Package ‘multilevelPSA’

October 13, 2022

License GPL (>= 2)

Title Multilevel Propensity Score Analysis

Type Package

Author Jason Bryer <jason@bryer.org>

Maintainer Jason Bryer <jason@bryer.org>

Description Conducts and visualizes propensity score analysis for multilevel,
or clustered data. Bryer & Pruzek (2011) <doi:10.1080/00273171.2011.636693>.

Version 1.2.5

URL `http://github.com/jbryer/multilevelPSA`

BugReports `https://github.com/jbryer/multilevelPSA/issues`

Depends ggplot2, xtable, R (>= 3.0)

Imports PSAgraphics, plyr, psych, reshape, grid, party, MASS

Suggests testthat

Date 2018-03-22

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-03-22 20:21:26 UTC

R topics documented:

`multilevelPSA-package` *Multilevel Propensity Score Analysis*

Description

This packages provides functions to estimate and visualize multilevel propensity score analysis.

Details

This package extends the principles put forth by the `PSAgraphics` (Helmreich, Pruzek, & Xiong, 2010) for multilevel, or clustered, data.

Propensity score analyses are typically done in two phases. In phase I, a statistical model predicting treatment using the available individual covariates is estimated. This package currently currently provides functions to perform propensity score estimates using logistic regression (see `mlpsa.logistic`) and conditional inference trees (see `mlpsa.ctree`). The latter method provides explicit stratifications as defined by each leaf node. The former however, results in a numerical value ranging from zero to one (i.e. the fitted values). A common approach is to then create stratifications using quintiles. However, other approaches such as Loess regression are also provided.

Phase II of typical propensity score analyses concerns with the comparison of an outcome between the treatment and comparison groups. The `mlpsa` method will perform this analysis in a multilevel, or clustered, fashion. That is, the results of the `mlpsa` procedure produce summary results at level one (i.e. each strata within each cluster), level two (i.e. overall results for each cluster), and overall (i.e. overall results across all clusters).

This package also provides a number of visualizations that provide a critical part in presenting, understanding, and interpreting the results. See `plot.mlpsa` for details.

Author(s)

Jason Bryer `<jason@bryer.org>`

References

`https://CRAN.R-project.org/package=PSAgraphics` `http://www.jstatsoft.org/v29/i06/`

See Also

`PSAgraphics`

`align.plots` *Adapted from ggExtra package which is no longer available. This is related to an experimental mlpsa plot that will combine the circular plot along with the two individual distributions.*

Description

Adapted from ggExtra package which is no longer available. This is related to an experimental mlpsa plot that will combine the circular plot along with the two individual distributions.

Usage

```
## S3 method for class 'plots'
align(gl, ...)
```

Arguments

`gl`	grid.layout
`...`	graphic elements to combine.

as.data.frame.covariate.balance

Returns the overall effects as a data frame.

Description

Returns the overall effects as a data frame.

Usage

```
## S3 method for class 'covariate.balance'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

Arguments

`x`	results of `covariate.balance`.
`row.names`	unused.
`optional`	unused.
`...`	unused

Value

a data frame with overall covariate effects before and after adjustment.

covariate.balance

Estimate covariate effect sizes before and after propensity score adjustment.

Description

Estimate covariate effect sizes before and after propensity score adjustment.

Usage

```
covariate.balance(covariates, treatment, level2, strata, abs = TRUE)
```

Arguments

`covariates`	frame or matrix of covariates.
`treatment`	vector of treatment indicators.
`level2`	vector indicating level 2 membership.
`strata`	strata indicators.
`abs`	if TRUE absolute values of effect sizes will be plotted.

covariateBalance — *Calculate covariate effect size differences before and after stratification.*

Description

This function is modified from the cv.bal.psa function in the PSAgrpahics package.

Usage

