Vignette PlasmodeSim

2022-10-19

Welcome to the vignette about the R package PlasmodeSim. This package is still under development. This package goal is to simulate a new outcomes for real patients data where the outcomes follow a specified model.

installing plasmodeSim using remotes

To install using `remotes` run:

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

Setting up

To start we need a plpModel and plpData. For information how to obtain these one can look at: https://ohdsi.github.io/PatientLevelPrediction/articles/BuildingPredictiveModels.html In this documents we load them from a save file:

```
plpResultLogistic <- PatientLevelPrediction::loadPlpResult( "yourpathForPlpResult")
plpData <- PatientLevelPrediction::loadPlpData( "yourPathForPlpData" )
```

Example 1 Simulate from a plpModel

In this example we obtain new outcomes following a fitted logistic model. We start from a plpModel, then run predictPlp. At last we generate new outcomes with the function `newOutcomes` that uses the plpPrediction.

```
plpModelLog <- plpResultLogistic$model

plpPrediction <- PatientLevelPrediction::predictPlp(
  plpModel =plpModelLog,
  plpData= plpData,
  population = plpData$cohorts
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.204 secs
## Prediction took 0.186 secs
```

When running the function predictPlp it returnes some information.

```
newOut <- PlasmodeSim::newOutcomes(
  noPersons = 200,
  props=  plpPrediction
)
head(newOut)
```

```
##   rowId outcomeCount
## 1     7            0
## 2    18            1
## 3    36            0
## 4    40            0
## 5    44            0
## 6    52            0
```

The rowId in the output of `newOutcomes` are the rowId of patients that are drawn randomly with the same probability, the patients could be drawn multiple times. If a rowId happens to be in the output twice they can have a different outcome. The function `newOutcomes` needs a data set that contains the columns `rowId` and `value`. The column called `value` contains the probabilities used in generating the new outcomes.

Example 2 simulation from unfittedmodel

We here we show how to simulate outcomes from an unfitted logistic model. We use the function `makeLogisiticModel` to specify a logistic model.

```
Parameters <- plpModelLog$model$coefficients
UnfittedParameters <- Parameters
UnfittedParameters[1,1] <- -0.4
UnfittedParameters[3:5,1] <- 0.4
head(UnfittedParameters)
```

```
##   betas covariateIds
## 1  -0.4  (Intercept)
## 2   0.0         6003
## 3   0.4         8003
## 4   0.4         9003
## 5   0.4      8507001
## 6   0.0     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`. The function `makeLogisitcModel` makes a plpModel from the specified parameters.

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

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

