Making patient-level predictive network study packages

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

2018-10-07

Contents

1 Introduction

The OHDSI Patient Level Prediction (PLP) package provides the framework to implement prediction models at scale. This can range from developing a large number of models across sites (methodology and study design insight) to extensive external validation of existing models in the OHDSI PLP framework (model insight). This vignette describes how you can use the `PatientLevelPrediction` package to create a network study package.

2 Running a Network study Process

2.1 Step 1 – developing the study

- Design the study: target/outcome cohort logic, concept sets for medical definitions, settings for developing new model or validation of adding existing models to framework. Suggestion: look in literature for validated definitions.
- Write a protocol that motivates the study and provides full details (sufficient for people to replicate the study in the future).
- Write an R package for implementing the study across diverse computational environments [see guidance below for structure of package and use the skeleton github package here: ...]

2.2 Step 2 – implementing the study part 1

- Get contributors to install the package and dependencies. Ensure the package is installed correctly by running the checkInstall functions.
- Get contributors to run the createCohort function to inspect the target/outcome definitions. If the definitions are not suitable for a site, go back to step 1 and revise the cohort definitions.

2.3 Step 3 – implementing the study part 2 [make sure package checks outputs the package is functioning as planned and the definitions are valid across sites]

- Get contributors to run the main.R with the settings configured to their environment
- Get the contributors to submit the results

2.4 Step 4 – Publication

- The study creator has the first option to be first author, if he/she does not wish to be first author then he/she can pick the most suitable person from the contributors. All contributors will be listed as authors on the paper. The last author will be the person who lead/managed the study, if this was the first author then the first author can pick the most suitable last author. All authors between the first and last author will be alphabetical by last name.

3 Package Skeleton - File Structure

- DESCRIPTION – This file describes the R package and the dependencies
- NAMESPACE – This file is created automatically by Roxygen
- Readme.md – This file should provide the step by step guidance on implementing the package

- R
- helpers.r – all the custom functions used by the package should be in this file (e.g., checkInstall)
- main.r – this file will call the functions in helpers.r to execute the full study
- submit.r – this file will be called at the end the submit the compressed folder to the study creator/manager.
- Man – this folder will contain the documentation for the functions in helpers.r (this should be automatically generated by roxygen)
- Inst
- sql/sql_sever
  - targetCohort – the target cohort parameterised sql code
  - outcomeCohort – the outcome cohort parameterised sql code
- extdata – place any data required for the package here
- plp_models – place any PLP models here
- existing_models – place the files for existing models here
- Extras

4 Package Skeleton - Output of Running Package

The output should contain three folders inside the study directory such as `outputLoc <- (file.path(getwd(), paste0(studyName_database_date)))`: * Plots – containing the test/train or validation ROC plot, calibration plot, precision recall plot and optionally the demographic calibration plot. * Results – The output of running savePlpResult * Summary – a summary csv of performance and the table 1 csv

Then there should also be a zip file of the folder in the working directory containing the same folders and files but with sensitive results removed (this will be created using the packageResults function). Once the contributor has inspected the zipped file and is happy with the content being shared, he/she can then finally run the submit function with the details provided in the readme.md.

5 Example Code To Make Package For External Validation of PLP Model

First you need to make a copy of the PatientLevelPrediciton skeleton package found here:

Assuming you ran a sucessful PatientLevelPrediction model development and saved the output of `runPlp()` to to location 'goodModel' in your working directory then:

```
library(PatientLevelPrediction)
plpResult <- loadPlpResult("goodModel")

# add the model to the skeleton package with sensitive information removed
exportPlpResult(plpResult = plpResult, modelName = "Model Name", packageName = "Your Package Name",
    gitHubLocation = "location/of/github", includeEvaluationStatistics = T,
    includeThresholdSummary = T, includeDemographicSummary = T, includeCalibrationSummary = T,
    includePredictionDistribution = T, includeCovariateSummary = F)
```

