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")
```

```
## Skipping install of ’PlasmodeSim’ from a github remote, the SHA1 (1ac0bf55) has not changed since las
##   Use ‘force = TRUE‘ to force installation
```

Setting up

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

```
library(dplyr)

modelSettings <- PatientLevelPrediction::setLassoLogisticRegression()

plpResultLogistic <- PatientLevelPrediction::loadPlpResult("~/R/internshipErasmusMC/simulate-new-patien

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

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 0.194 secs
## Prediction took 0.178 secs
```

The function predictPlp returned this information.

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

```
##   rowId newOutcomes
## 1  2201           0
## 2  1212           0
## 3   348           0
## 4  2451           0
## 5  2530           0
## 6   373           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 0.188 secs
## Prediction took 0.164 secs
```

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

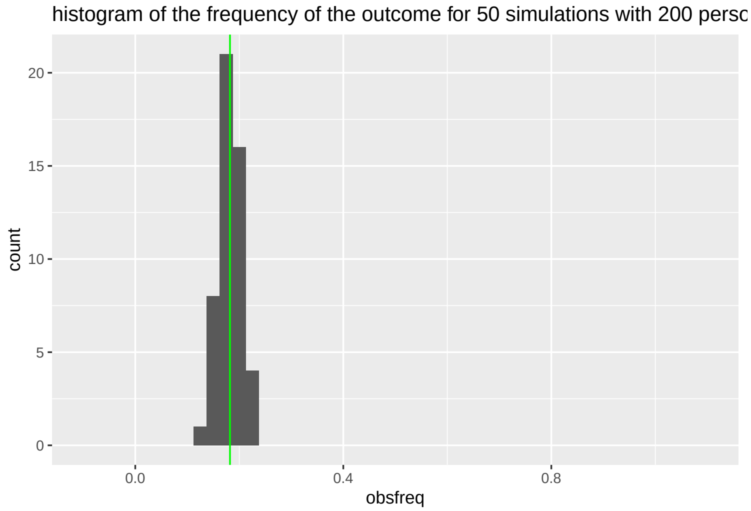
```
##   rowId newOutcomes
## 1   565           1
## 2  1783           1
## 3  1202           1
## 4  2448           0
## 5  1951           0
## 6  1386           1
```

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.

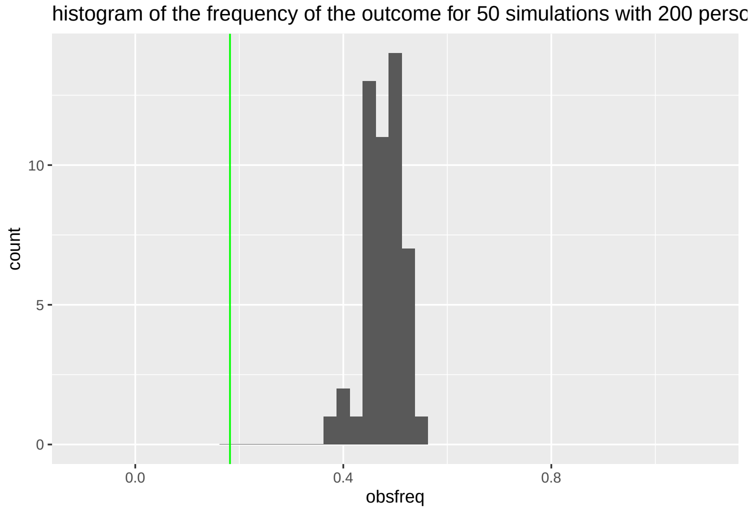
```
PlasmodeSim::visualOutcome(plpData,50,200,Parameters)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.178 secs
## Prediction took 0.181 secs
```



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

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.209 secs
## Prediction took 0.177 secs
```



