Package ‘condvis2’

September 25, 2020

Title  Interactive Conditional Visualization for Supervised and Unsupervised Models in Shiny

Version  0.1.1

Description  Constructs a shiny app function with interactive displays for conditional visualization of models, data and density functions. An extended version of package 'condvis'.

License  GPL (>= 2.0)

Encoding  UTF-8

LazyData  true

Imports  shiny, RColorBrewer, ggplot2, scales, cluster, DendSer, methods, pplyr, colorspace, gower

RoxygenNote  7.1.1

Suggests  knitr, rmarkdown, hdrcde, scagnostics, keras, kernlab, mclust, MASS, ks, mgcv, randomForest, parsnip, mlr, C50, bartMachine, caret, e1071, gbm, glmnet, glmnetUtils, mlr3, nnet, rpart, tree, testthat

VignetteBuilder  knitr

BugReports  https://github.com/cbhurley/condvis2/issues

URL  https://github.com/cbhurley/condvis2

NeedsCompilation  no

Author  Catherine Hurley [aut, cre], Mark O'Connell [aut], Katarina Domijan [aut]

Maintainer  Catherine Hurley <catherine.hurley@mu.ie>

Repository  CRAN

Date/Publication  2020-09-25 09:20:09 UTC
R topics documented:

`arrangeC` . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  2
clusPath . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  3
conditionPlot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  4
condvis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  5
createCVServer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  7
createCVUI . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  8
CVpredict . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
fitPath . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
medoid . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
pathInterpolate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
plotTourDiagnostics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
sectionPlot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
similarityweight . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
tours . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
weightcolor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

Index 29

---

`arrangeC`  
*Make a list of variable pairings for condition selecting plots*

**Description**

This function arranges a number of variables in pairs, ordered by their bivariate relationships. The goal is to discover which variable pairings are most helpful in avoiding extrapolations when exploring the data space. Variable pairs with strong bivariate dependencies (not necessarily linear) are chosen first. The bivariate dependency is measured using `savingby2d`. Each variable appears in the output only once.

**Usage**

```r
arrangeC(data, method = "default")
```

**Arguments**

- `data`  
  A dataframe

- `method`  
  The character name for the method to use for measuring bivariate dependency, passed to `savingby2d`.

**Details**

If `data` is so big as to make `arrangeC` very slow, a random sample of rows is used instead. The bivariate dependency measures are rough, and the ordering algorithm is a simple greedy one, so it is not worth allowing it too much time.
clusPath

Value
A list containing character vectors giving variable pairings.

References

clusPath Constructs tours of data space based on centers of clusters

Description
Constructs tours of data space based on centers of clusters

Usage
kmeansPath(data, length = 10, reorder = TRUE, conditionvars = NULL, ...)
pamPath(
  data,
  length = 10,
  reorder = TRUE,
  conditionvars = NULL,
  maxn = 4000,
  ...
)
c ClaraPath(data, length = 10, reorder = TRUE, conditionvars = NULL, ...)
medoidPath(data, cl, reorder = FALSE)
centroidPath(data, cl, reorder = FALSE)

Arguments
data A dataframe
length Path length, defaults to 10
reorder If TRUE uses DendSer to reorder the path dser
conditionvars A vector of variable names. The returned tour is for this subset of variables.
maxn (pamPath only) For datasets with more than maxn rows, use maxn randomly selected rows.
cl A vector specifying cluster membership for rows of data.
ConditionPlot

Value

A dataframe with the path

Functions

- `kmeansPath()`: Constructs a tour of data space following length k-means centroids
- `pamPath()`: Constructs a tour of data space following length pam medoids
- `claraPath()`: Constructs a tour of data space following length clara medoids
- `medoidPath()`: Returns a path visiting cluster medoids
- `centroidPath()`: Returns a path visiting cluster centroids

Examples

```r
kmeansPath(mtcars,length=4)
pamPath(mtcars,length=4)
claraPath(mtcars,length=4)
medoidPath(mtcars,cl=rep(1:3, length.out=nrow(mtcars)))
```

---

ConditionPlot

Plots a conditionPlot.

