Package ‘dsld’

July 22, 2025

Version 0.2.2

Title Data Science Looks at Discrimination

Maintainer Norm Matloff <`nsmatloff@ucdavis.edu`>

VignetteBuilder knitr

Imports Kendall, ranger, ggplot2, plotly, freqparcoord, fairness,sandwich

Depends R (>= 3.5.0), fairml, gtools, regtools,qeML,rmarkdown

Suggests knitr,bnlearn,Matching,randomForest

License GPL (>= 2)

Description Statistical and graphical tools for detecting and measuring discrimination and bias, be it racial, gender, age or other. Detection and remediation of bias in machine learning algorithms. 'Python' interfaces available.

URL `https://github.com/matloff/dsld`

BugReports `https://github.com/matloff/dsld/issues`

NeedsCompilation no

Author Norm Matloff [aut, cre] (ORCID:
<`https://orcid.org/0000-0001-9179-6785`>),
Taha Abdullah [aut],
Arjun Ashok [aut],
Shubhada Martha [aut],
Aditya Mittal [aut],
Billy Ouattara [aut],
Jonathan Tran [aut],
Brandon Zarate Estrada [aut]

Repository CRAN

Date/Publication 2024-09-13 18:20:09 UTC

Contents

`compas1` *Criminal Offenders Screened in Florida*

Description

A collection of criminal offenders screened in Florida (US) during 2013-14. This data was used to predict recidivism.

Additional details for this dataset can be found via the **fairml** package.

`dsldBnlearn` *dsldBnlearn*

Description

Wrappers for functions in the **bnlearn** package. (Just (Presently, just `iamb`.)

Usage

```
dsldIamb(data)
```

Arguments

`data`	Data frame.

Details

Under very stringent assumptions, `dsldIamb` performs causal discovery, i.e. fits a causal model to `data`.

Value

Object of class 'bn' (**bnlearn** object). The generic `plot` function is callable on this object.

Author(s)

N. Matloff

Examples

```
data(svcensus)
# iamb does not accept integer data
svcensus$wkswrkd <- as.numeric(svcensus$wkswrkd)
svcensus$wageinc <- as.numeric(svcensus$wageinc)
iambOut <- dsldIamb(svcensus)
plot(iambOut)
```

`dsldCHunting and dsldOHunting` *Confounder and Proxy Hunting*

Description

Confounder hunting: searches for variables C that predict both Y and S. Proxy hunting: searches for variables O that predict S.

Usage

```
dsldCHunting(data,yName,sName,intersectDepth=10)
dsldOHunting(data,yName,sName)
```

Arguments

`data`	Data frame.
`yName`	Name of the response variable column.
`sName`	Name of the sensitive attribute column.
`intersectDepth`	Maximum size of intersection of the Y predictor set and the S predictor set

Details

`dsldCHunting`: The random forests function `qeML:qeRF` will be run on the indicated data to indicate feature importance in prediction of Y (without S) and S (without Y). Call these "important predictors" of Y and S.

Then for each `i` from 1 to `intersectDepth`, the intersection of the top `i` important predictors of Y and the the top `i` important predictors of S will be reported, thus suggesting possible confounders. Larger values of `i` will report more potential confounders, though including progressively weaker ones.

The analyst then may then consider omitting the variables C from models of the effect of S on Y.

Note: Run times may be long.

`dsldOHunting`: Factors, if any, will be converted to dummy variables, and then the Kendall Tau correlations will be calculated betwene S and potential proxy variables O, i.e. every column other than Y and S. (The Y column itself doesn't enter into computation.)

In fairness analyses, in which one desires to either eliminate or reduce the impact of S, one must consider the indirect effect of S via O. One may wish to eliminate or reduce the role of O.

Value

The function `dsldCHunting` returns an R list, one component for each confounder set found.

The function `dsldOHunting` returns an R matrix of correlations, one row for each level of S.

Author(s)

N. Matloff

Examples

```
data(lsa)
dsldCHunting(lsa,'bar','race1')
# e.g. suggests confounders 'decile3', 'lsat'

data(mortgageSE)
dsldOHunting(mortgageSE,'deny','black')
# e.g. suggests using loan value and condo purchase as proxies
```

`dsldConditDisparity` *dsldConditDisparity*

Description

Plots (estimated) mean Y against X, separately for each level of S, with restrictions `condits`. May reveal Simpson's Paradox-like differences not seen in merely plotting mean Y against X.

