Package 'OurTools'

February 26, 2020

Title OurTools for Structured Statistical Analysis

Version 0.24

Date 2020-02-26

Description
A collection of functions designed to support structured statistical analysis of clinical studies.
OurTools supports data preparation, data import and export,
and provides smart functions for recurrent statistical tasks
implementing a common methodological standard.
Functions return either data.frames or plots
that can be readily knitred in analysis reports.

Author Dirk Hasenclever <dirk.hasenclever@imise.uni-leipzig.de>

Maintainer Dirk Hasenclever <dirk.hasenclever@imise.uni-leipzig.de>

Imports graphics, stats, utils, boot, visreg, survival, robust, cmprsk, gplots

License MIT + file LICENSE

EnableCodeIndexing Yes

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

R topics documented:

BinaryAssociation *BinaryAssociation*

Description

BinaryAssociation produces a data.framed matrix of association measures between binary variables.

Usage

```
BinaryAssociation(X, Y = NULL, TYPE = "YuleQ", DIGITS = 3,
  ASMATRIX = F)
```

Arguments

X	either a binary variable or a data.frame containing binary variables (non binaries are discarted)
Y	=NULL only used of X is a binary variable
TYPE	="YuleQ". Choose association measure among c("OddsRatio", "logOddsRatio", "YuleY", "YuleQ", "RR")
DIGITS	=3 Number of digits of association measures
ASMATRIX	= F if TRUE returned result is a KxK matrix not a data.frame with rownames as first column (default)

Value

a data.frame with colnames[1] indicating the TYPE or matrix if ASMATRIX = T)

Author(s)

Dirk Hasenclever 2018-01-07, 2019-02-05

Examples

```
## Not run:
require(MASS)
BinaryAssociation(birthwt)
Table(birthwt$smoke,birthwt$low)
BinaryAssociation(birthwt$smoke,birthwt$low)
BinaryAssociation(birthwt$smoke,birthwt$low,TYPE="logOddsRatio")

## End(Not run)
```

CompareMetricBySplit *CompareMetricBySplit*

Description

compares one metric variable by a split variable with t.test-oneway.test and wilcox.test-kruskal.test. For two groups confidence intervals for mean difference, difference in pseudomedians (Hodge-Lehmann), and proversion probabilities are provided.

Usage

```
CompareMetricBySplit(DFR, METRIC, SPLIT, DIGITS = 3, CONF.LEVEL = 0.95)
```

Arguments

DFR,	is a dataframe
METRIC,	name of metric variable in DFR
SPLIT,	name of split factor (grouping variable) in DFR
DIGITS	= 3, Number of digits for estimates
CONF.LEVEL	= 0.95, Confidence Level for confidence interval

Details

CompareMetricBySplit

Value

a data.frame with results

Author(s)

Dirk Hasenclever 2017-12-10

Examples

```
x<-c(rnorm(100,2,1),rnorm(200,2.4,1))
x[sample(1:300,13)]<-NA
x<-c(rnorm(100,2,1),rnorm(200,2.5,1))
gr2<-factor(c(rep("Gr1",100),rep("Gr2",200)))
gr3<-factor(c(rep("Gr1",100),rep("Gr2",100),rep("Gr3",100)))
dfr<-data.frame(X=x,GR2=gr2,GR3=gr3)#
CompareMetricBySplit(dfr,METRIC="X",SPLIT="GR2")
CompareMetricBySplit(dfr,METRIC="X",SPLIT="GR3")
```

CompareSuccessRatesBySplit
CompareSuccessRatesBySplit

Description

CompareSuccessRatesBySplit compares one factor variable by a split variable with prop.test and glm. For two groups confidence intervals for difference in success rate, odds ratio, and risk ratio are provided.