Description

Plots a conditionPlot, showing one, two or many predictors. The predictor setting in `varVal` is drawn in magenta.

Usage

```r
conditionPlot(
  CVdata,
  var,
  varVal,
  pointColor = "steelblue",
  sim = NULL,
  resetpar = TRUE,
  plotrows = NULL
)
```

Arguments

- `CVdata`: the dataset used for the fit
- `var`: one more condition vars. Draws a parallel coordinate plot for more than two.
- `varVal`: the current setting of the conditionvars, shown in magenta.
- `pointColor`: a color, vector of colors, or the name of variable to be used for coloring
- `sim`: If non-NULL should be a vector of similarity weights.
- `resetpar`: For use with shiny app.
- `plotrows`: If non-NULL should be a vector of case indices
**Examples**

```r
conditionPlot(mtcars, c("wt","hp"), c("wt"=3, "hp"=200), pointColor="am")

conditionPlot(mtcars, c("wt","hp"), mtcars[1,], pointColor="am")
```

# Calculate similarity using wt, hp observations from first case
```r
sim <- similarityweight(mtcars[1, c("wt","hp")], mtcars[, c("wt","hp")], threshold=1)
```

# Marks points with black border with positive sim values. These are points within 1 (threshold) sd of pink cross.
```r
conditionPlot(mtcars, c("wt","hp"), mtcars[1,], pointColor="am", sim=sim)
```

```r
sim <- similarityweight(mtcars[1, ], mtcars, threshold=2)
conditionPlot(mtcars, names(mtcars), mtcars[1,], sim=sim)
```

