	1
2.1	Define the model	1
2.2	Create the model	3
2.3	Run the model	3
3	Acknowledgments	5

1 Introduction

This vignette describes how you can implement existing logistic regression models in the Observational Health Data Sciences (OHDSI) `PatientLevelPrediction` framework. This allows you to for example externally validate them at scale in the OHDSI data network.

As an example we are going to implement the CHADS2 model as described in:

Gage BF, Waterman AD, Shannon W, Boehler M, Rich MW, Radford MJ. Validation of clinical classification schemes for predicting stroke: results from the National Registry of Atrial Fibrillation. *JAMA*. 2001 Jun 13;285(22):2864-70

To implement the model you need to create three tables: the model table, the covariate table, and the intercept table. The model table specifies the `modelId` (sequence number), the `modelCovariateId` (sequence number) and the `covariateValue` (beta for the covariate). The covariate table specifies the mapping between the covariates from the published model and the standard covariates, i.e. it maps to a combination of an `analysisid` and a `concept_id` (see below). The intercept table specifies per `modelId` the intercept.

2 Model implementation

2.1 Define the model

The CHADS2 is a score based model with:

##	Points	Covariate
## 1	1	Congestive heart failure
## 2	1	Hypertension
## 3	1	Age >= 75 years
## 4	1	Diabetes mellitus
## 5	2	Stroke/transient ischemic attack

The model table should therefore be defined as:

```
##   modelId modelCovariateId covariateValue
## 1      1          1          1
## 2      1          2          1
## 3      1          3          1
## 4      1          4          1
## 5      1          5          2
```

The covariateTable will then specify what standard covariates need to be included in the model.

In this case we choose the following Standard SNOMED concept_ids: 319835 for congestive heart failure, 316866 for hypertensive disorder, 201820 for diabetes, and 381591 for cerebrovascular disease. It is allowed to add multiple concept_ids as separate rows for the same modelCovariateId if concept sets are needed. These concept_ids can be found using the vocabulary search in ATLAS.

The standard covariates are of the form: conceptid*1000 + analysisid. The analysisid specifies the domain of the covariate and its lookback window. Examples can be found here: <https://github.com/OHDSI/FeatureExtraction/blob/master/inst/csv/PrespecAnalyses.csv>

Our example of CHADS2 uses agegroup and conditions in the full history. Therefore we need to define the standard covariates using the FeatureExtraction::createCovariateSettings as follows:

```
library(PatientLevelPrediction)
covSet <- FeatureExtraction::createCovariateSettings(useDemographicsAgeGroup = T,
                                                    useConditionOccurrenceLongTerm = T,
                                                    includedCovariateIds = NULL,
                                                    longTermStartDays = -9999,
                                                    endDays = 0)
```

In the above code we used the useConditionOccurrenceLongTerm (these have an analysis id of 102) and we defined the longTermStartDays to be -9999 days relative to index (so we get the full history). We include the index date in our lookback period by specifying endDays = 0. The includeCovariateIds is set to NULL here, but this will be updated automatically later on. As we picked analysis id 102, the standard covariate for anytime prior congestive heart failure is 319835102. The same logic follows for the other conditions, so the covariate table will be:

```
##   modelCovariateId covariateId
## 1          1      319835102
## 2          2      316866102
## 3          3         15003
## 4          3         16003
## 5          3         17003
## 6          3         18003
## 7          3         19003
## 8          4      201820102
## 9          5      381591102
```

modelCovariateId 3 was age >= 75, as the standard covariate age groups are in 5 year groups, we needed to add the age groups 75-80, 80-85, 85-90, 90-95 and 95-100, these correspond to the covariateIds 15003, 16003, 17003, 18003 and 19003 respectively.

To create the tables in R for CHADS2 you need to make the following dataframes:

```
model_table <- data.frame(modelId = c(1,1,1,1,1),
                          modelCovariateId = 1:5,
                          coefficientValue = c(1, 1, 1, 1, 2)
                          )

covariate_table <- data.frame(modelCovariateId = c(1,2,3,3,3,3,3,4,5),
                              covariateId = c(319835102, 316866102,
```

```
15003, 16003, 17003, 18003, 19003,
201820102, 381591102)

)

interceptTable <- data.frame(modelId = 1,
                             interceptValue = 0)
```

2.2 Create the model

Now you have everything in place to actually create the existing model. First specify the current environment as executing `createExistingModelSql` creates two functions for running the existing model into the specified environment. You need to specify the type of model (either logistic or score), in our example we are calculating a score. We finally need to specify the `analysisId` for the newly created CHADS2 covariate.

```
e <- environment()
PatientLevelPrediction::createExistingModelSql(modelTable = model_table,
                                              modelNames = 'CHADS2',
                                              interceptTable = data.frame(modelId = 1, interceptValue = 0),
                                              covariateTable = covariate_table,
                                              type = 'score',
                                              analysisId = 112, covariateSettings = covSettings, e = e)
```

Once run you will find two new functions in your environment:

- `createExistingmodelsCovariateSettings()`
- `getExistingmodelsCovariateSettings()`