```
covariateBalance(covariates, treatment, propensity, strata = NULL,
  int = NULL, tree = FALSE, minsize = 2, universal.psd = TRUE,
  trM = 0, absolute.es = TRUE, trt.value = NULL, use.trt.var = FALSE,
  verbose = FALSE, xlim = NULL, plot.strata = TRUE, na.rm = TRUE, ...)
```

Arguments

`covariates`	dataframe of interest
`treatment`	binary vector of 0s and 1s (necessarily? what if character, or 1, 2?)
`propensity`	PS scores from some method or other.
`strata`	either a vector of strata number for each row of covariate, or one number n in which case it is attempted to group rows by ps scores into n strata of size approximately 1/n. This does not seem to work well in the case of few specific propensity values, as from a tree.
`int`	either a number m used to divide [0,1] into m equal length subintervals, or a vector of cut points between 0 an 1 defining the subintervals (perhaps as suggested by loess.psa). In either case these subintervals define strata, so strata can be of any size.
`tree`	logical, if unique ps scores are few, as from a recursively partitioned tree, then TRUE will force each ps value to define a stratum.
`minsize`	smallest allowable stratum-treatment size. If violated, strata is removed.
`universal.psd`	If 'TRUE', forces standard deviations used to be unadjusted for stratification.
`trM`	trimming proportion for mean calculations.
`absolute.es`	logical, if 'TRUE' routine uses absolute values of all effect sizes.
`trt.value`	allows user to specify which value is active treatment, if desired.
`use.trt.var`	logical, if true then Rubin-Stuart method using only treatment variance with be used in effect size calculations.
`verbose`	logical, controls output that is visibly returned.
`xlim`	limits for the x-axis.
`plot.strata`	logical indicating whether to print strata.
`na.rm`	should missing values be removed.
`...`	currently unused.

Details

Note: effect sizes are calculated as treatment 1 - treatment 0, or treatment B - treatment A.

Author(s)

Robert M. Pruzek RMPruzek@yahoo.com

James E. Helmreich James.Helmreich@Marist.edu

KuangNan Xiong harryxkn@yahoo.com

Jason Bryer jason@bryer.org

`cv.trans.psa` *Transformation of Factors to Individual Levels*

Description

The function `cv.trans.psa` takes a covariate data frame and replaces each categorical covariate of `n` >=3 levels with `n` new binary covariate columns, one for each level. Transforms covariate dataframe for use with the function `cv.bal.psa`.

Usage

```
cv.trans.psa(covariates, fcol = NULL)
```

Arguments

`covariates`	A dataframe of covariates, presumably some factors.
`fcol`	An optional vector containing the factor columns in the covariate data frame. In `NULL` (default) routine to identify factors internally.

Details

NOTE: This function originated in the `PSAgraphics` package. It has been adapted here for the `multilevelPSA` package.

Author(s)

James E. Helmreich James.Helmreich@Marist.edu

Robert M. Pruzek RMPruzek@yahoo.com

KuangNan Xiong harryxkn@yahoo.com

Jason Bryer jason@bryer.org

`difftable.plot` *This function produces a ggplot2 figure containing the mean differences for each level two, or cluster.*

Description

This function produces a ggplot2 figure containing the mean differences for each level two, or cluster.

Usage

```
difftable.plot(x, fill.colors = NULL, legendlab = NULL, ...)
```

Arguments

`x`	the results of `mlpsa`.
`fill.colors`	the colors to use for each level two.
`legendlab`	the label to use for the legend, or NULL to exclude.
`...`	currently unused.

Value

a ggplot2 figure

`getPropensityScores` *Returns a data frame with two columns corresponding to the level 2 variable and the fitted value from the logistic regression.*

Description

Returns a data frame with two columns corresponding to the level 2 variable and the fitted value from the logistic regression.

Usage

```
getPropensityScores(lr.results, nStrata = 5)
```

Arguments

`lr.results`	the results of `mlpsa.logistic`
`nStrata`	number of strata within each level.

Value

a data frame

See Also

mlpsa.logistic

`getStrata` *Returns a data frame with two columns corresponding to the level 2 variable and the leaves from the conditional inference trees.*

Description

Returns a data frame with two columns corresponding to the level 2 variable and the leaves from the conditional inference trees.

Usage

```
getStrata(party.results, data, level2)
```

Arguments

`party.results`	the results of `mlpsa.ctree`
`data`	the data frame to merge results to
`level2`	the name of the level 2 variable.

Value

a data frame

See Also

mlpsa.ctree

`is.mlpsa` *Returns true if the object is of type* `mlpsa`

Description

Returns true if the object is of type `mlpsa`

Usage