Usage

```
dsldConditDisparity(data, yName, sName, xName, condits = NULL,
    qeFtn = qeKNN, minS = 50, useLoess = TRUE)
```

Arguments

`data`	Data frame or equivalent.
`yName`	Name of predicted variable Y. Must be numeric or dichtomous R factor.
`sName`	Name of the sensitive variable S, an R factor
`xName`	Name of a numeric column for the X-axis.
`condits`	An R vector; each component is a character string for an R logical expression representing a desired condition involving `names(data)` other than S and Y.
`qeFtn`	qeML predictive function (not quoted; only default arguments will be used.)
`minS`	Minimum size for an S group to be retained in the analysis.
`useLoess`	If TRUE, do loess smoothing on the fitted regression values.

Value

No value; plot.

Author(s)

N. Matloff, A. Ashok, S. Martha, A. Mittal

Examples

```
data(compas)
# graph probability of recidivism by race given age, among those with at
# most 4 prior convictions and COMPAS decile score at least 6
compas$two_year_recid <- as.numeric(compas$two_year_recid == "Yes")
dsldConditDisparity(compas,"two_year_recid", "race", "age",
    c("priors_count <= 4","decile_score>=6"), qeKNN)

dsldConditDisparity(compas,"two_year_recid", "race", "age",
    "priors_count == 0", qeGBoost)
```

`dsldConfounders` *dsldConfounders*

Description

Plots estimated densities of all continuous features X, conditioned on a specified categorical feature C.

Usage

```
dsldConfounders(data, sName, graphType = "plotly", fill = FALSE)
```

Arguments

`data`	Dataframe, at least 2 columns.
`sName`	Name of the categorical column, an R factor. In discrimination contexts, Typically a sensitive variable.
`graphType`	Either "plot" or "plotly", for static or interactive graphs. The latter requires the **plotly** package.
`fill`	Only applicable to graphType = "plot" case. Setting to true will color each line down to the x-axis.

Value

No value; plot.

Author(s)

N. Matloff, T. Abdullah, A. Ashok, J. Tran

Examples

```
data(svcensus)
dsldConfounders(svcensus, "educ")
```

`dsldDensityByS` *dsldDensityByS*

Description

Graphs densities of a response variable, grouped by a sensitive variable. Similar to `dsldConfounders`, but includes sliders to control the bandwidth of the density estimate (analogous to controlling the bin width in a histogram).

Usage

```
dsldDensityByS(data, cName, sName, graphType = "plotly", fill = FALSE)
```

Arguments

`data`	Datasetwith at least 1 numerical column and 1 factor column
`cName`	Possible confounding variable column, an R numeric
`sName`	Name of the sensitive variable column, an R factor
`graphType`	Type of graph created. Defaults to "plotly".
`fill`	To fill the graph. Defaults to "FALSE".

Value

No value; plot.

Author(s)

N. Matloff, T. Abdullah, A. Ashok, J. Tran

Examples

```
data(svcensus)
dsldDensityByS(svcensus, cName = "wageinc", sName = "educ")
```

`dsldEDFFair Wrappers` *dsldEDFFair Wrappers*

Description

Explicitly Deweighted Features: control the effect of proxies related to sensitive variables for prediction.

Usage

```
dsldQeFairKNN(data, yName, sNames, deweightPars=NULL, yesYVal=NULL,k=25,
  scaleX=TRUE, holdout=floor(min(1000,0.1*nrow(data))))
dsldQeFairRF(data,yName,sNames,deweightPars=NULL, nTree=500, minNodeSize=10,
  mtry = floor(sqrt(ncol(data))),yesYVal=NULL,
  holdout=floor(min(1000,0.1*nrow(data))))
dsldQeFairRidgeLin(data, yName, sNames, deweightPars = NULL,
  holdout=floor(min(1000,0.1*nrow(data))))
dsldQeFairRidgeLog(data, yName, sNames, deweightPars = NULL, holdout =
  floor(min(1000, 0.1 * nrow(data))), yesYVal = levels(data[, yName])[2])
## S3 method for class 'dsldQeFair'
predict(object,newx,...)
```

Arguments

`data`	Dataframe, training set.
`yName`	Name of the response variable column.
`sNames`	Name(s) of the sensitive attribute column(s).
`deweightPars`	Values for de-emphasizing variables in a split, e.g. 'list(age=0.2,gender=0.5)'. In the linear case, larger values means more deweighting, i.e. less influence of the given variable on predictions. For KNN and random forests, smaller values mean more deweighting.
`scaleX`	Scale the features. Defaults to TRUE.
`yesYVal`	Y value to be considered "yes," to be coded 1 rather than 0.

