Package 'Causata'

February 19, 2015

Type Package

Title Analysis utilities for binary classification and Causata users.

Version 4.2-0

Date 2013-07-18

Author Justin Hemann, David Barker, Suzanne Weller, Jason McFall

Maintainer Justin Hemann <justinh@causata.com>

Description The Causata package provides utilities for
extracting data from the Causata application, training binary classification
models, and exporting models as PMML for scoring.

Depends R (>= 2.15.1)

Imports XML, R.utils, rjson, RMySQL, RCurl, stringr, yaml, boot,
foreach, data.table, glmnet, ggplot2

Suggests doMC, testthat, pROC, knitr

License GPL

LazyLoad yes

VignetteBuilder knitr

URL www.causata.com

NeedsCompilation no

Repository CRAN

Date/Publication 2013-07-18 18:01:32

R topics documented:

`Causata-package` *Causata analysis utilities*

Description

Causata analysis utilities

`BinaryCut` *Cuts a numeric independent variable into bins.*

Description

A numeric independent variable is discretized and returned as a factor. A binary dependent variable is used to select the bins using a simple, fast algorithm based on quantiles.

Usage

```
BinaryCut(iv, dv, nbins=10,
  minBin=ceiling(min(table(dv))/50),
  woeDelta=0.1, bins=FALSE, debug=FALSE)
```

Arguments

`iv`	A numeric independent variable that will be cut into bins. Missing values will be ignored during binning and replaced using `CleanNaFromFactor`.
`dv`	The dependent variable must be an array of values with the same length as `iv`. It can be numeric with only two unique values, or a factor with two levels. Missing values are not allowed.
`nbins`	The number of bins to break `iv` into. The actual number of bins returned may be lower due to merging. Must be >=2.
`minBin`	Each bin will have at least `minBin` values for each of the classes in the binary dependent variable, subject to the constraint that at least two bins are returned. The default is 2% of the data in the smaller class of the dependent variable. Set to 0 to disable merging by counts. Optionally, a function can be provided to calculate `minBin`. The function must accept `iv` and `dv` as the only two arguments, in that order.
`woeDelta`	If the absolute value of the Weight Of Evidence for adjacent bins falls below this threshold, then the bins are merged. See `Woe` for more information. Set to 0 to disable merging.
`bins`	If TRUE the breaks are returned, along with the factor, in a list.
`debug`	If TRUE debug information will be printed to the screen.

Details

This function is similar to cut, but it uses a dependent variable to inform the binning. The algorithm is designed to be fast and simple; it is a slightly modified version of an equal frequency approach (quantiles).

The algorithm works as follows:

1. The independent variable is filtered to include only non-missing values, and values from the smaller class of the dependent varaible.

2. The filtered independent variable is used to compute nbins quantiles. For the special case where there are fewer unique values than bins the unique values are used as the quantiles.
3. The first and last quantiles are adjusted, if necessary, to include all independent variable values regardless of their dependent variable class.
4. The independent variable is cut into bins using the quantiles as boundaries.
5. Each class of the dependent variable is counted in each bin. If the count is below minBin for either class then the bin is merged with the smallest adjacent bin. This merge process continues until all bins have a sufficient count of dependent variable values, or until there are 2 bins left.
6. The Weight of Evidence is calculated for each bin. If the difference in the WOE for adjacent bins falls below a threshold defined in terms of woeDelta then the bins are merged.

Value

If bins is FALSE then a factor with up to nbins levels is returned, where the level names are as found from cut. Missing values in the independent variable are returned as missing values in the output, and are not counted as a bin.

If bins is TRUE then a list is returned with two elements:

1. fiv A factor representation of the independent variable, as described above.
2. breaks A vector of breaks or cutpoints used to discretize the independent variable.

Author(s)

Justin Hemann <support@causata.com>

See Also

cut, Woe.

Examples

```
data(df.causata)
dv <- df.causata$has.responded.mobile.logoff_next.hour_466
iv <- df.causata$online.number.of.page.views_last.30.days_3
f <- BinaryCut(iv,dv)

# compute the weight of evidence for each bin
woe <- Woe(f, dv)

# adjust plot margins to increase space for bin labels
par(oma=c(1,8,1,1))

# plot the bins against the weight of evidence
barplot(woe$woe.levels, names.arg=levels(f), horiz=TRUE, las=1,
  main="Weight of Evidence for clicking a banner for a mobile app.",
  sub="WOE vs. Page View Count, Last 30 Days" )
```

BinaryPredictor *Univariate analysis for binary classification.*

Description

An independent variable is evaluated as a predictor for a binary dependent variable. The independent variable may be numeric, a factor, or a data frame containing numeric and factor columns.

Usage

```
## S3 method for class 'factor'
BinaryPredictor(iv, dv, min.power=0.01, min.robustness=0.5,
  max.missing=0.99, max.levels=20, civ=NULL, copy.data=FALSE, name=NULL, ...)

## S3 method for class 'numeric'
BinaryPredictor(iv, dv, min.power=0.01, min.robustness=0.5,
  max.missing=0.99, copy.data=FALSE, name=NULL, ...)

## S3 method for class 'data.frame'
BinaryPredictor(iv, dv, min.power=0.01, min.robustness=0.5,
  max.missing=0.99, verbose=FALSE, copy.data=FALSE, ...)

## Default S3 method:
BinaryPredictor(iv, dv, ...)

## S3 method for class 'BinaryPredictor'
plot(x, y=NULL, type="bin", plot.missing=TRUE, ...)

## S3 method for class 'BinaryPredictorList'
print(x, file=NULL, silent=FALSE, ...)
```

Arguments

iv	The independent variable(s). May be a factor, numeric, or a data frame.
dv	The dependent variable, which may have only two unique values. The length / number of rows in iv must match the length of dv.
min.power	The minimum predictive power from PredictivePowerCv for a variable to be kept.
min.robustness	The minimum robustness from PredictivePowerCv for a variable to be kept.
max.missing	The maxmimum allowable fraction of missing values for a variable to be kept.
max.levels	For factors, this controls the merging of small bins using MergeLevels.
civ	When a continuous variable is discretized, the original continuous data can be provided in civ so that linearity can be computed. See Woe for more information.
copy.data	Reserved for future use, indicates if the data should be copied.
name	The variable name. If NULL it will be extracted from the deparsed input iv.

...	For the BinaryPredictor functions the extra arguments are passed to PredictivePowerCv. If iv is numeric then extra arguments are also passed to BinaryCut. For plot the extra arguments are passed to ShortenStrings, which is used to shorten the names of factor levels in plots.
verbose	If true then calculation information is printed.
x	Output from one of the BinaryPredictor functions.
y	Unused argument for the generic plot function.
plot.missing	When plotting numeric variables a TRUE value will add a horizontal line representing the log odds associated with missing values.
type	Reserved for future use, indicates the type of plot to be generated. The only valid value now is 'bin'.
file	If a filename is provided then summary information will be written to a text file.
silent	If set to TRUE then nothing is printed to the screen.

Details

The BinaryPredictor family of functions are used to evaluate predictors of a binary outcome. Checks are executed for the variable class (only numeric, integer, and factor are allowed), missing values, predictive power, and robustness. If any checks fail then a "keep" flag is set to FALSE, otherwise it's TRUE.

The plot function generates a summary plot of the predictor. Predictive power and robustness are printed in the plot title, along with the smallest and largest bin sizes used during discretization. For numeric variables a count of missing values is also printed.

The print function writes a table of variable summary information to the screen or to a file.

Value

If iv is a vector then an object of class BinaryPredictor is returned with the folowing items:

name	The variable name.
keep	A boolean indicating if the variable meets the criteria for missing values, predictive power, etc.
reason	If keep=FALSE then this field contains a text string indicating the first criteria the variable failed to meet.
missing	The fraction of values that are missing / NA.
class	The variable class.
predictivePower	Results from PredictivePowerCv.
woe	Results from Woe.

If iv is a data frame then a list of BinaryPredictor objects is returned with class BinaryPredictorList.

The print.BinaryPredictorList function returns a data frame with columns for the values in the BinaryPredictor output. The values include the variable name, predictive power, robustness, etc.

Author(s)

Justin Hemann <support@causata.com>

See Also

PredictivePowerCv, BinaryCut, MergeLevels, Woe,ShortenStrings.