```
is.mlpsa(x)
```

Arguments

`x`	the object to test

loess.plot — *Loess plot with density distributions for propensity scores and outcomes on top and right, respectively.*

Description

Loess plot with density distributions for propensity scores and outcomes on top and right, respectively.

Usage

```
loess.plot(x, response, treatment, responseTitle = "",
  treatmentTitle = "Treatment", percentPoints.treat = 0.1,
  percentPoints.control = 0.01, points.treat.alpha = 0.1,
  points.control.alpha = 0.1, plot.strata, plot.strata.alpha = 0.2, ...)
```

Arguments

`x`	vector of propensity scores.
`response`	the response variable.
`treatment`	the treatment variable as a logical type.
`responseTitle`	the label to use for the y-axis (i.e. the name of the response variable)
`treatmentTitle`	the label to use for the treatment legend.
`percentPoints.treat`	the percentage of treatment points to randomly plot.
`percentPoints.control`	the percentage of control points to randomly plot.
`points.treat.alpha`	the transparency level for treatment points.
`points.control.alpha`	the transparency level for control points.
`plot.strata`	an integer value greater than 2 indicating the number of vertical lines to plot corresponding to quantiles.
`plot.strata.alpha`	the alpha level for the vertical lines.
`...`	other parameters passed to `geom_smooth` and `stat_smooth`.

Value

a ggplot2 figure

See Also

plot.mlpsa

Examples

```
## Not run:
require(multilevelPSA)
require(party)
data(pisana)
data(pisa.psa.cols)
cnt = 'USA' #Can change this to USA, MEX, or CAN
pisana2 = pisana[pisana$CNT == cnt,]
pisana2$treat <- as.integer(pisana2$PUBPRIV) %% 2
lr.results <- glm(treat ~ ., data=pisana2[,c('treat',pisa.psa.cols)], family='binomial')
st = data.frame(ps=fitted(lr.results),
math=apply(pisana2[,paste('PV', 1:5, 'MATH', sep='')], 1, mean),
pubpriv=pisana2$treat)
st$treat = as.logical(st$pubpriv)
loess.plot(st$ps, response=st$math, treatment=st$treat, percentPoints.control = 0.4,
           percentPoints.treat=0.4)

## End(Not run)
```

`lsos` *Nicer list of objects in memory. Particularly useful for analysis of large data.* `#http://stackoverflow.com/questions/1358003/tricks-to-manage-the-available-memory-in-an-r-session`

Description

Nicer list of objects in memory. Particularly useful for analysis of large data. `#http://stackoverflow.com/questions/1358003/tricks-to-manage-the-available-memory-in-an-r-session`

Usage

```
lsos(..., n = 10)
```

Arguments

`...`	not used.
`n`	the number of objects to return.

Value

a list of objects loaded sorted by size.

missing.plot — *Returns a heat map graphic representing missingness of variables grouped by the given grouping vector.*

Description

NOTE: This is an experimental function and the results may vary depending on the nature of the dataset.

Usage

```
missing.plot(x, grouping, grid = FALSE, widths = c(ggplot2::unit(3, "null"),
  ggplot2::unit(1, "inches")), heights = c(ggplot2::unit(1, "inches"),
  ggplot2::unit(3, "null")), color = "red", ...)
```

Arguments

`x`	a data frame containing the variables to visualize missingness
`grouping`	a vector of length nrow(vars) corresponding to how missing will be grouped by
`grid`	whether to draw a grid between tiles
`widths`	the ratio of the widths of the heatmap and histogram.
`heights`	the ratio of the heights of the heatmap and histogram.
`color`	the color used for indicating missingness.
`...`	currently unused.

Value

a ggplot2 expression

See Also

plot.mlpsa

mlpsa — *This function will perform phase II of the multilevel propensity score analysis.*

Description

TODO: Need more details

Usage