---

### condvis

*Creates Condvis Shiny app*

**Description**

Creates Condvis Shiny app

**Usage**

```r
condvis(
  data,
  model = NULL,
  response = NULL,
  sectionvars = NULL,
  conditionvars = NULL,
  predsInit = NULL,
  pointColor = c("steelblue", "grey0"),
  cPlotPCP = FALSE,
  cPlotn = 1000,
  orderConditionVars = "default",
  threshold = 1,
  thresholdmax = NULL,
  linecols = NULL,
  showsim = NULL,
  theta3d = 45,
  phi3d = 20,
  dataplot = "pcp",
  tours = NULL,
  predictArgs = NULL,
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
)```
density = FALSE,
showdata = density == FALSE,
displayHeight = 950
)

Arguments

data the dataset used for the fit. Should not have NAs for response, sectionvars or conditionvars.
model A fitted model or list of models. May be NULL.
response name of response variable. If null, tries to extract from model.
sectionvars names of sectionvars. If null, extracts from data.
conditionvars names of condition vars. If null, extracts from data.
predsInit Optionally provide starting value for some predictors. Defaults to medoid.
pointColor a color, or the name of variable to be used for coloring. If the named variable is numeric, it is first converted to a factor with 3 levels.
cPlotPCP if TRUE, conditionplots are drawn as a single PCP (for more than two conditionvars)
cPlotn Defaults to 1000. Shows a sample of this number of points in conditionplots. Non-numeric values are ignored.
orderConditionVars If supplied, a function to order the Condition Vars
threshold used for similarity weights, defaults to 1.
thresholdmax maximum value allowed of threshold.
linecols vector of colors to be used for fits
showsim if TRUE, shows sim in conditionplots with points/lines. Defaults to TRUE with 150 or fewer cases.
theta3d, phi3d Angles defining the viewing direction for 3d surface. theta3d gives the azimuthal direction and phi3d the colatitude. See persp.
dataplot "pcp" or "pairs". Used when there is no response, or more than two sectionvars.
tours A list of pre-calculated tours
predictArgs a list with one entry per fit, giving arguments for CVpredict
xlim passed on to sectionplot
ylim passed on to sectionplot
zlim passed on to sectionplot
density default FALSE. Use TRUE if model is a density function.
showdata defaults to density==TRUE. If FALSE, data on section not shown.
displayHeight supply a value for the display height

Examples

fit <- lm(mpg ~ wt+hp+am, data=mtcars)
if(interactive(){
  condvis(mtcars,fit, response="mpg",sectionvars="wt", conditionvars=c("am", "hp"), pointColor ="red")
})
createCVServer

Title Creates a shiny server

Description
Title Creates a shiny server

Usage
createCVServer(
CVfit,
CVdata = NULL,
response = NULL,
sectionvars,
conditionvars,
predsInit = NULL,
pointColor = NULL,
cPlotPCP = FALSE,
cPlotn = 1000,
orderConditionVars,
threshold = 1,
thresholdmax,
tours = NULL,
linecols = NULL,
dataplot = "pcp",
probs,
view3d,
theta3d,
phi3d,
predictArgs,
xlim = NULL,
ylim = NULL,
zlim = NULL,
density = FALSE,
showdata = TRUE
)

Arguments
CVfit a list of fits
CVdata the dataset used for the fit
response name of response variable
sectionvars names of at most two sectionvars
conditionvars names of conditionvars
predsInit starting value for predicts. Defaults to medoid.
pointColor a color, or the name of variable to be used for coloring. If the named variable is numeric, it is first converted to a factor with 3 levels.
cPlotPCP if TRUE, conditionplots are drawn as a single PCP (for more than two conditionvars)
cPlotn Shows a sample of this number of points in conditionplots.
orderConditionVars If supplied, a function to order the Condition Vars
threshold used for similarity weights, defaults to 1.
