Automatically Build Multiple Patient-Level Predictive Models

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

2020-06-03

Contents

1 Introduction

In our `paper`, we propose a standardised framework for patient-level prediction that utilizes the OMOP CDM and standardized vocabularies, and describe the open-source software that we developed implementing the framework's pipeline. The framework is the first to enforce existing best practice guidelines and will enable open dissemination of models that can be extensively validated across the network of OHDSI collaborators.

One our best practices is that we see the selection of models and all study setting as an emperical question, i.e. we should use a data-driven approach in which we try many settings. This vignette describes how you can use the Observational Health Data Sciencs and Informatics (OHDSI) `PatientLevelPrediction` package to automatically build multiple patient-level predictive models, e.g. different population settings, covariate settings, and modelsetting. This vignette assumes you have read and are comfortable with building single patient level prediction models as described in the `BuildingPredictiveModels` vignette.

Note that it is also possible to generate a Study Package directly in Atlas that allows for multiple patient-level prediction analyses this is out-of-scope for this vignette.

2 Creating the setting lists

To develop multiple models the user has to create a list of Study Populations Settings, Covariate Settings, and Model Settings. These lists will then be combined in a Model Analysis List and all combinations of the elements in this list will be automatically run by the package.

2.1 Study population settings

Suppose we like to make the following three population settings:

- study population 1: allows persons who have the outcome to leave the database before the end of time-at-risk and only those without the outcome who are observed for the whole time-at-risk period (requireTimeAtRisk = T).
- study population 2: does not impose the restriction that persons who do not experience the outcome need to be observed for the full time-at-risk period (requireTimeAtRisk = F).
- study population 3: does impose the restriction that persons who do not experience the outcome need to be observed for the full time-at-risk period (requireTimeAtRisk = T) and allows persons that had the outcome before (removeSubjectsWithPriorOutcome = F)

The create a study population setting list we use the function `createStudyPopulationSettings` as described below:

```
# define all study population settings
studyPop1 <- createStudyPopulationSettings(binary = T,
                                           includeAllOutcomes = F,
                                           removeSubjectsWithPriorOutcome = T,
                                           priorOutcomeLookback = 99999,
                                           requireTimeAtRisk = T,
                                           minTimeAtRisk=364,
                                           riskWindowStart = 1,
                                           riskWindowEnd = 365,
                                           verbosity = "INFO")

studyPop2 <- createStudyPopulationSettings(binary = T,
                                           includeAllOutcomes = F,
                                           removeSubjectsWithPriorOutcome = T,
                                           priorOutcomeLookback = 99999,
                                           requireTimeAtRisk = F,
                                           minTimeAtRisk=364,
                                           riskWindowStart = 1,
                                           riskWindowEnd = 365,
                                           verbosity = "INFO")

studyPop3 <- createStudyPopulationSettings(binary = T,
                                           includeAllOutcomes = F,
                                           removeSubjectsWithPriorOutcome = F,
                                           priorOutcomeLookback = 99999,
                                           requireTimeAtRisk = T,
                                           minTimeAtRisk=364,
                                           riskWindowStart = 1,
                                           riskWindowEnd = 365,
                                           verbosity = "INFO")

# combine these in a population setting list
populationSettingList <- list(studyPop1,studyPop2,studyPop3)
```

2.2 Covariate settings

The covariate settings are created using `createCovariateSettings`. We can create multiple covariate settings and then combine them in a list:

```
covSet1 <- createCovariateSettings(useDemographicsGender = T,
                                   useDemographicsAgeGroup = T,
                                   useConditionGroupEraAnyTimePrior = T,
                                   useDrugGroupEraAnyTimePrior = T)

covSet2 <- createCovariateSettings(useDemographicsGender = T,
                                   useDemographicsAgeGroup = T,
                                   useConditionGroupEraAnyTimePrior = T,
                                   useDrugGroupEraAnyTimePrior = F)

covariateSettingList <- list(covSet1, covSet2)
```

2.3 Algorithm settings

The model settings requires running the setModel functions for the machine learning algorithms of interest and specifying the hyper-parameter search and then combining these into a list. For example, if we wanted to try a logistic regression, gradient boosting machine and ada boost model then:

```
gbm <- setGradientBoostingMachine()
lr <- setLassoLogisticRegression()
ada <- setAdaBoost()

modelList <- list(gbm, lr, ada)
```

2.4 Model analysis list

To create the complete plp model settings use `createPlpModelSettings` to combine the population, covariate and model settings.

```
modelAnalysisList <- createPlpModelSettings(modelList = modelList,
                                   covariateSettingList = covariateSettingList,
                                   populationSettingList = populationSettingList)
```

3 Running multiple models

As we will be downloading loads of data in the multiple plp analysis it is useful to set the Andromeda temp folder to a directory with write access and plenty of space. `options(andromedaTempFolder = "c:/andromedaTemp")`

To run the study requires setting up a connectionDetails object

```
dbms <- "your dbms"
user <- "your username"
pw <- "your password"
server <- "your server"
port <- "your port"

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

Next you need to specify the cdmDatabaseSchema where your cdm database is found and workDatabaseSchema where your target population and outcome cohorts are and you need to specify a label for the database name:

a string with a shareable name of the database (this will be shown to OHDSI researchers if the results get transported).