```
mlpsa(response, treatment = NULL, strata = NULL, level2 = NULL,
  minN = 5, reverse = FALSE, ci.level = 0.05)
```

Arguments

`response`	vector containing the response values
`treatment`	vector containing the treatment conditions
`strata`	vector containing the strata for each response
`level2`	vector containing the level 2 specifications
`minN`	the minimum number of subjects per strata for that strata to be included in the analysis.
`reverse`	reverse the order of treatment and control for the difference calculation.
`ci.level`	the confidence level to use for confidence intervals. Defaults to a 95% confidence level.

Details

The ci.adjust provides a Bonferroni-Sidak adjusted confidence intervals based on the number of levels/clusters.

Value

a mlpsa class

See Also

`mlpsa.ctree` `mlpsa.logistic`

Examples

```
## Not run:
require(multilevelPSA)
require(party)
data(pisana)
data(pisa.colnames)
data(pisa.psa.cols)
mlctree = mlpsa.ctree(pisana[,c('CNT','PUBPRIV',pisa.psa.cols)], formula=PUBPRIV ~ ., level2='CNT')
student.party = getStrata(mlctree, pisana, level2='CNT')
student.party$mathscore = apply(student.party[,paste0('PV', 1:5, 'MATH')], 1, sum) / 5
results.psa.math = mlpsa(response=student.party$mathscore,
       treatment=student.party$PUBPRIV,
       strata=student.party$strata,
       level2=student.party$CNT, minN=5)
results.psa.math
summary(results.psa.math)

## End(Not run)
```

mlpsa.circ.plot — *Plots the results of a multilevel propensity score model.*

Description

The plot created uses the ggplot2 framework. As such, additional modifications can be made. This plot is an extension of the circ.psa function in the PSAgraphics package for multilevel models.

Usage

```
mlpsa.circ.plot(x, xlab = names(multilevelPSA$level2.summary)[4],
  ylab = names(multilevelPSA$level2.summary)[5], legendlab = "Level 2",
  title = NULL, overall.col = "blue", overall.ci.col = "green",
  level1.plot = FALSE, level1.point.size = NULL, level1.rug.plot = NULL,
  level1.projection.lines = FALSE, level2.plot = TRUE,
  level2.point.size = NULL, level2.rug.plot = "tr",
  level2.projection.lines = TRUE, level2.label = FALSE,
  unweighted.means = FALSE, weighted.means = FALSE, fill.colors = NULL,
  ...)
```

Arguments

`x`	the results of mlpsa.
`xlab`	label for the x-axis.
`ylab`	label for the y-axis.
`legendlab`	the label for the legend, or NULL to exclude.
`title`	title for the figure.
`overall.col`	the color used for the overall results.
`overall.ci.col`	the color used for the confidence intervals.
`level1.plot`	logical value indicating whether level 1 points should be plotted.
`level1.point.size`	the size of level 1 points
`level1.rug.plot`	the placement for plotting a level 2 rug. Possible values are `bl` (for left and bottom), `tr` (for top and right), or NULL (to exclude).
`level1.projection.lines`	logical value indicating whether level 1 project lines (parallel to the unit line) are drawn.
`level2.plot`	logical value indicating whether level 2 points should be plotted.
`level2.point.size`	the size of level 2 points
`level2.rug.plot`	the placement for plotting a level 2 rug. Possible values are `bl` (for left and bottom), `tr` (for top and right), or NULL (to exclude).

`level2.projection.lines`	logical value indicating whether level 2 project lines (parallel to the unit line) are drawn.
`level2.label`	logical value indicating whether level 2 points should be labeled.
`unweighted.means`	logical value indicating whether horizontal and vertical lines are drawn representing the unweighted (i.e. unadjusted from phase I of PSA) means for each level 2, or cluster.
`weighted.means`	logical value indicating whether horizontal and vertical lines are drawn representing the weighted means for each level 2, or cluster.
`fill.colors`	if specified, the colors to use for level 2 points.
`...`	currently unused.

See Also

plot.mlpsa

Examples