Usage

```
CompareSuccessRatesBySplit(DFR, FAC, SPLIT, SUCCESS = 2, DIGITS = 3,
  CONF.LEVEL = 0.95)
```

Arguments

DFR	is a dataframe
FAC	name of factor in DFR. Will be converted to a factor if not a factor.
SPLIT	name of split factor (grouping variable) in DFR
SUCCESS	=2 factor level that defines success. If numeric, the respective level of FAC is assumed.
DIGITS	=3 Number of digits for estimates
CONF.LEVEL	=0.95 Confidence Level for confidence interval

Details

CompareSuccessRatesBySplit

Value

a data.frame: two groups: proabability difference, odds ration and relative risk with CI; more than two groups: chisqr.test and likelihood ratio test of glm

Author(s)

Dirk Hasenclever 2017-12-12; 2018-02-05; 2018-08-14

Examples

```
## Not run:
require(MASS)
DAT<-survey
colnames(DAT)<-toupper(colnames(DAT))
CrossTabs(DAT$SMOKE,DAT$SEX,TYPE="COL")
DAT$Smoker<-DAT$SMOKE
levels(DAT$Smoker)<-c("Smoker","NonSmoker","Smoker","Smoker")
CrossTabs(DAT$Smoker,DAT$SEX,TYPE="COL")
CompareSuccessRatesBySplit(DAT,FAC="SMOKE",SUCCESS="Never",SPLIT="SEX")
CrossTabs(DAT$SMOKE,DAT$FOLD)
CompareSuccessRatesBySplit(DAT,FAC="SMOKE",SUCCESS="Never",SPLIT="FOLD")
```

```
## End(Not run)
```

`ConfusionTableStatistics` *ConfusionTableStatistics*

Description

ConfusionTableStatistics describes the concordance of two raters

Usage

```
ConfusionTableStatistics(ctab, DIGITS = 1)
```

Arguments

`ctab`	is a KxK table of categorisations by two rater. MUST be square.
`DIGITS`	= 1, Number of digits for percentages

Value

data.frame of statistics, Overall concordance, Discrepancies > 1, #' Concorances given at least one rater gives category i

Author(s)

Dirk Hasenclever 2017-07-14

Examples

```
## Not run:
ctab<-matrix(c(23,2,0,1,2,34,1,0,0,2,22,3,0,1,4,18),nrow=4)
rownames(ctab) = paste("rater1:",1:4)
colnames(ctab) = paste("rater2:",1:4)
ctab
ConfusionTableStatistics(ctab)

## End(Not run)
```

CorDiffCI *CorDiffCI*

Description

CorDiffCI calculates a confidence interval for the difference of two independent correlation coefficients.

Usage

```
CorDiffCI(R1, R2, N1, N2 = N1, CONF.LEVEL = 0.95, DIGITS = 3)
```

Arguments

R1	estimate of first correlation coefficient
R2	estimate of second correlation coefficient
N1	number of valid values for R1
N2	=N1 number of valid values for R2
CONF.LEVEL	=0.95 desired confidence level
DIGITS	=3 Number of digits for correlation difference

Value

data.frame of R1, OR, CorDiff, Lower and Upper

Author(s)

Dirk Hasenclever 2018-01-02 based on Zou, G. Y. (2007). Toward using confidence intervals to compare correlations. Psychological Methods, 12, 399-413.

Examples

```
## Not run:
CorDiffCI(.49,.36,145,87,CONF.LEVEL=.95)  ## Example from Zou 2007 reproduced
x<-c(rnorm(50),rnorm(50,5))
y<-c(rnorm(50),rnorm(50,5)+10*x[51:100])*2
Split<-factor(paste("Group",rep(LETTERS[1:2],each=50),sep="-"))
ScatterPlot(x,y,SPLIT=Split)
CorDiffCI(cor(x[Split=="Group-A"],y[Split=="Group-A"]),
          cor(x[Split=="Group-B"],y[Split=="Group-B"]),
          50,50)

## End(Not run)
```

CountBarplot *CountBarplot*

Description

Plot a barplot of a count variable avoiding left-out bars at zero counts in [min; MAX] A frequency table including zeros is returned invisibly. There are three options: Frequency, Proportion, and Percent on y-axis.

Usage

```
CountBarplot(COUNTS, TYPE = "Frequency", MAIN = "Barplot of",
  XLAB = "Counts", YLAB = NULL, COL = 4)
```

Arguments

COUNTS	is a prepared data matrix with cumulative frequencies of ORDLEVELS and COURSLABEL
TYPE	="Frequency other options for y-axis: "Proportion" and "Percent
MAIN	="Barplot of" is default. If left as default name of variable COUNTS is appended automatically.
XLAB	Label on x-axis set to "Counts" by default
YLAB	=NULL y-axis label. Set to TYPE if left NULL. (Nvalid = xxx) appended automatically
COL	=4 Colour of bars, default = 4 (blue).

