Package ‘RiskStratifiedEstimation’

July 12, 2019

Type Package

Title Risk Stratified Effect Estimation

Version 0.9.0

Author Alexandros Rekkas

Maintainer Alexandros Rekkas `<a.rekkas@lumc.nl>`

Description RiskStratifiedEstimation is an R package for implementing risk stratified analyses of treamtnet effect in an observational database in the OMOP Common Data Model. It combines functionality of PatientLevelPrediction and CohortMethod R packages to extract and analyze data from a database in the OMOP Common Data Model format. Analyses are performed in two distinct steps. First, a prediction step is implemented, where personalized risks are derived from a large set of covariates applying one of various possible prediction methods. In the second step, risk stratified treatment comparisons are performed. Large scale regularized regression is used to fit the propensity model. Inverse probability of treatment weights are calculated and truncated within strata of predicted risk. Only weighted Cox regression models are supported for the outcome model.

License Apache License 2.0

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports PatientLevelPrediction,
CohortMethod,
Cyclops,
ParallelLogger,
DatabaseConnector,
SqlRender,
foreach,
FeatureExtraction,
dplyr,
ff,
reshape,
caret,
ffbase,
ggplot2,
gridExtra,
survival,
tidyr,
doSNOW,

ggpubr,
survminer

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

`absoluteRiskReduction` *Absolute risk reduction*

Description

Calculates absolute risk reduction based on the Kaplan-Meier estimates within risk strata

Usage

```
absoluteRiskReduction(population, timePoint, psMethod)
```

Arguments

`population`	The study population generated by `matchOnPs` when using propensity score matching or by `stratifyByPs` when stratifying on the propensity score. In case of inverse probability of treatment weighting approach, it is a datframe with a `weights` column.
`timePoint`	The time at which the absolute risk difference is estimated
`psMethod`	Can be one of "matchOnPs", "stratifyByPs" or "inversePtWeighted".

Value

A dataframe with the absolute risk-stratum specific absolute risk difference estimates, along with 95 percent confidence interval.

createGetPlpDataArgs *Create a parameter object for the function getPlpData*

Description

Create a parameter object for the function getPlpData

Usage

```
createGetPlpDataArgs(studyStartDate = "", studyEndDate = "",
  firstExposureOnly = FALSE, washoutPeriod = 0,
  excludeDrugsFromCovariates = FALSE)
```

Arguments

`studyStartDate` A calendar date specifying the minimum date that a cohort indexdate can appear. Date format is 'yyyymmdd'.

`studyEndDate` A calendar date specifying the maximum date that a cohort indexdate can appear. Date format is 'yyyymmdd'. Important: the studyend data is also used to truncate risk windows, meaning no outcomesbeyond the study end date will be considered.

`firstExposureOnly` Should only the first exposure per subject be included? Note thatthis is typically done in the createStudyPopulation function,but can already be done here for efficiency reasons.

`washoutPeriod` The mininum required continuous observation time prior to indexdate for a person to be included in the at risk cohort. Note thatthis is typically done in the createStudyPopulation function,but can already be done here for efficiency reasons.

`excludeDrugsFromCovariates` A redundant option

Details

Create an object defining the parameter values.

createIPW *Creates Inverse Probability Weights*

Description

Calcuates inverse probability weights based on the propensity score

Usage

```
createIPW(ps, weightsType = "ATE", useStabilizedWeights = TRUE,
  truncationLevels = c(0.01, 0.99))
```

Arguments

`ps` A propensity score data frame as created from `createPs`

`weightsType` The type of the weights to be used. Allowed options are 'ATE' for average treatment effect and 'ATT' for average treatment effect on the treated weights

`useStabilizedWeights` Should stabilized weights be used?

`truncationLevels` The level of truncation expressed in percentiles of the propensity score.

