vignetteSurvivalTimes

2022-11-10

Contents

first we make sure we have the uptodate package

```
remotes::install_github("GidiusVanDeKamp/PlasmodeSim")
```

loading the plpData

```
connectionDetails <- Eunomia::getEunomiaConnectionDetails()

# create the database
Eunomia::createCohorts(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = 'main',
  cohortDatabaseSchema = 'main',
  cohortTable = 'cohort'
)
```

```
## Creating cohort: Celecoxib
##   |                                                                                   |
## Creating cohort: Diclofenac
##   |                                                                                   |
## Creating cohort: GiBleed
##   |                                                                                   |
## Creating cohort: NSAIDs
##   |                                                                                   |
## Cohorts created in table main.cohort
##   cohortId       name
## 1        1  Celecoxib
## 2        2 Diclofenac
## 3        3    GiBleed
## 4        4     NSAIDs
##                                                                                          description
## 1    A simplified cohort definition for new users of celecoxib, designed specifically for Eunomia.
## 2    A simplified cohort definition for new users ofdiclofenac, designed specifically for Eunomia.
## 3 A simplified cohort definition for gastrointestinal bleeding, designed specifically for Eunomia.
## 4       A simplified cohort definition for new users of NSAIDs, designed specifically for Eunomia.
##   count
## 1  1844
## 2   850
## 3   479
## 4  2694
```

```
# ------------------------------------------------------------------------------
# Points PatientLevelPredictionPackage to the Eunomia database
# Tells Eunomia to extract the cohort stored with id = 4 as the target cohort
# and cohort with id = 3 as the outcome cohort. The other settings (...Schema)
# tell the database where to look for the target and the outcome cohorts
# ------------------------------------------------------------------------------
databaseDetails <- PatientLevelPrediction::createDatabaseDetails(
  connectionDetails = connectionDetails,
  cdmDatabaseId = "eunomia",
  cdmDatabaseSchema = 'main',
  cdmDatabaseName = 'Eunomia',
  cohortDatabaseSchema = 'main',
  cohortTable = 'cohort',
  target = 4,
  outcomeDatabaseSchema = 'main',
  outcomeTable = 'cohort',
  outcomeId = 3,
  cdmVersion = 5
)

# Use ?FeatureExtraction::createCovariateSettings to see what the options are
# There are a lot...
covariateSettings <- FeatureExtraction::createCovariateSettings(
  useDemographicsGender = TRUE,
  useDemographicsAgeGroup = TRUE,
  useConditionGroupEraLongTerm = TRUE,
  useDrugGroupEraLongTerm = TRUE,
  endDays = -1,
  longTermStartDays = -365
)

restrictPlpDataSettings <- PatientLevelPrediction::createRestrictPlpDataSettings(
  studyStartDate = '20000101',
  studyEndDate = '20200101',
  firstExposureOnly = TRUE,
  washoutPeriod = 30
)
```

```
# issue with studyStartDate/studyEndDate
restrictPlpDataSettings <- PatientLevelPrediction::createRestrictPlpDataSettings(
  firstExposureOnly = TRUE,
  washoutPeriod = 30
)

plpData <- PatientLevelPrediction::getPlpData(
  databaseDetails = databaseDetails,
  covariateSettings = covariateSettings,
  restrictPlpDataSettings = restrictPlpDataSettings
)
```

```
##   |                                                                              |
```

```
## Warning: The 'oracleTempSchema' argument is deprecated. Use 'tempEmulationSchema' instead.
## This warning is displayed once every 8 hours.
```

```
## Constructing features on server
##   |                                                                              |
## Fetching data from server
## Fetching data took 0.339 secs
```

defining a training set.