```
cdmDatabaseSchema <- "your cdmDatabaseSchema"
workDatabaseSchema <- "your workDatabaseSchema"
cdmDatabaseName <- "your cdmDatabaseName"
```

Now you can run the multiple patient-level prediction analysis by specifying the target cohort ids and outcome ids

```
allresults <- runPlpAnalyses(connectionDetails = connectionDetails,
                             cdmDatabaseSchema = cdmDatabaseSchema,
                             cdmDatabaseName = cdmDatabaseName,
                             oracleTempSchema = cdmDatabaseSchema,
                             cohortDatabaseSchema = workDatabaseSchema,
                             cohortTable = "your cohort table",
                             outcomeDatabaseSchema = workDatabaseSchema,
                             outcomeTable = "your cohort table",
                             cdmVersion = 5,
                             outputFolder = "./PlpMultiOutput",
                             modelAnalysisList = modelAnalysisList,
                             cohortIds = c(2484,6970),
                             cohortNames = c('visit 2010','test cohort'),
                             outcomeIds = c(7331,5287),
                             outcomeNames =  c('outcome 1','outcome 2'),
                             maxSampleSize = NULL,
                             minCovariateFraction = 0,
                             normalizeData = T,
                             testSplit = "stratified",
                             testFraction = 0.25,
                             splitSeed = NULL,
                             nfold = 3,
                             verbosity = "INFO")
```

This will then save all the plpData objects from the study into "./PlpMultiOutput/plpData", the populations for the analysis into "./PlpMultiOutput/population" and the results into "./PlpMultiOutput/Result". The csv named settings.csv found in "./PlpMultiOutput" has a row for each prediction model developed and points to the plpData and population used for the model development, it also has descriptions of the cohorts and settings if these are input by the user.

Note that if for some reason the run is interrupted, e.g. because of an error, a new call to `RunPlpAnalyses` will continue and not restart until you remove the output folder.

4 Validating multiple models

If you have access to multiple databases on the same server in different schemas you could evaluate accross these using this call:

```
val <- evaluateMultiplePlp(analysesLocation = "./PlpMultiOutput",
                           outputLocation = "./PlpMultiOutput/validation",
                           connectionDetails = connectionDetails,
                           validationSchemaTarget = list('new_database_1.dbo',
                                                         'new_database_2.dbo'),
                           validationSchemaOutcome = list('new_database_1.dbo',
                                                          'new_database_2.dbo'),
                           validationSchemaCdm = list('new_database_1.dbo',
```

```
                                                    'new_database_2.dbo'),
                          databaseNames = c('database1','database2'),
                          validationTableTarget = 'your new cohort table',
                          validationTableOutcome = 'your new cohort table')
```

This then saves the external validation results in the validation folder of the main study (the outputLocation you used in runPlpAnalyses).

5 Viewing the results

To view the results for the multiple prediction analysis:

```
viewMultiplePlp(analysesLocation="./PlpMultiOutput")
```

If the validation directory in "./PlpMultiOutput" has results, the external validation will also be displayed.

6 Acknowledgments

Considerable work has been dedicated to provide the `PatientLevelPrediction` package.

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

```
##
## To cite PatientLevelPrediction in publications use:
##
## Reps JM, Schuemie MJ, Suchard MA, Ryan PB, Rijnbeek P (2018). "Design and
## implementation of a standardized framework to generate and evaluate patient-level
## prediction models using observational healthcare data." _Journal of the American
## Medical Informatics Association_, *25*(8), 969-975. <URL:
## https://doi.org/10.1093/jamia/ocy032>.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     author = {J. M. Reps and M. J. Schuemie and M. A. Suchard and P. B. Ryan and P. Rijnbeek},
##     title = {Design and implementation of a standardized framework to generate and evaluate patient-l
##     journal = {Journal of the American Medical Informatics Association},
##     volume = {25},
##     number = {8},
##     pages = {969-975},
##     year = {2018},
##     url = {https://doi.org/10.1093/jamia/ocy032},
##   }
```

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.