vignette PlasmodeSim

2022-10-19

Welcome to the vignette about the R package PlasmodeSim. This package is still under development.

installing plasmodeSim using remotes

To install using `remotes` run:

```
#install.packages("remotes")
#remotes::install_github("GidiusVanDeKamp/PlasmodeSim")
```

Setting up

This documents skips some parts, we have skipped the steps to obtain the plpResults and the plpData.

```
# library(dplyr)
# library(PlasmodeSim)

modelSettings <- PatientLevelPrediction::setLassoLogisticRegression()

plpResultLogistic <- PatientLevelPrediction::loadPlpResult(
  "~/R/internshipErasmusMC/simulate-new-patients-outcomes/plp_demolog/Result")

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

Example 1

In this example we obtain new outcomes of a fitted logistic model.

```
plpModelLog <- plpResultLogistic$model
probabilites <- PlasmodeSim::newPropsParametersPlpModel(plpModelLog,
                                                        plpData,
                                                        plpData$cohorts)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.198 secs
## Prediction took 0.177 secs
```

The function predictPlp returned this information.

```
newOut <- PlasmodeSim::newOutcomes(200, probabilites)
head(newOut)
```

```
##   rowId newOutcomes
## 1   199           1
## 2   408           0
## 3  1207           0
## 4  2327           0
## 5  1653           0
## 6   992           0
```

In the output of newOut patients are drawed randomly with the same chance, the patients could be drawed multiple times. If this happens they can have a different outcome. The function `newOutcomes` needs a data set where the column that contains the probabilities is called `value`.

Example 2

We here we show how to simulate new outcomes from an unfitted logistic model.

```
Parameters <- plpModelLog$model$coefficients
UnfittedParameters <- Parameters
UnfittedParameters[1,1] <- -0.4
UnfittedParameters[2:4,1] <- 0.4
head(UnfittedParameters)
```

```
##     betas covariateIds
## 1 -0.4000  (Intercept)
## 2  0.4000         6003
## 3  0.4000         8003
## 4  0.4000         9003
## 5  0.0092      8507001
## 6  0.0000     28060210
```

For the logistic model it is necessary that the parameters are stored in a dataset with a column called `betas` and a column called `covariateIds`.

```
plpModelunfitted <- PlasmodeSim::makeLogisticModel(UnfittedParameters)
newprobs <- PatientLevelPrediction::predictPlp(plpModelunfitted,
                                               plpData,
                                               plpData$cohorts)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.175 secs
## Prediction took 0.167 secs
```