Examples

```
library(ggplot2)
data(diamonds)
# set a dependent variable that is TRUE when the price is above $5000
dv <- diamonds$price > 5000

# convert ordered to factor
diamonds$cut <- as.factor(as.character(diamonds$cut))
diamonds$color <- as.factor(as.character(diamonds$color))
diamonds$clarity <- as.factor(as.character(diamonds$clarity))

# evaluate diamond cut and carats, and generate a plot for each
bp.cut <- BinaryPredictor(diamonds$cut, dv)
plot(bp.cut)
bp.carat <- BinaryPredictor(diamonds$carat, dv)
plot(bp.carat)

# Evaluate all predictors, print summary to screen
# note that price does not have 100% predictive
# power since the discreatization boundary is not $5000.
# Using a sample of 10k records and 3 folds of cross validation
# for greater speed.
set.seed(98765)
idx <- sample.int(nrow(diamonds), 10000)
bpList <- BinaryPredictor(diamonds[idx, ], dv[idx], folds=3)
df.summary <- print(bpList)
```

CausataConfig *Creates an object of class CausataConfig for working with Causata from within R.*

Description

The CausataConfig object stores information that is used to connect to a configuration server to upload models, add variables, delete variables, etc.

Usage

```
CausataConfig(config.server.host, config.server.port, config.username,
  config.password, protocol="https://", group=NULL)
is.CausataConfig(this)
```

Arguments

config.server.host	The host server URL.
config.server.port	The host server port.
config.username	Username for Causata configuration access.
config.password	Password for a Causata configuration access.
protocol	The protocol for connecting to the server.
group	Configuration data from the group provided will be loaded from the configuration file. See LoadCausataConfig. If the same parameter is provided in the configuration file and the function argument then the function argument will take precedence.
this	An argumement that will be tested, see is.

Details

Contact your Causata engagement manager for the settings required for these parameters.

Value

An object of class CausataConfig is returned.

Author(s)

David Barker <support@causata.com>

See Also

Config.DeleteVariable, Config.CreatePrimaryVariable, UploadModel, LoadCausataConfig, is.

Examples

```
# The settings below are not for an actual server,
# they are for illustration purposes only.
config <- CausataConfig("server.causata.com","8002","causatauser","Bg20qydd6")
```

CausataData *Creates an object of class CausataData for scoring in Causata.*

Description

Used for creating an object of class CausataData for scoring in Causata. This is essentially a dataframe with modeling data and a list of objects of class CausataVariable. Note that the variable names in the data frame must match

Usage

```
CausataData(dataframe, dependent.variable=NULL, query=NULL)
```

Arguments

dataframe	A data frame containing independent variables for modeling.
dependent.variable	An array of dependent variable values, or the name of a column in the data frame that will be used as the dependent variable. If NULL then dataframe must have a column named dependent.variable.
query	An optional Query object that can be stored with the data.

Details

A CausataData object is a container for objects from the CausataVariable class. The CausataData object is passed into ToPMML to create a PMML representation of a model.

Note that column names in the dataframe have to follow Causata naming conventions so that they can be matched to variables in Causata when scoring. See RToCausataNames for details. Columns that do not follow conventions will remain in the data frame, but will not have a corresponding CausataVariable generated.

The dependent variable name is set depending on the dependent.variable argument:

- If a name is provided then that is used.
- If a vector of data is provided then the name attribute of the vector will be used. If the name attribute is not set then the default name of "dependent.variable" will be used.

Value

CausataData returns an object of class CausataData. The object has the following fields:

df	The dataframe. If there is not a column for the dependent variable then it is added.
variableList	A list of variables of class CausataVariable.
dvName	The name of the dependent variable.

`skippedVariables`
: Variables in `df` with names that don't meet naming conventions and do not have corresponding `CausataVariable` objects.

`query`
: The object passed in through the query argument is stored here.

`GetQuery` returns the query object.

Author(s)

Justin Hemann <support@causata.com>

See Also

`CausataVariable`, `ToPmml`, `Query`, `RToCausataNames`.

Examples

```
df <- data.frame(f1__AP=factor(c("a","b",NA)), f2__AP=factor(c("x","y",NA)))
causataData <- CausataData(df, rep(0,nrow(df)))
```

CausataToRNames *Converts Causata system names to R-friendly column names.*

Description

Causata system names are converted to R-friendly column names in a reversible way.

Usage

```
CausataToRNames(name.vector)
```

Arguments

`name.vector`
: A character vector of Causata system names. E.g. c("total-spend$All Past")

Details

Causata system names may include lowercase letters, numbers, and dashes. Dashes (-)are not allowed in R names, so they are mapped to dots (.).

Causata uses a dollar sign ($) to separate a system name and a time domain. Dollar signs are not allowed in R names, so they are mapped to a double underscore (__).

Value

A list, mapping each input name to the corresponding R name.

Author(s)

David Barker <support@causata.com>

See Also

CausataData, CausataVariable, RToCausataNames.

Examples

```
CausataToRNames(c("variable-name$Time Domain"))
```

CausataVariable *Creates an object of class CausataVariable for scoring in Causata.*

Description

Preprocessing steps are recorded in a CausataVariable object so they can be written to PMML and executed in Causata.

Usage

```
CausataVariable(variableName, values, causata.name=variableName)
```

Arguments

variableName	A character string containing the variable name.
values	An array of values for the variable, or a factor.
causata.name	A character string containing the variable name in a Causata format.

Details

The CausataVariable class is used to record preprocessing steps such as replacing outliers, binning, and replacing missing values. Typically CausataVariable is not called directly – instead it is called repeatedly by CausataData.

Value

An object of class CausataVariable is returned.

Author(s)

Justin Hemann <support@causata.com>

See Also

CausataData.

Examples

```
CausataVariable("variable.name__All.Past", c(1,2,3,4,5))
```

CleanNaFromContinuous *Replaces missing values in an array of numeric values.*

Description

Replaces missing values in an array of numeric values.

Usage

```
## S3 method for class 'numeric'
CleanNaFromContinuous(x, method="median", replacement.value=NULL,
  return.replacement=FALSE, ...)

## S3 method for class 'POSIXct'
CleanNaFromContinuous(x, method="median", replacement.value=NULL,
  return.replacement=FALSE, ...)
```

Arguments

x	An array of numerical (continuous) values and missing values.
method	Sets the method used to replace missing values. Valid values are "median" and "mean".
replacement.value	If this argument is not NULL then missing values will be replaced with the value provided instead of a mean or median.
return.replacement	If FALSE then an array is returned with missing values replaced. If TRUE then a list is returned with the array and the replacement value.
...	Unused arguments for other methods.

Details

The default behavior is that missing numeric values are replaced with the median.

Value

Returns an array with missing values replaced. Optionally a list can be returned with the array and the replacement value.

Author(s)

Justin Hemann <support@causata.com>

Examples

```
# numeric, median is 2, mean is 3.7
x <- c(NA, 1,1.5,2,3,11)
CleanNaFromContinuous(x)
```

`CleanNaFromContinuous.CausataData` *Replaces missing values in an array of numeric values in a CausataData object.*

Description

Missing values are replaced in a `CausataData` object, and the process is recorded so that it can be reapplied during scoring.

Usage

```
## S3 method for class 'CausataData'
CleanNaFromContinuous(x, variableName=NULL, method="median", ...)
```

Arguments

`x`	An object of class `CausataData`.
`variableName`	If a name is supplied then the variable matching the name will have `CleanNaFromContinuous` applied. If no name is provided then `CleanNaFromContinuous` is applied to every continuous variable in `causataData`.
`method`	See `CleanNaFromContinuous`.
`...`	Unused arguments for other methods.

Details

The default behavior is that missing numeric values are replaced with the median.

Value

Returns a `CausataData` object.

Author(s)

Justin Hemann <support@causata.com>

See Also

`CleanNaFromContinuous`, `CausataData`.

Examples

```
# median of x__AP is 2
df <- data.frame(x__AP=c(NA, 1,1.5,2,3,11), y__AP=c(NA, 1,2,3,4,5))
causataData <- CausataData(df, rep(0, nrow(df)))
causataData <- CleanNaFromContinuous(causataData)
```

`CleanNaFromFactor` *Replaces missing values in a factor.*

Description

Missing values are replaced with the string "BLANK", or any other string supplied as an input.

Usage

```
## S3 method for class 'factor'
CleanNaFromFactor(x, replacement="BLANK", ...)
```

Arguments

`x`	A factor that may contain missing values.
`replacement`	Levels with missing values will be replaced with this string.
`...`	Unused arguments for other methods.

Value

Returns a factor that matches the input factor except that missing values are replaced.

Author(s)

Justin Hemann <support@causata.com>

Examples

```
f <- as.factor(c("a","b","c",NA))
CleanNaFromFactor(f)
```

`CleanNaFromFactor.CausataData`
Given a factor in a CausataData object, this replaces missing values.

Description

Missing values are replaced with the string "BLANK", or any other string supplied as an input.