```
## Not run:
data(pisana)
data(pisa.colnames)
data(pisa.psa.cols)
mlctree = mlpsa.ctree(pisana[,c('CNT','PUBPRIV',pisa.psa.cols)],
                      formula=PUBPRIV ~ ., level2='CNT')
student.party = getStrata(mlctree, pisana, level2='CNT')
student.party$mathscore = apply(student.party[,paste0('PV', 1:5, 'MATH')], 1, sum) / 5
results.psa.math = mlpsa(response=student.party$mathscore,
       treatment=student.party$PUBPRIV,
       strata=student.party$strata,
       level2=student.party$CNT, minN=5)
mlpsa.circ.plot(results.psa.math, legendlab=FALSE)

## End(Not run)
```

`mlpsa.ctree` *Estimates propensity scores using the recursive partitioning in a conditional inference framework.*

Description

This function will estimate propensity scores using the conditional inference framework as outlined in the `party` package. Specifically, a separate tree will be estimated for each level 2 (or cluster). A key advantage of this framework over other methods for estimating propensity scores is that this method will work on data sets containing missing values.

Usage

```
mlpsa.ctree(vars, formula, level2, ...)
```

Arguments

`vars`	a data frame containing the covariates to use for estimating the propensity scores.
`formula`	the model for estimating the propensity scores. For example, treat ~ .
`level2`	the name of the column in `vars` specifying the level 2 (or cluster).
`...`	currently unused.

Value

a list of BinaryTree-class classes for each level 2

References

Torsten Hothorn, Kurt Hornik and Achim Zeileis (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics, 15(3), 651–674.

See Also

`getStrata`

`tree.plot`

`mlpsa.difference.plot` *Creates a graphic summarizing the differences between treatment and comparison groups within and across level two clusters.*

Description

Creates a graphic summarizing the differences between treatment and comparison groups within and across level two clusters.

Usage

```
mlpsa.difference.plot(x, xlab, ylab = NULL, title = NULL,
  overall.col = "blue", overall.ci.col = "green",
  level2.point.size = NULL, level1.points = TRUE, errorbars = TRUE,
  errorbars.adjusted.ci = TRUE, level2.rug.plot = TRUE, jitter = TRUE,
  reorder = TRUE, labelLevel2 = TRUE, sd = NULL, xlim, ...)
```

Arguments

`x`	the results of `mlpsa`.
`xlab`	label for the x-axis, or NULL to exclude.
`ylab`	label for the y-axis, or NULL to exclude.
`title`	title of the figure, or NULL to exclude.
`overall.col`	the color of the overall results line.
`overall.ci.col`	the color of the overall confidence interval.
`level2.point.size`	the point size of level 2 points.
`level1.points`	logical value indicating whether level 1 strata should be plotted.
`errorbars`	logical value indicating whether error bars should be plotted for for each level 1.
`errorbars.adjusted.ci`	whether the Bonferroni adjusted error bars should be plotted (these will be dashed lines).
`level2.rug.plot`	logical value indicating whether a rug plot should be plotted for level 2.
`jitter`	logical value indicating whether level 1 points should be jittered.
`reorder`	logical value indicating whether the level two clusters should be reordered from largest difference to smallest.
`labelLevel2`	logical value indicating whether the difference for each level 2 should be labeled.
`sd`	If specified, effect sizes will be plotted instead of difference in the native unit.
`xlim`	the limits of the x-axis.
`...`	currently unused.

See Also

plot.mlpsa

Examples

```
## Not run:
data(pisana)
data(pisa.colnames)
data(pisa.psa.cols)
mlctree = mlpsa.ctree(pisana[,c('CNT','PUBPRIV',pisa.psa.cols)], formula=PUBPRIV ~ ., level2='CNT')
student.party = getStrata(mlctree, pisana, level2='CNT')
student.party$mathscore = apply(student.party[,paste0('PV', 1:5, 'MATH')], 1, sum) / 5
results.psa.math = mlpsa(response=student.party$mathscore,
       treatment=student.party$PUBPRIV,
       strata=student.party$strata,
       level2=student.party$CNT, minN=5)
mlpsa.difference.plot(results.psa.math, sd=mean(student.party$mathscore, na.rm=TRUE))

## End(Not run)
```

`mlpsa.distribution.plot` *Plots distribution for either the treatment or comparison group.*

Description

Plots distribution for either the treatment or comparison group.

Usage