```
newOut <- PlasmodeSim::newOutcomes(200, newprobs)
head(newOut)
```

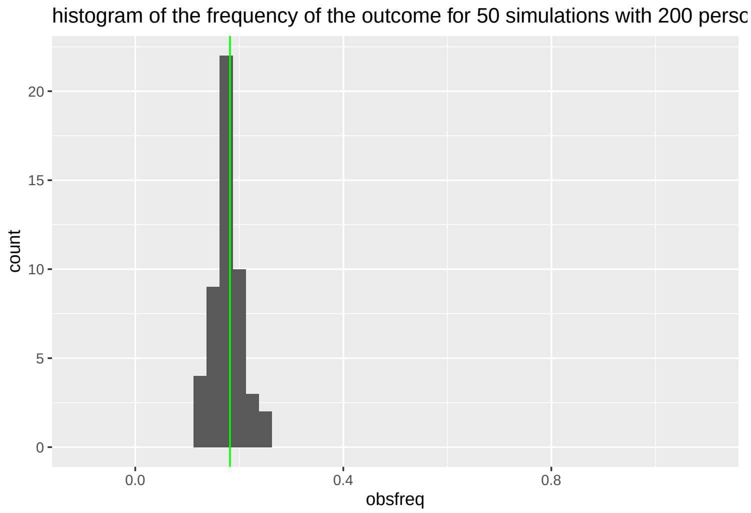
```
##   rowId newOutcomes
## 1   591           1
## 2   211           0
## 3   387           0
## 4  2361           0
## 5  1456           1
## 6  1354           0
```

Visual simulations

The function `visualOutcome` simulated new data and then plots the frequency of the outcome. Right now the function `visualOutcome` only works for a logistic model. The green line in the plots is the average outcome in the original dataset.

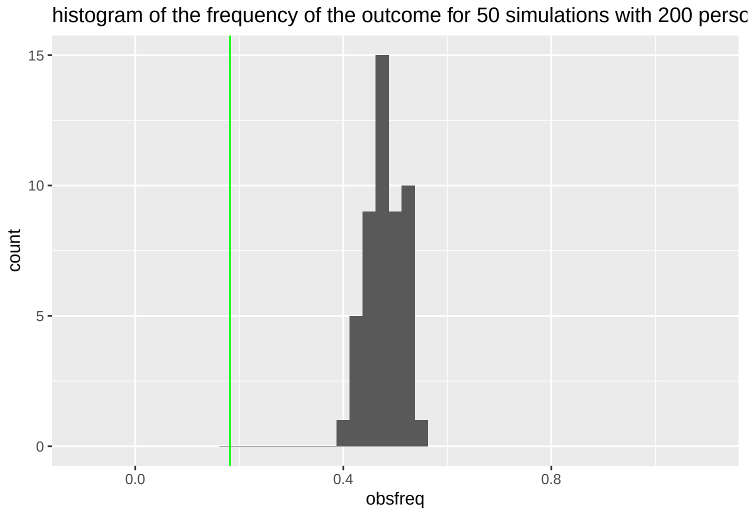
```
PlasmodeSim::visualOutcome(plpData,50,200,Parameters)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.178 secs
## Prediction took 0.171 secs
```



```
PlasmodeSim::visualOutcome(plpData,50,200,UnfittedParameters)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.173 secs
## Prediction took 0.172 secs
```



Here we have plotted 50 times the frequency of the outcome for a simulated dataset with 200 people.

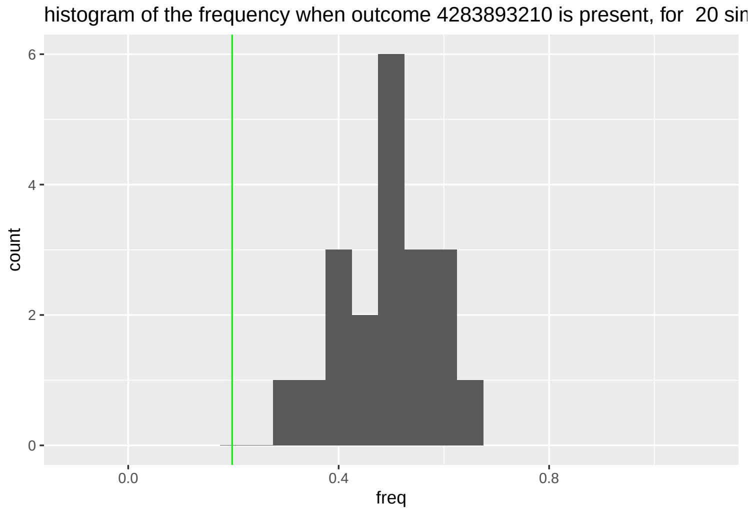
Visual of a specific covariate

```
covariateIdToStudy<- plpResultLogistic$covariateSummary$covariateId[3]
UnfittedParameters[3,]
```