Usage

```
## S3 method for class 'CausataData'
CleanNaFromFactor(x, variableName=NULL, replacement="BLANK", ...)
```

Arguments

x	An object from CausataData.
variableName	If a name is supplied then the variable matching the name will have CleanNaFromFactor applied. If no name is provided then CleanNaFromFactor is applied to every factor in causataData.
replacement	Levels with missing values will be replaced with this string.
...	Unused arguments for other methods.

Value

Returns a CausataData object with the missing value replacements recorded.

Author(s)

Justin Hemann <support@causata.com>

See Also

CleanNaFromFactor, CausataData.

Examples

```
df <- data.frame(f1__AP=factor(c("a","b",NA)), f2__AP=factor(c("x","y",NA)))
causataData <- CausataData(df, rep(0,nrow(df)))
causataData <- CleanNaFromFactor(causataData)
```

Config.CreatePrimaryVariable

Defines a Causata primary variable.

Description

This function attempts to add a new variable to variable set configured in Causata.

Usage

```
Config.CreatePrimaryVariable(causata.config, variable.name,
                             variable.display.name=variable.name,
                             variable.expression)
```

Arguments

causata.config	An object from CausataConfig.
variable.name	The internal system-name of the variable to create.
variable.display.name	The user-visible name of the variable to create.
variable.expression	The variable expression that defines the variable.

Value

TRUE if the variable definition succeeded. Otherwise, the response from the server (including error messages) is returned.

Author(s)

David Barker <support@causata.com>

See Also

CausataConfig.

Examples

```
# The settings below are not for an actual server,
# they are for illustration purposes only.
causata.config <- CausataConfig("server.causata.com","8002",
  "causatauser","Bg20qydd6*")

# the command below is commented out since it requires a live server to run
#result <- Config.CreatePrimaryVariable(causata.config,
#  variable.name="online-has-page-view",
#  variable.display.name="Online: Has Page View",
#  variable.expression="INCLUDES page-view")
```

Config.DeleteVariable *Deletes a Causata variable.*

Description

This function attempts to delete the named variable from Causata's configuration.

Usage

```
Config.DeleteVariable(causata.config, variable.name)
```

Arguments

causata.config	An object from CausataConfig.
variable.name	The internal system-name of the variable to create.

Details

This function attempts to delete the named variable from Causata's configuration.

Value

TRUE if the variable deletion succeeded. Otherwise, the response from the server (including error messages) is returned. This may occour if other variables are derived from the variable you're trying to delete, or if it is used in live decision engines or camapigns.

Author(s)

David Barker <support@causata.com>

See Also

CausataConfig.

`Connect` *Connect to Causata and extract data.*

Description

These functions open a connection to a Causata server, extract data, and close the connection.

Usage

```
Connect(sql.server.host, sql.server.port, sql.username, sql.password,
  group=NULL, verbose=FALSE)

## S3 method for class 'Connect'
GetData(conn, query, ...)

## S3 method for class 'Connect'
GetCausataData(conn, query, dependent.variable, ...)

## S3 method for class 'Connect'
GetRawData(conn, query, ...)

## S3 method for class 'CausataData'
GetQuery(this, ...)

## S3 method for class 'Connect'
GetNames(this, kind, ...)

## S3 method for class 'Connect'
Close(conn, ...)
```

Arguments

sql.server.host	The SQL server hostname. Contact your Causata Engagement Manager for the correct value.
sql.server.port	The SQL server port. Contact your Causata Engagement Manager for the correct value.
sql.username	Your Causata username. Contact your Causata Engagement Manager for the correct value.
sql.password	Your Causata password. Contact your Causata Engagement Manager for the correct value.
group	Configuration data from the group provided will be loaded from the configuration file. See LoadCausataConfig. If the same parameter is provided in the configuration file and the function argument then the function argument will take precedence.
verbose	If TRUE then the connection information will be printed to the screen.
conn	A connection object from Connect.
query	A query object from Query.
this	A CausataData object.
kind	The kind of variable name to return, valid values are 'display' and 'system'.
dependent.variable	See CausataData.
...	Unused arguments for other methods.

Details

- Connect opens a SQL connection to a Causata server using functions from the RMySQL package.
- GetData executes the provided query and returns data in a data frame.
- GetCausataData executes the provided query and returns data in a CausataData object.
- GetRawData returns unaltered data from the lower level query commands.
- Close simply closes a database connection.

GetData and GetRawData return data encoded in different formats using the Causata SQL interface. GetData is consistent with Causata data exported into CSV files, so it should generally be used instead of GetRawData.

The default behavior when using SQL queries of Causata within R is that boolean data in Causata is returned as numeric (1=TRUE, 0=FALSE), and text data is returned as characters. The other functions, GetData and GetCausataData, convert boolean data to factors with level names "true" and "false", and return character data as factors. This is the same format as data exported from Causata to CSV files.

GetNames is used to return the system names or display names for Causata variables.

Value

Connect returns an object of class Connect. GetData and GetRawData return data frames, while GetCausataData returns an object of class CausataData. Finally, GetNames returns a character vector of variable names.

Author(s)

David Barker <support@causata.com>

See Also

Connect, Query, CausataData, RMySQL, LoadCausataConfig.

Examples

```
# This command requires a connection to a Causata server
# the example below is for illustration only
## Not run:
conn <- Connect(hostname="example.causata.com", username="exampleuser",
  password="examplepassword")
data <- GetData(conn, Query())
Close(conn)
## End(Not run)
```

---

df.causata *Example data for the Causata package.*

Description

An example data set containing a non-random sample of anonymized Causata records.

Usage

```
data(df.causata)
```

Details

The data frame contains 16904 observations on 150 variables. The query used to generate this data has a focal point of a decision event where a mobile ad banner was selected for a web page. The dependent variable measures whether a user responds (clicks) on the banner within the hour following the decision.

The column has.responded.mobile.logoff_next.hour_466 is used as a dependent variable.

Examples

```
data(df.causata)
```

`Discretize.CausataData` *Discretizes a continous variable in a CausataData object.*

Description

Discretize a continuous variable in a `CausataData` object, and record the process so that it can be reapplied during scoring.

Usage

```
## S3 method for class 'CausataData'
Discretize(this, variableName, breaks, discrete.values, verbose=FALSE, ...)
```

Arguments

`this`	An object from `CausataData`.
`variableName`	The name of the numeric `CausataVariable` to discretize.
`breaks`	A numeric vector of two or more cut points. This is used by `cut` to discretize the variable. See *Details* below for more information.
`discrete.values`	A numeric vector of discrete values that the continuous values will replaced with. See *Details* below for more information.
`verbose`	If `TRUE` then binning information is printed to the console.
`...`	Unused arguments for other methods.

Details

This function uses `cut` to discretize the variable; it is called with `include.lowest=TRUE` and `right=TRUE`. If *N* discrete bins are desired then `breaks` should have *N+1* values for cut points.

Missing values are permitted, they will be mapped to a separate bin during discretization. This arrangement has three important conditions:

First, missing values must *not* be replaced (as in `CleanNaFromContinous`). Executing `Discretize` on a variable that was treated with `CleanNaFromContinous` will generate an error.

Second, `ReplaceOutliers` must be executed *before* `Discretize`, and the upper limit must be less than or equal to the last `breaks` value. Missing values are mapped to an artificial bin that is greater than the last value of `breaks`. Using `ReplaceOutliers` ensures that outliers are mapped to the existing values and not the missing values.

Third, if missing values are present in the variable and there are *N* bins, then *N+1* `discrete.values` are required. By convention missing values are mapped to the last value of `discrete.values`.

Value

Returns a `CausataData` object.

Author(s)

Justin Hemann <support@causata.com>.

See Also

CausataData, CausataVariable, cut, CleanNaFromContinuous, ReplaceOutliers.

Examples

```
# create a random variable and a dependent variable
set.seed(1234)
ivn <- rnorm(1e5) # random data, normally distributed, no missing values
ivm <- ivn  # create a copy, but replace the first 100 values with NA (missing)
ivm[1:100] <- NA
dvn <- rep(0, 1e5)
dvn[(ivn + rnorm(1e5, sd=0.5))>0] <- 1
causataData <- CausataData(data.frame(ivn__AP=ivn, ivm__AP=ivm), dependent.variable=dvn)

# plot data before discretization
hist(causataData$df$ivn__AP, main="Before discretization.", col="gray")

# the replace outliers step is required
causataData <- ReplaceOutliers(causataData, 'ivn__AP',
  lowerLimit=min(causataData$df$ivn__AP),
  upperLimit=max(causataData$df$ivn__AP))

# discretize with deciles, 1st decile is mapped to 1, 2nd to 2, etc.
breaks <- quantile(ivn, probs=seq(0,1,0.1))
causataData <- Discretize(causataData, 'ivn__AP', breaks, 1:10, verbose=TRUE)

# plot data after discretization
hist(causataData$df$ivn__AP, main="After discretization.", col="gray", breaks=seq(0.5,10.5,1))

# Discretize data where missing values are present.
# One extra value is required for discrete.values, map missing to 0.
# By convention missing values are mapped to the last element in discrete.values
causataData <- ReplaceOutliers(causataData, 'ivm__AP',
  lowerLimit=min(causataData$df$ivm__AP, na.rm=TRUE),
  upperLimit=max(causataData$df$ivm__AP, na.rm=TRUE))
causataData <- Discretize(causataData, 'ivm__AP', breaks, c(1:10,0), verbose=TRUE)
```

FocalPointQuery *Build a focal point query.*

Description

The FocalPointQuery class is used to generate SQL queries for Causata. The data in a focal point query is built around a particular event in a customer profile.