thresholdmax maximum value allowed of threshold.
tours A list of pre-calculated tours
linecols vector of colors to be used for fits
dataplot "pcp" or "pairs". Used when there is no response, or more than two sectionvars.
probs Logical; if TRUE, shows predicted class probabilities instead of just predicted classes.
view3d Logical; if TRUE, includes option for a three-dimensional regression surface if possible.
theta3d, phi3d Angles defining the viewing direction. theta3d gives the azimuthal direction and phi3d the colatitude. See persp.
predictArgs a list with one entry per fit, giving arguments for CVpredict
xlim passed on to sectionplot
ylim passed on to sectionplot
zlim passed on to sectionplot
density default FALSE. Use TRUE if model is a density function.
showdata If FALSE, data on section not shown.

Value

a function

createCVUI Constructs UI for Condvis

Description

Constructs UI for Condvis
createCVUI

Usage

createCVUI(
    CVfit,
    data,
    response,
    sectionvars,
    preds = NULL,
    pointColor,
    threshold = 1,
    thresholdmax,
    tours,
    probs,
    view3d,
    showsim,
    cPlotPCP
)

Arguments

CVfit a list of fits
data a dataset
response name of response variable
sectionvars names of sectionvars
preds names of predictors
pointColor a color, or the name of variable to be used for coloring
threshold used for similarity weights, defaults to 1.
thresholdmax maximum value allowed of threshold.
tours A list of pre-calculated tours
probs Logical; if TRUE, shows predicted class probabilities instead of just predicted classes.
view3d Logical; if TRUE, includes option for a three-dimensional regression surface if possible.
showsim if TRUE, shows sim in conditionplots with points/lines. Defaults to TRUE with 150 or fewer cases.
cPlotPCP if TRUE, conditionplots are drawn as a single PCP (for more than two conditionvars)

Value

a dataframe of conditions
CVpredict

A predict generic function for condvis

Description
A predict generic function for condvis

Usage
CVpredict(
  fit,
  newdata,
  ...,  
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL,
  pinterval = NULL,
  pinterval_level = 0.95
)

## Default S3 method:
CVpredict(
  fit,
  newdata,
  ...,  
  ptype = "pred",
  pthreshold = NULL,
  pinterval = NULL,
  pinterval_level = 0.95,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'lm'
CVpredict(
  fit,
  newdata,
  ...,  
  ptype = "pred",
  pthreshold = NULL,
  pinterval = NULL,
  pinterval_level = 0.95,
  ylevels = NULL,
  ptrans = NULL
)
## S3 method for class 'glm'
CVpredict(
  fit,
  ...,
  type = "response",
  ptype = "pred",
  pthreshold = NULL,
  pinterval = NULL,
  pinterval_level = 0.95,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'lda'
CVpredict(
  fit,
  ...,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'qda'
CVpredict(
  fit,
  ...,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'nnet'
CVpredict(
  fit,
  ...,
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'randomForest'
CVpredict(
  fit,
  ...,
type = NULL,
ptype = "pred",
pthreshold = NULL,
ylevels = NULL,
ptrans = NULL
)

## S3 method for class 'ranger'
CVpredict(
  fit,
  ..., 
  type = NULL, ptype = "pred",
  pthreshold = NULL,
ylevels = NULL,
ptrans = NULL
)

## S3 method for class 'rpart'
CVpredict(
  fit,
  ..., 
  type = NULL, ptype = "pred",
  pthreshold = NULL,
ylevels = NULL,
ptrans = NULL
)

## S3 method for class 'tree'
CVpredict(
  fit,
  ..., 
  type = NULL, ptype = "pred",
  pthreshold = NULL,
ylevels = NULL,
ptrans = NULL
)

## S3 method for class 'C5.0'
CVpredict(
  fit,
  ..., 
  type = NULL, ptype = "pred",
  pthreshold = NULL,
ylevels = NULL,
CVpredict