2.3 Run the model

Now you can use the functions you previously created to extract the existing model risk scores for a target population:

```
plpData <- PatientLevelPrediction::getPlpData(connectionDetails,
                                              cdmDatabaseSchema = 'databasename.dbo',
                                              cohortId = 1,
                                              outcomeIds = 2,
                                              cohortDatabaseSchema = 'databasename.dbo',
                                              cohortTable = 'cohort',
                                              outcomeDatabaseSchema = 'databasename.dbo',
                                              outcomeTable = 'cohort',
                                              covariateSettings = createExistingmodelsCovariateSettings(),
                                              sampleSize = 20000
                                              )
```

To implement and evaluate an existing model you can use the function:

```
PatientLevelPrediction::evaluateExistingModel()
```

with the following parameters:

- `modelTable` - a data.frame containing the columns: `modelId`, `covariateId` and `coefficientValue`
- `covariateTable` - a data.frame containing the columns: `covariateId` and `standardCovariateId` - this provides a set of `standardCovariateId` to define each model covariate.
- `interceptTable` - a data.frame containing the columns `modelId` and `interceptValue` or `NULL` if the model doesn't have an intercept (equal to zero).

- type - the type of model (either: score or logistic)
- covariateSettings - this is used to determine the startDay and endDay for the standard covariates
- customCovariates - a data.frame with the covariateId and sql to generate the covariate value.
- riskWindowStart - the time at risk starts at target cohort start date + riskWindowStart
- addExposureDaysToEnd - if true then the time at risk window ends at the cohort end date + riskWindowEnd rather than cohort start date + riskWindowEnd
- riskWindowEnd - the time at risk ends at target cohort start/end date + riskWindowStart
- requireTimeAtRisk - whether to add a constraint to the number of days observed during the time at risk period in including people into the study
- minTimeAtRisk - the minimum number of days observation during the time at risk a target population person needs to be included
- includeAllOutcomes - Include outcomes even if they do not satisfy the minTimeAtRisk? (useful if the outcome is associated to death or rare)
- removeSubjectsWithPriorOutcome - remove target population people who have the outcome prior to the time at risk period?
- connectionDetails - the connection to the CDM database

Finally you need to add the settings for downloading the new data:

- cdmDatabaseSchema
- cohortDatabaseSchema
- cohortTable
- cohortId
- outcomeDatabaseSchema
- outcomeTable
- outcomeId
- oracleTempSchema

To run the external validation of an existing model where the target population are those in the cohort table with id 1 and the outcome is those in the cohort table with id 2 and we are looking to predict first time occurrence of the outcome 1 day to 365 days after the target cohort start date (assuming you have the modelTable, covariateTable and interceptTable in the format explained above):

```
# in our example the existing model uses gender and condition groups looking back 200 days:
covSet <- FeatureExtraction::createCovariateSettings(useDemographicsGender = T,
                                                    useConditionGroupEraMediumTerm = T,
                                                    mediumTermStartDays = -200)

result <- evaluateExistingModel(modelTable = modelTable,
                               covariateTable = covariateTable,
                               interceptTable = NULL,
                               type = 'score',
                               covariateSettings = covSet,
                               riskWindowStart = 1,
                               addExposureDaysToEnd = F,
                               riskWindowEnd = 365,
                               requireTimeAtRisk = T,
                               minTimeAtRisk = 364,
                               includeAllOutcomes = T,
                               removeSubjectsWithPriorOutcome = T,
                               connectionDetails = connectionDetails,
                               cdmDatabaseSchema = 'databasename.dbo',
                               cohortId = 1,
                               outcomeId = 2,
                               cohortDatabaseSchema = 'databasename.dbo',
                               cohortTable = 'cohort' ,
```

```
outcomeDatabaseSchema = 'databasename.dbo',  
outcomeTable = 'cohort'  
)
```

Result will contain the performance and the predictions made by the model.

3 Acknowledgments

Considerable work has been dedicated to provide the PatientLevelPrediction package.

```
citation("PatientLevelPrediction")
```

```
##  
## Jenna Reps, Martijn J. Schuemie, Marc A. Suchard, Patrick B.  
## Ryan and Peter R. Rijnbeek (2018). PatientLevelPrediction:  
## Package for patient level prediction using data in the OMOP  
## Common Data Model. R package version 3.0.0.  
##  
## A BibTeX entry for LaTeX users is  
##  
## @Manual{,  
## title = {PatientLevelPrediction: Package for patient level prediction using data in the OMOP Com  
## Model},  
## author = {Jenna Reps and Martijn J. Schuemie and Marc A. Suchard and Patrick B. Ryan and Peter R.  
## year = {2018},  
## note = {R package version 3.0.0},  
## }
```

Please reference this paper if you use the PLP Package in your work:

Reps JM, Schuemie MJ, Suchard MA, Ryan PB, Rijnbeek PR. Design and implementation of a standardized framework to generate and evaluate patient-level prediction models using observational healthcare data. J Am Med Inform Assoc. 2018;25(8):969-975.

This work is supported in part through the National Science Foundation grant IIS 1251151.