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 data, that could be used for testing statistical methods that use observational data instead of randomised trials.

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 a plpModel and plpData.

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

Example 1

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 out comes with the function `newOutcomes` that needs the plpPrediction.

```
plpModelLog <- plpResultLogistic$model

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

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.196 secs
## Prediction took 0.178 secs
```

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

The function predictPlp returned this information.

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

```
##   rowId outcomeCount
## 1    14            0
## 2    28            1
## 3    43            0
## 4    46            0
## 5    57            0
## 6   113            0
```

In the output of newOut patients are drawn randomly with the same probability, the patients could be drawn multiple times. If this happens 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 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`.

```
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 0.171 secs
## Prediction took 0.18 secs
```

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

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

```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'
)

#
# labels <- data.frame(newOut, survivalTime = 12*newOut$outcomeCount)
# trainData <- list(covariateData = plpData$covariateData,
#                   labels= labels,
#                   folds= data.frame(rowId = newOut$rowId,
#                                     index = rep(c(1,2,3,4),length(newOut$rowId)/4)))

population <- PatientLevelPrediction::createStudyPopulation(plpData , 3, populationSettings)
```

```
## Outcome is 0 or 1
```

```r
population <- dplyr::filter(population, rowId %in% newOut$rowId)
population <- dplyr::left_join(population, newOut, by = 'rowId')
head(population)
```

```
##   rowId subjectId targetId cohortStartDate daysFromObsStart daysToCohortEnd
## 1     2         2        4      1956-12-04            13335               0
## 2     3         3        4      1957-12-08            15315               0
## 3     4         5        4      2009-05-30            14920               0
## 4     5         6        4      2005-07-13            15170               0
## 5     7         9        4      2014-08-05            13165               0
## 6     8        11        4      1987-06-15            12550               0
##   daysToObsEnd ageYear gender outcomeCount.x timeAtRisk daysToEvent
## 1        18739      36   8532              0        365          NA
## 2        22240      41   8507              1        365          52
## 3         3378      41   8507              0        365          NA
## 4          573      42   8532              0        365          NA
## 5         1542      36   8532              0        365          NA
## 6        11649      34   8507              0        365          NA
##   survivalTime outcomeCount.y
## 1          365              0
## 2           52              0
```

```
## 3          365              0
## 4          365              0
## 5          365              0
## 6          365              1
```

```
population <- dplyr::mutate(population, outcomeCount = outcomeCount.y)
population <- dplyr::select(population,-outcomeCount.y, -outcomeCount.x)


population$outcomeCount
```

```
##    [1] 0 0 0 0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1
##   [38] 1 0 1 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 1 0 1
##   [75] 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 1 0
##  [112] 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1
##  [149] 1 1 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0 1
##  [186] 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 0 1
##  [223] 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0 0 0 1 1
##  [260] 1 1 1 1 1 0 0 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0
##  [297] 1 0 1 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1
##  [334] 0 1 1 1 1 0 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 0 0 1
##  [371] 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1
##  [408] 1 0 0 1 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 1 0 1
##  [445] 0 0 1 1 0 1 1 1 0 0 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
##  [482] 1 1 1 1 1 1 0 1 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 1 0 1 1 1 0 0 0 0 0 0
##  [519] 0 1 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 1 1 0 1 1
##  [556] 0 0 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1
##  [593] 0 0 0 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 0 1
##  [630] 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0 1 0 1 0 0 1 1 0 0
##  [667] 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0
##  [704] 1 1 1 1 0 1 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 0 1 1 0
##  [741] 1 0 1 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 1 0 0
##  [778] 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 0 0 0 1 0 0
##  [815] 1 1 1 1 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 0 1 0 1 0 1 1
##  [852] 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 1 0
##  [889] 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 0 0
##  [926] 1 0 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 1 0 1 0 0 0 1 1 0 1 1 0 0 1 1
##  [963] 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0
## [1000] 0 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1
## [1037] 1 0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1 1 1 0 1
## [1074] 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0
## [1111] 1 0 0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 0 0 0 1 1 1 1 0 1 1 1
## [1148] 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 0 0
## [1185] 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 0 1 0 1 1
## [1222] 1 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 0 1 0
## [1259] 0 1 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 1 1 1 0 0
## [1296] 1 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 0 1 1 1
## [1333] 0
```

```
trainData <- PatientLevelPrediction::splitData(plpData= plpData, population = population, splitSettings
```

```
## seed: 65374
## Creating a 25% test and 75% train (into 3 folds) random stratified split by class
## Data split into 333 test cases and 1000 train cases (334, 333, 333)
```

```r
weirdFit <- PatientLevelPrediction::fitPlp(trainData$Train,
                                           modelSettings,
                                           analysisId = 'firstTry')
```

```
## Running Cyclops
## Done.
## GLM fit status:  OK
## Creating variable importance data frame
## Prediction took 0.128 secs
```

```r
weirdFit$model$coefficients
```

```
##            betas covariateIds
## 1   0.1463163157  (Intercept)
## 2  -0.1484418732         6003
## 3  -0.0078146707         7003
## 4   0.0235032962         8003
## 5   0.2888982699         9003
## 6   0.0809517447      8507001
## 7  -0.0749128132      8532001
## 8   0.0000000000     28060210
## 9   0.0954271471     30753210
## 10  0.0000000000     78272210
## 11  0.0000000000     80809210
## 12  0.0000000000     81151210
## 13  0.0000000000     81893210
## 14  0.0000000000    134438210
## 15  0.0000000000    195588210
## 16  0.0000000000    196456210
## 17  0.0000000000    198809210
## 18  0.0000000000    257012210
## 19 -0.6227859216    260139210
## 20  0.0000000000    261325210
## 21  0.0000000000    372328210
## 22  0.0000000000    375671210
## 23  0.0000000000    378001210
## 24  0.0000000000    381316210
## 25  0.0000000000    439777210
## 26  0.0000000000    708298410
## 27  0.0000000000    738818410
## 28  0.0000000000    753626410
## 29  0.0000000000    782043410
## 30 -0.6227518866    920293410
## 31 -0.0257578355    933724410
## 32  0.0000000000    967823410
## 33  0.0000000000    975125410
## 34  0.0000000000   1000560410
## 35  0.0000000000   1102527410
## 36  0.0000000000   1110410410
## 37  0.0000000000   1112807410
## 38  0.0000000000   1115008410
## 39  0.0000000000   1119510410
## 40  0.0000000000   1124957410
```