Value

The ps data frame provided as input along with a weights column

createRunPlpArgs *Create a parameter object for the function runPlp*

Description

Create a parameter object for the function runPlp

Usage

```
createRunPlpArgs(minCovariateFraction = 0.001, normalizeData = TRUE,
  testSplit = "person", testFraction = 0.25, trainFraction = NULL,
  splitSeed = NULL, nfold = 3, indexes = NULL, savePlpData = TRUE,
  savePlpResult = TRUE, savePlpPlots = TRUE, saveEvaluation = TRUE,
  verbosity = "INFO", timeStamp = FALSE, analysisId = NULL)
```

Arguments

`minCovariateFraction` The minimum fraction of target population who must have a covariate for it to be included in the model training

`normalizeData` Whether to normalise the covariates before training (Default: TRUE) `testSplit` Either 'person' or 'time' specifying the type of evaluation used.'time' find the date where testFraction of patients had an index after the date and assigns patients with an index prior to this date into the training set and post the date into the test set'person' splits the data into test (1-testFraction of the data) andtrain (validationFraction of the data) sets. The split is stratified by the class label.

`testFraction` The fraction of the data to be used as the test set in the patientsplit evaluation.

`trainFraction` A real number between 0 and 1 indicating the train set fraction of the data.If not set trainFraction is equal to 1 - test

`splitSeed` The seed used to split the test/train set when using a person type testSplit

`nfold` The number of folds used in the cross validation (default 3)

`indexes` A dataframe containing a rowId and index column where the index value of -1 means in the test set, and positive integer represents the cross validation fold (default is NULL)

`savePlpData` Binary indicating whether to save the plpData object (default is T)

`savePlpResult` Binary indicating whether to save the object returned by runPlp (default is T)

`savePlpPlots` Binary indicating whether to save the performance plots as pdf files (default is T)

`saveEvaluation` Binary indicating whether to save the oerformance as csv files (default is T)

`verbosity` Sets the level of the verbosity. If the log level is at or higher in priority than the logger threshold, a message will print. The levels are:DEBUGHighest verbosity showing all debug statementsTRACEShowing information about start and end of stepsINFOShow informative information (Default)WARNShow warning messagesERRORShow error messagesFATALBe silent except for fatal errors

`timeStamp` If TRUE a timestamp will be added to each logging statement. Automatically switched on for TRACE level.

`analysisId` Identifier for the analysis. It is used to create, e.g., the result folder. Default is a timestamp.

Details

Create an object defining the parameter values.

createStudyPopulationCmSettings

Create a parameter object for the function createStudyPopulation

Description

Create a parameter object for the function createStudyPopulation

Usage

```
createStudyPopulationCmSettings(population = NULL,
  firstExposureOnly = FALSE, restrictToCommonPeriod = FALSE,
  washoutPeriod = 0, removeDuplicateSubjects = FALSE,
  removeSubjectsWithPriorOutcome = TRUE, priorOutcomeLookback = 99999,
  minDaysAtRisk = 1, riskWindowStart = 0,
  addExposureDaysToStart = FALSE, riskWindowEnd = 0,
  addExposureDaysToEnd = TRUE, censorAtNewRiskWindow = FALSE)
```

Arguments

`population` If specified, this population will be used as the startingpoint instead of the cohorts in the cohortMethodDataobject.

`firstExposureOnly` Should only the first exposure per subject be included? Notethat this is typically done in thecreateStudyPopulation function,

`restrictToCommonPeriod` Restrict the analysis to the period when both exposures are observed?

`washoutPeriod` The mininum required continuous observation time prior toindex date for a person to be included in the cohort.

`removeDuplicateSubjects` Remove subjects that are in both the target and comparatorcohort? See details for allowed values.

`removeSubjectsWithPriorOutcome` Remove subjects that have the outcome prior to the riskwindow start?

`priorOutcomeLookback` How many days should we look back when identifying prioroutcomes?

`minDaysAtRisk` The minimum required number of days at risk.

`riskWindowStart` The start of the risk window (in days) relative to the indexdate (+ days of exposure if theaddExposureDaysToStart parameter is specified).

`addExposureDaysToStart` Add the length of exposure the start of the risk window?

`riskWindowEnd` The end of the risk window (in days) relative to the indexdata (+ days of exposure if the addExposureDaysToEndparameter is specified).

`addExposureDaysToEnd` Add the length of exposure the risk window?

`censorAtNewRiskWindow` If a subject is in multiple cohorts, should time-at-risk be censoredwhen the new time-at-risk starts to prevent overlap?

Details

Create an object defining the parameter values.

`fitOutcomeModels` *Fit outcome models*

Description

Fits outcome models within risk strata, estimating relative and absolute differences. Designed to be performed within a parellelized analysis.