```
mlpsa.distribution.plot(x, treat, fill.colours = NULL, flip = TRUE,
  label = treat, level2.label = NULL, legendlab = NULL,
  axis.text.size = 8, fill.colors = NULL, ...)
```

Arguments

`x`	the results of `mlpsa`.
`treat`	the group to plot. This must be one of the two levels of the treatment variable.
`fill.colours`	if specified, the colors to use for level 2 points.
`flip`	if TRUE, the level 2 clusters will be on the y-axis and the outcome variable on the x-axis. Otherwise reversed.
`label`	the label to use for the axis.
`level2.label`	the axis label for the level 2 indicators.
`legendlab`	the label for the legend, or NULL to exclude a legend.
`axis.text.size`	the size of the axis text
`fill.colors`	if specified, the colors to use for level 2 points.
`...`	currently unused.

See Also

plot.mlpsa

`mlpsa.logistic` *Estimates propensity scores using logistic regression.*

Description

This method will estimate a separate logistic regression model for each level 2 (or cluster).

Usage

```
mlpsa.logistic(vars, formula, level2, stepAIC = FALSE, ...)
```

Arguments

`vars`	data frame containing the variables to estimate the logistic regression
`formula`	the logistic regression formula to use
`level2`	the name of the column containing the level 2 specification
`stepAIC`	if true, the `stepAIC` from the `MASS` package will be used within each level.
`...`	currently unused.

Value

a list of glm classes for each level 2 or stepwise-selected model if stepAIC is true.

See Also

getPropensityScores

`pisa.colnames` *Mapping of variables in `pisana` with full descriptions.*

Description

This data frame provides three variables, `Variable` corresponding to the column names in `pisana`, `ShortDesc` providing a short description of the variable as a valid R object name, and `Desc` providing a longer description of the variable.

Format

a data frame with 50 rows of 3 variables.

`pisa.countries` *Data frame mapping PISA countries to their three letter abbreviation.*

Description

This data frame has two columns, `CNT3` for the three letter abbreviation of each country and `Country` that provides the full country name in English.

Format

data frame with 65 rows of 2 variables.

`pisa.psa.cols` *Character vector representing the list of covariates used for estimating propensity scores.*

Description

Character vector representing the list of covariates used for estimating propensity scores.

Format

a character vector with covariate names for estimating propensity scores.

`pisana` *North American (i.e. Canada, Mexico, and United States) student results of the 2009 Programme of International Student Assessment.*

Description

Student results from the 2009 Programme of International Student Assessment (PISA) as provided by the Organization for Economic Co-operation and Development (OECD). See `http://www.pisa.oecd.org/` for more information including the code book.

Format

a data frame with 66,548 obvservations of 65 variables.

Details

Note that missing values have been imputed using the http://cran.r-project.org/web/packages/mice/index.html package. Details on the specific procedure are in the `pisa.impute` function in the `pisa` package.

Source

Organization for Economic Co-operation and Development

References

Organization for Economic Co-operation and Development (2009). Programme for International Student Assessment (PISA). `http://www.pisa.oecd.org/`

plot.covariate.balance

Multiple covariate balance assessment plot.

Description

A graphic based upon `cv.bal.psa` function in the `PSAgraphics` package. This graphic plots the effect sizes for multiple covariates before and after propensity score adjustement.

Usage

```
## S3 method for class 'covariate.balance'
plot(x, plot.strata = FALSE,
  order = c("unadjusted", "adjusted"), strata.size = 3,
  strata.legend.guide = "none", point.size = 3, point.alpha = 1,
  line.color = "black", line.alpha = 0.2, legend.position = c(0.8, 0.2),
  ...)
```

Arguments

`x`	results of `covariate.balance`.
`plot.strata`	whether individual strata should be plotted.
`order`	how to order the y-axis. Possible values are adjusted, unadjusted, or NULL (don't reorder).
`strata.size`	text size for strata if plotted.
`strata.legend.guide`	guide for legend placement for strata.
`point.size`	size of the overall effect size points.
`point.alpha`	transparency level of the overall effect size points.
`line.color`	the color of the line connecting the overall effect size points.
`line.alpha`	transparency level of the line connecting the overall effect size points.
`legend.position`	where to position the legend.
`...`	currently unused.

Value

a ggplot2 with an attribute, `effects`, that is the data frame used to create the plot.

`plot.mlpsa` *Plots the results of a multilevel propensity score model.*

Description

The plot created uses the `ggplot2` framework. As such, additional modifications can be made. This plot is an extension of the `circ.psa` function in the `PSAgraphics` package for multilevel models.

Usage