For simulating new censored survival times we need more than one probability, we use the baselinehazard, stored in the plpModel, we also need a model for the censoring. First we make the traing set.

```
populationSettings <- PatientLevelPrediction::createStudyPopulationSettings(
  binary = TRUE,
  includeAllOutcomes = FALSE,
  firstExposureOnly = FALSE,
  washoutPeriod = 180,
  removeSubjectsWithPriorOutcome = FALSE,
  priorOutcomeLookback = 99999,
  requireTimeAtRisk = TRUE,
  minTimeAtRisk = 1,
  riskWindowStart = 1,
  startAnchor = 'cohort start',
  riskWindowEnd = 7300,
  endAnchor = 'cohort start'
)
executeSettings <- PatientLevelPrediction::createExecuteSettings(
  runSplitData = TRUE,
  runSampleData = FALSE,
  runfeatureEngineering = FALSE,
  runPreprocessData = TRUE,
  runModelDevelopment = TRUE,
  runCovariateSummary = TRUE
)
splitSettings <- PatientLevelPrediction::createDefaultSplitSetting(
  testFraction = 0.25,
  trainFraction = 0.75,
```

```r
  splitSeed = 123,
  nfold = 3,
  type = 'stratified'
)
sampleSettings <- PatientLevelPrediction::createSampleSettings(
  type = 'none'
)
featureEngineeringSettings <-
  PatientLevelPrediction::createFeatureEngineeringSettings(
  type = 'none'
)
preprocessSettings <- PatientLevelPrediction::createPreprocessSettings(
  minFraction = 0,
  normalize = TRUE,
  removeRedundancy = TRUE
)
TrainingSet <- PlasmodeSim::MakeTraingSet(
  plpData = plpData,
  executeSettings = executeSettings,
  populationSettings = populationSettings,
  splitSettings = splitSettings,
  sampleSettings = sampleSettings,
  preprocessSettings = preprocessSettings,
  featureEngineeringSettings = featureEngineeringSettings,
  outcomeId = 3
)
```

```
## Outcome is 0 or 1
## seed: 123
## Creating a 25% test and 75% train (into 3 folds) random stratified split by class
## Data split into 656 test cases and 1974 train cases (658, 658, 658)
## Train Set:
## Fold 1 658 patients with 120 outcomes - Fold 2 658 patients with 120 outcomes - Fold 3 658 patients
## 103 covariates in train data
## Test Set:
## 656 patients with 119 outcomes
## Removing 2 redundant covariates
## Normalizing covariates
## Tidying covariates took 0.512 secs
## Train Set:
## Fold 1 658 patients with 120 outcomes - Fold 2 658 patients with 120 outcomes - Fold 3 658 patients
## 101 covariates in train data
## Test Set:
## 656 patients with 119 outcomes
```

Fitting the model

To fit the model we make the modelsettings and we run the function `fitModelWithCensoring`.

```r
modelSettings <- PatientLevelPrediction::setCoxModel()

fitCensor <- PlasmodeSim::fitModelWithCensoring(
  Trainingset = TrainingSet$Train,
  modelSettings = modelSettings
  # now i have only one model setting
  # should i change this to two seperate settings
)
```

```
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.146 secs
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.137 secs
```

Generating new outcomes times

```
population <- PatientLevelPrediction::createStudyPopulation(
  plpData = plpData,
  outcomeId = 3,
  populationSettings = populationSettings
)
```

```
## Outcome is 0 or 1
```

```
NewOutcomes <- PlasmodeSim::simulateSurvivaltimesWithCensoring(
  censorModel = fitCensor,
  plpData = plpData,
  population = population,
  populationSettings = populationSettings,
  numberToSimulate = 10
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.174 secs
## Prediction took 0.266 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.171 secs
## Prediction took 0.261 secs
```

```
head(NewOutcomes)
```

```
##   rowId survivalTime outcomeCount
## 1  2367           18            1
## 2   500         7300            0
## 3  2086           20            1
## 4  2279         7300            0
## 5   287         7300            0
## 6   290         5862            0
```

```
#for simulation the uncensored data

newdata <- PlasmodeSim::simulateSurvivaltimes(
  plpModel = fitCensor$outcomesModel,
  plpData = plpData,
  numberToSimulate = 10,
  population = population,
  populationSettings = populationSettings
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.183 secs
## Prediction took 0.263 secs
```

```
head(newdata)
```

```
##   rowId outcome
## 1  1961    7300
## 2  1134    7300
## 3  1720    7300
## 4  2369    7300
## 5   418    7300
## 6  2599    7300
```