Usage

```
fitOutcomeModels(outcomeId, analysisPath, cohortMethodDataFolder,
  outcomeIdList, compareAllOutcomes, timePoint, psMethod, weightsType,
  useStabilizedWeights, truncationLevels, populationCmSettings)
```

Arguments

`outcomeId` The outcome of interest for which the esitmation is performed. That is the outcome for which risk stratification is performed.

`analysisPath` The path to the `RSEE` analysis results.

`cohortMethodDataFolder` The directory where the `cohortMethodData` object is stored.

`outcomeIdList` The other outcomes for which risk stratified estimates need to derived. Only required `compareAllOutcomes` is `TRUE`.

`compareAllOutcomes` Compare all the outcomes within the `outcomeIdList` within risk strata of code-outcomeId?

`timePoint` The time point at which absolute risk differences will be calculated.

`psMethod` Select the propensity score method for the estimation of treatment effects within risk strata. It can be "matchOnPs", "stratifyByPs" or "inversePtWeighted".

`weightsType` Only required if `weightsType` is "inversePtWeighted". The type of weights for the balancing of covariates. Should be either 'ATE' or 'ATT'

`useStabilizedWeights` Only required if `weightsType` is "inversePtWeighted". Should stabilized weights be used?

`truncationLevels` Only required if `weightsType` is "inversePtWeighted". The level of truncation expressed in percentiles of the propensity score.

`populationCmSettings` A parameter object for the function `createStudyPopulation`. Can be generated from function `createStudyPopulationCmSettings`.

Value

`NULL`. The results are all saved.

fitPsModel *Calculate propensity scores for a specific outcome*

Description

Fits a large-scale regularized regression model to estimate propensity scores within predicted risk strata. Designed to be applied in a parallelized analysis.

Usage

```
fitPsModel(cohortMethodDataFolder, plpDataFolder, outcomeId,
  populationCmSettings, populationPlpSettings, saveDirectory, riskStrata,
  analysisId, analysisPath)
```

Arguments

`cohortMethodDataFolder`
The directory where the `cohortMethodData` object is stored.

`plpDataFolder` The directory where the `plpData` object is stored.

`outcomeId` The outcome of interest for which the risk stratification is performed.

`populationCmSettings`
A parameter object for the function `createStudyPopulation`. Can be generated from function `createStudyPopulationCmSettings`.

`populationPlpSettings`
A parameter object for the function `createStudyPopulation`. Can be generated from unction `createStudyPopulationSettings`.

`saveDirectory` The direcotry of the RSEE analysis where the prediction models are located.

`riskStrata` The considered number of risk strata.

`analysisId` The analysis ID of the prediction model used to stratify the population.

`analysisPath` The directory where the propensity scores will be stored.

Value

`NULL`. The results are all saved.

`loadRSEE` *Load the result of a risk stratified analysis*

Description

Loads the result of a risk stratified analysis.

Usage

```
loadRSEE(file, mainOutcomes = NULL, loadOtherOutcomes = TRUE,
  otherOutcomes = NULL, loadPs = TRUE)
```

Arguments

`file` The file location where the results are stored. It should point at the analysisId folder of a risk stratified analysis

`mainOutcomes` The main outcomes for which the results should be loaded. If set to `NULL` the results for all the outcomes are loaded.

`loadOtherOutcomes`
Logical: Whether to load results for other outcomes. If `FALSE`, only the main outcome results are loaded.

`otherOutcomes` The Ids of the other outcomes to be loaded.

`loadPs` Should the propensity scores along with the matrices mapping risk stratification be loaded?

Value

The result of a previous risk stratified analysis.

`outcomeModelWeighted` *Fits a weighted cox regression model*

Description

Fits a weighted cox regression model using an inverse probability of treatment weighting approach

Usage

```
outcomeModelWeighted(ps, calculateWeights = TRUE, weightsType = "ATE",
  useStabilizedWeights = TRUE, truncationLevels)
```

Arguments

`ps` A dataframe wiht propensity scores as generated from `createPs`.

`calculateWeights` Should weights be calculated?

`weightsType` The type of weights for the balancing of covariates. Should be either 'ATE' or 'ATT'

`useStabilizedWeights` Should stabilized weights be used?

`truncationLevels` The level of truncation expressed in percentiles of the propensity score.

Value

A weighted cox regression model.

