Package 'lmSupport'

January 27, 2015

Type Package

Title Support for Linear Models

Version 2.9.1

Date 2014-09-26

Author John Curtin <jjcurtin@wisc.edu>

Maintainer John Curtin <jjcurtin@wisc.edu>

Description
Accompanies Markus Brauer, Bas Rokers, and John Curtin's two-course graduate series in General, Generalized, and Multi-level Linear Models (PSY 610/710).

License GPL (>= 2)

Depends car

Imports psych, gvlma, AICcmodavg, lme4, pbkrtest

LazyLoad yes

URL http://dionysus.psych.wisc.edu/

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-26 18:50:31

R topics documented:

`dfMerge` *Merges two data frames*

Description

Merges variables from two data frames (DataX, DataY). By default matches on row names but can use other variable names in DataX (ByX) and DataY (ByY) as needed. By default, includes all cases in DataX and DataY but can limit to only matching (AllX=FALSE, AllY=FALSE) or left join (AllY=FALSE) or right join (AllX=FALSE).

Usage

```
dfMerge(DataX, DataY, ByX = 0, ByY = 0, AllX = TRUE, AllY = TRUE)
```

Arguments

`DataX`	first data frame for merge
`DataY`	second data frame for merge
`ByX`	Name of variable in DataX to match cases on. Column can be specified by name or number. Default is 0 which uses rownames
`ByY`	Name of variable in DataY to match cases on. Column can be specified by name or number. Default is 0 which uses rownames
`AllX`	logical; if TRUE, then extra rows will be added to the output, one for each row in DataX that has no matching row in DataY These rows will have NAs in those columns that are usually filled with values from dY. The default is TRUE, so that all rows with data from both dX and dY are included in the output. In other words, it is the union of these two dataframes
`AllY`	analagous to AllX but for DataY

Details

see merge() for more details

Value

Returns merged data frame

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

merge

Examples

```
dX <- data.frame(v1=c(1,2,3,4,5), v2=c(1,NA,NA,2,4), data=1:5)
rownames(dX) = c(1,2,3,4,5)
dY <- data.frame(v3=c(3,2,1,4,15), v4=c(2,4,5,6,7), data=6:10)
rownames(dY) = c(1,2,3,4,6)
dNew = dfMerge(dX,dY)
```

---

dfReadDat *Opens a tab-delimited dat file with typical Curtin lab settings*

Description

Opens a tab-delimited data file with standard Curtin lab format which include using a header and setting delimiter to tab.

If variable named SubID (default) or other text supplied by SubID variable exists in dat file, row names will be set with this variable and then variable is removed from new data frame.

Usage

```
dfReadDat(File, SubID = "SubID")
```

Arguments

File	File name for .dat file including extension
SubID	String to indicate name of SubID variable. Default is 'SubID'. If set to NULL, rownames will not be altered

Value

returns a data frame

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

read.table(), read.delim(), write.table()

Examples

```
##dfReadDat('Sample.dat)   #not executable unless Sample.dat exists in path
##dfReadDat('Sample.dat, SubID = 'subnum')  #not executable unless Sample.dat exists in path
```

---

dfRemoveCases *Removes cases from dataframe*

Description

Removes cases from dataframe. Cases can be numeric or character. If numeric, rownames must be able to be converted to numeric. Returns warning if cases not found in dataframe.

Usage

```
dfRemoveCases(Data, Cases)
```

Arguments

Data	a dataframe
Cases	a vector of numeric or character case IDs

Value

Returns dataframe with cases removed.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
d = dfRemoveCases(Prestige, c('bookbinders')) #data from car package
d = dfRemoveCases(Davis, c(1,2,4))        #data from car package
```

dfRownames *Sets rownames to SubID*

Description

Sets the row names of the data frame to the variable name listed as SubID. SubID should be text name of variable. Also keeps number of characters constant by default and removes SubID by default

Usage