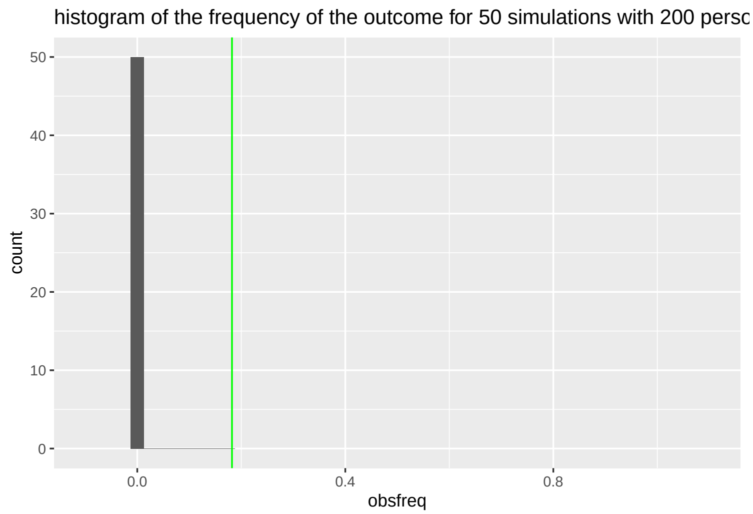
```
## 41  0.0000000000   1125315410
## 42  0.0000000000   1154029410
## 43  0.0000000000   1174888410
## 44  0.0000000000   1177480410
## 45  0.0000000000   1305058410
## 46  0.0000000000   1322184410
## 47  0.0000000000   1332418410
## 48  0.0000000000   1347450410
## 49  0.0000000000   1361711410
## 50  0.0000000000   1367571410
## 51  0.0000000000   1539403410
## 52  0.0000000000   1551099410
## 53  0.0000000000   1713332410
## 54  0.0000000000   1729720410
## 55  0.0000000000   1738521410
## 56  0.0000000000   1741122410
## 57  0.0000000000   1746114410
## 58 -1.3545379214   1759842410
## 59 -0.3338658379   4001336210
## 60  0.0000000000   4029498210
## 61  0.0000000000   4035415210
## 62  0.0000000000   4048171210
## 63  0.0000000000   4084167210
## 64  0.0000000000   4109685210
## 65  0.0000000000   4112343210
## 66  0.0000000000   4113008210
## 67  0.0000000000   4116491210
## 68 -0.0009220533   4132546210
## 69  0.0000000000   4134304210
## 70  0.0000000000   4152936210
## 71  0.0000000000   4155034210
## 72  0.0000000000   4156265210
## 73 -0.1198654678   4218389210
## 74  0.0000000000   4237458210
## 75  0.0000000000   4278672210
## 76  0.0000000000   4280726210
## 77  0.2362965091   4283893210
## 78 -0.3545071392   4285898210
## 79  0.0000000000   4294548210
## 80  0.0000000000   4296204210
## 81  0.0000000000   4296205210
## 82  0.0000000000   4310024210
## 83  0.0000000000  19003953410
## 84  0.0000000000  19010482410
## 85  0.0000000000  40481087210
## 86  0.0000000000  40486433210
```

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.