Usage

```
FocalPointQuery(focalpoint.event,
  cardinality=if (length(event.attribute)) {"using.all.values"} else {"using.all.events"},
   event.attribute=NULL)
```

Arguments

`focalpoint.event`	The event to be used as a focal point.
`cardinality`	This controls which records are returned when there are multiple events in a profile. The default settings will return one or more records for a profile. See the Details section below.
`event.attribute`	If supplied, the focal point will be at the time given on this event attribute of the given event.

Details

The FocalPointQuery object builds a focal point query for profile records. A blank FocalPointQuery translates to the following:

```
"SELECT * FROM Scenarios variable, `focalpoint.event` WHERE variable.focal_point = `focalpoint.event`
```

This focal point query can be made more specific by adding variables with `WithVariables`, adding where clauses with `Where` or setting a row limit with `Limit`. The SQL query can be generated with as.character, e.g. `as.character(FocalPointQuery("focalpoint.event"))`.

When a customer event stream is loaded, there may be multple events that match the focal point event and where clauses. In this case one or more profile records (rows) may be returned for a single customer. The cardinality argument defines how which records are output.

When there is no event.attribute specified, the legal values for this argument and their meanings are:

- `"using.all.events"` For each occurance of the focal point event (that match the where clauses) there will be a customer record with the timestamp of that event as the focal point.
- `"using.oldest.event"` When multiple events match the focal point query, the event with the smallest (oldest) timestamp is used to build the customer record.
- `"using.newest.event"` When multiple events match the focal point query, the event with the largest (newest) timestamp is used to build the customer record.

When an event.attribute is specified, then the legal values for this argument and thier meanings are:

- `"using.all.values"` All values for the attribute on all match focal point events are used to build customer records, with the value of those attributes as the focal point point for the record.
- `"using.earliest.value"` The value of the attribute on the event with the smallest (earliest) timestamp is used as the focal point time.
- `"using.most.recent.value"` The value of the attribute on the event with the largest (most recent) timestamp is used as the focal point time.

Value

A FocalPointQuery object based on the supplied event (and optional event attribute).

Author(s)

David Barker (support@causata.com)

See Also

FocalPointQuery, Connect, WithVariables, Variables, Where, Limit, CausataData.

Examples

```
# This example builds a query returning a profile at each purchase event
# with a price greater than $30
query <- FocalPointQuery("purchase") + WithVariables(c("some", "variables")) +
  Where("purchase-price$Same Session", GreaterThan(30)) + Limit(1000)
as.character(query)
```

GetStratifiedSample *Gets a stratified sample of data from Causata*

Description

Extracts a stratified sample of data

Usage

```
GetStratifiedSample(connect, query, stratification.variable,
                    stratification.variable.name,
                    stratification.value=0)
```

Arguments

connect	Causata connect object - used to resample at the stratified sampling rates.
query	Causata query object - used to resample at the stratified sampling rates. Note that the Limit must be defined.
stratification.variable	A vector of values on which to base the stratification.
stratification.variable.name	The name of the Causata variable that is used as the basis of stratification.
stratification.value	Value of the stratification.variable which will determine the stratum for a record.

Details

This function gets a stratified sample of data from Causata. The population will be split into two strata based on whether the stratification.variable value for a record matches the stratification.value. Sampling rates for the two strata are then calculated where the rate for the larger strata, strata.A is:

```
sample.rate.A = sqrt((# records in strata.B) / (# records in strata.A))
```

New queries are run to resample the Causata data at these sample rates.

Value

Returns a list with two elements as follows:

df	A dataframe of sampled data containing all of the variables found in query.
weights	A vector of weight values. The weights are the inverse of the probability of selecting a record in the sample.

Author(s)

Suzanne Weller <support@causata.com>

See Also

Connect, Query, Limit.

Examples

```
# create some variables to query for
variables <- c('customer-id', 'total-spend')

# create a stratified sample given an initial query
# The commands below are commented out since they require an actual server connection
#connection <- Connect(hostname="server.causata.com",
#  username="user@gmail.com", password="enw8Q!mN")
#query <- Query() + Limit(500)
#df <- GetData(connection, query)

# The commands below are commented out since they require an actual server connection
#sampled.data <- GetStratifiedSample(connection, query,
#  df[['has.purchase__Next.30.Days']], 'has.purchase__Next.30.Days', "true")
#table(sampled.data$weights)
```

GetTransforms *GetTransforms*

Description

Returns a function that will re-apply the transformations that have been applied to a CausataData obejct to another data frame.

Usage

```
## S3 method for class 'CausataData'
GetTransforms(this, ...)
```

Arguments

this	A CausataData object.
...	Unused extra arguments.

Details

As transformations are applied to the CausataData object, they are recorded. This function returns a function that will apply these transformations, in order, as they were applied to the data frame in the CausataData object. This can be used to validate that transformations work as expected on new data, and are used in model validation.

Value

A function that accepts a data.frame as an argument and returns a transformed data.frame.

Author(s)

David Barker (support@causata.com)

See Also

CausataData.

Examples

```
# Create a data frame with a factor that has 5 levels.
df <- data.frame(var__AP=c("a", "a", "a",  "b", "b",  "c", "d", "e", NA))
caustaData <- CausataData(df, rep(0,nrow(df)))

# Merge the smaller levels so the factor has 3 levels.
# The remaining levels will be a, b, and Other.
caustaData <- MergeLevels(caustaData, max.levels=3)

# Get a function that will re-apply any transformations in caustaData.
transformer <- GetTransforms(caustaData)

# Now, create a new data.frame and apply the same transformation to it.
# Any levels in the factor that are not "a", or "b" will be replaced
# with "Other"
new.df <- data.frame(var__AP=c("a", "b", "c", "c", "c", "d", "a", NA))
transformed.df <- transformer(new.df)
transformed.df$var__AP
```

GetVariable.CausataData

Get the CausataVariable for the named variable

Description

Returns the CausataVariable object for the given column in a CausataData object. The name passed in must match a column name in the `CausataData$df` data frame.

Usage

```
## S3 method for class 'CausataData'
GetVariable(this, r.name=NULL, ...)
```

Arguments

this	A CausataData object.
r.name	A column name in the CausataData$df data frame.
...	Unused extra arguments.

Value

The CausataVariable object for the given column, or NULL if there is no such column.

Author(s)

David Barker <support@causata.com>

See Also

CausataData, CausataVariable, CausataToRNames, RToCausataNames.

GrepLoop *Searches for a list of patterns within a list of strings.*

Description

Given a vector of patterns and a vector of strings, this searches for the patterns within the strings and returns the matching locations.

Usage

```
GrepLoop(patternVec, x, ignore.case=TRUE, boolean=FALSE)
```

Arguments

patternVec	A vector of pattern strings to search for.
x	A vector of strings to search through using grep.
ignore.case	Indicates if the matches are case-sensitive.
boolean	Controls whether an index of integers or booleans is returned.

Details

This function applies grep multiple times.

Value

A vector of indices indicating which elements in x match any of the patterns in patternVec.

Author(s)

Justin Hemann <support@causata.com>

See Also

grep.

Examples

```
pats <- c("gray", "grey")
x <- c("dark gray", "yellow", "light grey", "red")
# The first and third elements in x match the patterns
GrepLoop(pats, x)
```

LoadCausataConfig *Load passwords and configuration data.*

Description

Loads Causata configuration data, including usernames and passwords, from a configuration file. This way scripts can be shared among users without compromising passwords.

Usage

```
LoadCausataConfig(group)
```

Arguments

group	A text string indicating a group name. In a configuration file, arguments under the will be returned in a list.

Details

Configuration data can be stored in a local file located in the user directory. The file must be named .causata-config.yaml, and it must be located in the user's home directory. This function will attempt to load the file using path.expand:

```
path.expand("~/.causata-config.yaml")
```

Two functions in the Causata package will call LoadCausataConfig if a group name is provided: CausataConfig, and Connect. The parameter names provided in the config file will be mapped to the input parameters of these functions.

