Creating Shiny App

Jenna Reps

2021-12-16

Contents

1 Introduction

In this vignette we will show with example code how to create a shiny app and add the shiny app online for other researcher around the whole to explore.

There are two ways to create the shiny app: 1) Using the atlas R generated prediction package 2) Manually using the PatientLevelPrediction functions in a script

We assume you have experience with using the OHDSI PatientLevelPrediction package to develop and externally validate prediction models using data in the OMOP CDM. If you do not have experience with this then please first read our general vignette `BuildingPredictiveModels` vignette.

2 Atlas Development Shiny App

2.1 Step 1: Run the model development package to get results

To create a shiny app project via the Atlas auto-generated prediction R package you named 'myPackage' you need to run the execute function:

```
library(myPackage)
myPackage::execute(connectionDetails = connectionDetails,
        cdmDatabaseSchema = 'myDatabaseSchema.dbo',
        cdmDatabaseName = 'MyDatabase',
        cohortDatabaseSchema = 'myDatabaseSchema.ohdsi_results',
        cohortTable = 'cohort',
```

```
        outputFolder = 'C:/myResults',
        createProtocol = F,
        createCohorts = F,
        runAnalyses = T,
        createResultsDoc = F,
        packageResults = F,
        createValidationPackage = F,
        minCellCount= 5,
        createShiny = F,
        createJournalDocument = F,
        analysisIdDocument = 1)
```

This will extract data based on the settings you supplied in the Atlas prediction design from cohort tables already generated in your CDM database schema. The PatientLevelPrediction framework will then run and develop/evaluate models saving the results to the location specified by outputFolder (e.g., 'C:/myResults').

2.2 Step 2: Create the shiny app

To create a shiny app project with these results you can then simply run:

```
myPackage::execute(connectionDetails = connectionDetails,
        cdmDatabaseSchema = 'myDatabaseSchema.dbo',
        cdmDatabaseName = 'MyDatabase',
        cohortDatabaseSchema = 'myDatabaseSchema.ohdsi_results',
        cohortTable = 'cohort',
        outputFolder = 'C:/myResults',
        minCellCount= 5,
        createShiny = T)
```

making sure the outputFolder is the same location used when you ran the analysis. This code populates a shiny app project with the results but removes sensitive date such as connection settings, the cdmDatabaseSchema setting, the predicton matrix and any sensitive counts less than 'minCellCount' from the covariate summary and performance evalaution.

The shiny app project populated with the model development results can then be found at '[outputFolder]/ShinyApp' e.g., 'C:/myResults/ShinyApp'.

2.2.1 Testing (Optional but recommended)

You can test the app by opening the shiny project within the [outputFolder]/ShinyApp' folder, double click on the file named 'PLPViewer.Rproj'. This will open an R studio session with the shiny app project loaded. Now load the 'ui.R' files within this R studio session and you will see a green arrow with the words 'Run App' at the top right of the script. Click on this and the shiny app with open. Note: You may need to install some R pacakge dependancies for the shiny app to work.

2.3 Step 3: Sharing the shiny app

Once you are happy with your app, you can publish it onto https://data.ohdsi.org by adding the folder 'ShinyApp' to the OHDSI githib ShinyDeploy (https://github.com/OHDSI/ShinyDeploy/). Continuing the example, we would copy the folder '[outputFolder]/ShinyApp' and paste it to the local github clone of ShinyDeploy. We recommend renaming the folder from 'ShinyApp' to a name that describes your prediction, e.g., 'StrokeinAF'. Then commit the changes and make a pull request to ShinyDeploy. Once accepted your shiny app will be viewable at 'https://data.ohdsi.org'. If you commited the folder named 'StrokeInAF' then the shiny app will be hosted at 'https://data.ohdsi.org/StrokeInAF'.