`k`	Number of nearest neighbors. In functions other than `dsldQeFairKNN` for which this is an argument, it is the number of neighbors to use in finding conditional probabilities via knnCalib.
`holdout`	How many rows to use as the holdout/testing set. Can be NULL. The testing set is used to calculate s correlation and test accuracy.
`nTree`	Number of trees.
`minNodeSize`	Minimum number of data points in a tree node.
`mtry`	Number of variables randomly tried at each split.
`object`	An object returned by the dsld-EDFFAIR wrapper.
`newx`	New data to be predicted. Must be in the same format as original data.
`...`	Further arguments.

Details

The sensitive variables S are removed entirely, but there is concern that they still affect prediction indirectly, via a set C of proxy variables.

Linear EDF reduces the impact of the proxies through a shinkage process similar to that of ridge regression. Specifically, instead of minimizing the sum of squared errors SSE with respect to a coefficient vector b, we minimize SSE + the squared norm of Db, where D is a diagonal matrix with nonzero elements corresponding to C. Large values penalizing variables in C, thus shrinking them.

KNN EDF reduces the weights in Euclidean distance for variables in C. The random forests version reduces the probabilities that a proxy will be used in splitting a node.

By using various values of the deweighting parameters, the user can choose a desired position in the Fairness-Utility Tradeoff.

More details can be found in the references.

Value

The EDF functions return objects of class 'dsldQeFair', which include components for test and base accuracy, summaries of inputs and so on.

Author(s)

N. Matloff, A. Mittal, J. Tran

References

https://github.com/matloff/EDFfair

See Also

Matloff, Norman, and Wenxi Zhang. "A novel regularization approach to fair ML."
`arXiv preprint arXiv:2208.06557` (2022).

Examples

```
data(compas1)
data(svcensus)

# dsldQeFairKNN: deweight "decile score" column with "race" as
# the sensitive variable
knnOut <- dsldQeFairKNN(compas1, "two_year_recid", "race",
   list(decile_score=0.1), yesYVal = "Yes")
knnOut$testAcc
knnOut$corrs
predict(knnOut, compas1[1,-8])

# dsldFairRF: deweight "decile score" column with "race" as sensitive variable
rfOut <- dsldQeFairRF(compas1, "two_year_recid", "race",
   list(decile_score=0.3), yesYVal = "Yes")
rfOut$testAcc
rfOut$corrs
predict(rfOut, compas1[1,-8])

# dsldQeFairRidgeLin: deweight "occupation" and "age" columns
lin <- dsldQeFairRidgeLin(svcensus, "wageinc", "gender", deweightPars =
  list(occ=.4, age=.2))
lin$testAcc
lin$corrs
predict(lin, svcensus[1,-4])

# dsldQeFairRidgeLin: deweight "decile score" column
log <- dsldQeFairRidgeLog(compas1, "two_year_recid", "race",
  list(decile_score=0.1), yesYVal = "Yes")
log$testAcc
log$corrs
predict(log, compas1[1,-8])
```

`dsldFairML Wrappers` *dsldFairML Wrappers*

Description

Fair machine learning models: estimation and prediction. The following functions provide wrappers for some functions in the **fairML** package.

Usage

```
dsldFrrm(data, yName, sName, unfairness, definition = "sp-komiyama",
   lambda = 0, save.auxiliary = FALSE)
dsldFgrrm(data, yName, sName, unfairness, definition = "sp-komiyama",
   family = "binomial", lambda = 0, save.auxiliary = FALSE)
dsldNclm(data, yName, sName, unfairness, covfun = cov, lambda = 0,
```

```
   save.auxiliary = FALSE)
dsldZlm(data, yName, sName, unfairness)
dsldZlrm(data, yName, sName, unfairness)
```

Arguments

`data`	Data frame.
`yName`	Name of the response variable column.
`sName`	Name(s) of the sensitive attribute column(s).
`unfairness`	A number in (0, 1]. Degree of unfairness allowed in the model. A value (very near) 0 means the model is completely fair, while a value of 1 means the model is not constrained to be fair at all.
`covfun`	A function computing covariance matrices.
`definition`	Character string, the label of the definition of fairness. Currently either 'sp-komiyama', 'eo-komiyama' or 'if-berk'.
`family`	A character string, either 'gaussian' to fit linear regression, 'binomial' for logistic regression, 'poisson' for log-linear regression, 'cox' for Cox proportional hazards regression, or 'multinomial' for multinomial logistic regression.
`lambda`	Non-negative number, a ridge-regression penalty coefficient.
`save.auxiliary`	A logical value, whether to save the fitted values and the residuals of the auxiliary model that constructs the debiased predictors.