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

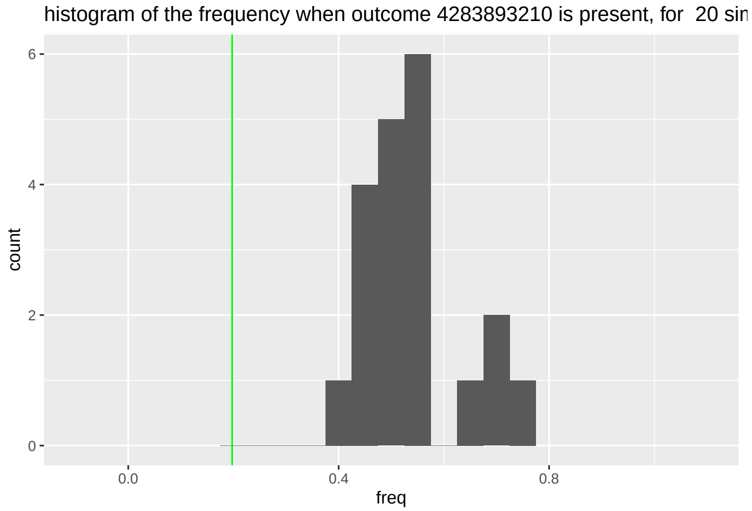
Visual of a specific covariate

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

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

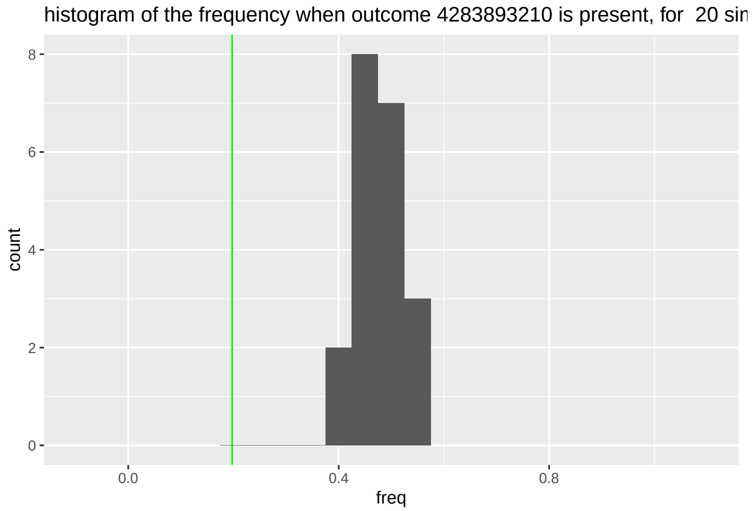
```
PlasmodeSim::visualOutcomeCovariateId(plpData, covariateIdToStudy, 20, 200, UnfittedParameters)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.194 secs
## Prediction took 0.215 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 0.175 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.

For simulating new survival times we need more than one probability, we use the baselinehazard, stored in the plpModel.

```
plpResult <- PatientLevelPrediction::loadPlpResult("~/R/internshipErasmusMC/simulate-new-patients-outcom
plpData <- PatientLevelPrediction::loadPlpData("~/R/internshipErasmusMC/simulate-new-patients-outcomes/p
plpModel<- plpResult$model #a fitted Cox model

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'
)

newdata <- PlasmodeSim::simulateSurvivaltimes(plpModel, plpData, 3, populationSettings)
head(newdata)
# function that simulates uncensored data,  simulateSurvivaltimes(plpModel, plpData, 20)
```

Simulation censored survival data

first we have to set some settings.

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

now that we have out settings. We define a training set and fit the model. The model acctually consist of two plpModels stored in a list.

```
TrainingSet <- PlasmodeSim::MakeTraingSet(plpData, executeSettings, populationSettings, splitSettings,
```

```
## 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 0.501 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
```

```
fitcensor <- PlasmodeSim::fitCensoring(TrainingSet$Train, modelSettings)
```

```
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.139 secs
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.127 secs
```

```
NewOutcomes <- PlasmodeSim::simulateSurvivaltimesWithCensoring(fitcensor, plpData, populationSettings, 2
```

```
## Outcome is 0 or 1
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.176 secs
## Prediction took 0.26 secs
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.176 secs
## Prediction took 0.263 secs
```

```
head(NewOutcomes)
```

```
##   rowId outcomeCount outcomes
## 1  1290            1       57
## 2   500            1        8
## 3  1864            1       27
## 4   428            0      226
## 5   315            0     7300
## 6   392            0     6024
```

Make an unfitted model

```
# makeCoxModel.
coeff <- plpModel$model$coefficients
survival <- plpModel$model$baselineSurvival$surv
times <- plpModel$model$baselineSurvival$time

unfittedmodel <- PlasmodeSim::makeCoxModel(coeff, survival, times)

#newdata <- PlasmodeSim::simulateSurvivaltimes(unfittedmodel, plpData, 3, populationSettings)
#head(newdata)
```