vignetteSurvivalTimes

2022-11-10

Contents

Loading the plpData 2
Defining a training set. 3
Fitting the model 5
Generating new outcomes times 5
Defining an unfitted model 6
Defining an unfitted model with censoring 7

Adjusting the BaselineSurvival 8

Plotting Kaplan Meier estimates 9

first we make sure we have the uptodate package

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

```
## cli (3.3.0 -> 3.4.1) [CRAN]
## package 'cli' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'cli'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying C:
## \Users\gidiu\AppData\Local\R\win-library\4.2\00LOCK\cli\libs\x64\cli.dll to C:
## \Users\gidiu\AppData\Local\R\win-library\4.2\cli\libs\x64\cli.dll: Permission
## denied

## Warning: restored 'cli'

##
## The downloaded binary packages are in
##  C:\Users\gidiu\AppData\Local\Temp\RtmpYPR2fZ\downloaded_packages
## * checking for file 'C:\Users\gidiu\AppData\Local\Temp\RtmpYPR2fZ\remotes260c17cd3a71\GidiusVanDeKam
## * preparing 'PlasmodeSim':
## * checking DESCRIPTION meta-information ... OK
## * checking for LF line-endings in source and make files and shell scripts
## * checking for empty or unneeded directories
## Omitted 'LazyData' from DESCRIPTION
## * building 'PlasmodeSim_0.1.0.tar.gz'
##
```

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 0.379 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,
  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 0.536 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
```

Fitting 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 0.186 secs
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.141 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 0.175 secs
## Prediction took 0.268 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.181 secs
## Prediction took 0.268 secs
```

```
head(NewOutcomes)
```

```
##   rowId survivalTime outcomeCount
## 1  2367           18            1
## 2   500         7293            0
## 3  2086           20            1
## 4  2279         7300            0
## 5   287         7300            0
## 6   290         5888            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 0.273 secs
```

```
head(newdata)
```

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

Defining an unfitted model

```r
# 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 0.207 secs
```

```r
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

```r
#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 0.183 secs
## Prediction took 0.266 secs
## Prediction took 0.202 secs
```

```r
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

```r
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 0.18 secs
## Prediction took 0.246 secs
```

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

```
## Prediction took 0.197 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.184 secs
## Prediction took 0.275 secs
```

```r
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 0.19 secs
## Prediction took 0.266 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.191 secs
## Prediction took 0.27 secs
```

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

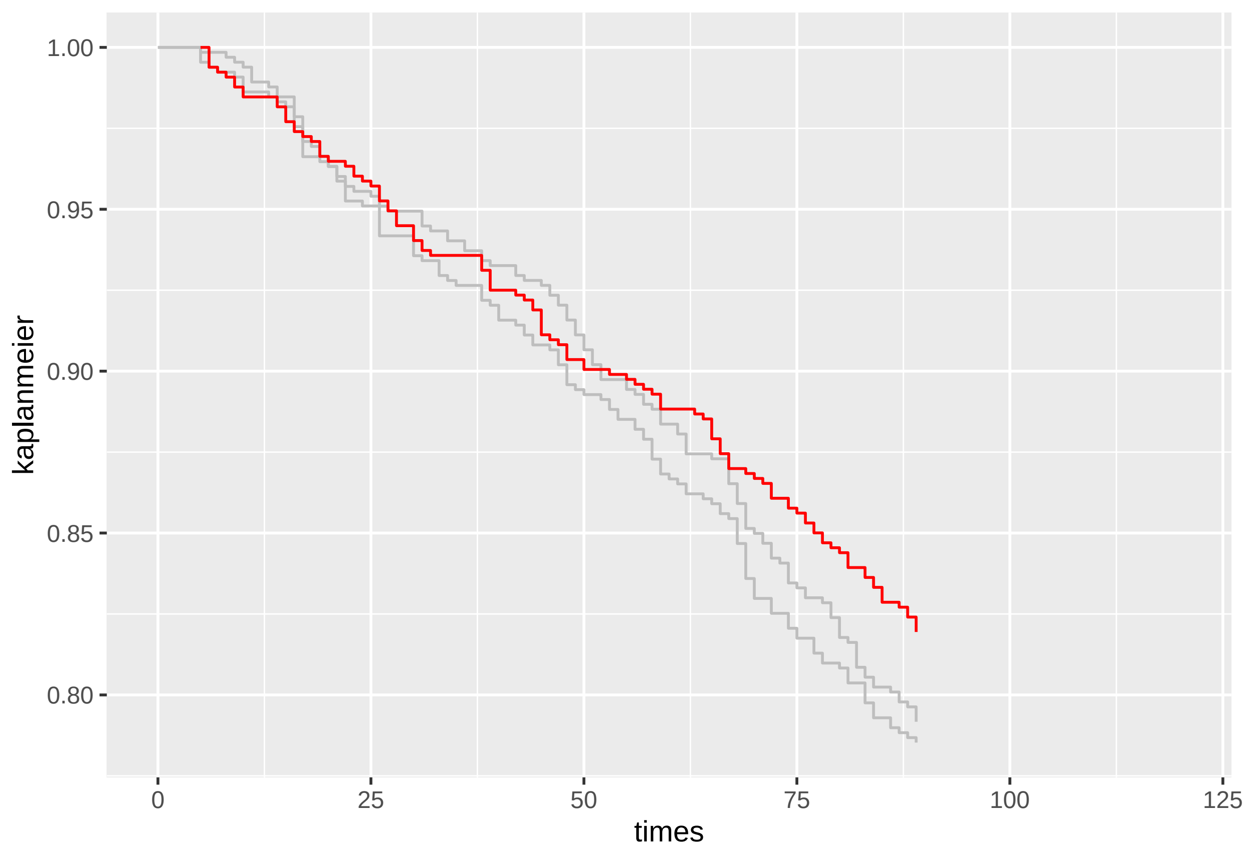
```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.179 secs
## Prediction took 0.269 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.178 secs
## Prediction took 0.27 secs
```

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

```
## Warning: Removed 223 rows containing missing values (`geom_step()`).

## Warning: Removed 236 rows containing missing values (`geom_step()`).

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



1.00
0.95
0.90
0.85
0.80
kaplanmeier
0
25
50
75
100
125
times