The format of the configuration file is simple. There are one or more groups with a set of indented parameters below. The group and parameter names are followed by a colon.

If the user has a configuration file as shown in the example below, then calling this function returns a list as follows:

Example .causata-config.yaml file

```
# Use hash for comments
example.1:
  sql.server.host : 123.456.789.219
  sql.server.port : 33060
  config.server.host : 123.456.789.100
  config.server.port : 8003
  username : example@causata.com
  password : ni83jfH

example.2:
  sql.server.host : 123.456.789.999
  sql.server.port : 33060
  config.server.host : 123.456.789.500
  config.server.port : 8003
  username : example@causata.com
  password : 972hfgHB
```

If "example.1" is used then the first set of parameters is returned in a list:

```
  LoadCausataConfig("example.1")
  $sql.server.host
  [1] "123.456.789.219"

  $sql.server.port
  [1] 33060

  $config.server.host
  [1] "123.456.789.100"

  $config.server.port
  [1] 8003

  $username
  [1] "example@causata.com"

  $password
  [1] "ni83jfH"
```

Similarly, if the "example.2" group is supplied then the second set of parameters is returned in a list:

```
  LoadCausataConfig("example.2")
  $sql.server.host
  [1] "123.456.789.999"
```

```
$sql.server.port
[1] 33060

$config.server.host
[1] "123.456.789.500"

$config.server.port
[1] 8003

$username
[1] "example@causata.com"

$password
[1] "972hfgHB"
```

Value

A list of parameters for the group is returned.

Author(s)

Justin Hemann <support@causata.com>

References

http://www.yaml.org/

See Also

path.expand, yaml.load_file, CausataConfig, Connect.

`MergeLevels` *Combines least-frequently occurring levels of a factor into an "Other" category.*

Description

Take a nominal variable and merge the least-frequently occurring levels into an Other category, to leave only max.levels distinct categories (including Other). For example, if there are 15 levels in the data and we request max.levels = 10, then the leading 9 levels will be retained, and the least frequent 6 levels will be merged into Other.

Usage

```
## S3 method for class 'factor'
MergeLevels(this, max.levels, other.name="Other", ...)
```

Arguments

this	A a factor, ie a nominal variable.
max.levels	The maximum number of levels required. eg If we request 10 levels, then there will be 9 distinct levels, plus Other. max.levels must be at least 2. If max.levels is greater than the number of levels in the data then no merging is done.
other.name	The merged levels will be assigned to a new level with the name provided.
...	Unused extra arguments.

Value

Returns a new factor with the smaller levels merged.

Author(s)

Jason McFall, Justin Hemann <support@causata.com>

Examples

```
library(stringr)
f <- factor(str_split("a a a b b b c c c d e f g h", " ")[[1]])
# d,e,f,g,h are merged into Other
MergeLevels(f, max.levels=4)
```

MergeLevels.CausataData

Combines least-frequently occurring levels of a factor into an "Other" category.

Description

MergeLevels is applied to a CausataData object, and the merge process is recorded so that it can be repeated during scoring.

Usage

```
## S3 method for class 'CausataData'
MergeLevels(this, variableName=NULL, max.levels,
  other.name="Other", verbose=FALSE, ...)
```

Arguments

this	An object from CausataData.
variableName	If a name is supplied then the variable matching the name will have MergeLevels applied. If no name is provided then MergeLevels is applied to every factor in causataData.
max.levels	See MergeLevels.

other.name	See MergeLevels.
verbose	If TRUE then summary information will be printed to the screen.
...	Unused extra arguments.

Value

Returns an object of class CausataData.

Author(s)

Justin Hemann <support@causata.com>

See Also

CausataData, MergeLevels.

Examples

```
library(stringr)
df <- data.frame(
  f1__AP=factor(str_split("a a a b b b c c c d e f g h", " ")[[1]]),
  f2__AP=factor(c(rep("x",7),rep("y",7))))
causataData <- CausataData(df, rep(0,nrow(df)))
# For the factor f1__AP, the levels d,e,f,g are merged into Other.
# f2__AP is not altered since it has only two levels.
causataData <- MergeLevels(causataData, max.levels=4)
```

ModelDefinition *Define model metadata.*

Description

This function defines model metadata for models that will be uploaded to Causata.

Usage

```
## S3 method for class 'cv.glmnet'
ModelDefinition(model, causata.data, formula,
  lambda = model$lambda.1se, ...)
```

Arguments

model	A model object from cv.glmnet.
causata.data	Model training data from CausataData.
formula	The formula used to train the model.
lambda	The lambda parameter from glmnet.
...	Extra unused arguments.

Details

The package vignette illustrates how the ModelDefinition function is used.

Value

An object of class `ModelDefinition` is returned is returned by the ModelDefinition function.

Author(s)

David Barker <support@causata.com>

See Also

`UploadModel`, `cv.glmnet`, `CausataData`, `formula`, `is`.

`predict.GlmnetModelDefinition`
Generate predictions for a glmnet model.

Description

Generate predictions for a glmnet model.

Usage

```
## S3 method for class 'GlmnetModelDefinition'
predict(object, data, verbose = FALSE, ...)
```

Arguments

`object`	An object from `ModelDefinition`.
`data`	A dataframe that will be used to generate predictions.
`verbose`	If TRUE then prediction information will be printed to the console.
`...`	Extra unused arguments.

Details

This function generates predictions using the data provided. If columns in the model matrix are missing then columns of zeros will be inserted and a warning will be generated. Typically transformations are applied using `GetTransforms` before `predict` is applied.

Value

A list is returned with elements as follows:

model.matrix	The model matrix used to calculate predicted values.
predicted	Predicted values.
lambda	The lambda value used by glmnet.
missing.cols	Column names that are missing from the prediction matrix.

Author(s)

Justin Hemann <support@causata.com>

See Also

ModelDefinition, glmnet, model.matrix, glmnet, GetTransforms.

PredictivePower *Predictive power for a single variable.*

Description

This function computes predictive power for a single independent variable and a binary dependent variable.

Usage

```
## S3 method for class 'factor'
PredictivePower(iv, dv, warn.levels=30, cv=NULL, debug=FALSE, ...)

## S3 method for class 'numeric'
PredictivePower(iv, dv, warn.levels=30, cv=NULL, debug=FALSE, ...)

PredictivePowerCv(iv, dv, warn.levels=30, debug=FALSE, folds=10, ...)
```

Arguments

iv	The independent variable.
dv	The dependent variable, which may have only two unique values.
warn.levels	If the number of levels in iv exceeds this value then a warning will be issued.
debug	If set to TRUE then debugging information is printed to the screen.
cv	If NULL then all data are used to compute the predictive power. If an index of boolean values is provided then they are used to separate the data into two parts for cross validation. See the Details below for more information.
...	Additional arguments are passed to BinaryCut.

`folds` This argument is used to specify the folds used for cross validation. If a number between 2 and 10 is provided then data will be assigned to the selected number of folds at random. If a vector of values is provided then it will be used as an index to assign data to folds. The number of unique values must be between 2 to 10, and the vector length must match `iv`.

Details

Predictive power is defined as the area under the gains chart for the provided independent variable divided by the area under the gains chart for a perfect predictor. A random predictor would have a predictive power value of 0, and a perfect predictor would have a value of 1.

The power calculation is derived from a discretized gains chart. As such it only works with categorical variables. Numeric variables are discretized before power is computed. The `PredictivePower.numeric` function discretizes continuous data using the `BinaryCut` function. Note that the predictive power will depend, in part, on the discretization method.

By default the second level of `dv` is used as the "positive" class during power calculations. This can be controlled by ordering the levels in a factor supplied as `dv`.

Missing values in `iv` are allowed in `PredictivePower.factor` – they are ignored during the calculations, as are the corresponding dependent variable values. The missing values can be used in the power calculations if the missing values are mapped to a non-missing level in the factor. See `CleanNaFromFactor`. Missing values are not allowed in `dv`.

Cross validation is executed using the `PredictivePowerCv` function as a wrapper for the `PredictivePower` functions. When constructing the gains chart the bins are ordered by the odds for a "positve" within each bin. During cross validation the ordering is derived from one set of data, and the area under the curve is calculated with the other set.

Value

The `PredictivePower` functions returns a numeric value representing the predictive power, between 0 and 1.

`PredictivePowerCv` returns a list as follows:

`predictive.power` An array of predictive power values, one for each fold of cross validation.

`mean` The mean predictive power value.

`sd` The standard deviation of predictive power values.

`robustness` A measure of stability defined as `1-sd/mean`. Values will be between zero (unstable) and 1 (stable).

Author(s)

Justin Hemann <support@causata.com>

References

Inspired by Miller, H. (2009) *Predicting customer behaviour: The University of Melbourne's KDD Cup report*.