Details

See documentation for the **fairml** package.

Value

An object of class 'dsldFairML', which includes the model information, `yName`, and `sName`.

Author(s)

S. Martha, A. Mittal, B. Ouattara, B. Zarate, J. Tran

Examples

```
data(svcensus)
data(compas1)

yName <- "wageinc"
sName <- "age"
frrmOut <- dsldFrrm(svcensus, yName, sName, 0.2, definition = "sp-komiyama")
summary(frrmOut)
predict(frrmOut, svcensus[1:10,])

yName <- "two_year_recid"
sName <- "age"
```

```
fgrrmOut <- dsldFgrrm(compas1, yName, sName, 0.2, definition = "sp-komiyama")
summary(fgrrmOut)
predict(fgrrmOut, compas1[c(1:10),])
```

`dsldFairUtilTrade` *dsldFairUtilTrade*

Description

Exploration of the Fairness-Utility Tradeoff. Finds predictive accuracy and correlation between S and predicted Y.

Usage

```
dsldFairUtilTrade(data,yName,sName,dsldFtnName,
   unfairness=NULL,deweightPars=NULL,yesYVal=NULL,yesSVal=NULL,
   corrType='kendall', holdout = floor(min(1000, 0.1 * nrow(data))))
```

Arguments

`data`	Data frame.
`yName`	Name of the response variable Y column. Y must be numeric or binary (two-level R factor).
`sName`	Name of the sensitive attribute S column. S must be numeric or binary (two-level R factor).
`dsldFtnName`	Quoted name of one of the **fairML** or EDF functions.
`unfairness`	Nonnull for the **fairML** functions.
`deweightPars`	Nonnull for the EDF functions.
`yesYVal`	Y value to be treated as Y = 1 for binary Y.
`yesSVal`	S value to be treated as S = 1 for binary S.
`corrType`	Either 'kendall' or 'probs'.
`holdout`	Size of holdout set.

Details

Tool for exploring tradeoff between utility (predictive accuracy, Mean Absolute Prediction Error or overall probability of misclassification) and fairness. Roughly speaking, the latter is defined as the strength of relation between S and predicted Y (the smaller, the better). The main issue is definition of "relation" in the case of binary Y or S:

In the 'kendall' case, binary predicted Y or S is recoded to 1s and 0s, and Kendall correlation is used. In the 'probs' case, binary Y or S is replaced by P(Y = 1 | X) and P(S = 1 | X); squared Pearson correlation is then computed.

Value

A two-component vector, consisting of predictive accuracy and strength of relation between S and predicted Y.

Author(s)

N. Matloff

Examples

```
data(svcensus)
dsldFairUtilTrade(svcensus,'wageinc','gender','dsldFrrm',0.2,yesSVal='male')
data(lsa)
race1 <- lsa$race1
lsabw <- lsa[race1 == 'black' | race1 == 'white',]
# need to get rid of excess levels
race1 <- lsabw$race1
race1 <- as.character(race1)
lsabw$race1 <- as.factor(race1)
dsldFairUtilTrade(lsabw,'bar','race1','dsldQeFairRidgeLog',
   deweightPars=list(fam_inc=0.1),yesYVal='TRUE',yesSVal='white')
```

`dsldFreqPCoord` *dsldFreqPCoord*

Description

Wrapper for the `freqparcoord` function from the **freqparcoord** package.