`plotCovariateBalance` *Plots the covariate balance before and after balancing*

Description

Plots covariate before and after weighting using the inverse of the propensity score

Usage

```
plotCovariateBalance(ps, cohortMethodData, calculateWeights = TRUE,
  weightsType = "ATE", useStabilizedWeights = TRUE, truncationLevels,
  showNotBalancedCovariateIds = TRUE)
```

Arguments

`ps` A propensity score data frame as created from `createPs`

`cohortMethodData`
A cohortMethodData object

`calculateWeights`
Should the weights be calculated?

`weightsType` The type of the weights to be used. Allowed options are 'ATE' for average treatment effect and 'ATT' for average treatment effect on the treated weights

`useStabilizedWeights`
Should stabilized weights be used?

`truncationLevels`
The level of truncation expressed in percentiles of the propensity score.

`showNotBalancedCovariateIds`
Show covariate ids that were not balanced after weighting?

Value

The covariate balance plot

plotRSEE *Plot the results of a risk stratified analysis*

Description

Plots the overall results of a risk stratified analysis.

Usage

```
plotRSEE(rseeResult, mapOutcomes = NULL, mapTreatments = NULL)
```

Arguments

`rseeResult` The overall result of a risk stratified analysis.

`mapOutcomes` A dataframe with the outcome labels. It should have 2 columns called "outcome" and "labelOutcome". The former should contain character values of the form "outcome_x" and the latter should contain the outcome label. If set to `NULL`, the outcome defintion id's appear in the graphs.

`mapTreatments` A dataframe with the treatment labels. It should have 2 columns named "cohort" and "labelTreatments". The former should contain the values "treatment" and "comparator" and the latter should contain the treatment labels. If set to `NULL`, the labels treatment and comaprator appear in the graph,

Value

A 3-level graph. In the first level observed outcome rates are presented for all treatment-outcome combinations across risk strata. In the second level, hazard ratios across risk strata are given. In the final level, absolute risk difference across risk strata are presented.

prepareForPlpData *Prepares for the running the PatientLevelPrediction package*

Description

Prepares for running the PatientLevelPrediction package by merging the treatment and comparator cohorts and defining a new covariate for treatment.

Usage

```
prepareForPlpData(treatmentCohortId, comparatorCohortId, targetCohortId,
  cohortDatabaseSchema, cohortTable, resultsDatabaseSchema,
  mergedCohortTable, attributeDefinitionTable, cohortAttributeTable,
  connectionDetails)
```

Arguments

`treatmentCohortId`
The treatment cohort id

`comparatorCohortId`
The comparator cohort id

`targetCohortId` The id of the merged cohorts

`cohortDatabaseSchema`
The name of the database schema that is the location where the cohort data used to define the at risk cohort is available

`cohortTable` The table that contains the treatment and comparator cohorts.

`resultsDatabaseSchema`
The name of the database schema to store the new tables. Need to have write access.

`mergedCohortTable`
The table that will contain the merged cohorts.

`attributeDefinitionTable`
The table that will contain the definition of the treatment variable.

`cohortAttributeTable`
The table that will contain the patients along with their new covariate values.

`connectionDetails`
The connection details required to connect to a database.

Value

Creates the tables resultsDatabaseSchema.mergedCohortTable, resultsDatabaseSchema.attributeDefinitionTable and resultsDatabaseSchema.cohortAttributeTable

A covariate settings object for the treatment covariate.

relativeRiskReduction *Relative risk reduction*

Description

Calculates hazard ratios within risk strata.

Usage

```
relativeRiskReduction(model)
```

Arguments

model	The model that was used to fit a cox regression model to the data.

Value

A dataframe with hazard ratios for treatment effect across risk strata along with 95 percent confidence intervals

runRiskStratifiedEstimation *Runs a risk stratified analysis*

Description

Runs a risk stratified analysis in two stages. It first runs a prediction algorithm using PatientLevelPrediction to derive baseline patient risks and then derives estimates within risk strata using CohortMethod package.