```r
newOut <- PlasmodeSim::newOutcomes(2000, newprobs)
head(newOut)
```

```
##   rowId outcomeCount
## 1     1            1
## 2     2            1
## 3     4            1
## 4     6            0
## 5     6            0
## 6     6            1
```

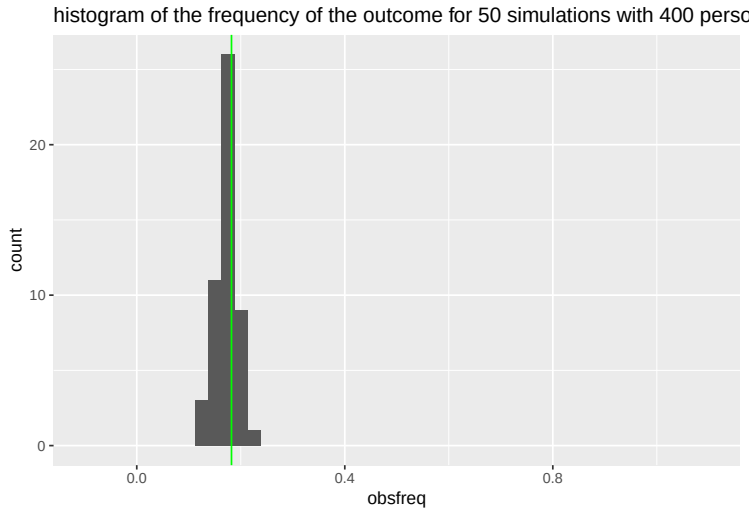
```r
#
# newOut <- dplyr::distinct(newOut,rowId, .keep_all= TRUE)
#
# modelSettings <- PatientLevelPrediction::setLassoLogisticRegression()
# splitSettings <- PatientLevelPrediction::createDefaultSplitSetting()
#
# populationSettings <- PatientLevelPrediction::createStudyPopulationSettings(
#   binary = T,
#   includeAllOutcomes = FALSE,
#   firstExposureOnly = FALSE,
#   washoutPeriod = 180,
#   removeSubjectsWithPriorOutcome = FALSE,
#   priorOutcomeLookback = 99999,
#   requireTimeAtRisk = TRUE,
#   minTimeAtRisk = 364,
#   riskWindowStart = 1,
#   startAnchor = 'cohort start',
#   riskWindowEnd = 365,
#   endAnchor = 'cohort start'
# )
#
# population <- PatientLevelPrediction::createStudyPopulation(plpData , 3, populationSettings)
# population <- dplyr::filter(population, rowId %in% newOut$rowId)
# population <- dplyr::left_join(population, newOut, by = 'rowId')
# head(population)
# population <- dplyr::mutate(population, outcomeCount = outcomeCount.y)
# population <- dplyr::select(population,-outcomeCount.y, -outcomeCount.x)
#
#
# population$outcomeCount
#
#
# trainData <- PatientLevelPrediction::splitData(plpData= plpData, population = population, splitSettin
# weirdFit <- PatientLevelPrediction::fitPlp(trainData$Train,
#                                            modelSettings,
#                                            analysisId = 'firstTry')
#
# weirdFit$model$coefficients
```

Visual simulations

The function `visualOutcome` simulates 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.

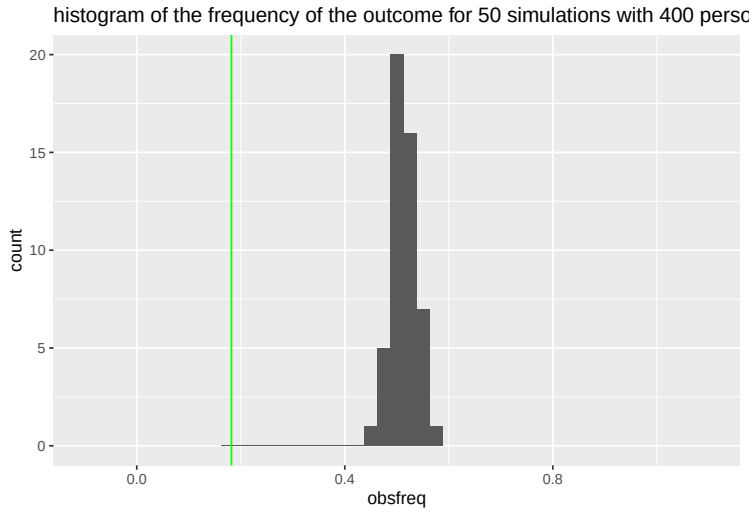
```
PlasmodeSim::visualOutcome(
  plpData = plpData,
  noSimulations = 50,
  noPersons = 400,
  parameters = Parameters
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.18 secs
## Prediction took 0.175 secs
```




```
PlasmodeSim::visualOutcome(
  plpData = plpData,
  noSimulations = 50,
  noPersons = 400,
  parameters = UnfittedParameters
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.188 secs
## Prediction took 0.174 secs
```



Here we have plotted 50 times the frequency of the outcome for a simulated dataset with 200 people. We can see that the outcome count for the fitted parameters is similar tot the original dataset, but when changeing the parameters the outcome count also changes.

Visual of a specific covariate

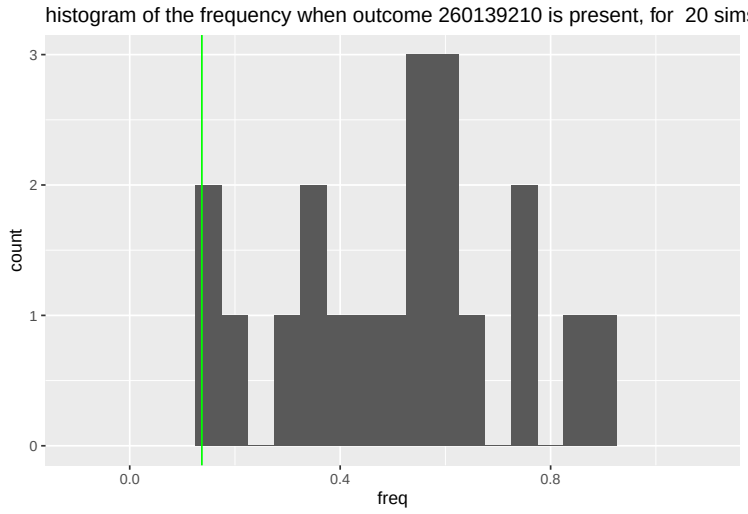
Say we are interested in the outcomes of a group with a specific covariate. Here we picked the third covariate in the model to visualise.