Usage

```
dsldFreqPCoord(data, m, sName = NULL, method
    = "maxdens", faceting = "vert", k = 50, klm = 5 * k, keepidxs = NULL,
    plotidxs = FALSE, cls = NULL, plot_filename = NULL)
```

Arguments

`data`	Data frame or matrix.
`m`	Number of lines to plot for each group. A negative value in conjunction with the method `maxdens` indicates that the lowest-density lines are to be plotted. If method is `locmax`, then `m` is forced to 1.
`sName`	Column for the grouping variable, if any (if none, all the data is treated as a single group); the column must be a vector or factor. The column must not be in `dispcols`. If method is `locmax`, `grpvar` is forced to NULL
`method`	What to display: 'maxdens' for plotting the most (or least) typical lines, 'locmax' for cluster hunting, or 'randsamp' for plotting a random sample of lines.

`faceting`	How to display groups, if present. Use 'vert' for vertical stacking of group plots, 'horiz' for horizontal ones, or 'none' to draw all lines in one plot, color-coding by group.
`k`	Number of nearest neighbors to use for density estimation.
`klm`	If method is "locmax", number of nearest neighbors to use for finding local maxima for cluster hunting. Generally needs to be much larger than `k`, to avoid "noise fitting."
`keepidxs`	If not NULL, the indices of the rows of `data` that are plotted will be stored in a component `idxs` of the return value. The rows themselves will be in a component `xdisp`, ordered by `data[,dispcols[1]`.
`plotidxs`	If TRUE, lines in the display will be annotated with their case numbers, i.e. their row numbers within `data`. Use only with small values of `m`, as overplotting may occur.
`cls`	Cluster, if any (see the `parallel` package) for parallel computation.
`plot_filename`	Name of the file that will hold the saved graph image. If NULL, the graph will be generated and displayed without being saved. If a filename is provided, the graph will not be displayed, only saved.

Details

The `dsldFreqPCoord` function wraps `freqparcoord`, which uses a frequency-based parallel coordinates method to vizualize multiple variables simultaneously in graph form.

This is done by plotting either the "most typical" or "least typical" (i.e. highest or lowest estimated multivariate density values respectively) cases to discern relations between variables.

The Y-axis represents the centered and scaled values of the columns.

Value

Object of type 'gg' (**ggplot2** object), with components `idxs` and `xdisp` added if `keepidxs` is not NULL (see argument `keepidxs` above).

Author(s)

N. Matloff, T. Abdullah, B. Ouattara, J. Tran, B. Zarate

References

https://cran.r-project.org/web/packages/freqparcoord/index.html

Examples

```
data(lsa)
lsa1 <- lsa[,c('fam_inc','ugpa','gender','lsat','race1')]
dsldFreqPCoord(lsa1,75,'race1')
# a number of interesting trends among the most "typical" law students in the
# dataset: remarkably little variation among typical
# African-Americans; typical Hispanic men have low GPAs, poor LSAT
```

```
# scores there is more variation; typical Asian and Black students were
# female; Asians and Hispanics have the most variation in family income
# background
```

`dsldFrequencyByS` *dsldFrequencyByS*

Description

Informal assessment of C as a possible confounder in a relationship between a sensitive variable S and a variable Y.

Usage

```
dsldFrequencyByS(data, cName, sName)
```

Arguments

`data`	Data frame or equivalent.
`cName`	Name of the "C" column, an R factor.
`sName`	Name of the sensitive variable column, an R factor

Details

Essentially an informal assessment of the between S and C.

Consider the `svcensus` dataset. If for instance we are studying the effect of gender S on wage income Y, say C is occupation. If different genders have different occupation patterns, then C is a potential confounder. (Y does not explicitly appear here.)

Value

Data frame, one for each level of the sensitive variable S, and one column for each level of the confounder C. Each row sums to 1.0.

Author(s)

N. Matloff, T. Abdullah, A. Ashok, J. Tran

Examples

```
data(svcensus)
dsldFrequencyByS(svcensus, cName = "educ", sName = "gender")
# not much difference in education between genders
dsldFrequencyByS(svcensus, cName = "occ", sName = "gender")
# substantial difference in occupation between genders
data(lsa)
lsa$faminc <- as.factor(lsa$fam_inc)
dsldFrequencyByS(lsa,'faminc','race1')
# distribution of family income by race
```

`dsldLinear` *dsldLinear*

Description

Comparison of sensitive groups via linear models, with or without interactions with the sensitive variable.