```
dfRownames(Data, SubID = "SubID", FixedWidth = TRUE, Remove = TRUE)
```

Arguments

Data	a data frame with a variable containing subject ID numbers
SubID	Text name of subject ID variable. Default is SubID
FixedWidth	logical. If TRUE (default), all rowames will be the same length by padding with leading 0's
Remove	logical. If TRUE (default), the subject ID variable will be removed from data frame after setting rownames

Value

Returns data frame with rownames set (and SubID removed if requested)

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
d <- data.frame(SubID = c(1,2,3,10,20), v1=c(1,2,3,4,5), v2=c(1,NA,NA,2,4), data=1:5)
d=dfRownames(d)
```

dfWriteDat *Saves dataframe as tab-delimited text file with typical Curtin lab parameters*

Description

Saves a dataframe as a tab-delimited data file with standard Curtin lab format. Will add rownames as a first column in .dat file and label this column with SubID

Usage

```
dfWriteDat(Data, File, SubID = "SubID")
```

Arguments

Data	a dataframe
File	file name for .dat file
SubID	Name for new column with data from rownames. If NULL, rownames will not be added to .dat file) Default is 'SubID'

Details

Uses these parameters with write.table no append, no quote, separator is tab, no rownames, yes for columnns.

Value

no return value but creates .dat file in current wd

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

read.table(), read.delim(), write.table()

Examples

```
dfWriteDat(Ornstein, File="Test1.dat")
dfWriteDat(Ornstein, File="Test2.dat", SubID = 'ID')
dfWriteDat(Ornstein, File="Test3.dat", SubID = NULL)
```

---

figConfidenceBand *Creates confidence band for regression line*

Description

Adds a confidence band around a regression line in a plot

Usage

```
figConfidenceBand(X, Y, CILo, CIHi, Color)
```

Arguments

X	Vector of data for X to plot
Y	Vector of data for Y to plot
CILo	Vector of data for lower bound of confidence interval
CIHi	Vector of data for upper bound of confidence interval
Color	String to indicate R color. Will be .15 transparent in plot

Value

No value is returned

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

---

figStripChart *Create strip chart on plot*

Description

Adds a strip chart (variant of a rug plot that includes density info) to X (or other) axis on a plot

Usage

```
figStripChart(x, side=1, sshift=0.3, col='gray', pch=15, cex= 0.2, adjoffset=1)
```

Arguments

x	vector of data to plot
side	axis for plot, 1=bottom (default), 2=left, 3= top, 4= right
sshift	scaling parameter for location of plot. Use default
col	color of dots. Default is gray
pch	point type for dots. Default is 15 (small dot)
cex	scaling parameter for size of dots
adjoffset	scaling parameter for dot spacing

Value

No value is returned

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

lmSupport-deprecated *Deprecated Functions in lmSupport Package*

Description

These functions are provided for compatibility with older versions of the **lmSupport** package and may be removed eventually. These functions may not necessarily work as in previous versions of the **lmSupport** package. It is strongly recommended that you update your code to use the new functions.

Usage

```
lm.boxCox(...)
lm.codeRegressors(...)
lm.correctSE(...)
lm.deltaR2(...)
lm.describeData(...)
lm.describeGroups(...)
lm.figSum(...)
lm.mergeData(...)
lm.pointEstimates(...)
lm.readDat(...)
lm.removeCases(...)
lm.renameVar(...)
lm.setContrasts(...)
lm.setRownames(...)
lm.stripChart(...)
lm.sumSquares(...)
lm.writeDat(...)
```

Arguments

...	pass arguments down.

Details

lm.boxCox is now a synonym for the modelBoxCox function. lm.codeRegressors is now a synonym for the varRegressors function. lm.correctSE is now a synonym for the modelCorrectSE function. lm.deltaR2 is now a synonym for the modelCompare function. lm.describeData is now a synonym for the varDescribe function. lm.describeGroups is now a synonym for the varDescribeBy function. lm.figSum is now a synonym for the varPlot function. lm.mergeData is now a synonym for the dfMerge function. lm.pointEstimates is now a synonym for the modelPredictions function. lm.readDat is now a synonym for the dfReadDat function. lm.removeCases is now a synonym for the dfRemoveCases function. lm.renameVar is now a synonym for the varRename function. lm.setContrasts is now a synonym for the varContrasts function. lm.setRownames is now a synonym for the dfRownames function. lm.sumSquares is now a synonym for the modelEffectSizes function. lm.stripChart is now a synonym for the figStripChart function. lm.writeDat is now a synonym for the dfWriteDat function.

