Creating A Prediction Study Package in R

Jenna M. Reps

2021-03-26

Contents

1 Introduction

ATLAS can be used to design a prediction study and create an R package. However, sometimes you may wish to add extra features that are not available in ATLAS. In this guide you will find insructions for designing and creating R packages outside of ATLAS (with additional features).

The additional features include:

- Ability to use cohort features, measurements and more advanced age features (or any custom feature)
- Ability to specify which Outcomes should be paired with each population setting (useful if some outcomes require a 1 year time-at-risk but others require a 30-day time-at-risk)
- Ability to specify which models should be paired with each covariate setting (useful if you have temporal feature models and non-temporal feature models)

2 Setting up

- Make sure you have PatientLevelPrediction installed and all dependancies (see installation guide in PatientLevelPrediction)
- Install the dependencies for the skeleton (source the file at: https://github.com/OHDSI/SkeletonPredictionStudy/blob/master/extras/packageDeps.R)
- Install Hydra (https://github.com/OHDSI/Hydra) - this package will use the settings json object you create and generate a Study Package to develop the specified models.

3 Creating Study Package

3.1 Step 1: Designing the prediction study

The components that are needed are:

- The target populations (this is the ATLAS Ids for the target cohorts and the corresponding targetName)

```
targets <- data.frame(targetId = c(16425,16377,16387),
                      cohortId = c(16425,16377,16387),
                      targetName = c('diabetes','liver issues','fall'))
```

- The outcome definitions (this is the ATLAS Ids for the outcome cohorts and the corresponding outcomeName)

```
outcomes <- data.frame(outcomeId = c(16428,16435),
                       cohortId = c(16428,16435),
                       outcomeName = c('congestive heart failure','afib'))
```

We currently only support ATLAS cohorts as the corresponding json specifications will be inserted into the study package so people can create these when executing the study. If you want to add custom SQL that is possible but will require manually editing the generated study package.

- The population settings that specify the time-at-risk, this must be a list and can contain multiple populations settings:

```
populationSettings <- list(PatientLevelPrediction::createStudyPopulationSettings(riskWindowEnd = 365),
                           PatientLevelPrediction::createStudyPopulationSettings(riskWindowEnd = 730))
```

- The model settings list, this is a list of all the prediction model settings:

```
modelList <- list(PatientLevelPrediction::setLassoLogisticRegression(),
                  PatientLevelPrediction::setAdaBoost())
```

- The covariate settings - a list of the covariate setting options (if you want multiple covariate settings per plpData then you can create a lists of settings). In the example below we create two covariate settings, the first settings contains standard FeatureExtraction features and a cohort covariate, the second setting contains standard FeatureExtracion features only. You must specify the function name that is used to create the features and the settings for that function. It is possible to add any custom function here as long as you add it into the study package:

```
covariateSettings <- list(list(list(fnct = 'createCovariateSettings',
                                    settings = FeatureExtraction::createCovariateSettings(useDemographi
                               list(fnct = 'createCohortCovariateSettings',
                                    settings = list(covariateName = 'test', covariateId = 16442956,
                                                    cohortId = 16442,
                                                    startDay=-30, endDay=0, count=F,
                                                    ageInteraction = F))),
                          list(fnct = 'createCovariateSettings',
                               settings = FeatureExtraction::createCovariateSettings(useDemographicsGen
)
```

- settings to restrict certain populations to an outcome

```
resrictOutcomePops <- data.frame(outcomeId = c(16428,16435),
                                 populationSettingId = c(1,2))
```

or certainb models to covariate settings.

```
resrictModelCovs = data.frame(modelSettingId = c(1,1,2),
                              covariateSettingId = c(1,2,1))
```

3.2 Step 2: Creating the JSON

For the json creation you will need to source the code found at: https://github.com/OHDSI/SkeletonPredictionStudy/blob/master/extras/createPredSkelJson.R to load the function 'createDevelopmentStudyJson'.