Value

Invisible return of the count variable as ordered factor with non observed counts intercalated. Side-effect: generates a barplot.

Author(s)

Dirk Hasenclever 2014-04-13, 2018-09-10

Examples

```
## Not run:
Counts<-sample(1:28,52,replace=T)
CountBarplot(Counts)
Counts[3]<-NA
CountBarplot(Counts,TYPE="Proportion",YLAB="Hallo World")
Countsfactor<-CountBarplot(Counts,TYPE="Percent",COL=2)
Countsfactor

## End(Not run)
```

CountValue *CountValue*

Description

CountValue counts the number positive occurrences of a specified value in a vector X.

Usage

```
CountValue(X, VALUE)
```

Arguments

X	vector possibly containing NAs
VALUE	specified value to be counted. VALUE=NA work.

Details

CountValue

Value

number of components == VALUE

Author(s)

Dirk Hasenclever 2019-02-03

Examples

```
## Not run:
CountValue(c(1,2,NA,NA,5), NA)
CountValue(c(1,2,2,2,NA,5), VALUE=2)
## Typical application
df<-matrix(sample(LETTERS[1:5], 50, replace=T), ncol=10)
df
apply(df,1,CountValue,"A")
apply(df,2,CountValue,"A")

## End(Not run)
```

Cox2Df *Cox2Df*

Description

Cox2Df produces a table with results of COX models as printable data.frame.

Usage

```
Cox2Df(MOD, DIGITS = 3)
```

Arguments

MOD,	fit of a Cox Model
DIGITS	=3 round to DIGITS

Value

data.frame with Covariates, coef, se(coef), exp(coef), lower.95 upper.95 p.value

Author(s)

Dirk Hasenclever 2017-12-20

Examples

```
## Not run:
require(MASS)
mod<-coxph(Surv(stime,status)~treat+age+Karn+prior,data=VA)
Cox2Df(mod,DIGITS=3)

## End(Not run)
```

CrossTabs *CrossTabs*

Description

CrossTables with margins and either overall, row or column percentages.

Usage

```
CrossTabs(fac1, fac2 = NULL, TYPE = "ALL", NOZEROS = T, DIGITS = 1,
  PVALUE = F, VARNAME = F, LONGNAME = NULL, ALL = NULL)
```

Arguments

fac1	factor or variable convertible to factor
fac2	= NULL factor or variable convertible to factor. If not provided a dummy factor is generated.
TYPE	= "ALL" chooses between "ALL", "ROW", "COL"
NOZEROS	= T Option to discard factor levels which do not occur
DIGITS	= 1 Number of decimals for percentages
PVALUE	= F p-value from Chi2-test
VARNAME	= F Option to add a column on the left with the varname of fac1
LONGNAME	= NULL alternative to varname derived from function call
ALL	= NULL If ALL=NULL 'All' columns on the right, if ALL=T on the left and if ALL=F omitted

Value

CrossTable as data.frame with Counts and Percentages intercalated.

Author(s)

Dirk Hasenclever 2017-03-08; 2017-03-14, 2018-01-05

See Also

FactorsBySplit

Examples

```
## Not run:
a<-c(rep("M",240),rep("F",170))
b<-sample(c(rep("Stage1",110),rep("Stage2",180),rep("Stage3",120)),410)

CrossTabs(a,b,TYPE="ALL")
CrossTabs(a,b,TYPE="ROW")
CrossTabs(a,b,TYPE="ROW",ALL=F)
CrossTabs(a,b,TYPE="ROW",ALL=T)
CrossTabs(a,b,TYPE="COL")

## End(Not run)
```

DataMaturityPlot *DataMaturityPlot*

Description

Produces a DataMaturity Plot comparing an actual Kaplan Meier curve with the optimistic one assuming all events known.