defing an unfitted model

```
# makeCoxModel.

plpModel <- fitCensor$outcomesModel
coeff <- plpModel$model$coefficients
survival <- plpModel$model$baselineSurvival$surv
times <- plpModel$model$baselineSurvival$time

unfittedmodel <- PlasmodeSim::makeCoxModel(
  coefficients = coeff,
  baselinehazard = survival,
  timesofbaselinhazard = times,
  featureEngineering = NULL #  = NULL is the standart setting.
)

newdata <- PlasmodeSim::simulateSurvivaltimes(
  plpModel = unfittedmodel,
  plpData = plpData,
  numberToSimulate = 10,
  population = population,
  populationSettings = populationSettings
)
```

```
## Prediction took 0.21 secs
```

```
head(newdata)
```

```
##   rowId outcome
## 1    54      67
## 2  1229    7300
## 3   552    7300
## 4  2376    7300
## 5  1447    7300
## 6  1709      30
```

defining an unfitted model with censoring

```
#we can swap outcomes with censoring.
unfittedcensor<- list(censorModel = unfittedmodel,
                      outcomesModel = fitCensor$outcomesModel)

NewOutcomes <- PlasmodeSim::simulateSurvivaltimesWithCensoring(
  censorModel = unfittedcensor,
  plpData = plpData,
  population =  population,
  populationSettings = populationSettings,
  numberToSimulate = 200
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.175 secs
## Prediction took 0.263 secs
## Prediction took 0.195 secs
```

```
head(NewOutcomes)
```

```
##   rowId survivalTime outcomeCount
## 1   658           51            0
## 2  1864         7300            0
## 3  1317         7300            0
## 4  1193           86            1
## 5  1518         7300            0
## 6  1883         7300            0
```

Adjusting the BaselineSurvival

```
adjustedModel <- PlasmodeSim::AdjustBaselineSurvival(
  plpModel = plpModel,
  TrainingSet = TrainingSet$Train,
  plpData = plpData,
  populationSettings = populationSettings,
  timeTofixat = 3592,
  proptofixwith = 0.87,
  intervalSolution= c(-100,100)
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.178 secs
## Prediction took 0.244 secs
```

```
NewOutcomes <- PlasmodeSim::simulateSurvivaltimesWithCensoring(
  censorModel = list(censorModel = fitCensor$outcomesModel,
                      outcomesModel = adjustedModel),
  plpData = plpData,
  population =  population,
  populationSettings = populationSettings,
  numberToSimulate = 2000
)
```

```
## Prediction took 0.195 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.173 secs
## Prediction took 0.267 secs
```

```
head(NewOutcomes)
```

```
##   rowId survivalTime outcomeCount
## 1   705           56            1
## 2   306           14            1
## 3  1066         7300            0
## 4   678         7300            0
## 5   532           64            0
## 6  1942           58            1
```

plotting Kaplan Meier estimates

the function `kaplanMeierPlot` visualised the kamplanmeier estimate of a given dataset. It works with ggplot. we can easily compare the simulated data sets with the real dataset by putting them in one plot. For the true data set we set the colour to red.

```
NewOutcomes <- PlasmodeSim::simulateSurvivaltimesWithCensoring(
  censorModel = fitCensor,
  plpData = plpData,
  population = population,
  populationSettings = populationSettings,
  numberToSimulate = 656
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.176 secs
## Prediction took 0.263 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.179 secs
## Prediction took 0.263 secs
```

```
NewOutcomes2 <- PlasmodeSim::simulateSurvivaltimesWithCensoring(
  censorModel = fitCensor,
  plpData = plpData,
  population = population,
  populationSettings = populationSettings,
  numberToSimulate = 656
)
```