See Also

CleanNaFromFactor, BinaryCut.

Examples

```
library(stringr)

# Power is 1/3 where levels differ by 1/3, missing values in iv are ignored.
PredictivePower(factor(c(str_split("a a a b b b", " ")[[1]], NA,NA)),
              c(                   1,1,0,0,0,1,              1, 1 ) )

# Power is 1.0 for perfect predictor
PredictivePower(factor(c(str_split("a a a a a b b b b b", " "))[[1]]),
                factor(c(str_split("1 1 1 1 1 0 0 0 0 0", " "))[[1]]) )

# Power is 0 for random predictor
PredictivePower(factor(c(str_split("a a a a b b b b", " "))[[1]]),
                factor(c(str_split("1 1 0 0 1 1 0 0", " "))[[1]]) )

# compute power for random data, power and robustness should be low
set.seed(1234)
fl <- as.factor(sample(letters, size=1e5, replace=TRUE))
dv <- sample(c(0,1), size=1e5, replace=TRUE)
PredictivePowerCv(fl,dv)

# compute power for numeric data, send nbins arguments to BinaryCut
ivn <- rnorm(1e5)
dvn <- rep(0, 1e5)
dvn[(ivn + rnorm(1e5, sd=0.5))>0] <- 1
PredictivePower(ivn,dvn, nbins=10)
```

`Query` *Build queries to extract data from Causata.*

Description

The Query class is used to generate SQL queries for Causata. The queries are built with the helper objects `WithVariables`, `Where` and `Limit`. SQL is generated when `as.character()` is invoked on the query object.

Usage

```
Query()

## S3 method for class 'Query'
Limit(this, ...)

## S3 replacement method for class 'Query'
Limit(this) <- value
```

```
## S3 method for class 'Query'
Variables(this, ...)

## S3 replacement method for class 'Query'
Variables(this) <- value


WithVariables(...)
```

Arguments

this	A query object.
value	For Query this is a number indicating the maximum number of records to return. For Variables this is one or more variable names in a list.
...	For WithVariables this is a single variable, or a list of variables. For Query and Variables this is unused extra arguments.

Details

The Query object builds a query for customer data, a blank Query corresponds to

```
SELECT * FROM Customers variable
```

This query can be made more specific by adding variables with WithVariables, adding where clauses with Where or setting a row limit with Limit. The actual SQL query can be generated with as.character, e.g. as.character(Query()).

The variables and limit can be retrieved and modified with Variables(query) and Limit(query) respectively.

Value

Query returns a blank Query object.

Author(s)

David Barker, Justin Hemann (support@causata.com)

See Also

FocalPointQuery, Connect, Variables, Where, CausataData, is.

Examples

```
q <- Query()
q <- q + WithVariables(c("var1", "var2"))
q <- q + Where("variable-one", GreaterThan(30))
Variables(q) # returns c("var1", "var2")
Variables(q) <- c("var2", "var3") # set the variables for this query
Limit(q) # since the limit has not been set this returns NULL
Limit(q) <- 1000 # Sets the limit to 1000
```

```
as.character(q)

q <- Query() + WithVariables("variable-one", "variable-two") +
  Where("variable-one", GreaterThan(5))
# The example below is commented out since it requires a server connection.
# With a connection this would retrieve data and return it in a dataframe df.
## Not run:
conn <- Connect(hostname, port, username, password)
data <- GetData(conn, q)

## End(Not run)
```

ReadCausataCsv *Loads data from a Causata CSV file.*

Description

Loads data exported from a Causata CSV file into a data frame. Metadata from Causata is used to set variable names and classes. The function arguments allow for selective filtering of rows and / or columns.

Usage

```
ReadCausataCsv(causataR, include=c(), exclude=c(), maxMb=1000,
  colFilterFunc=NA, rowIndex=NA, nrows=NA, metadata=FALSE,
  debug=FALSE, ...)
```

Arguments

causataR	An output list from the ReadCausataR function.
include	A list of variable names or patterns to match against the variables in the CSV data. Matches are kept. See 'Details' for more information.
exclude	A list of variable names or patterns to match against the variables in the CSV data. Matches are excluded. See 'Details' for more information.
maxMb	Specifies the maximum megabytes of data to load in one pass, which is computed before rows and columns are filtered out. This constraint is applied only if nrows is specified. See 'Details' for more information.
colFilterFunc	An optional function that is applied to each column of data. The function must take the independent variable as its first argument, and it must return a logical (TRUE/FALSE) value OR a list including an element named keep. If the value is TRUE then the variable is kept, if FALSE the variable is discarded.
rowIndex	An optional vector of logical values where TRUE indicates which rows should be kept.
nrows	The maximum number of rows to read from the csv file. This is applied before rows are filtered.

metadata	If FALSE then a data frame is returned. If TRUE then a list of outputs is returned.
debug	If TRUE the column filter is applied with a for loop instead of doMC, which is easier to debug.
...	Extra arguments are sent to the colFilterFunc.

Details

CSV data from Causata is read into a data frame. The arguments allow filtering by column names, row index, or filtering by column calculations when a function is provided.

The include and exclude arguments are used to select which columns to load from the csv file. If these arguments are left at their default values then all columns are loaded. If include and exclude are set then exclude is applied first, followed by include.

The maxMb parameter can be used to load and filter data in several passes, which would reduce the total memory required if row / column filters are specified in colFilterFunc or rowIndex. If the estimated required memory exceeds maxMb, then the load will be broken into multiple passes, each no larger than maxMb. The default estimate is 12 bytes per cell of a data frame, so when MaxMb=1000 (about a gigabyte) that corresponds to a data frame with 100k rows and 833 columns.

Value

A data frame of CSV data, or a list containing the data frame and metadata as follows:

df	A data frame of CSV data.
metadata	A list of outputs returned from the colFilterFunc.

Author(s)

Justin Hemann <support@causata.com>

ReadCausataR *Parses an R file exported with Causata data.*

Description

Parses an R file exported with Causata data. The information in the R file can be used to filter or select variables.

Usage

```
ReadCausataR(rFile, countRows=FALSE)
```

Arguments

rFile	The R file to process, including path.
countRows	If TRUE the rows in the CSV file will be counted. This step can reduce memory requirements.

Details

This function parses the R code exported with Causata data. File names, column names, and column classes are extracted.

Value

A list is returned with elements as follows:

`fileR`	A filename of the R file, including the path.
`fileData`	A filename of the CSV data file, including the path.
`colClasses`	The classes of each column, e.g. numeric, factor, etc.
`col.names`	The column names of each column.
`nrows`	The number of rows in the CSV data. If `countRows` is FALSE then the value is set to `-1`.

Author(s)

Justin Hemann <support@causata.com>

`ReplaceOutliers` *Replaces outliers in a continuous variable.*

Description

Given a vector of integer or numeric values, outliers exceeding user-defined limits are replaced.

Usage

```
## S3 method for class 'numeric'
ReplaceOutliers(this, lowerLimit=NULL, upperLimit=NULL, ...)
```

Arguments

`this`	An array of numeric values. Missing values will be ignored and retained.
`lowerLimit`	If a value is provided then values less than this in `this` will be replaced with the value of `lowerLimit`.
`upperLimit`	If a value is provided then values greater than this in `this` will be replaced with the value of `upperLimit`.
`...`	Extra unused arguments.

Details

A new array is returned with outliers replaced with limit values. Any value that is less than `lowerLimit` or greater than `upperLimit` is considered an outlier. Missing values are ignored.

Value

An array with outliers replaced.

Author(s)

Justin Hemann <support@causata.com>

Examples

```
ReplaceOutliers(c(-1000, 1, 2, 3, NA, 1000), lowerLimit=1, upperLimit=3)
```

ReplaceOutliers.CausataData

Replace outliers in a CausataData object.

Description

Outliers are replaced in a CausataData object, and the limits are stored so that they can be re-applied when scoring.

Usage

```
## S3 method for class 'CausataData'
ReplaceOutliers(this, variableName,
  lowerLimit=NULL, upperLimit=NULL, ...)
```

Arguments

this	A CausataData object.
variableName	The name of the variable within the causataData object that will have outliers replaced.
lowerLimit	See ReplaceOutliers.
upperLimit	See ReplaceOutliers.
...	Extra unused arguments.

Value

Returns a CausataData object.

Author(s)

Justin Hemann <support@causata.com>

See Also

CausataData, CausataVariable, ReplaceOutliers.