Usage

```
DataMaturityPlot(DFR, TIMES, STATUS, IDEALTIMES, TIMEPOINT = NULL,
  MAIN = "Data Maturity Plot: fudged versus actual", XLAB = "time",
  YLAB = "proportion still under observation", YLIM = c(0, 1),
  UNITS = "months", COLS = c(3, 2), ABLINES = NULL, CEX = 1.2)
```

Arguments

DFR	data.frame with variables TIMES, STATUS, IDEALTIMES (colnames). Alternatively, if DFR set to NULL, TIMES, STATUS, IDEALTIMES will be treated as data vectors.
TIMES	name of times component of time to event, alternatively data vector
STATUS	name of status component of time to event, alternatively data vector
IDEALTIMES	name of ideal observation time component of time to event, alternatively data vector
TIMEPOINT	=NULL, time where to read off curves, set at about 80 percent percentile if not specified
MAIN	="Data Maturity Plot - fudged versus actual", plot title
XLAB	="time", label for time axis
YLAB	="proportion still under observation", label for y axis
YLIM	=c(0,1), you may shorten y-axis if desired
UNITS	="months", unit of time appendend at XLAB if not NULL or blank. In addition, 'months', 'Months', 'days', 'Days' are recognised and used to find appropriate ticks.
COLS	=c(3,2), colours for curves based on ideal and actual observations
ABLINES	=NULL, horizontal dashed lines
CEX	=1.2, scaling factor of fonts on axes and legend

Details

DataMaturityPlot

Value

returns data.frame with estimates and 95percent-CI of survival times at TIMEPOINT (Fudged and Actual)

Author(s)

Dirk Hasenclever 2017-09-21, 2018-01-31

Examples

```
## Not run:
TIMES<-rexp(100)*60
CENS<-runif(100,0,80)
STATUS<-(TIMES <= CENS)
TIMES[TIMES > CENS]<-CENS[TIMES > CENS]
IDEALTIMES<-TIMES+12
IDEALTIMES[STATUS]<-TIMES[STATUS]
```

```
DFR<-data.frame(TIMES,STATUS,IDEALTIMES)
DataMaturityPlot(DFR=DFR,"TIMES","STATUS","IDEALTIMES",UNITS="months")
DataMaturityPlot(DFR=DFR,"TIMES","STATUS","IDEALTIMES",UNITS="years")
DataMaturityPlot(DFR=NULL,TIMES,STATUS,IDEALTIMES,UNITS="months",COLS=c(4,3))
DataMaturityPlot(DFR=NULL,TIMES,STATUS,IDEALTIMES,UNITS="days",CEX=1.0)

## End(Not run)
```

`DescribeMetricBySplit-deprecated` *DescribeMetricBySplit produces a data.frame describing one or many metric variables*

Description

This OUT_DATED function produces a data.frame describing one or many metric variables selected from a data.frame DFR by their names in a character vector COLUMNS SPLIT is a group variable which may be left out

Usage

```
DescribeMetricBySplit(DFR, COLUMNS = NULL, SPLIT = NULL, LONGNAMES = NULL,
  SEL = "All", DIGITS = 3, ALL = T, FORTRANSPOSITION = F)
```

Arguments

`DFR`	is a dataframe (if vector, COLUMNS must remain NULL)
`COLUMNS`	=NULL colnames in DFR to be described (at least one, unless DFR is a vector)
`SPLIT`	=NULL split factor (variable), if NULL simple description, NULL is default
`LONGNAMES`	=NULL optional long names for variables specified by COLUMNS
`SEL`	="ALL" if specified select statistics from c("MEAN", "SD", "P25", "MEDIAN", "P75", "MIN", "MAX", "MAD", "NVALID", "NMISSING")
`DIGITS`	Number of digits for percentages , digits=1 is default
`ALL`	=T If TRUE, also overall statistics are calculated
`FORTRANSPOSITION`	=F Set to T if you later want to transpose the resulting data.frame - VARIABLE stays redundant

Value

a data.frame with statistics for each COLUMN by the group variable SPLIT

See Also

`OurTools-deprecated`

Df2Csv *Df2Csv writes a data.frame to a csv file*

Description

Df2Csv writes a data.frame to a csv file such that it can be opended in excel and excel interprets numerics with "." as decimal