```r

ptrans = NULL

## S3 method for class 'svm'
CVpredict(
  fit,
  ...,
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'gbm'
CVpredict(
  fit,
  ...,
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  n.trees = fit$n.trees,
  ptrans = NULL
)

## S3 method for class 'loess'
CVpredict(fit, newdata = NULL, ...)

## S3 method for class 'ksvm'
CVpredict(
  fit,
  newdata,
  ...,
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'glmnet'
CVpredict(
  fit,
  newdata,
  ...,
  type = "response",
  ptype = "pred",
```
CVpredict

```r
CVpredict(fit, newdata, ...
  type = "response",
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL,
  makex = NULL
)

## S3 method for class 'cv.glmnet'
CVpredict(fit,
  newdata,
  ...
  type = "response",
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL,
  makex = NULL
)

## S3 method for class 'glmnet.formula'
CVpredict(fit,
  newdata,
  ...
  type = "response",
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL,
  s = NULL
)

## S3 method for class 'cv.glmnet.formula'
CVpredict(fit,
  newdata,
  ...
  type = "response",
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'keras.engine.training.Model'
CVpredict(
  fit,
```
newdata,
..., ptype = "pred",
pthreshold = NULL,
ylevels = NULL,
ptrans = NULL,
batch_size = 32,
response = NULL,
predictors = NULL
)

## S3 method for class 'kde'
CVpredict(fit, newdata = fit$x, ..., scale = TRUE)

## S3 method for class 'densityMclust'
CVpredict(
  fit,
  newdata = NULL,
  ..., ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
ptrans = NULL,
scale = TRUE
)

## S3 method for class 'MclustDA'
CVpredict(
  fit,
  newdata,
  ..., ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
ptrans = NULL
)

## S3 method for class 'MclustDR'
CVpredict(
  fit,
  newdata,
  ..., ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
ptrans = NULL
)
## S3 method for class 'Mclust'
CVpredict(
  fit,
  newdata,
  ..., 
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'train'
CVpredict(
  fit,
  newdata,
  ..., 
  type = "response",
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'bartMachine'
CVpredict(
  fit,
  newdata,
  ..., 
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL
)

## S3 method for class 'model_fit'
CVpredict(
  fit,
  ..., 
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL,
  pinterval = NULL,
  pinterval_level = 0.95
)
## S3 method for class 'WrappedModel'
CVpredict(
  fit,
  newdata,
  ..., 
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL,
  pinterval = NULL,
  pinterval_level = 0.95
)

## S3 method for class 'Learner'
CVpredict(
  fit,
  newdata,
  ..., 
  type = NULL,
  ptype = "pred",
  pthreshold = NULL,
  ylevels = NULL,
  ptrans = NULL,
  pinterval = NULL,
  pinterval_level = 0.95
)

### Arguments

- **fit**: A fitted model
- **newdata**: Where to calculate predictions.
- **...**: extra arguments to predict
- **ptype**: One of "pred", "prob" or "probmatrix"
- **pthreshold**: Used for calculating classes from probs, in the two class case
- **ylevels**: The levels of the response, when it is a factor
- **ptrans**: A function to apply to the result
- **pinterval**: NULL, "confidence" or "prediction". Only for lm, parsnip, mlr(regression, confidence only)
- **pinterval_level**: Defaults to 0.95
- **type**: For some predict methods
- **n.trees**: Used by CVpredict.gbm, passed to predict
- **s**: Used by CVpredict.glmnet and CVpredict.cv.glmnet, passed to predict
- **makex**: Used by CVpredict.glmnet and CVpredict.cv.glmnet. A function to construct xmatrix for predict.
batch_size  Used by CVpredict.keras.engine.training.Model, passed to predict
response    Used by CVpredict.keras.engine.training.Model. Name of response (optional)
predictors  Used by CVpredict.keras.engine.training.Model. Name of predictors
scale       Used by CVpredict for densities. If TRUE (default) rescales the conditional density to integrate to 1.

Details

This is a wrapper for predict used by condvis. When the model response is numeric, the result is a vector of predictions. When the model response is a factor the result depends on the value of ptype. If ptype="pred", the result is a factor. If also threshold is numeric, it is used to threshold a numeric prediction to construct the factor when the factor has two levels. For ptype="prob", the result is a vector of probabilities for the last factor level. For ptype="probmatrix", the result is a matrix of probabilities for each factor level.

Value

a vector of predictions, or a matrix when type is "probmatrix"

Methods (by class)

- default: CVpredict method
- lm: CVpredict method
- glm: CVpredict method
- lda: CVpredict method
- qda: CVpredict method
- nnet: CVpredict method
- randomForest: CVpredict method
- ranger: CVpredict method
- rpart: CVpredict method
- tree: CVpredict method
- C5.0: CVpredict method
- svm: CVpredict method
- gbm: CVpredict method
- loess: CVpredict method
- ksvm: CVpredict method
- glmnet: CVpredict method
- cv.glmnet: CVpredict method
- glmnet.formula: CVpredict method
- cv.glmnet.formula: CVpredict method
- keras.engine.training.Model: CVpredict method
- kde: CVpredict method
• densityMclust: CVpredict method
• MclustDA: CVpredict method
• MclustDR: CVpredict method
• Mclust: CVpredict method
• train: CVpredict method for caret
• bartMachine: CVpredict method
• model_fit: CVpredict method for parsnip
• WrappedModel: CVpredict method for mlr
• Learner: CVpredict method for mlr3

Examples

# Fit a model.
f <- lm(Fertility~ ., data=swiss)
CVpredict(f)

# Fit a model with a factor response
swiss1 <- swiss
swiss1$Fertility <- cut(swiss$Fertility, c(0, 80, 100))
levels(swiss1$Fertility) <- c("lo", "hi")
f <- glm(Fertility ~ ., data=swiss1, family="binomial")
CVpredict(f) # by default gives a factor
CVpredict(f, ptype="prob") # gives prob of level hi
CVpredict(f, ptype="probmatrix") # gives prob of both levels

fitPath

Description

Constructs tours of data space based on fits.

Usage

lofPath(
  data,
  fits,
  length = 10,
  reorder = TRUE,
  conditionvars = NULL,
  predictArgs = NULL,
  response = NULL,
  ...
)

diffsfitPath(
data,  
fits,  
length = 10,  
reorder = TRUE,  
conditionvars = NULL,  
predictArgs = NULL,  
...  
)

hiresponsePath(  
data,  
response = NULL,  
length = 10,  
reorder = TRUE,  
conditionvars = NULL,  
...  
)

loresponsePath(  
data,  
response = NULL,  
length = 10,  
reorder = TRUE,  
conditionvars = NULL,  
...  
)

Arguments

data  A dataframe
fits  A model fit or list of fits
length  Path length, defaults to 10
reorder  If TRUE (default) uses DendSer to reorder the path dser
conditionvars  A vector of variable names. The returned tour is for this subset of variables.
predictArgs  Extra inputs to CVpredict
response  The name of the response variable
...  ignored

Value

A dataframe with the path

Functions

• lofPath: Constructs a tour of data space showing biggest absolute residuals from fits.
• diffitsPath: Constructs a tour of data space showing biggest differences in fits.
• hiresponsePath: Constructs a tour of data space showing high (numeric) response values
• loresponsePath: Constructs a tour of data space showing low (numeric) response values
Examples