`modelAssumptions` *Assess Linear Model Assumptions*

Description

Provides diagnositic graphs and score tests to evaluate linear model assumptions of normality, constant variance and linearity. Follows best practices and uses many functions from car package.

Usage

```
modelAssumptions(Model, Type = "NORMAL", ID=row.names(Model$model), one.page = TRUE)
```

Arguments

`Model`	a linear model produced by `lm`.
`Type`	Type =c('NORMAL', 'CONSTANT', 'LINEAR') for normally distributed residuals with constant variance, and linear (e.g., mean of residuals 0 for all Y')
`ID`	Use to identify points. Default = row.names(model$model). NULL = no identification
`one.page`	logical; display all graphs on one page if TRUE (Default).

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

References

Fox, J. (1991). Regression diagnostics. SAGE Series (79) Quantitative Applictions in the Social Science.

Examples

```
m = lm(interlocks~assets+nation, data=Ornstein)
modelAssumptions(m,'NORMAL')
modelAssumptions(m,'CONSTANT')
modelAssumptions(m,'LINEAR', ID=NULL)
```

`modelBoxCox` *Calculates lambda for Box-Cox power transformation*

Description

Calcluates and plots log-liklihoods lambda for power transformation of response variable. Reports chi-square test of lambda <> 1. All values of Y must > 0 or function will crash. Add offset to Y if necessary (see example). Default lambda range is -2 to 2. Uses boxCox() from car package.

Usage

```
modelBoxCox(Model, Lambdas = seq(-2, 2, by = 0.1))
```

Arguments

`Model`	an unweighted linear model, produced by `lm`.
`Lambdas`	a vector of lambda values to plot. Default is seq(-2,2,by=0.1)

Author(s)

John J. Curtin `<jjcurtin@wisc.edu>`

References

Box, G. E. P. & Cox, D. R. (1964). An analysis of transformations (with discussion). Journal of the Royal Statistical Society, 26, 211-252.

See Also

boxCox(), boxcox()

Examples

```
m = lm(interlocks + 1 ~ assets+nation, data=Ornstein)
modelBoxCox(m)
```

`modelCaseAnalysis` *Provides graphs and/or tests for problematic cases for a linear model*

Description

Provides diagnositic graphs and visual cut points for identification of points that are univaraite outliers, high leverage, regression outliers, and/or influential

Usage

```
modelCaseAnalysis(Model, Type = "RESIDUALS", Term = NULL, ID = row.names(Model$model))
```

Arguments

`Model`	a linear model produced by `lm`.
`Type`	Type = c('RESIDUALS', 'UNIVARIATE', 'HATVALUES', 'COOKSD', 'DFBETAS', 'INFLUENCEPLOT' 'COVRATIO') RESIDUALS (default) = regression outliers, UNIVARIATE = univariate outliers, HATVALUES = leverage, COOKSD = model influence, DFBETAS= individual parameter influence, INFLUENCEPLOT= leverage X influence, COVRATIO = inflation of SEs.
`Term`	Term from model to display. Used only by DFBETAS. DEFAULT is NULL with all terms displayed
`ID`	Use to identify points. Default = row.names(Model$model). NULL = no identification

Value

Side effect of plot is main goal for function. Also returns a list with Rownames and CaseAnalysis Values for cases identified. No list returned if DFBETAS without single term identified.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

References

Fox, J. (1991). Regression diagnostics. SAGE Series (79) Quantitative Applictions in the Social Science.