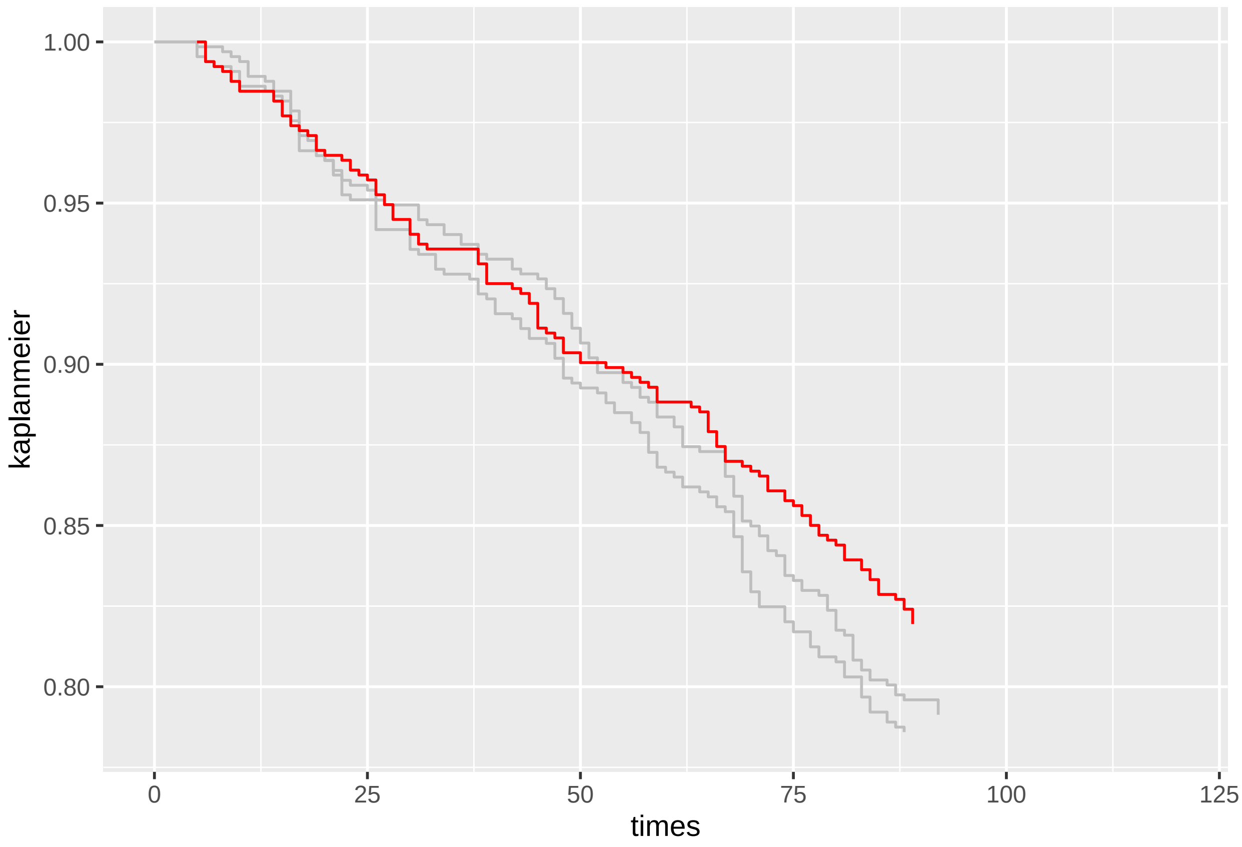
```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.183 secs
## Prediction took 0.264 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.175 secs
## Prediction took 0.259 secs
```

```
ggplot2::ggplot()+
  PlasmodeSim::KaplanMeierPlot( NewOutcomes )+
  PlasmodeSim::KaplanMeierPlot( NewOutcomes2 )+
  PlasmodeSim::KaplanMeierPlot( TrainingSet$Test$labels, colour = 'red' )+
  ggplot2::xlim(c(0,120))
```

```
## Warning: Removed 227 rows containing missing values (‘geom_step()‘).

## Warning: Removed 231 rows containing missing values (‘geom_step()‘).

## Warning: Removed 283 rows containing missing values (‘geom_step()‘).
```


kaplanmeier
times