```
##   betas covariateIds
## 3   0.4         8003
```

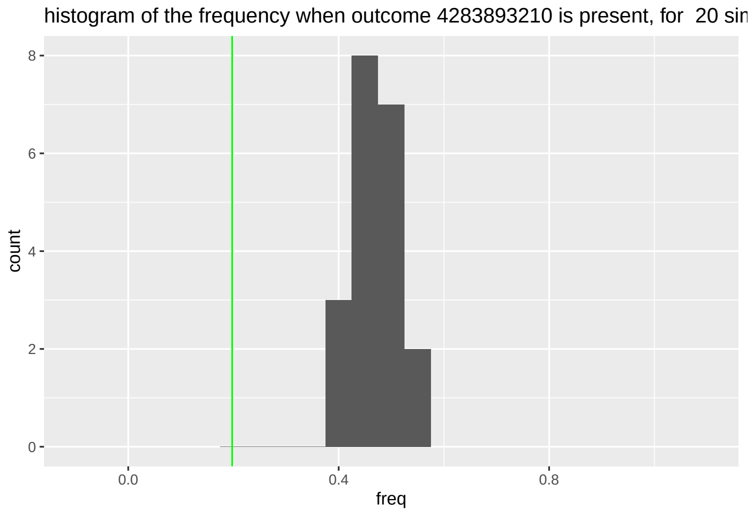
```
PlasmodeSim::visualOutcomeCovariateId(plpData,
                                      covariateIdToStudy,
                                      20,
                                      200,
                                      UnfittedParameters)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.189 secs
## Prediction took 0.173 secs
```



```
PlasmodeSim::visualOutcomeCovariateId2(plpData,
                                       covariateIdToStudy,
                                       20,
                                       200,
                                       UnfittedParameters)
```

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



As one can see `visualOutcomeCovariateId` and `visualOutcomeCovariateId2` are very similiar, they both calculate and plot the frequency for a group with a specific covariate present. The small difference is that `visualOutcomeCovariateId` filters a newly simulated dataset set to only keep the patients where the covariate is present, and `visualOutcomeCovariateId2` only simulates new outcomes for patients that have the covariate present. We see they are almost the identical only `visualOutcomeCovariateId2` is spread out less because the groups for calculating the frequency with are bigger.

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,
```

```r
  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 0.496 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
```

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 0.135 secs
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.132 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 = 20
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.17 secs
## Prediction took 0.269 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.172 secs
## Prediction took 0.265 secs
```

```
head(NewOutcomes)
```

```
##   rowId Time outcomecount
## 1  2367 4096            0
## 2   500    7            1
## 3  2086 6598            0
## 4  2279 7293            0
## 5   287 7300            0
## 6   290 7300            0
```

```
#for simulation the uncensored data

newdata <- PlasmodeSim::simulateSurvivaltimes(
  plpModel = fitCensor$outcomesModel,
  plpData = plpData,
  numberToSimulate = 100,
  population = population,
  populationSettings = populationSettings
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.174 secs
## Prediction took 0.255 secs
```

```
head(newdata)
```

```
##   rowId outcome
## 1  2509    7300
## 2  1032    7300
## 3  2063    7300
## 4  1335    7300
## 5    25      47
## 6    54    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 0.188 secs
```

```
head(newdata)
```

```
##   rowId outcome
## 1  2467    7300
## 2   627    7300
## 3  1494      62
## 4  1179    7300
## 5   207    7300
## 6     5    7300
```

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 = 20
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.181 secs
## Prediction took 0.26 secs
## Prediction took 0.19 secs
```

```
head(NewOutcomes)
```

```
##   rowId Time outcomecount
## 1  1557 7300            0
## 2  1113    8            1
## 3  2200 7300            0
## 4   603   68            1
## 5  1587 7300            0
## 6  1062   84            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 0.179 secs
## Prediction took 0.24 secs
```

```
NewOutcomes <- PlasmodeSim::simulateSurvivaltimesWithCensoring(
  censorModel = list(censorModel = fitCensor$outcomesModel,
                     outcomesModel = adjustedModel),
  plpData = plpData,
  population =  population,
  populationSettings = populationSettings,
  numberToSimulate = 20
)
```

```
## Prediction took 0.188 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.172 secs
## Prediction took 0.273 secs
```

```
head(NewOutcomes)
```

```
##   rowId Time outcomecount
## 1   109   14            1
## 2   845   73            1
## 3   719   79            1
## 4  1286    9            1
## 5   193   61            1
## 6  2122    0            1
```