```r
fit1 <- lm(mpg ~ wt+hp+am, data=mtcars)
fit2 <- lm(mpg ~ wt, data=mtcars)
lofPath(mtcars, fit1, response="mpg")
diffsPath(mtcars, list(fit1, fit2))
```

---

### medoid

**Finds medoid of data**

**Description**

Finds medoid of data

**Usage**

```r
medoid(data, maxn = 4000)
```

**Arguments**

- `data`: A dataframe
- `maxn`: For datasets with more than maxn rows, use maxn randomly selected rows.

**Value**

A dataframe with one row, which is the medoid of the data, based on (standardised) daisy dist

**Examples**

```r
medoid(mtcars)
```

---

### pathInterpolate

**Interpolation**

**Description**

Interpolation
Usage

pathInterpolate(x, ninterp = 4)

## Default S3 method:
pathInterpolate(x, ninterp = 4L)

## S3 method for class 'factor'
pathInterpolate(x, ninterp = 4L)

## S3 method for class 'data.frame'
pathInterpolate(x, ninterp = 4L)

Arguments

x a numeric or factor vector or dataframe
ninterp number of interpolated steps

Value

interpolated version of x

Methods (by class)

• default: Default interpolate method
• factor: pathInterpolate method for factor
• data.frame: pathInterpolate method for data.frame

Description

Plots diagnostics for the tour supplied

Usage

plotTourDiagnostics(
  path,
  data,
  pathlen = nrow(path),
  threshold = 1,
  which = 1:3,
  ...
)

plotTourDiagnostics Plots diagnostics for the tour supplied
sectionPlot

Arguments

- **path**: the tour
- **data**: the dataset
- **pathlen**: the pathlength
- **threshold**: used for similarityweight
- **which**: subset of 1:3
- **...**: other args for similarityweight

Details

The first plot shows approximately how much data are visible on each section, the second shows what proportion of data are *visited* by the tour, and the third a density estimate of max similarity values.

Value

Table of max sims attained.

---

sectionPlot

*Plots the main condvis display*

Description

The section plot relates a fit or fits to one or two predictors (**sectionvar**), for fixed values of other predictors in **conditionvals**.

Usage

```r
sectionPlot(
  CVdata,
  CVfit = NULL,
  response = NULL,
  preds,
  sectionvar,
  conditionvals,
  pointColor = "steelblue",
  sim = NULL,
  threshold = 1,
  linecols = NULL,
  dataplot = "pcp",
  gridsize = 50,
  probs = FALSE,
  view3d = FALSE,
  theta3d = 45,
  phi3d = 20,
)```
sectionPlot

xlim = NULL,
ylim = NULL,
zlim = NULL,
pointsSize = 1.5,
predictArgs = NULL,
resetpar = TRUE,
density = FALSE,
showdata = density == FALSE,
returnInfo = FALSE,
pointColorFromResponse = FALSE,
pcolInfo = NULL
)