Usage

```
Df2Csv(df, FILENAME = NULL, TABLECAPTURE = NULL, DIGITS = 4,
  WIDTH = 12, JUSTIFY = "right", ROW.NAMES = T)
```

Arguments

df	as dataframe
FILENAME	as the saved name
TABLECAPTURE	as header for table
DIGITS,	4 is default
WIDTH,	width=12 is default
JUSTIFY	, right is default
ROW.NAMES,	TRUE is default

Value

csv file

Author(s)

Dirk Hasenclever

Examples

```
## Not run:
test<-data.frame(a=rnorm(10),b=rep(1:2,5), c=rep(rnorm(10)))
Df2Csv(df=test,TABLECAPTURE="Test of data frame output",DIGITS=2)

## End(Not run)
```

Df2Rdata *Df2Rdata*

Description

Df2Rdata writes a data.frame to a .Rdata file renaming the DFR corresponding to the DFNAME such that the DFR can be loaded for knitr. ATTENTION: If there exists a ResultDir string pointing to a Results directory in the global environment, DRF is saved there!

Usage

```
Df2Rdata(DFR, DFNAME, ROW.NAMES = T)
```

Arguments

DFR	as dataframe
DFNAME	is Filename (without the .Rdata !!)
ROW.NAMES	=TRUE is default

Value

nothing

Author(s)

Dirk Hasencever 2017-03-22, 2018-01-18

Examples

```
## Not run:
test<-data.frame(a=rnorm(10),b=rep(1:2,5), c=rep(rnorm(10)))
Df2Rdata(DFR=test, DFNAME="Example")

## End(Not run)
```

FacTable *Creates a frequency table of a factorial variable*

Description

The function creates a frequency table of a factorial variable.

Usage

```
FacTable(fac, VARNAME = NULL, NOZEROS = T, SORT = F)
```

Arguments

fac	is a factorial variable
VARNAME,	default=NULL, name the variable, if NULL, VARNAME generated from fac
NOZEROS,	removed if 0 observations, TRUE is default
SORT,	sort ascending, FALSE is default

Details

FacTable

Value

a frequency table

Author(s)

Dirk Hasenclever

Examples

```
## Not run:
fac<-c(rep("D",56),rep("F",23),rep("AT",5))
fac<-factor(fac,levels=c("AT","B","D","F"))
table(fac)
FacTable(fac,NOZEROS=F)
FacTable(fac,NOZEROS=F,SORT=F)
FacTable(fac,VARNAME="Country Codes",NOZEROS=F,SORT=F)
gr<-factor(c(rep("Gr1",100),rep("Gr2",200)))
FacTable(gr)

gg<-c(rep("Gr1",100),rep("Gr2",200),SORT=F)
FacTable(gg)

gn<-c(rep(1,100),rep(2,200))
FacTable(gn,VARNAME="NumerischerVektor")

## End(Not run)
```

FactorsBySplit *produces a data.frame describing one or many factor variables by a grouping factor SPLIT*

Description

The function produces a data.frame describing one or many Factor variables selected from a data.frame DFR by their names in a character vector COLUMNS SPLIT is a group variable which may be left out. FactorsBySplit is a wrapper function calling CrossTabs for each COLUMN, rbinding the results.

Usage

```
FactorsBySplit(DFR, COLUMNS, SPLIT = NULL, LONGNAMES = NULL,
  PVALUE = F, NOZEROS = T, ALL = NULL, DIGITS = 1)
```

Arguments

DFR	is a dataframe
COLUMNS	colnames in DFR to be described (at least one)
SPLIT	split factor (variable) of same length as DFR, or alternatively column name in DFR as string. If SPLIT=NULL simple factor description, NULL is default
LONGNAMES	=NULL alternative names for the COLUMNS
PVALUE	p-value from Chi2-test FALSE is default
NOZEROS	eliminates factor levels that were not obserevd, TRUE is default
ALL	= NULL If ALL=NULL 'All' columns on the right, if ALL=T on the left and if ALL=F omitted.
DIGITS	Number of digits for percentages , digits=1 is default

Details

FactorsBySplit

Value

a frequency table by group variable

Author(s)