3 Atlas External Validation

To include external validation results you can use the Atlas generated R study package to create the external validation package:

```
myPackage::execute(connectionDetails = connectionDetails,
        cdmDatabaseSchema = 'myDatabaseSchema.dbo',
        cdmDatabaseName = 'MyDatabase',
        cohortDatabaseSchema = 'myDatabaseSchema.ohdsi_results',
        cohortTable = 'cohort',
        outputFolder = 'C:/myResults',
        createValidationPackage = T)
```

This will create a new R package inside the 'outputFolder' location with the word 'Validation' appended the name of your development package. For example, if your 'outputFolder' was 'C:/myResults' and your development package was named 'myPackage' then the validation package will be found at: 'C:/myResults/myPackageValidation'. When running the valdiation package make sure to set the 'outputFolder' to the Validation folder within your model development outputFolder location:

```
myPackageValidation::execute(connectionDetails = connectionDetails,
                  databaseName = databaseName,
                  cdmDatabaseSchema = cdmDatabaseSchema,
                  cohortDatabaseSchema = cohortDatabaseSchema,
                  oracleTempSchema = oracleTempSchema,
                  cohortTable = cohortTable,
                  outputFolder = 'C:/myResults/Validation',
                  createCohorts = T,
                  runValidation = T,
                  packageResults = F,
                  minCellCount = 5,
                  sampleSize = NULL)
```

Now you can rerun Steps 2-3 to populate the shiny app project that will also include the validation results (as long as the validation results are in the Validation folder found in the Step 1 outputFolder location e.g., in 'C:/myResults/Validation').