Examples

```
df <- data.frame(variable1__All.Past=c(1,2,3,4,1000))
# create CausataData object
causataData <- CausataData(df, rep(0,nrow(df)))
# max is 1000 before outliers are replaced
max(causataData$df$variable1__All.Past)
causataData <- ReplaceOutliers(causataData, 'variable1__All.Past', upperLimit=4)
# now max is 4 after outliers are replaced
max(causataData$df$variable1__All.Past)
```

`RToCausataNames` *Converts R-friendly causata column names to the corresponding Causata system name*

Description

Converts R-friendly causata column names to the corresponding Causata system name

Usage

```
RToCausataNames(name.vector)
```

Arguments

`name.vector`	A character vector of column names.

Details

Causata variables follow two naming conventions. The first is found in data exported from within Causata using the "R Formated CSV" option:

`variable.name_Time.Domain_id` where id is a number, e.g. `variable.name_Time.Domain_123`

The second convention is found in data exported from the SQL interface:

`variable.name__Time.Domain`

Example conversions:

`variable.name__Time.Domain` becomes `variable-name$Time Domain`

`variable.name_Time.Domain_123` is unchanged.

Variables that do not conform to these conventions will be mapped to `"No Causata Name"` and a warning will be triggered.

Value

An character vector of mapped variable names.

Author(s)

David Barker <support@causata.com>

See Also

CausataData, CausataVariable, CausataToRNames.

Examples

```
RToCausataNames(c("variable.name__Time.Domain", "variable.name_Time.Domain_123"))
RToCausataNames("bad-name-doesn't fit convention")
```

SampleStratified *Draws a random, stratified sample from a vector of indices.*

Description

Given a vector of logical values, this returns an index where TRUE values are kept and FALSE values are sampled.

Usage

```
SampleStratified(idxTrue, scale=1, verbose=TRUE)
```

Arguments

idxTrue	An array of logical TRUE / FALSE values. All TRUE values are kept (their index is always TRUE), and FALSE values are sampled (their index may be TRUE or FALSE).
scale	Controls the sampling rate for FALSE values. See the Details section below for more information.
verbose	If TRUE then summary information is printed to the screen.

Details

All TRUE values from the input index are kept. The number of FALSE values that are kept is computed as follows:

$$sampleRate = \sqrt{\frac{nFalse}{nTrue}} \frac{1}{scale}$$

$$numKeep = round(\frac{nFalse}{sampleRate})$$

Here nFalse and nTrue are the number of FALSE and TRUE values provided in the array idxTrue. Note that if sampleRate is less than 1 then then no sampling is performed – all FALSE values are kept. Values of scale greater than 1 result in more FALSE values being kept; values below 1 result in fewer.

Value

An array of logical values indicating which records should be kept.

Author(s)

Justin Hemann <support@causata.com>

Examples

```
data(df.causata)
idx <- SampleStratified(df.causata$has.responded.mobile.logoff_next.hour_466=="true")
table(df.causata$has.responded.mobile.logoff_next.hour_466, idx)
```

ShortenStrings *Shortens strings by replacing the middle with a separator.*

Description

Strings are shortened by replacing characters from the middle of the string with a separator.

Usage

```
ShortenStrings(strings, max.len=40, end.len=floor(max.len/2), sep='...')
```

Arguments

strings	A string of characters, or an array of strings.
max.len	The maximum length of the string. Must be at least 1+str_length(sep).
end.len	The length of the string retained after the separator.
sep	A separator that will be used to replace the middle of the string.

Details

If the input string is longer than max.len, then the string is shortened as follows. First, leading and trailing whitespace is removed. If the string is still longer than max.len then a start length is computed as:

start.len = max.len - end.len - str_length(sep)

Characters after start.len and before end.len are replaced with sep. If end.len is too large (as in the third example below) then it is silently reset to the largest allowable value given sep and max.len.

Value

A single string or an array of strings with the same length as strings.

Author(s)

Justin Hemann <support@causata.com>

See Also

abbreviate.

Examples

```
# only leading / trailing whitespace is removed
ShortenStrings(' abcdefghijklmnopqrstuvwxyz ', max.len=26)

# the middle is replaced with ...
ShortenStrings(' abcdefghijklmnopqrstuvwxyz ', max.len=20)

# the beginning is replaced with ...
# note that end.len is too long, it is silently set to 17 here.
ShortenStrings(' abcdefghijklmnopqrstuvwxyz ', max.len=20, end.len=20)

# the end is replaced with ...
ShortenStrings(' abcdefghijklmnopqrstuvwxyz ', max.len=20, end.len=0)
```

ToPmml *Generates a PMML representation of a model.*

Description

Encodes a glmnet model in a string of PMML text for importing into Causata.

Usage

```
## S3 method for class 'GlmnetModelDefinition'
ToPmml(model.definition, variable.definition, verbose=FALSE, ...)
```

Arguments

model.definition	An object from GlmnetModelDefinition.
variable.definition	An object from VariableDefinition.
verbose	If TRUE then translation information is printed to the console.
...	Extra unused arguments.

Details

The PMML text string can be written to a file using the saveXML function from the XML package.

Value

Returns an object with classes from the XML package. The classes are XMLInternalElementNode, XMLInternalNode, and XMLAbstractNode.

Use print with this object to return a string of PMML text, and saveXML to write the PMML to a file.

Author(s)

David Barker, Justin Hemann <support@causata.com>

See Also

UploadModel, ModelDefinition, VariableDefinition, saveXML.

UploadModel *Loads a model definition into Causata for scoring.*

Description

Three different sets of configuration information are combined to upload a model to Causata for scoring.

Usage

```
UploadModel(causata.config, model.definition, variable.definition, verbose=FALSE)

UploadModelWithValidation(causata.config, model.definition, variable.definition,
  connection, query.function, record.error.max, verbose=FALSE, ...)
```

Arguments

causata.config	An object from CausataConfig.
model.definition	An object from ModelDefinition.
variable.definition	An object from VariableDefinition.
verbose	If TRUE then information is printed to the console.
connection	An object from Connect.
query.function	A function that returns a query string or Query object. The first argument to this function must accept a character string representing a variable name that will be added to the query. See the Details section below for more information.
record.error.max	The absolute value of the largest acceptable error.
...	Extra arguments are passed to the query.function.

Details

UploadModel translates a model into PMML and uploads it to Causata, where it will become available as a new variable.

UploadModelWithValidation adds validation to the upload process. The process works as follows:

1. The model is uploaded to a random variable name.
2. A new query is executed using the provided `query.function`. The new query will include the variables originally used to train the model, and the new model variable from Causata. The R scoring process is re-applied to the new data, and the results from R and Causata are compared. The validation is deemed successful if the difference in results is below the value provided in `record.error.max`.

   If the validation was successful then the model is re-uploaded using the variable name provided in `model.definition`. If the validation failed then

There are two important requirements for the query function:

1. The query function must accept a variable name as its first argument – this argument is used to add the score variable to the query.
2. The query function must return a query including all of the variables that were originally used to train the model. The recommended best-practice is to use a function to extract the training data, then re-use the same function for the validation process.

Value

For `UploadModel`, if the upload was successful then a boolean TRUE is returned. If the upload failed then an error message is returned.

`UploadModelWithValidation` returns a list with the following elements:

`result`	A boolean that is TRUE if the validation was successful and FALSE otherwise.
`validation.data`	A dataframe containing data used in the validation process.
`errors`	An array of error values, which are the absolute value of the difference between `prediction` and `actuals`.
`prediction`	The model scores as calculated by R.
`model.matrix`	The model matrix used by R to generate scores.
`actuals`	The model scores as calcualted by Causata.
`problematic.indices`	An array of indices that are TRUE if the error value exceeds the `record.error.max` and FALSE otherwise.

Author(s)

David Barker, Justin Hemann <support@causata.com>

See Also

`CausataConfig`, `ModelDefinition`, `VariableDefinition`, `Connect`, `Query`, `CausataData`.

Examples

```
# An example query function for UploadModelWithValidation
# The focal point query below returns profiles from the most recent
# ad impression where the product name is "Test Product".
query.function <- function(variables, more.variables=c(), limit=100){
  query <- paste(
    "select", BacktickCollapse(c(variables, more.variables)),
    "from Scenarios S,",
    "     `ad-impression` E",
    "where S.profile_id = E.profile_id",
    "  and S.focal_point = E.timestamp",
    "  and is_last(E.timestamp)",
    "and exists",
    "( select *",
    "  from `ad-impression` A",
    "  where A.`product-name` = 'Test Product'",
    ")",
    "Limit", limit)
  return(query)
}
```

`VariableDefinition` *Defines information for creating variables in Causata.*

Description

This function defines variable information, including the name, description, and labels, for creating new variables in Causata.