Usage

```
runRiskStratifiedEstimation(plpDataFolder = NULL,
  cohortMethodDataFolder = NULL, cdmDatabaseSchema, cohortDatabaseSchema,
  outcomeDatabaseSchema, resultsDatabaseSchema, cohortTable, outcomeTable,
  mergedCohortTable, attributeDefinitionTable, cohortAttributeTable,
  treatmentCohortId, comparatorCohortId, outcomeIds, targetCohortId,
  connectionDetails, getDbCohortMethodDataArgs, covariateSettingsCm,
  populationCmSettings, exposureTable, psControl = NULL,
  psPrior = NULL, psMethod = "matchOnPs", createPsThreads = 1,
  exposureDatabaseSchema, getPlpDataArgs, covariateSettingsPlp,
  modelSettings, populationPlpSettings, cdmVersion = "5", runPlpArgs,
  riskStrata = 4, weightsType = "ATE", useStabilizedWeights = TRUE,
  truncationLevels = c(0.01, 0.99), timePoint,
  compareAllOutcomes = TRUE, predictionThreads = 1, saveResults,
  saveDirectory = NULL, fftempdir, fitOutcomeModelsThreads = 1,
  saveMapMatrix = TRUE, savePs = TRUE, verbosity = "INFO",
  analysisId = NULL)
```

Arguments

`plpDataFolder` Folder where the `plpData` object is stored. If `NULL`, it will be constructed within the function.

`cohortMethodDataFolder`
Folder where the `cohortMethodData` object is stored. If `NULL`, it will be constructed within the function.

`cdmDatabaseSchema`
The name of the database schema that contains the vocabulary files. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.

`cohortDatabaseSchema`
The name of the database schema that contains the treatment and comaparator cohorts. Requires read permissions to this database.

`outcomeDatabaseSchema`
The name of the database schema that contains the outcome cohorts. Requires read permissions to this database.

`resultsDatabaseSchema`
The name of the database schema with write permissions.

`cohortTable` The name of the table holding the treatment and comparator cohorts.

`outcomeTable` The name of the table hodling the outcome cohorts.

`mergedCohortTable`
The name of the table where the merged treatment and comparator cohorts will be stored.

`attributeDefinitionTable`
The table where the definition of the treatment covariate will be stored.

`cohortAttributeTable`
The table where the covariate values with regard to treatment will be stored.

`treatmentCohortId`
The cohort definition id of the treatment cohort in the cohortTable.

`comparatorCohortId`
The cohort definition id of the comparator cohort in the cohortTable.

`outcomeIds` A list of cohort definition ids used to define the outcomes in the outcome table.

`targetCohortId` The cohrt definition id of of the merged cohort in the mergedCohortTable.

`connectionDetails`
An R object of type `connectionDetails` created using function `createConnectionDetails`. Either the `connection` or the `connectionDetails` argument should be specified.

`getDbCohortMethodDataArgs`
A parameter object for the function `getDbCohortMethodData`. Can be generated from function `createGetDbCohortMethodDataArgs`.

`covariateSettingsCm`
An object of type `covariateSettings` as created using the `createCovariateSettings` to be used for the definition of the `cohortMethodData` object.

`populationCmSettings`
A parameter object for the function `createStudyPopulation`. Can be generated from function `createStudyPopulationCmSettings`.

`exposureTable` Input of function `getDbCohortMethodData`: The tablename that contains the exposure cohorts.

`psControl` An object of the type `cyclopsControl` generated from `createControl`.

`psPrior` An object of the type `cyclopsPrior` generated from `createPrior`.

`psMethod` Select the propensity score method for the estimation of treatment effects within risk strata. It can be "matchOnPs", "stratifyByPs" or "inversePtWeighted".

`createPsThreads` The number of threads for the calculation of the propensity scores.

`exposureDatabaseSchema` Input of function `getDbCohortMethodData`: The name of the database schema that is the location where the exposure data used to define the exposure cohorts is available.

`getPlpDataArgs` A parameter object for the function `getPlpData`. It can be generated from function `createGetPlpDataArgs`.

`covariateSettingsPlp` An object of type `covariateSettings` as created using the `createCovariateSettings` to be used for the definition of the `plpData` object. note that a covariate indicating treatment will be added.

`modelSettings` An object of the class `modelSettings` to be used as input for `runPlp`.

`populationPlpSettings` A parameter object for the function `createStudyPopulation`. Can be generated from function `createStudyPopulationSettings`.

`cdmVersion` Define the OMOP CDM version used: currently supported is "5".