Usage

```
dsldLinear(data, yName, sName, interactions = FALSE, sComparisonPts = NULL,
    useSandwich = FALSE)
## S3 method for class 'dsldLM'
summary(object,...)
## S3 method for class 'dsldLM'
predict(object,xNew,...)
## S3 method for class 'dsldLM'
coef(object,...)
## S3 method for class 'dsldLM'
vcov(object,...)
```

Arguments

`data`	Data frame.
`yName`	Name of the response variable Y column.
`sName`	Name of the sensitive attribute S column.
`interactions`	Logical value indicating whether or not to model interactions with the sensitive variable S.
`sComparisonPts`	If `interactions` is TRUE, a data frame of new cases for which mean Y | X will be compared across each pair of S levels. Must be in the same format as original data.
`useSandwich`	If TRUE, use the "sandwich" variance estimator.
`object`	An object returned by the `dsldLinear` function.
`xNew`	New data to be predicted. Must be in the same format as original data.
`...`	Further arguments.

Details

The `dsldLinear` function fits a linear model to the response variable Y using all other variables in `data`. The user may select for interactions with the sensitive variable S.

The function produces an instance of the ‘dsldLM‘ class (an S3 object). Instances of the generic functions `summary` and `coef` are provided.

If `interactions` is TRUE, the function will fit m separate models, where m is the number of levels of S. Then `summary` will contain m+1 data frames; the first m of which will be the outputs from the individual models.

The m+1st data frame will compare the differences in conditional mean Y|X for each pair of S levels, and for each value of X in `sComparisonPts`. The intention is to allow users to see the comparisons of conditions for sensitive groups via linear models, with interactions with S.

The `dsldDiffS` function allows users to compare mean Y at that X between each pair of S level for additional new unseen data levels using the model fitted from `dsldLinear`.

Value

The `dsldLinear` function returns an S3 object of class 'dsldLM', with one component for each level of S. Each component includes information about the fitted model.

Author(s)

N. Matloff, A. Mittal, A. Ashok

Examples

```
data(svcensus)

newData <- svcensus[c(1, 18), -c(4,6)]
lin1 <- dsldLinear(svcensus, 'wageinc', 'gender', interactions = TRUE,
    newData)
coef(lin1)
vcov(lin1)
summary(lin1)
predict(lin1, newData)

lin2 <- dsldLinear(svcensus, 'wageinc', 'gender', interactions = FALSE)
summary(lin2)
```

`dsldLogit` *dsldLogit*

Description

Comparison of conditions for sensitive groups via logistic regression models, with or without interactions with the sensitive variable.

Usage

```
dsldLogit(data, yName, sName, sComparisonPts = NULL, interactions = FALSE,
   yesYVal)
## S3 method for class 'dsldGLM'
summary(object,...)
## S3 method for class 'dsldGLM'
predict(object,xNew,...)
## S3 method for class 'dsldGLM'
coef(object,...)
## S3 method for class 'dsldGLM'
vcov(object,...)
```

Arguments

`data`	Data frame used to train the linear model; will be split according to each level of `sName` in output if `interactions` is TRUE.
`yName`	Name of the response variable column.
`sName`	Name of the sensitive attribute column.
`interactions`	If TRUE, fit interactions with the sensitive variable.
`sComparisonPts`	If `interactions` is TRUE, a a data frame of new cases (minus Y,S) for which P(Y = 1| X) will be compared between each pairs of S levels. Must be in the same format as the original data.
`yesYVal`	Y value to be considered 'yes', to be coded 1 rather than 0.
`object`	An object returned by `dsldLogit`.
`xNew`	Dataframe to predict new cases. Must be in the same format as `data`.
`...`	Further arguments.

Details

The `dsldLogit` function fits a logistic regression model to the response variable. Interactions are handled as in `dsldLinear`.

Value

The `dsldLog` function returns an S3 object of class 'dsldGLM', with one component for each level of S. Each component includes information about the fitted model.

Author(s)

N. Matloff, A. Mittal, A. Ashok

Examples

```
data(lsa)
newData <- lsa[c(2,22,222,2222),-c(8,11)]
log1 <- dsldLogit(lsa,'bar','race1', newData, interactions = TRUE, 'TRUE')

coef(log1)
vcov(log1)
summary(log1)
predict(log1, newData)

log2 <- dsldLogit(data = lsa,
    yName = 'bar',sName = 'gender',
    interactions = FALSE, yesYVal = 'TRUE')
summary(log2)
```

`dsldMatchedATE` *dsldMatchedATE*

Description

Causal inference via matching models. Wrapper for `Matching::Match`.

