vignetteSurvivalTimes

2022-11-10

Survival times outcomes.

first we make 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.

```
plpData <- PatientLevelPrediction::loadPlpData(
  "~/R/internshipErasmusMC/simulate-new-patients-outcomes/plp_democox/Data" )



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,
  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 w
## 103 covariates in train data
## Test Set:
## 656 patients with 119 outcomes
## Removing 2 redundant covariates
## Normalizing covariates
## Tidying covariates took 0.52 secs
## Train Set:
## Fold 1 658 patients with 120 outcomes - Fold 2 658 patients with 120 outcomes - Fold 3 658 patients w
## 101 covariates in train data
## Test Set:
## 656 patients with 119 outcomes
```

Fit the model

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

```
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 0.132 secs
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.135 secs
```

Generate new outcomes from a population.

```
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 0.182 secs
## Prediction took 0.262 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.18 secs
## Prediction took 0.267 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 0.181 secs
## Prediction took 0.261 secs
```

```
head(newdata)
```

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

Make 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 0.184 secs
```

```
head(newdata)
```

```
##   rowId outcome
```

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

Make 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 0.171 secs
## Prediction took 0.266 secs
## Prediction took 0.191 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
```

Adjust 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 0.175 secs
## Prediction took 0.25 secs
```

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

```
## Prediction took 0.186 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.181 secs
## Prediction took 0.262 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 the survival

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 0.182 secs
## Prediction took 0.266 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.179 secs
## Prediction took 0.262 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 0.175 secs
## Prediction took 0.269 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.173 secs
## Prediction took 0.269 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()`).
```