Usage

```
VariableDefinition(name, display.name = name, description = name,
  labels = list(),
  author = Sys.info()[["user"]],
  timestamp = as.integer(1000 * as.numeric(format(Sys.time(), "%H%M%OS3"))),
  archived = FALSE,
  categorizing.attribute = "",
  output.value.count = -1,
  data.type = "double")
```

Arguments

`name`	The variable system name. Only letters, numbers, and dashes are allowed in the name, e.g. `most-recent-product-viewed`.
`display.name`	The variable display name as it will be shown in Causata, e.g. `Most Recent Product Viewed`.
`description`	A brief description of the variable, which will be displayed in Causata.
`labels`	A list of optional variable labels, used for categorization.

`author` The variable author name.

`timestamp` The timestamp for when the variable was created. The format is milliseconds from the Unix epoch, Jan 1 1970, 00:00 UTC.

`archived` A boolean indicating if this variable is archived or not.

`categorizing.attribute` Name of the Causata attribute used to categorize the output.

`output.value.count` The number of output values.

`data.type` The data type of the output. Allowable values are "double", "float", "long", "integer", or "short".

Details

Consult your Causata documentation for more information about variables in Causata.

Value

An object of class VariableDefinition is returned.

Author(s)

David Barker <support@causata.com>

See Also

`UploadModel`

Examples

```
variable.definition <- VariableDefinition(name="most-recent-product-viewed",
  display.name="Most Recent Product Viewed",
  description="The most recent product viewed online.",
  labels=list("online","products"))
```

Vinclude *Create lists of variables from Causata for queries.*

Description

This collection of functions is used to create lists of variables from Causata. The lists can be inputs to queries using the Causata SQL interface.

Usage

```
## S3 method for class 'Connect'
Vinclude(this, name.patterns=NULL, label.patterns=NULL,
  and=TRUE, ...)

## S3 method for class 'Connect'
Vexclude(this, variable.names=NULL, name.patterns=NULL,
  label.patterns=NULL, and=TRUE, ...)

## S3 method for class 'Connect'
Vtime(this, variable.names, domains, ...)

BacktickCollapse(variable.names)
```

Arguments

this	An object from the class Connect. This object stores the list of available variable names and time domains.
name.patterns	A character vector of variable system name patterns that will be used with grep to find matches.
label.patterns	A character vector of variable label names patterns that will be used with grep to find matches.
and	If multiple filtering arguments are supplied, then this controls whether variables have to match all of the criteria (and=TRUE) or any of the criteria (and=FALSE).
variable.names	Character vector containing Causata variable system names.
domains	Character vector containing Causata variable time domains.
...	Extra unused arguments.

Details

These functions create lists of variables that are filtered according to the given criteria.

Vinclude returns a character array of all variables matching the provided criteria. If the criteria are left at their default values (NULL) then a list of all available variables is returned.

Vexclude works in the same manner as Vinclude, except that variables matching the criteria are excluded. If variable.names is NULL then the matching process begins with all avaible variables. If variable.names contains variable names then the matching process will select from the provided names.

Vtime appends time domains to the variables.

To see a list of the available variables and time domains, open a Connect object and view the embedded data frames of variable metadata. See the example below for details.

Value

Vinclude, Vexclude, and Vtime return a vector of character strings naming variables found in Causata.

BacktickCollapse returns a single character string with the variables names concatenated together and surrounded by backticks. This string can be used directly in a SQL query to Causata.

Author(s)

Justin Hemann <support@causata.com>

See Also

Connect, grep.

Examples

```
# Some of these examples require a Causata connection, so they are not run
## Not run:
conn <- Connect(group="example")

# View available variables
View(conn$variables)

# View available time ranges
View(conn$timeRanges)

# View available time points
View(conn$timePoints)

# Get a list of variables matching the given labels: all online and demographics
variables <- Vinclude(conn, label.patterns=c("online", "demographics"))

# Get a list of all variables except those with "test" in the name
variables <- Vexclude(conn, name.patterns="test")

# build a query string and extract data
query.str <- paste(
  "select", BacktickCollapse(variables),
  "from customers")
df <- GetData(conn, query.str)
Close(conn)

## End(Not run)

# simple example with BacktickCollapse
BacktickCollapse(c("variable-one", "variable-two"))
```

Where *Query WHERE clause and comparison operations*

Description

The Where function creates a WHERE clause that can be added to Query or FocalPointQuery objects.

Usage

```
Where(variable, operation=NULL)
```

Arguments

variable	A string matching the name of a Causata variable, or a string representing a SQL WHERE clause, like "variable <> 123".
operation	An operator of class RawOperator, such as GreaterThan(123) or EqualTo(123)

Details

There are three basic usage patterns. These are all equivalent:

- Where("variable.`variable-name` <> 'string value'")
- Where("variable-name", "<> 'string value'")
- Where("variable-name", NotEqualTo("string value")

The first form is a raw SQL WHERE clause, when using this form, you must correctly reference and escape the variable name, and the value for the operator you choose. The second form is (variable.name, operation). Using this form the variable name will be correctly referenced and escaped. However, the right hand side (the operation) must still be correctly escaped by you. The third form is (variable.name, operation.object). Using this form the variable name and operation are both correctly escaped. The allowable operations in the third form are as follows:

Operation	Arguments
BeginningWith	Single value
EqualTo	Single value
NotEqualTo	Single value
GreaterThan	Single value
GreaterThanOrEqualTo	Single value
LessThan	Single value
LessThanOrEqualTo	Single value
Like	Single value
In	One or more comma-separated values

Value

An object of class Where used for building queries.

Author(s)

David Barker (support@causata.com)

See Also

Query, FocalPointQuery.

Examples

```
q <- FocalPointQuery("page-view") + Where("page-view-count", GreaterThan(10))
q <- FocalPointQuery("page-view") +
     Where("total-spend", LessThanOrEqualTo(100)) +
     Where("total-spend", GreaterThan(1))
```

Woe *Weight of evidence for each level of a factor.*

Description

Computes the weight of evidence for each level of a factor and a dependent variable.

Usage

```
## S3 method for class 'factor'
Woe(iv, dv, maxOdds=10000, civ=NULL, ...)
```

Arguments

iv	A factor, the independent variable. Missing values, if present, are replaced using CleanNaFromFactor.
dv	The dependent variable, which may have only two unique values. Missing values are not allowed.
maxOdds	When the odds are greater than maxOdds or less than 1/maxOdds then the odds are replaced with the threshold value.
civ	If iv is a discretized version of a continuous variable, then the original continuos variable can be provided in this argument so that linearity can be calculated. See the Value section below for more information.
...	Extra unused arguments.

Details

This function computes the log odds (aka weight of evidence) for each level in a factor as follows:

$$woe = \log \frac{nPositive}{nNegative}$$

where nPositive is the number of "positive" values in the dependent variable, and nNegative is the number of "negative" values.

By default the second level of dv is used as the "positive" class during power calculations. This can be controlled by ordering the levels in a factor supplied as dv.

Other metrics returned include the information value and the log density ratio.

Value

A list with the following elements:

`woe.levels`	A vector of WOE values corresponding to each level of the factor `iv`. The values are ordered to match the input factor `iv`.
`woe`	A vector of WOE values with the same length as `iv`. Essentially each factor value is replaced with the associated log odds.
`odds`	A vector of odds values corresponding to each level of the factor `iv`. The values are ordered to match the input factor `iv`.
`bin.count`	A count of data points in each level of the factor `iv`.
`true.count`	A count of "true" dependent variable values in each level of the factor `iv`. The number of "false" values is `bin.count - true.count`.
`log.density.ratio`	A vector of log density ratio values corresponding to each level of the factor `iv`. The values are ordered to match the input factor `iv`.
`information.value`	A vector of information values corresponding to each level of the factor `iv`. The values are ordered to match the input factor `iv`.
`linearity`	A measure of correlation between the log-odds of the dependent variable and the binned values of the continuous independent variable `civ`. This is calculated if the `civ` argument was provided, otherwise it's NA.

Author(s)

Justin Hemann <support@causata.com>

See Also

`CleanNaFromFactor`.

Examples

```
library(stringr)

# create a factor with three levels
# - odds of 1 for a:  1:2 = 2.0
# - odds of 1 for b:  2:1 = 0.5
# - odds of 1 for NA: 1:1 = 1.0
f1  <- factor(c(str_split("a a a b b b", " ")[[1]], NA,NA))
dv1 <- c(                  1,1,0,0,0,1,              1, 0 )
fw1 <- Woe(f1,dv1)
fw1$odds

# discretize a continuous variable into a factor with 10 levels and compute WOE,
data(df.causata)
dv <- df.causata$has.responded.mobile.logoff_next.hour_466
f2 <- BinaryCut(df.causata$online.average.authentications.per.month_all.past_406, dv)
fw2 <- Woe(f2, dv, civ=df.causata$online.average.authentications.per.month_all.past_406)
fw2$odds
fw2$linearity
```

Index