Examples

```
m = lm(interlocks~assets+nation, data=Ornstein)
Cases = modelCaseAnalysis(m,'RESIDUALS')
Ornstein[Cases$Rownames,]

modelCaseAnalysis(m,'DFBETAS')
modelCaseAnalysis(m,'DFBETAS', 'assets')
```

`modelCompare` *F-tests for nested models*

Description

Calculates F-test to compare two models to determine if ModelA significantly reduces SSE from ModelC. Also reports Partial eta2 and Delta R2 for this model comparison. ModelC should contain subset of ModelA regressors. NOTE: Does not check that model C is subset of model A. User must use caution and verify this themselves.

Usage

```
modelCompare(ModelC, ModelA)
```

Arguments

`ModelC`	a linear model, produced by `lm`. This compact model should include a subset of regressors from ModelA
`ModelA`	a linear model, produced by `lm`. This augmented model should include all regressors from ModelC plus additional regressors.

Details

Calculates F test for model comparison F = ((sseC - sseA)/ (pA - pC)) / (sseA / (N-pA)) ndf = pA - pC ddf = N -P

Value

Returns a list with results for model comparison, sses, and other relevant fields

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
mC = lm(interlocks~assets, data=Ornstein)
mA = lm(interlocks~assets+nation, data=Ornstein)
modelCompare(mC, mA)
```

`modelCorrectSE` *Calculates White (1980)'s heteroscedascity-corrected SEs and Tests for a linear model*

Description

Calculates heteroscedascity-corrected SEs and associated tests for regression coefficients based on method described by White (1980) using hccm() from car package. Prints tables with orginal and corrected results and returns corrected coefficient table

Usage

```
modelCorrectSE(Model, Digits=3)
```

Arguments

`Model`	an unweighted linear model, produced by `lm`.
`Digits`	digits to print in table output. Default =3

Value

Returns the lm coefficients table with corrected SEs and associated tests

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

References

Fox, J. (2008). Applied Regression Analysis and Generalized Linear Models, Second Edition. Sage.

Cribari-Neto, F. (2004). Asymptotic inference under heteroskedasticity of unknown form. Computational Statistics and Data Analysis, 45, 215-233.

Long, J. S. and Ervin, L. H. (2000). Using heteroscedasity consistent standard errors in the linear regression model. The American Statistician, 54, 217-224.

White, H. (1980). A heteroskedastic consistent covariance matrix estimator and a direct test of heteroskedasicity. Econometrica, 48, 817-838.

See Also

hccm() in car package

Examples

```
m = lm(interlocks~assets+nation, data=Ornstein)
modelCorrectSE(m)
```

`modelEffectSizes` *Calculates effect size indices based on Sums of Squares*

Description

Calculates unique SSRs, SSE, SST. Based on these SSs, it calculates partial eta2 and delta R2 for all effects in a linear model object. For categorical variables coded as factors, it calculates these for multi-df effect. Manually code regressors to get 1 df effects Uses car::Anova() with Type 3 error

Usage

```
modelEffectSizes(Model, Print = TRUE, Digits = 4)
```

Arguments

`Model`	a linear model, produced by `lm`
`Print`	Display results to screen. Default = TRUE
`Digits`	Number of digits for printing effect sizes

Value

Returns a list with fields for effect sizes, SSE, and SST.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

Anova()

Examples

```
m = lm(interlocks~assets+nation, data=Ornstein)
modelEffectSizes(m)
```

modelErrors *Returns model errors (residuals) from lm object*

Description

Simple wrapper to return model errors using residuals() function. Implemented simply to match terminology to 610/710 GLM course. Also prints(but does not return) model SSE

Usage

```
modelErrors(Model)
```

Arguments

Model	an lm model obect

Value

Returns vector of model errors (residuals) from sample

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

residuals, lm

Examples

```
m = lm(interlocks~assets+nation, data=Ornstein)
modelErrors(m)
```

modelPredictions *Provides predicted values for sample or new data. New predictions include SEs*

Description

If no data are provided, modelPredictions returns a numeric vector predicted values for the sample, functioning as a simple wrapper for fitted.values(). If a dataframe with new values for Xs are provided, modelPredictions adds prediced values and SEs for these new data to the dataframe using predict() from car package.

Usage