Arguments

CVdata the dataset used for the fit
CVfit a fit or list of fits
response name of response variable
preds names of predictors
sectionvar section variable, or variables.
conditionvals conditioning values. A vector/list or dataframe with one row
pointColor a color, vector of colors, or the name of variable to be used for coloring
sim vector of similarity weights
threshold used for similarity weights, defaults to 1.
linecols vector of line colours
dataplot "pcp" or "pairs". Used when there is no response, or more than two sectionvars.
gridsize used to construct grid of fitted values.
probs Logical; if TRUE, shows predicted class probabilities instead of just predicted classes. Only available with two numeric sectionvars and the model’s predict method provides this.
view3d Logical; if TRUE plots a three-dimensional regression surface if possible.
theta3d, phi3d Angles defining the viewing direction. theta3d gives the azimuthal direction and phi3d the colatitude. See `persp`.
xlim passed on to plot
ylim passed on to plot
zlim passed on to plot
pointSize used for points
predictArgs a list with one entry per fit, giving arguments for predict
resetpar When TRUE (the default) resets pars after drawing.
density default FALSE. Use TRUE if model is a density function.
showdata If FALSE, data on section not shown.
returnInfo If TRUE, returns coordinates for some plots
pointColorFromResponse ignore–For interactive use only
pcolInfo ignore–For interactive use only
Details

The type of plot depends on the fit and the section variables. Observations with non zero values of the similarity weights \( sim \) are shown. If no fit is provided, the data are shown as a parallel coordinate plot or pairs plot, depending on \( dataplot \). The fit could also be a density estimate.

Value

plotted coordinates, for some plots

Examples

```r
# Fit a model.
 f <- lm(Fertility~ ., data=swiss)
 svar <- "Education"
 preds <- variable.names(f)[-1]
 sectionPlot(swiss,f, "Fertility",preds,svar, swiss[12,])
 sectionPlot(swiss,f, "Fertility",preds,svar, apply(swiss,2,median))
 sectionPlot(swiss,f, "Fertility",preds,preds[1:2], apply(swiss,2,median))
 sectionPlot(swiss,f, "Fertility",preds,preds[1:2], apply(swiss,2,median), view3d=TRUE)

# PCP of swiss data, showing only cases whose percent catholic and infant.mortality are
# similar to those of the first case
 sectionPlot(swiss,preds=names(swiss),
            sectionvar= names(swiss)[1:4],conditionvals=swiss[1,] )
 # Use dataplot="pairs" to switch to a pairs plot

# A density estimate example
## Not run:
 library(ks)
 fde <-kde(iris[,1:3],
 sectionPlot(iris,list(kde=fde), response=NULL,
            preds=names(iris)[1:3],
            sectionvar=names(iris)[1],
            conditionvals=iris[1,],density=TRUE)

## End(Not run)
```

similarityweight

Calculate the similarity weight for a set of observations

Description

Calculate the similarity weight for a set of observations, based on their distance from some arbitrary points in data space. Observations which are very similar to the point under consideration are given weight 1, while observations which are dissimilar to the point are given weight zero.
similarityweight

Usage

```r
similarityweight(
  x, 
  data, 
  threshold = 1, 
  distance = "euclidean", 
  lambda = NULL, 
  scale = TRUE
)
```

Arguments

- **x**: A dataframe describing arbitrary points in the space of the data (i.e., with same colnames as `data`).
- **data**: A dataframe representing observed data.
- **threshold**: Threshold distance outside which observations will be assigned similarity weight zero. This is numeric and should be > 0. Defaults to 1.
- **distance**: The type of distance measure to be used, currently just three types of Minkowski distance: "euclidean" (default), "maxnorm", "manhattan" and also "gower".
- **lambda**: A constant to multiply by the number of categorical mismatches, before adding to the Minkowski distance, to give a general dissimilarity measure. If left `NULL`, behaves as though `lambda` is set larger than `threshold`, meaning that one factor mismatch guarantees zero weight.
- **scale**: defaults to `TRUE`, in which case numeric variables are scaled to unit sd.