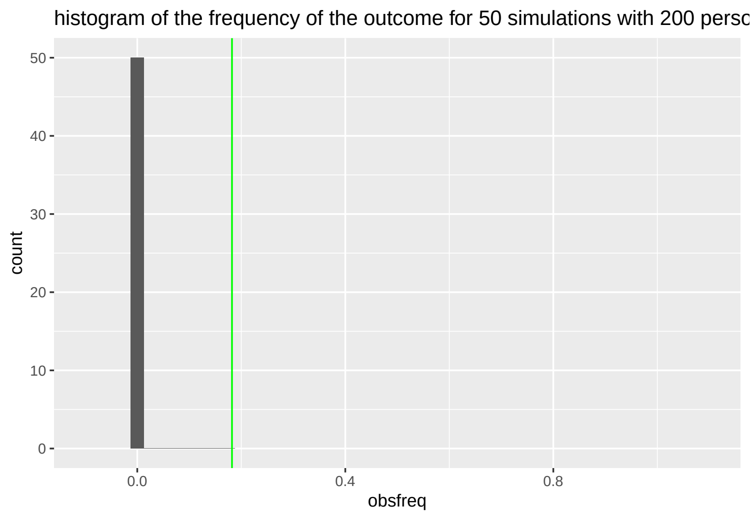
```r
PlasmodeSim::visualOutcome(plpData,50,200,Parameters)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.191 secs
## Prediction took 0.172 secs
```




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

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.17 secs
## Prediction took 0.176 secs
```



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

Visual of a specific covariate

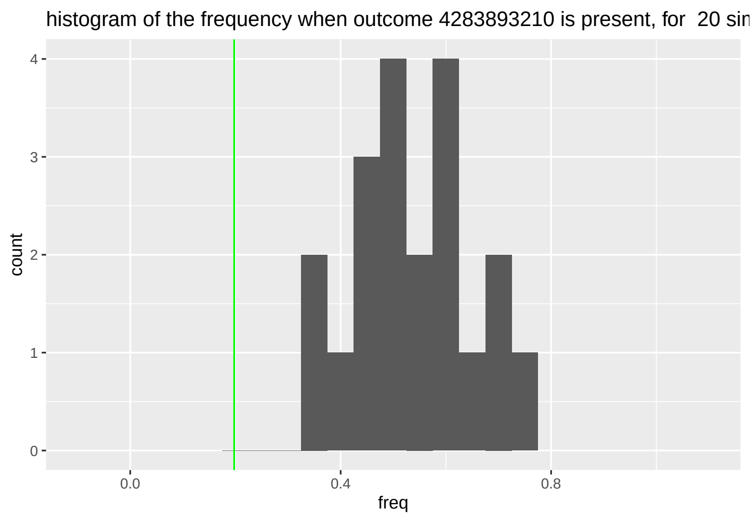
Say we are interested in the outcomes of a group with a specific covariate. Here we picked the third covariate in the model to visualise.

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

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

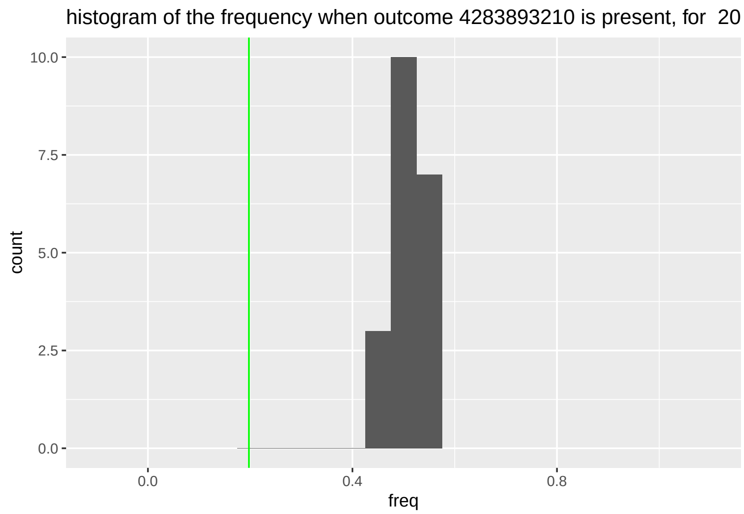
```
PlasmodeSim::visualOutcomeCovariateId(plpData,
                                      covariateIdToStudy,
                                      20,
                                      200,
                                      UnfittedParameters)
```

```
## Removing infrequent and redundant covariates and normalizing
## Removing infrequent and redundant covariates covariates and normalizing took 0.197 secs
## Prediction took 0.19 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 0.18 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.