```
modelPredictions(Model, Data=NULL, Label = NULL, Type = 'response')
```

Arguments

Model	a linear model, produced by lm.
Data	a dataframe containg cases for predictions. Must include all regressors from model. Default is NULL with predictions returned for the current sample.
Label	A string label to append to variable names for predicted values, CIs and SE. Default is NULL with no append
Type	'response' or 'link'. Used only for glm objects. see predict()

Value

If Data=NULL, returns a numeric vector of predicted values for sample. If Data are provided, adds four new columns at the front of the dataframe These variables are named Predicted (prediced value), CILo (lower bound of - 1 SE from Predicted), CIHi (upper bound of + 1 SE), and SE (Standard error of predicted value). If Label is not NULL, than Label is appended to end of these four variable names.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

predict(), fitted.values()

Examples

```
##make plot of predicted values with 1SE error bands for CAN
m = lm(interlocks~assets+nation, data=Ornstein)
dNew = data.frame(assets = seq(1000,100000, by=1000),nation='CAN')
dNew = modelPredictions(m, dNew)
plot(dNew$assets,dNew$Predicted, type = 'l', col= 'red')
lines(dNew$assets,dNew$CILo, type = 'l', col= 'gray', lwd =.5)
lines(dNew$assets,dNew$CIHi, type = 'l', col= 'gray', lwd =.5)

##Return predicted values for sample
P = modelPredictions(m)
```

---

modelR2 *Model R2, adjusted R2 and F-test*

Description

Reports model R2, adjusted R2, and F-test of model R2.

Usage

```
modelR2(Model, Print=TRUE)
```

Arguments

Model	an lm model object
Print	print results to screen. Default is TRUE

Value

Returns full list object from modelSummary() with many stats

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

lm, modelSummary, summary

Examples

```
m = lm(interlocks~assets+nation, data=Ornstein)
modelR2(m)
```

---

modelSummary *summary of results for lm model*

Description

This is a modified version of summary for use with an lm, glm, or lmer object. It provides results that align better with Brauer/Curtin perspective on these linear models from their graduate statistics series

Usage

```
modelSummary(Model, t = TRUE, Print= TRUE, Digits = 4)
```

Arguments

Model	a linear model, produced by lm.
t	Indicates if t-statistics (TRUE; Default) or F-statistics should be reported for tests of parameter estimates
Print	Print output to screen. Default is TRUE
Digits	Number of digits for values in coefficients table. Default = 4

Details

Reports model summary results from an lm object. Results include parameter estimates and their tests, SSE, model R2

Value

Returns a list with results for model.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

summary, modelR2

Examples

```
m = lm(interlocks~assets+nation, data=Ornstein)
modelSummary(m)
```

---

varContrasts *Set Factor Contrasts*

Description

Calculates contrast matrix for a specified contrast type. Options include DUMMY, POC, HELMERT, EFFECTS

Usage

```
varContrasts(TheFactor, Type = "DUMMY", RefLevel = length(levels(TheFactor)),
       POCList = NULL, Labels = NULL)
```

Arguments

TheFactor	factor from dataframe
Type	type of contrast, Options include DUMMY (default), POC, HELMERT, or EFFECTS
RefLevel	Reference level for contrast. Only applies to DUMMY, HELMERT, and EFFECTS. For DUMMY: RefLevel is numeric index of control/reference category (i.e. coded 0 for all regressors). For HELMERT: RefLevel = 1 indicates reverse HELMERT (i.e., last vs. earlier, second to last vs. earlier, etc), RefLevel = 'Highest Level' indicates forward HELMERT (i.e., first vs. later, second vs. later, etc). For EFFECTS: RefLevel is numeric index of excluded level.
POCList	if Type = POC, a list of Contrasts is required in POCList; e.g., list(c(1,0,-1), c(-1,2,-1)). Best to provide as whole numbers. Function will re-scale to unit weighted contrasts.
Labels	if Type = POC, Labels can be provided. If NULL (Default), contrast labels are POC1, POC2, etc.

Details

Use the contrast matrix with contrasts() to set contrast for a specific factor in dataframe.

Value

Returns contrast matrix for indicated type of contrast.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

contrasts

Examples