```
## S3 method for class 'mlpsa'
plot(x, ratio = c(1, 2), plotExtra = NULL, ...)
```

Arguments

`x`	the results of `mlpsa`.
`ratio`	the ratio of the size of the distribution plots (left and bottom) to the circular plot.
`plotExtra`	a plot to place in the lower left corner.
`...`	parameters passed to `mlpsa.circ.plot` and `mlpsa.distribution.plot`

Examples

```
## Not run:
require(multilevelPSA)
require(party)
data(pisana)
data(pisa.colnames)
data(pisa.psa.cols)
mlctree = mlpsa.ctree(pisana[,c('CNT','PUBPRIV',pisa.psa.cols)], formula=PUBPRIV ~ ., level2='CNT')
student.party = getStrata(mlctree, pisana, level2='CNT')
student.party$mathscore = apply(student.party[,paste0('PV', 1:5, 'MATH')], 1, sum) / 5
results.psa.math = mlpsa(response=student.party$mathscore,
       treatment=student.party$PUBPRIV,
       strata=student.party$strata,
       level2=student.party$CNT, minN=5)
plot(results.psa.math)

## End(Not run)
```

plot.psrange *Plots densities and ranges for the propensity scores.*

Description

Plots densities and ranges for the propensity scores.

Usage

```
## S3 method for class 'psrange'
plot(x, xlab = NULL, ylab = NULL,
  labels = c("Comparison", "Treatment"), text.ratio.size = 4,
  text.ncontrol.size = 3, point.size = 1, point.alpha = 0.6,
  line.width = 6, density.alpha = 0.2, rect.color = "green",
  rect.alpha = 0.2, ...)
```

Arguments

`x`	the result of psrange.
`xlab`	label for x-axis.
`ylab`	label for y-axis.
`labels`	labels for the comparison and treatment legend.
`text.ratio.size`	size of the text for the ratio.
`text.ncontrol.size`	size of the text for the number of control units.
`point.size`	size of the points for the minimum and maximum ranges for each model.
`point.alpha`	the alpha (transparency) level for the points.
`line.width`	the width of the line between the median of the minimum and maximum ranges.
`density.alpha`	the alpha (transparency) level of the density curves.
`rect.color`	the color of the rectangle surrounding the range of minimum and maximum ranges.
`rect.alpha`	the alpha (transparency) level of the rectangle.
`...`	currently unused.

Value

a ggplot2 object

`print.covariate.balance`
Prints the overall effects before and after propensity score adjustment.

Description

Prints the overall effects before and after propensity score adjustment.

Usage

```
## S3 method for class 'covariate.balance'
print(x, ...)
```

Arguments

`x`	results of `covariate.balance`.
`...`	unused.

`print.mlpsa` *Prints basic information about a* `mlpsa` *class.*

Description

Prints basic information about a `mlpsa` class.

Usage

```
## S3 method for class 'mlpsa'
print(x, ...)
```

Arguments

`x`	the `mlpsa` class.
`...`	unused.

print.psrange *Prints information about a psrange result.*

Description

Prints information about a psrange result.

Usage

```
## S3 method for class 'psrange'
print(x, ...)
```

Arguments

`x`	psrange to print info about.
`...`	currently unused

print.xmlpsa *Prints the results of `mlpsa` and `xtable.mlpsa`.*

Description

Print method for `xtable.mlpsa`.

Usage