Now you want to add the cohorts (generally the parameterized sql required to create one or more target and outcome cohorts). This should be added into the inst/sql/sql_server directory of your package. If you are using atlas to create the cohorts then you can use: `OhdsiRTools::insertCirceDefinitionInPackage()`. The settings for the cohort creation are defined in the inst/extdata directory in the file cohort_details.csv. this file contains two columns: cohortName and cohortId. The cohortName should contain the name of the sql file of the cohort in inst/sql/sql_server (e.g., a file called "targetCohort.sql" has the name "targetCohort") and the cohortId is the default cohort_definition_id that will be used when people run the study corresponding

to this cohort. The main.R file in the extras directory contains the vanilla code to run a study with the model eported into the package and the cohort files added.

```
library(PatientLevelPrediction)
# input settings for person running the study
connectionDetails <- " "
cdmDatabaseSchema <- "their_cdm_database"
databaseName <- "Name for database"
cohortDatabaseSchema <- "a_database_with_write_priv"
cohortTable <- "package_table"
outputLocation <- "location to save results"

cohortDetails <- createCohort(connectionDetails = connectionDetails, cdmDatabaseSchema = cdmDatabaseSch
    cohortDatabaseSchema = cohortDatabaseSchema, cohortTable = cohortTable,
    package = "Your Package Name")

plpResult <- loadPlpResult(system.file("model", package = "Your Package Name"))
result <- externalValidatePlp(plpResult = plpResult, connectionDetails = connectionDetails,
    validationSchemaTarget = cohortDatabaseSchema, validationSchemaOutcome = cohortDatabaseSchema,
    validationSchemaCdm = cdmDatabaseSchema, validationTableTarget = cohortTable,
    validationTableOutcome = cohortTable, validationIdTarget = target_cohort_id,
    validationIdOutcome = outcome_cohort_id)

# save results to standard output
resultLoc <- standardOutput(result = result, outputLocation = outputLocation,
    studyName = "external validation of ... model", databaseName = databaseName,
    cohortName = "your cohortName", outcomeName = "your outcomeName")

# package results ready to submit
packageResults(mainFolder = resultLoc, includeROCplot = T, includeCalibrationPlot = T,
    includePRPlot = T, includeTable1 = F, includeThresholdSummary = T, includeDemographicSummary = T,
    includeCalibrationSummary = T, includePredictionDistribution = T, includeCovariateSummary = F,
    removeLessThanN = F, N = 10)
```

Where the target_cohort_id and outcome_cohort_id should correspond to the cohort_details.csv file.

We recommend getting the network implementors to submit their results of `createCohort()` before continuing with the study to ensure definitions run across the network. After running the rest of main.R the implementor should inspect the files in the export folder created by the package to ensure there isn't sensitive data remaining. Once checked the implementor can run submit.R to send the results to the study organisor. The submit.R file is:

```
submitResults(exportFolder = outputLocation, dbName = databaseName, key, secret)
```

6 Example Code To Make Package For External Validation of Existing Model Converted to PLP framework

First you need to make a copy of the PatientLevelPrediciton skeleton package found here:

Add the modelTable, covariateTable and interceptTable into inst/extdata as csv files as described in the AddingExistingModels.pdf. Now you want to add the cohorts (generally the parameterized sql required to create one or more target and outcome cohorts). This should be added into the inst/sql/sql_server directory of your package. If you are using atlas to create the cohorts then you can use: `OhdsiRTools::insertCirceDefinitionInPackage()`. The settings for the cohort creation are defined in the