```
d = data.frame(f=factor(c('f1', 'f2', 'f3')))
contrasts(d$f)

##set as DUMMY with last category as reference
contrasts(d$f) = varContrasts(d$f, Type='DUMMY', RefLevel = 3)

##set as POC with user defined labels
contrasts(d$f) = varContrasts(d$f, Type='POC', POCList = list(c(2,-1,-1),c(0,1,-1)),
                 Labels = c('f1_v_f2f3', 'f2_v_f3'))

##set as reverse HELMERT
contrasts(d$f) = varContrasts(d$f, Type='HELMERT', RefLevel = 1)

##set as EFFECTS, excluding f3 vs. grand mean contrast
contrasts(d$f) = varContrasts(d$f, Type='EFFECTS', RefLevel = 3)
```

---

varDescribe *Provides typical descriptive statistics for data frame*

Description

Provides three levels of detail regarding descriptive statistics for a data frame. Based on describe() function from psych package

Usage

```
varDescribe(Data, Detail = 2, Digits=2)
```

Arguments

Data	a data frame
Detail	Indicates level of detail for descriptives, 1=minimal, 2=typical (default), 3= detailed
Digits	Number of decimal places to display; NULL = display all sig digits. Default =2.

Value

Returns table with descriptive statistics rounded to digits.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

describe lm.describeGroups describe.by

Examples

```
varDescribe(Ornstein)
varDescribe(Ornstein, Detail=3)
varDescribe(Ornstein, Detail=2, Digits=1)
```

---

varDescribeBy *Provides common desriptives for dataframe by factor(s)*

Description

Provides commons descriptive statistics for a data frame split on some factor or combination of factors. Essentially a wrapper for varDescribe() and by().

Usage

```
varDescribeBy(Data, IVList)
```

Arguments

Data	a dataframe
IVList	list of one or more factors from data frame

Value

An object of class "by", giving the results from varDescribe() applied to each subset.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
varDescribeBy(Adler,list(Adler$expectation,Adler$instruction))
```

---

varParse *Returns a subset of digits from a Number*

Description

Returns a subset of digits from a Number.

Usage

```
varParse(Number, UpperDigit=1, LowerDigit=1)
```

Arguments

Number	Number to parse
UpperDigit	Location in base ten of upper end of digits to return
LowerDigit	Location in base ten of lower end of digits to return

Value

Returns a subset of the digits in Number

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
varParse(1234,100,10)
varParse(1234,1,1)
varParse(1234,1000,1000)
```

varPlot *Creates histogram, optional rug/strip and density plots, and generates univariate descriptive statistics*

Description

Represents important aspects of a variable/vector both visually (histogram, rug or strip, and density plots) and with descriptive statistics of varying detail

Usage

```
varPlot(TheVar,  VarName = '', IDs = NULL, AddPoints = 'Strip',
        AddDensity = TRUE, Detail = 2)
```

Arguments

TheVar	A variable/vector to visualize
VarName	The variable name of TheVar as string. Default = ”
IDs	Rownames for interactive identification of data points, Default is NULL with no identification done
AddPoints	Strip (default), Rug, or None
AddDensity	TRUE (default) or FALSE to include density plot
Detail	1-3 of increasing detail for descriptives using varDescribe()

Value

Prints descriptive statistics table and creates graphic as side effect. Returns list with Indices, Rownames, and Values if identify is not NULL

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

hist(), rug(), varStripPlot(), density(), varDescribe(), describe(), identify()

Examples

```
varPlot(Prestige$income, 'Income') #default use strip
varPlot(Prestige$income, AddPoints='RUG')
varPlot(Prestige$income, IDs=rownames(Prestige))
```

varRecode *Recode levels of variable*

Description

Recodes levels of variable from old values to new values. Levels in Old are recoded to levels in New by matching position in these two vectors.

Usage