```
## S3 method for class 'xmlpsa'
print(x, tabular.environment = "longtable",
  floating = FALSE, ...)
```

Arguments

`x`	result of `xtable.mlpsa`
`tabular.environment`	see `print.xtable`.
`floating`	see `print.xtable`.
`...`	other parameters passed to `print.xtable`

psrange — *Estimates models with increasing number of comparison subjects starting from 1:1 to using all available comparison group subjects.*

Description

Estimates models with increasing number of comparison subjects starting from 1:1 to using all available comparison group subjects.

Usage

```
psrange(df, treatvar, formula, nsteps = 10, nboot = 10, samples,
  type = c("logistic", "ctree"), ...)
```

Arguments

`df`	data frame with variables to pass to glm
`treatvar`	vector representing treatment placement. Should be coded as 0s (for control) and 1s (for treatment).
`formula`	formula for logistic regression model
`nsteps`	number of steps to estimate from 1:1 to using all control records.
`nboot`	number of models to execute for each step.
`samples`	the sample sizes to draw from control group for each step.
`type`	either logistic for Logistic regression (using `glm` function) or ctree for Conditional Inference Trees (using the `ctree` function).
`...`	other parameters passed to glm.

Value

a class of psrange that contains a summary data frame, a details data frame, and a list of each individual result from glm.

summary.mlpsa — *Provides a summary of a `mlpsa` class.*

Description

Provides a summary of a `mlpsa` class.

Usage

```
## S3 method for class 'mlpsa'
summary(object, overall.label = "Overall", ...)
```

Arguments

`object`	the mlpsa object.
`overall.label`	the label to place in the strata column for the overall results.
`...`	unused.

`summary.psrange` *Prints the summary results of psrange.*

Description

Prints the summary results of psrange.

Usage

```
## S3 method for class 'psrange'
summary(object, ...)
```

Arguments

`object`	psrange to print summary of.
`...`	currently unused.

`tree.plot` *Heat map representing variables used in a conditional inference tree across level 2 variables.*

Description

This figure provides a summary of the covariates used within each level two cluster along with their relative importance. Covariates are listed on the y-axis and level two clusters along the x-axis. Cells that are shaded indicate that that covariate was present in the conditional. The shade of the color represents the highest level within the tree that covariate appeared. That is, the darkest color, or depth 1, corresponds to the covariate used at the root of the tree, or the first split.

Usage

```
tree.plot(x, colNames, level2Col, colLabels = NULL, color.high = "azure",
  color.low = "steelblue", color.na = "white", ...)
```

Arguments

`x`	the results of `mlpsa.ctree`
`colNames`	the columns to include in the graphic
`level2Col`	the name of the level 2 column.
`colLabels`	column labels to use. This is a data frame with two columns, the first column should match the values in `trees` and the second column the description that will be used for labeling the variables.
`color.high`	color for variables with less relative importance as determined by occurring later in the tree (further from the root split).
`color.low`	color for variables with greater relative importance as determined by occurring sooner in the tree (closer to the root split).
`color.na`	color for variables that do not occur in the tree.
`...`	currently unused.

Value

a ggplot2 expression

See Also

plot.mlpsa

Examples

```
## Not run:
require(party)
data(pisana)
data(pisa.colnames)
data(pisa.psa.cols)
mlctree = mlpsa.ctree(pisana[,c('CNT','PUBPRIV',pisa.psa.cols)], formula=PUBPRIV ~ ., level2='CNT')
student.party = getStrata(mlctree, pisana, level2='CNT')
tree.plot(mlctree, level2Col=pisana$CNT)

## End(Not run)
```

`xtable.mlpsa` *Prints the results of `mlpsa` as a LaTeX table.*

Description

This function implements the `xtable` method for `mlpsa`.

Usage

```
## S3 method for class 'mlpsa'
xtable(x, caption, label, align, digits = 2, display = NULL,
  auto = FALSE, include.note = TRUE, ...)
```

Arguments

`x`	results of `mlpsa`
`caption`	passed through to `xtable`.
`label`	passed through to `xtable`.
`align`	Not used.
`digits`	number of digits to print.
`display`	passed through to `xtable`.
`auto`	passed through to `xtable`.
`include.note`	include a table note indicating how many rows were removed due to insufficient data within a strata.
`...`	other parameters passed to `summary.mlpsa`

Index