inst/extdata directory in the file cohort_details.csv. this file contains two columns: cohortName and cohortId. The cohortName should contain the name of the sql file of the cohort in inst/sql/sql_server (e.g., a file called "targetCohort.sql" has the name "targetCohort") and the cohortId is the default cohort_definition_id that will be used when people run the study corresponding to this cohort. The main.R file in the extras directory contains the vanilla code to run a study with the model eported into the package and the cohort files added.

```
library(PatientLevelPrediction)
library(FeatureExtraction)
# input settings for person running the study
connectionDetails <- " "
cdmDatabaseSchema <- "their_cdm_database"
databaseName <- "Name for database"
cohortDatabaseSchema <- "a_database_with_write_priv"
cohortTable <- "package_table"
outputLocation <- "location to save results"

cohortDetails <- createCohort(connectionDetails = connectionDetails, cdmDatabaseSchema = cdmDatabaseSch
    cohortDatabaseSchema = cohortDatabaseSchema, cohortTable = cohortTable,
    package = "Your Package Name")

# load the model details and insert into function to run exisitng model
modelTable <- loadPlpResult(system.file("extdata/modelTable", package = "Your Package Name"))
covariateTable <- loadPlpResult(system.file("extdata/covariateTable", package = "Your Package Name"))
interceptTable <- loadPlpResult(system.file("extdata/interceptTable", package = "Your Package Name"))
# enter the model type - e.g., score/logistic
type <- "score"
# add the covariates used in the covariateTable and the temporal information
# e.g., if you used age, gender and mediumTermLookback conditionEraGroups
# where you want 100 day lookback
covariateSettings <- createCovariateSettings(useDemographicsGender = T, useDemographicsAge = T,
    useConditionGroupEraMediumTerm = T, mediumTermStartDays = -100)
result <- evaluateExistingModel(modelTable = modelTable, covariateTable = covariateTable,
    interceptTable = interceptTable, type = type, covariateSettings = covariateSettings,
    riskWindowStart = 1, riskWindowEnd = 365, requireTimeAtRisk = T, minTimeAtRisk = 364,
    includeAllOutcomes = T, removeSubjectsWithPriorOutcome = T, connectionDetails = connectionDetails,
    cdmDatabaseSchema = cdmDatabaseSchema, cohortDatabaseSchema = cohortDatabaseSchema,
    outcomeDatabaseSchema = cohortDatabaseSchema, cohortTable = cohortTable,
    outcomeTable = cohortTable, cohortId = target_cohort_id, outcomeId = outcome_cohort_id)

# save results to standard output
resultLoc <- standardOutput(result = result, outputLocation = outputLocation,
    studyName = "external validation of ...", databaseName = databaseName, cohortName = "your cohortNam
    outcomeName = "your outcomeName")

# package results ready to submit
packageResults(mainFolder = resultLoc, includeROCplot = T, includeCalibrationPlot = T,
    includePRPlot = T, includeTable1 = F, includeThresholdSummary = T, includeDemographicSummary = T,
    includeCalibrationSummary = T, includePredictionDistribution = T, includeCovariateSummary = F,
    removeLessThanN = F, N = 10)
```

Where the target_cohort_id and outcome_cohort_id should correspond to the cohort_details.csv file.

Finally get the implementor to check the export file created by the code into te outputLocation and to submit the results but running:

```
submitResults(exportFolder = outputLocation, dbName = databaseName, key, secret)
```

7 Useful PatientLevelPrediction Functions

The functions to aid the creation of a network study are:

Function	Description	Usage
`checkPlpInstallation()`	This function checks the connection, and various aspects of the PLP package to check it is set up correctly	This should be run with the appropriate settings to check the contributor is set up correctly for the study
`createCohortTables()`	This function requires a connection, cohort ids and cdm/work database details then creates any cohort in the inst/sql/sql_server. It then summarises the cohorts.	This should be used to create and check any target/outcome cohorts for the study. The cohort summary should be emailed to the study creator.
`getPlpData()`	This function extracts the data from the cdm for model development	This should be used if developing new models
`runPlp()`	This function trains and tests a new PLP model	This should be used if developing new models

Function	Description	Usage
`exportPlpResults()`	This function exports the output of runPlp into an R package while removing sensitive objects	This should be used when saving a model into a study package to validate the model
`externalValidatePlp()`	This function requires the user to inpute an existing model and then extracts the required data on a new database and applies/evaluates the model.	This should be used if validating a PLP model
`evaluateExistingModel()`	This function enables existing models to be added into the PLP framework	This should be used if validating an existing model by converting it into the PLP framework.
`getTable1()`	This function creates the table 1 characteristic for journal papers	This should be used to characterise the data for any study
`standardOutput()`	This function saves the model development/evaluation as the standard package output	This should be used to save the study results
`packageResults()`	This function takes a folder with standard output as input, removes sensitive data and moves the remaining data into a zipped file	This should be used to package up the study results

Function	Description	Usage
`submitResults()`	This sends a zipped file to the OHDSI amazon repository	This should be run after a study (once the zipped file has been checked) to submit the results for a data site to the study creator