4 Combining multiple atlas results into one shiny app:

The code below can be used to combine multiple Atlas packages' results into one shiny app:

```
populateMultipleShinyApp <- function(shinyDirectory,
                             resultDirectory,
                             minCellCount = 10,
                             databaseName = 'sharable name of development data'){

  #check inputs
  if(missing(shinyDirectory)){
    shinyDirectory <- system.file("shiny", "PLPViewer", package = "SkeletonPredictionStudy")
  }
  if(missing(resultDirectory)){
    stop('Need to enter the resultDirectory')
  }


    for(i in 1:length(resultDirectory)){
```

```
    if(!dir.exists(resultDirectory[i])){
      stop(paste('resultDirectory ',i,' does not exist'))
    }
  }

outputDirectory <- file.path(shinyDirectory,'data')

# create the shiny data folder
if(!dir.exists(outputDirectory)){
  dir.create(outputDirectory, recursive = T)
}


# need to edit settings ...
files <- c()
for(i in 1:length(resultDirectory)){
# copy the settings csv
file <- utils::read.csv(file.path(resultDirectory[i],'settings.csv'))
file$analysisId <- 1000*as.double(file$analysisId)+i
files <- rbind(files, file)
}
utils::write.csv(files, file.path(outputDirectory,'settings.csv'), row.names = F)

for(i in 1:length(resultDirectory)){
# copy each analysis as a rds file and copy the log
files <- dir(resultDirectory[i], full.names = F)
files <- files[grep('Analysis', files)]
for(file in files){

  if(!dir.exists(file.path(outputDirectory,paste0('Analysis_',1000*as.double(gsub('Analysis_','',file)
    dir.create(file.path(outputDirectory,paste0('Analysis_',1000*as.double(gsub('Analysis_','',file))
  }

  if(dir.exists(file.path(resultDirectory[i],file, 'plpResult'))){
    res <- PatientLevelPrediction::loadPlpResult(file.path(resultDirectory[i],file, 'plpResult'))
    res <- PatientLevelPrediction::transportPlp(res, n= minCellCount,
                                                save = F, dataName = databaseName[i])

    res$covariateSummary <- res$covariateSummary[res$covariateSummary$covariateValue!=0,]
    covSet <- res$model$metaData$call$covariateSettings
    res$model$metaData <- NULL
    res$model$metaData$call$covariateSettings <- covSet
    res$model$predict <- NULL
    if(!is.null(res$performanceEvaluation$evaluationStatistics)){
    res$performanceEvaluation$evaluationStatistics[,1] <- paste0('Analysis_',1000*as.double(gsub('Anal
    } else{
      writeLines(paste0(resultDirectory[i],file, '-ev'))
    }
    if(!is.null(res$performanceEvaluation$thresholdSummary)){
    res$performanceEvaluation$thresholdSummary[,1] <- paste0('Analysis_',1000*as.double(gsub('Analysis
    }else{
      writeLines(paste0(resultDirectory[i],file, '-thres'))
    }
```

```
    if(!is.null(res$performanceEvaluation$demographicSummary)){
    res$performanceEvaluation$demographicSummary[,1] <- paste0('Analysis_',1000*as.double(gsub('Analys
    } else{
      writeLines(paste0(resultDirectory[i],file, '-dem'))
    }
    if(!is.null(res$performanceEvaluation$calibrationSummary)){
    res$performanceEvaluation$calibrationSummary[,1] <- paste0('Analysis_',1000*as.double(gsub('Analys
    }else{
      writeLines(paste0(resultDirectory[i],file, '-cal'))
    }
    if(!is.null(res$performanceEvaluation$predictionDistribution)){
    res$performanceEvaluation$predictionDistribution[,1] <- paste0('Analysis_',1000*as.double(gsub('An
    }else{
      writeLines(paste0(resultDirectory[i],file, '-dist'))
    }
    saveRDS(res, file.path(outputDirectory,paste0('Analysis_',1000*as.double(gsub('Analysis_','',file)
  }
  if(file.exists(file.path(resultDirectory[i],file, 'plpLog.txt'))){
    file.copy(from = file.path(resultDirectory[i],file, 'plpLog.txt'),
              to = file.path(outputDirectory,paste0('Analysis_',1000*as.double(gsub('Analysis_','',fil
  }
}
}



for(i in 1:length(resultDirectory)){
# copy any validation results
if(dir.exists(file.path(resultDirectory[i],'Validation'))){
  valFolders <-  dir(file.path(resultDirectory[i],'Validation'), full.names = F)

  if(length(valFolders)>0){
    # move each of the validation rds
    for(valFolder in valFolders){

      # get the analysisIds
      valSubfolders <- dir(file.path(resultDirectory[i],'Validation',valFolder), full.names = F)
      if(length(valSubfolders)!=0){
        for(valSubfolder in valSubfolders ){
          valSubfolderUpdate <- paste0('Analysis_', as.double(gsub('Analysis_','', valSubfolder))*1000
          valOut <- file.path(valFolder,valSubfolderUpdate)
          valOutOld <- file.path(valFolder,valSubfolder)
          if(!dir.exists(file.path(outputDirectory,'Validation',valOut))){
            dir.create(file.path(outputDirectory,'Validation',valOut), recursive = T)
          }


          if(file.exists(file.path(resultDirectory[i],'Validation',valOutOld, 'validationResult.rds'))
            res <- readRDS(file.path(resultDirectory[i],'Validation',valOutOld, 'validationResult.rds
            res <- PatientLevelPrediction::transportPlp(res, n= minCellCount,
                                                        save = F, dataName = databaseName[i])
            res$covariateSummary <- res$covariateSummary[res$covariateSummary$covariateValue!=0,]
            saveRDS(res, file.path(outputDirectory,'Validation',valOut, 'validationResult.rds'))
```

```
            }
          }
        }

      }

    }

  }
  }

  return(outputDirectory)

}
```

4.1 Example code to combine multiple results

The following code will combine the results found in 'C:/myResults', 'C:/myResults2' and 'C:/myResults3' into the shiny project at 'C:/R/library/myPackage/shiny/PLPViewer':

```
populateMultipleShinyApp(shinyDirectory = 'C:/R/library/myPackage/shiny/PLPViewer',
                                      resultDirectory = c('C:/myResults',
                                                          'C:/myResults2',
                                                          'C:/myResults3'),
                                      minCellCount = 0,
                                      databaseName = c('database1','database2','database3'))
```

5 Manual App Creation

[instructions coming soon]

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.