Usage

```
dsldMatchedATE(data,yName,sName,yesSVal,yesYVal=NULL,
   propensFtn=NULL,k=NULL)
```

Arguments

`data`	Data frame.
`yName`	Name of the response variable column.
`sName`	Name of the sensitive attribute column. The attribute must be dichotomous.
`yesSVal`	S value to be considered "yes," to be coded 1 rather than 0.
`yesYVal`	Y value to be considered "yes," to be coded 1 rather than 0.
`propensFtn`	Either 'glm' (logistic), or 'knn'.
`k`	Number of nearest neighbors if `propensFtn='knn'`.

Details

This is a **dsld** wrapper for `Matching::Match`.

Matched analysis is typically applied to measuring "treatment effects," but is often applied in situations in which the "treatment," S here, is an immutable attribute such as race or gender. The usual issues concerning observational studies apply.

The function `dsldMatchedATE` finds the estimated mean difference between the matched Y pairs in the treated/nontreated (exposed and non-exposed) groups, with covariates X in `data` other than the `yName` and `sName` columns.

In the propensity model case, we estimate P(S = 1 | X), either by a logistic or k-NN model.

Value

Object of class 'Match'. See documentation in the **Matching** package.

Author(s)

N. Matloff

Examples

```
data(lalonde,package='Matching')
ll <- lalonde
ll$treat <- as.factor(ll$treat)
ll$re74 <- NULL
ll$re75 <- NULL
summary(dsldMatchedATE(ll,'re78','treat','1'))
summary(dsldMatchedATE(ll,'re78','treat','1',propensFtn='glm'))
summary(dsldMatchedATE(ll,'re78','treat','1',propensFtn='knn',k=15))
```

`dsldML` *dsldML*

Description

Nonparametric comparison of sensitive groups.

Usage

```
dsldML(data,yName,sName,qeMLftnName,sComparisonPts="rand5",
    opts=NULL,holdout=NULL)
```

Arguments

`data`	A data frame.
`yName`	Name of the response variable column.
`sName`	Name(s) of the sensitive attribute column(s).
`qeMLftnName`	Quoted name of a prediction function in the `qeML` package.
`sComparisonPts`	Data frame of one or more data points at which the regression function is to be estimated for each level of S. If this is 'rand5', then the said data points will consist of five randomly chosen rows in the original dataset.
`opts`	An R list specifying arguments for the above `qeML` function.
`holdout`	The size of holdout set.

Details

In a linear model with no interactions, one can speak of "the" difference in mean Y given X across treatments, independent of X. In a nonparametric analysis, there is interaction by definition, and one can only speak of differences across treatments for a specific X value. Hence the need for the argument `sComparisonPts`.

The specified `qeML` function will be called on the indicated data once for each level of the sensitive variable. For each such level, estimated regression function values will be obtained for each row in `sComparisonPts`.

Value

An R list. The first component consists of the holdout-set prediction accuracies, while the second is a data frame predicted values for each sensitive group.

Author(s)

N. Matloff

Examples

```
data(svcensus)
w <- dsldML(svcensus,'wageinc','gender',qeMLftnName='qeKNN',
   opts=list(k=50))
print(w)
```

`dsldScatterPlot3D` *ScatterPlot3D in dsld*

Description

Plotly 3D visualization of a dataset on 3 axes, with points color-coded on a 4th variable.

Usage

```
dsldScatterPlot3D(data, yNames, sName, sGroups = NULL, sortedBy =
  "Name", numGroups = 8, maxPoints = NULL, xlim = NULL,
  ylim = NULL, zlim = NULL, main = NULL, colors =
  "Paired", opacity = 1, pointSize = 8)
```

Arguments

`data`	Data frame with at least 4 columns.
`yNames`	Vector of the indices or names of the columns of the data frame to be graphed on the 3 axes.
`sName`	Index or name of the column that contains the groups for which the data will be grouped by. This will affect the colors of the points of the graph. This column must be an R factor.
`sGroups`	Vector of the names of the groups for which the data will be grouped by. Every value in the vector must exist in the `sName` column of the data frame. If not supplied or is NULL, the function will create this automatically according to the `sortedby` and `numgrps` parameters. By default, the function uses the 8 alphabetically first distinct groups in the `sName` column.
`sortedBy`	Controls how `sGroups` is created automatically. If `sGroups` is supplied, this does nothing. One of three values: "Name", "Frequency", "Frequency-Descending". "Name" gets the first values alphabetically. "Frequency" gets the most frequently occuring values. "Frequency-Descending" gets the least frequently occuring values.