`runPlpArgs` A parameter object for the function `runPlp`. Can be generated from function `createRunPlpArgs`.

`riskStrata` The number of risk strata to divide the study population.

`weightsType` Only required if `weightsType` is "inversePtWeighted". The type of weights for the balancing of covariates. Should be either 'ATE' or 'ATT'

`useStabilizedWeights` Only required if `weightsType` is "inversePtWeighted". Should stabilized weights be used?

`truncationLevels` Only required if `weightsType` is "inversePtWeighted". The level of truncation expressed in percentiles of the propensity score.

`timePoint` The time point of interest for the calculation of the absolute risk reduction.

`compareAllOutcomes` Should all the outcomes be analyzed within all stratifications?

`predictionThreads` The number of threads to be used to run the predictions.

`saveResults` Should the results of the entire analysis be saved?

`saveDirectory` The file path to the directory where the results of the analysis will be saved.

`fftempdir` The directory where the temporary `ff` files will be saved.

`fitOutcomeModelsThreads` The number of threadss to be used for the calculation of the risk stratified results.

`saveMapMatrix` Should the map matrix with the risk sratum allocations be saved?

`savePs` Should the propensity scores be saved?

`verbosity` Sets the level of the verbosity. If the log level is at or higher in priority than the logger threshold, a message will print. The levels are:

- DEBUGHighest verbosity showing all debug statements
- TRACEShowing information about start and end of steps
- INFOShow informative information (Default)
- WARNShow warning messages
- ERRORShow error messages
- FATALBe silent except for fatal errors

.

`analysisId` The identifier of the analysis.

Value

An object containing two large lists:

- The propensity scores within risk strata for each outcome in `outcomeIds`
- The results of the risk stratified analyis within risk strata for all outcomes in `outcomeIds`:
  - Weighted Kaplan-Meier estimates.
  - Relative risk reduction.
  - Absolute risk reduction.
  - Number of cases.

`singlePlotRSEE` *Plot a single outcome*

Description

Plots the result of a risk stratified analysis for a single outcome of interest

Usage

```
singlePlotRSEE(outcomeId, mapMatrix, title = NULL,
  overallResult = NULL, mapTreatments = NULL)
```

Arguments

`outcomeId` The outcome Id of interest

`mapMatrix` The matrix that maps patients to their risk quantiles

`title` The title of the graph

`overallResult` The hazard ratio from an analysis on the overall study population

`mapTreatments` A dataframe with the treatment labels. It should have 2 columns named "cohort" and "labelTreatments". The former should contain the values "treatment" and "comparator" and the latter should contain the treatment labels. If set to `NULL`, the labels treatment and comaprator appear in the graph.

Value

A 3-level graph for a single outcome. In the first level observed outcome rates are presented for all treatment-outcome combinations across risk strata. In the second level, hazard ratios across risk strata are given. In the final level, absolute risk difference across risk strata are presented.

`stratifiedKaplanMeier` *Calculate stratified Kaplan-Meier estimates*

Description

Calculate stratified Kaplan-Meier estimates

Usage

```
stratifiedKaplanMeier(population, timePoint)
```

Arguments

`population`	The population of interest stratified using the `stratifyByPs`
`timePoint`	The point in time for which the absolute risk difference is required

Value

A vector of the absolute risk difference along with the lowest and highest limits of the the 95 percent confidence interval

`weightedKM` *Calculates the weighted Kaplan-Meier estimates*

Description

Calculates the weighted Kaplan-Meier estimates.

Usage

```
weightedKM(ps, calculateWeights = TRUE, weightsType = "ATE",
  useStabilizedWeights = TRUE, truncationLevels)
```

Arguments

`ps`	A data frame including the propensity scores as generated from `createPs`
`calculateWeights`	Whether to calculate the weights using `createIPW`
`weightsType`	The type of the weights to be used. Allowed options are 'ATE' for average treatment effect and 'ATT' for average treatment effect on the treated weights
`useStabilizedWeights`	Should stabilized weights be used?
`truncationLevels`	The level of truncation expressed in percentiles of the propensity score.

Value

A data frame with the Kaplan-Meier estimates

References

Xie J, Liu C. Adjusted Kaplan-Meier estimator and log-rank test with inverse probability of treatment weighting for survival data. Statistics in Medicine 2005; 2:3089–3110.

Index