- The webApi for the ATLAS that contains the target and outcome cohorts (the cohort jsons will be extarcted using a get request)

```
webApi <- 'http://yourwebapi.com'
```

Settings for the study package - packageName - a string that will be the name of the R package e.g., 'exampleStudy' - packageDescription - a string describing the study e.g., 'an example of the skeleton' - createdBy - a string specifying who created the study 'add name' - organizationName - a string specifying the organization of the study creator 'add organization' - outputLocation - a location where the json settings will be saved e.g., 'D:/testing/' - jsonName - a string specifying the name of the json settings when it is saved e.g., 'predictionAnalysisList.json'

You can then use these components to generate the json settings that specify the prediction study by running (this will return the json object and also save it to a file named inside where these <> are specified by the user:

```
json <- createDevelopmentStudyJson(packageName = 'exampleStudy',
                           packageDescription = 'an example of the skeleton',
                           createdBy = 'add name',
                           organizationName = 'add organization',
                           targets = targets,
                           outcomes = outcomes,
                           populationSettings = populationSettings,
                           modelList = modelList,
                           covariateSettings = covariateSettings,
                           resrictOutcomePops = resrictOutcomePops,
                           resrictModelCovs = resrictModelCovs,
                           webApi = webApi,
                           outputLocation = 'D:/testing',
                           jsonName = 'predictionAnalysisList.json')
```

3.3 Step 3: Creating the study package

You can now use Hydra and the json you created to generate an R package for you study. You need to load the settings by specifying where the json you previously created is saved, for example if when running 'createDevelopmentStudyJson' you set outputLocation = 'D:/testing' and jsonName = 'predictionAnalysisList.json' then you would run:

```
specifications <- Hydra::loadSpecifications(file.path('D:/testing', 'predictionAnalysisList.json'))
```

Then execute Hydra using the specification and tell hydra where you want the package created, for example if I wanted the R package created at 'D:/testing/package' then I would run:

```
Hydra::hydrate(specifications = specifications, outputFolder = 'D:/testing/package')
```

4 Editing the R package

After completing steps 1-3 above you will now have your R study package found in the location specified. This package is ready to execute, but you can also make customisations.

5 Complete Example

```
# code to create the json prediction:
webApi = 'https://yourWebAPI'

populationSettings <- list(PatientLevelPrediction::createStudyPopulationSettings(riskWindowEnd = 365),
                           PatientLevelPrediction::createStudyPopulationSettings(riskWindowEnd = 730))
modelList <- list(PatientLevelPrediction::setLassoLogisticRegression(),
                  PatientLevelPrediction::setAdaBoost())
covariateSettings <- list(list(list(fnct = 'createCovariateSettings',
                                    settings = FeatureExtraction::createCovariateSettings(useDemographi
                               list(fnct = 'createCohortCovariateSettings',
                                    settings = list(covariateName = 'test', covariateId = 16442956,
                                                    cohortId = 16442,
                                                    startDay=-30, endDay=0, count=F,
                                                    ageInteraction = F))),
                          list(fnct = 'createCovariateSettings',
                               settings = FeatureExtraction::createCovariateSettings(useDemographicsGen
)

resrictOutcomePops <- data.frame(outcomeId = c(16428,16435),
                                 populationSettingId = c(1,2))
resrictModelCovs = data.frame(modelSettingId = c(1,1,2),
                              covariateSettingId = c(1,2,1))

json <- createDevelopmentStudyJson(packageName = 'exampleStudy',
                           packageDescription = 'an example of the skeleton',
                           createdBy = 'add name',
                           organizationName = 'add organization',
                           targets = data.frame(targetId = c(16425,16377,16387),
                                                cohortId = c(16425,16377,16387),
                                                targetName = c('diabetes','liver issues','fall')),
                           outcomes = data.frame(outcomeId = c(16428,16435),
                                                 cohortId = c(16428,16435),
                                                 outcomeName = c('congestive heart failure','afib')),
                           populationSettings = populationSettings,
                           modelList = modelList,
                           covariateSettings = covariateSettings,
                           resrictOutcomePops = resrictOutcomePops,
                           resrictModelCovs = resrictModelCovs,
                           webApi = webApi,
                           outputLocation = 'D:/testing',
                           jsonName = 'predictionAnalysisList.json')

specifications <- Hydra::loadSpecifications(file.path('D:/testing', 'predictionAnalysisList.json'))
Hydra::hydrate(specifications = specifications, outputFolder = 'D:/testing/package')
```