`numGroups`	Number of groups to be automatically generated by the function. If `grpnames` is supplied, this does nothing.
`maxPoints`	Limit to how many points may be displayed on the graph. There is no limit by default.
`xlim`, `ylim`, `zlim`	The x, y and z limits, each a vector with c(min, max).
`main`	The title of the graph. By default, the `sName` "vs. " `yNames`.
`colors`	Either a colorbrewer2.org palette name (e.g. "YlOrRd" or "Blues"), or a vector of colors to interpolate in hexadecimal "#RRGGBB" format, or a color interpolation function like colorRamp().
`opacity`	A value between 0 and 1.
`pointSize`	A value above 1.

Details

An interactive Plotly visualization will be created, with the three variables specified in `yNames`. Points will be color-coded according to `sName`. The plot can be rotated etc. using the mouse.

Value

No value, plot.

Author(s)

J. Tran and B. Zarate

References

https://plotly.com/r/3d-scatter-plots/

Examples

```
data(lsa)
dsldScatterPlot3D(lsa,sName = "race1",
   yNames=c("ugpa", "lsat","age"), xlim=c(2,4))
```

`dsldTakeALookAround` *dsldTakeALookAround*

Description

Evaluate feature sets for predicting Y while considering the Fairness-Utility Tradeoff.

Usage

```
dsldTakeALookAround(data, yName, sName, maxFeatureSetSize = (ncol(data) - 2),
    holdout = floor(min(1000,0.1*nrow(data))))
```

Arguments

`data`	Data frame.
`yName`	Name of the response variable column.
`sName`	Name of the sensitive attribute column.
`maxFeatureSetSize`	Maximum number of combinations of features to be included in the data frame.
`holdout`	If not NULL, form a holdout set of the specified size. After fitting to the remaining data, evaluate accuracy on the test set.

Details

This function provides a tool for exploring feature combinations to use in predicting an outcome Y from features X and a sensitive variable S.

The features in X will first be considered singly, then doubly and so on, up though feature combination size `maxFeatureSetSize`. Y is prediction from X either a linear model (numeric Y) or logit (dichotomous Y).

The accuracy (based on qeML holdout) will be computed for each of these cases: (a) Y predicted from the given feature combination C, (b) Y predicted from the given feature combination C plus S, and (c) S predicted from C. The difference between columns 'a' and 'b' shows the sacrifice in utility stemming from not using S in our prediction of Y. (Due to sampling variation, it is possible for column 'b' to be larger than 'a'.) The value in column 'c' shows fairness, the smaller the fairer.

Value

Data frame whose first column consists of the variable names, followed by columns 'a', 'b' and 'c' as described in 'details'.

Author(s)

N. Matloff, A. Ashok, S. Martha, A. Mittal

Examples

```
# investigate predictive accuracy for a continuous Y,
# 'wageinc', using the default arguments for maxFeatureSetSize = 4
data(svcensus)
dsldTakeALookAround(svcensus, 'wageinc', 'gender', 4)

# investigate the predictive accuracy for a categorical Y,
# 'educ', using the default arguments for maxFeatureSetSize = 4
dsldTakeALookAround(svcensus, 'educ', 'gender')
```

`lak` *Labor Market Discrimination*

Description

Fictional CVs sent to real employers to investigate discrimination via given names. See Mullainathan and Bertran (2004).

References

- Mullainathan, S. and Bertran, M. (2004). Are Emily and Greg More Employable Than Lakisha and Jamal? A Field Experiment on Labor Market Discrimination. American Economic Review, 94:991-1013

`mortgageSE` *Mortgage Denial*

Description

The dataset provides applicant information (including race, income, loan information, etc.) The response variable indicates whether or not the applicant was approved for the loan. Additional details can be found in the `SortedEffects` package.

`svcensus` *Silicon Valley programmers and engineers data*

Description

Via qeML: This data set is adapted from the 2000 Census, restricted to programmers and engineers in the Silicon Valley area.

`utilities` *Utitlities*

Description

Attempts to load the specified package, halting execution upon failure.

Usage

```
getSuggestedLib(pkgName)
```

Arguments

`pkgName`	Name of the package to be checked/loaded.

Value

No value, just side effects.

Index