Details

Similarity weight is assigned to observations based on their distance from a given point. The distance is calculated as Minkowski distance between the numeric elements for the observations whose categorical elements match, or else the Gower distance.

Value

A numeric vector or matrix, with values from 0 to 1. The similarity weights for the observations in `data` arranged in rows for each row in `x`.

References


Examples

```r
## Say we want to find observations similar to the first observation.
## The first observation is identical to itself, so it gets weight 1. The
## second observation is similar, so it gets some weight. The rest are more
## different, and so get zero weight.
```
```r
data(mtcars)
similarityweight(x = mtcars[1, ], data = mtcars)

## By increasing the threshold, we can find observations which are more
## approximately similar to the first row. Note that the second observation
## now has weight 1, so we lose some ability to discern how similar
## observations are by increasing the threshold.

similarityweight(x = mtcars[1, ], data = mtcars, threshold = 5)

## Can provide a number of points to 'x'. Here we see that the Mazda RX4 Wag
## is more similar to the Merc 280 than the Mazda RX4 is.

similarityweight(mtcars[1:2, ], mtcars, threshold = 3)
```

### tours

*Constructs a various tours of data space*

#### Description

Constructs a various tours of data space

#### Usage

```r
randomPath(data, length = 10, reorder = TRUE, conditionvars = NULL, ...)
```

```r
seqPath(data, length = 10, reorder = FALSE, conditionvars = NULL, ...)
```

```r
alongPath(data, var, length = 10, current = NULL, ...)
```

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A dataframe</td>
</tr>
<tr>
<td>length</td>
<td>Path length, defaults to 10</td>
</tr>
<tr>
<td>reorder</td>
<td>If TRUE (default) uses DendSer to reorder the path dser</td>
</tr>
<tr>
<td>conditionvars</td>
<td>A vector of variable names. The returned tour is for this subset of variables.</td>
</tr>
<tr>
<td>...</td>
<td>ignored</td>
</tr>
<tr>
<td>var</td>
<td>A variable name for alongPath</td>
</tr>
<tr>
<td>current</td>
<td>Default value of variables for alongPath</td>
</tr>
</tbody>
</table>

#### Value

A dataframe with the path
Functions

- `randomPath`: Constructs a tour of data space following random observations
- `seqPath`: Constructs a tour of data space following first length observations
- `alongPath`: Constructs a tour of data space of length equi-spaced values in the range of var. If var is a factor, its levels are used.

Examples

```r
randomPath(mtcars, length=5)
seqPath(mtcars, length=5)
alongPath(mtcars, var="mpg", length=5, current=mtcars[1,])
```

```
weightcolor(col, weights, nlevels = 5)
```

**Arguments**

- `col` A vector of colour
- `weights` A vector of weights, values between 0 and 1
- `nlevels` The number of groups

**Value**

A vector of colours
Index

alongPath (tours), 27
arrangeC, 2

centroidPath (clusPath), 3
ciara, 4
ciaraPath (clusPath), 3
clusPath, 3
conditionPlot, 4
condvis, 5
createCVServer, 7
createCVUI, 8
CVpredict, 10
diffsPath (fitPath), 19
dser, 3, 20, 27
fitPath, 19
hiresponsePath (fitPath), 19
kmeansPath (clusPath), 3
lofPath (fitPath), 19
loresponsePath (fitPath), 19

medoid, 21
medoidPath (clusPath), 3

pam, 4
pamPath (clusPath), 3
pathInterpolate, 21
persp, 6, 8, 24
plotTourDiagnostics, 22
randomPath (tours), 27
sectionPlot, 23
seqPath (tours), 27
similarityweight, 25
tours, 27

weightcolor, 28