```
covariateIdToStudy<- plpResultLogistic$covariateSummary$covariateId[4]
UnfittedParameters[4,]
```

```
##   betas covariateIds
## 4   0.4         9003
```

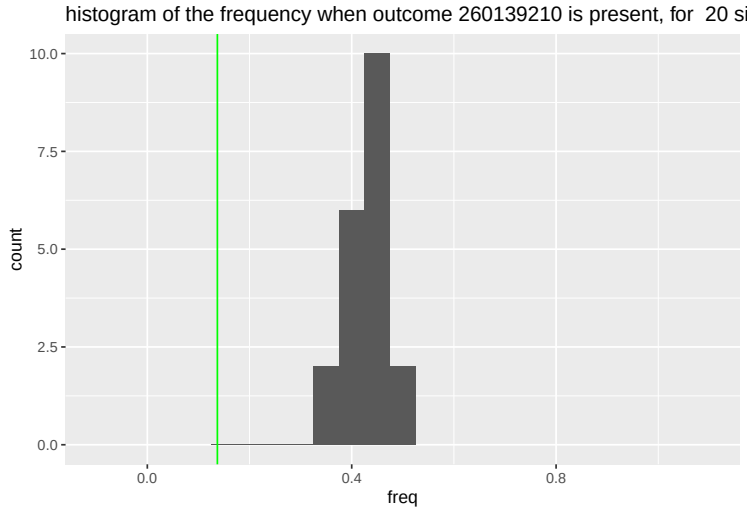
```
PlasmodeSim::visualOutcomeCovariateId(
  plpData=plpData,
  studyCovariateId= covariateIdToStudy,
  noSimulations = 20,
  noPersons = 200,
  parameters= UnfittedParameters
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.203 secs
## Prediction took 0.18 secs
```



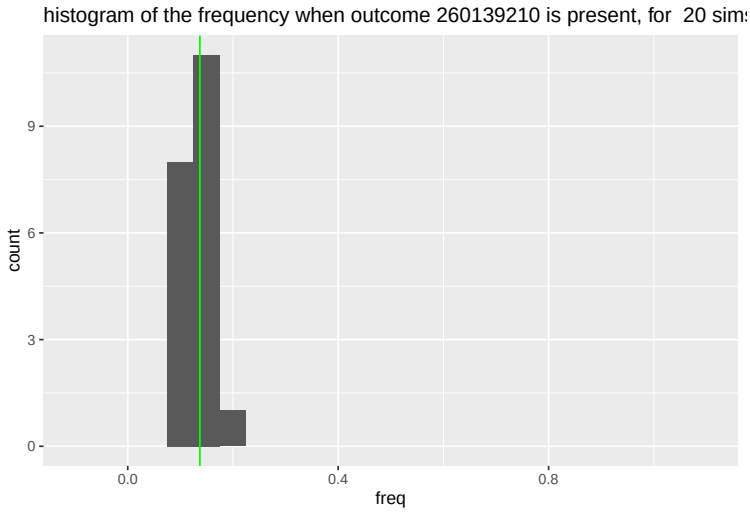

```
PlasmodeSim::visualOutcomeCovariateId2(
  plpData=plpData,
  restrictToCovariateId= covariateIdToStudy,
  noSimulations = 20,
  noPersons= 200,
  parameters= UnfittedParameters
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.213 secs
## Prediction took 0.18 secs
```




```
PlasmodeSim::visualOutcomeCovariateId2(
  plpData=plpData,
  restrictToCovariateId= covariateIdToStudy,
  noSimulations = 20,
  noPersons= 200,
  parameters= Parameters
)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.193 secs
## Prediction took 0.186 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 larger.