```
varRecode(Var, Old, New)
```

Arguments

Var	A variable to recode.
Old	Vector with original levels of Var
New	vector with new levels

Value

Returns variable with new levels

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

recode

Examples

```
##d$rIV1 = varRecode(d$IV1, c(-1,1), c(-.5, .5))
##d$rIV2 = varRecode(d$IV2, c(1,2,3), c(-.667, .333, .333))
##d$rIV3 = varRecode(d$IV3, c('A', 'B'), c('C', 'D'))
```

varRegressors *Adds actual numeric regressors for factor to dataframe as new variables*

Description

Adds new variables/columns in dataframe to represent numeric regressors for a factor. Factors are coded using their currently defined contrast codes. This function is useful for control of a factor covariate when graphing and ignoring this factor and/or other lower-level control variables. For this purpose, POC coding will typically be set for factor prior to using lm.codeRegressor

Usage

```
varRegressors(Data, VarName, RegressorNames = NULL)
```

Arguments

Data	The dataframe to add regressors
VarName	Character string name of variable to code regressor for
RegressorNames	Optional variable names for regressors.

Value

Returns original data frame (Data) with addition of new regressors.

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
d = Ornstein
contrasts(d$nation) = varContrasts(d$nation, Type='POC',
   POCList = list(c(3,-1,-1, -1),c(0,2,-1, -1), c(0,0,1,-1)))
d = varRegressors(d,'nation')
```

varRename *Rename Variable in Dataframe*

Description

Renames a variable in specified dataframe.

Usage

```
varRename(Data, From, To)
```

Arguments

Data	a dataframe object
From	vector of original name(s) of variable(s) as strings
To	vector of new name(s) of variable(s) as strings

Value

Returns dataframe with new variable names for specified variable(s)

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
d = data.frame(x=1:10, y=11:20)
names(d)
d = varRename(d, c('x', 'y'), c('x1', 'y1'))
names(d)
```

---

varReverse *Reverse score an ordinal or boolean scored item/variable*

Description

Reverse scores an item that was ordinal/interval scored or boolean.

Usage

```
varReverse(Var, LowAnchor, HighAnchor)
```

Arguments

Var	A variable to reverse score.
LowAnchor	Absolute low value for variable
HighAnchor	Absoluate high value for variable

Value

Returns variable new (reversed) scores

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

See Also

recode

Examples

```
##d$Item5r = varReverse(d$Item5, 1, 5)
```

---

varScore *Creates a total score from a sum of items*

Description

Creates a total score from a sum of items in a data frame. Can do range checking for items, reverse scoring of items, and prorating for missing data.

Usage

```
varScore(Data, Forward, Reverse=NULL, Range = NULL, Prorate = TRUE, MaxMiss = .20)
```

Arguments

Data	a dataframe that contains item scores among other variables
Forward	a vector of variable names to indicate the items that should be summed as is (in contrast to reverse scored). All items should be listed in EITHER Forward or Reverse argument
Reverse	a vector of variable names to indicate the items that should be summed after reverse scoring the items. Range arugment (see below) must also be specified to reverse score items. Default is NULL which indicates no items are reverse scored. All items should be listed in EITHER Forward or Reverse argument
Range	A numeric vector with two values for low and high anchor values for items. Must be specified if any items will be reverse scored. Used also to do range checking for all items. Default is NULL which indicates no range checking and no reverse scored items
Prorate	A boolean to indicate if total score should be prorated for missing data. Default is TRUE.
MaxMiss	Maximum acceptable percentage of missing data before total score will be set to missing. Implemented regardless if Prorate is TRUE or FALSE. However, if Prorate is false, should probably be set to 0

Details

This is a flexible routine to score measures that consist of sums of items.

Value

Returns vector of total scores for each participant

Author(s)

John J. Curtin <jjcurtin@wisc.edu>

Examples

```
##Same code not executable with sample dataframe
##varScore(d, c('I1', 'I3', 'I4'), Reverse= c('I2', 'I5'),
##         Range = c(1,5), Prorate=TRUE, MaxMiss = .25)
```

Index