Dirk Hasenclever 2017-03-14, 2018-01-05

See Also

CrossTabs

Examples

```
## Not run:
a<-c(rep("M",240),rep("F",170))
b<-c(rep("S1",110),rep("S2",190),rep("S3",110))
c<-factor(b,levels=c("S0","S1","S2","S3"))
gr<-sample(LETTERS[1:2],410,TRUE)
df<-data.frame(a,b,c,gr)

FactorsBySplit(df, COLUMNS = c("a","b","gr"))
FactorsBySplit(df,c("a","b","c"), SPLIT = "gr")
FactorsBySplit(df,c("a","b","c"), SPLIT = "gr",NOZERO=F)
FactorsBySplit(df,c("a","b"), SPLIT = "gr",ALL=NULL,PVALUE=T)
FactorsBySplit(df,c("a","b"), SPLIT = "gr",ALL=F,PVALUE=T)
FactorsBySplit(df,c("a","b"), SPLIT = "gr",ALL=T,PVALUE=T)
FactorsBySplit(df,c("a","b"), SPLIT = "gr",ALL=T,PVALUE=T,DIGITS=3)
FactorsBySplit(df,c("a","b"), SPLIT = "gr",ALL=T,PVALUE=T,
                        LONGNAMES=c("Gender","STAGE"))

## End(Not run)
```

findmode-deprecated *identifies the modal value of a numeric variable*

Description

The function determines the modal value of a numeric variable, which is needed for example in graphics.

Usage

```
findmode(x, ADJ = 1)
```

Arguments

x	is a numeric variable
ADJ	adjust, default=1

Value

mode as modal value of a numeric variable

#'

See Also

OurTools-deprecated

FollowUpPlot *FollowUpPlot*

Description

Produces a Follow-Up (or inverse Kaplan Meier Plot) comparing actual and ideal observation times. Patients with event are censored. Medians are indicated.

Usage

```
FollowUpPlot(DFR, TIMES, STATUS, IDEALTIMES,
  MAIN = "Distribution of ideal and actual observation times",
  XLAB = "follow-up time", YLAB = "proportion still under observation",
  UNITS = "months", COLS = c(3, 2), CEX = 1.2)
```

Arguments

DFR	data.frame with variables TIMES, STATUS, IDEALTIMES (colnames). Alternatively, if DFR set to NULL, TIMES, STATUS, IDEALTIMES will be treated as data vectors.
TIMES	name of times component of time to event, alternatively data vector
STATUS	name of status component of time to event, alternatively data vector
IDEALTIMES	name of ideal observation time component of time to event, alternatively data vector
MAIN	="Distribution of ideal and actual observation times", plot title
XLAB	="follow-up time", label for time axis
YLAB	="proportion still under observation",label for y axis
UNITS	="months", unit of time appendend at XLAB if not NULL or blank. In addition, 'months', 'Months', 'days', 'Days' are recognised and used to find appropriate ticks.
COLS	=c(3,2), colours for ideal and actual observation curves
CEX	=1.2, scaling factor of fonts on axes and legend

Details

FollowUpPlot

Value

returns data.frame with estimates and 95

Author(s)

Dirk Hasenclever 2017-09-21

Examples

```
## Not run:
### Test FollowUpPlot
TIMES<-rexp(100)*12
TIMES[1:5]<--TIMES[1:5]
STATUS<-rbinom(100,1,.2)
IDEALTIMES<-runif(100,min=1,max=70)
DFR<-data.frame(TIMES,STATUS,IDEALTIMES)
FollowUpPlot(DFR=DFR,"TIMES","STATUS","IDEALTIMES",UNITS="months")
FollowUpPlot(DFR=DFR,"TIMES","STATUS","IDEALTIMES",UNITS="years")
FollowUpPlot(DFR=NULL,TIMES,STATUS,IDEALTIMES,UNITS="months",COLS=c(4,3))
FollowUpPlot(DFR=NULL,TIMES,STATUS,IDEALTIMES,UNITS="days",CEX=1.5)

## End(Not run)
```

Glm2Df *Glm2Df*

Description

Glm2Df produces a table with results of glm models as printable data.frame.

Usage

```
Glm2Df(MOD, DIGITS = 3)
```

Arguments

MOD,	fit of a glm or lm regression Model (without random effects)
DIGITS	=3 round to DIGITS

Value

data.frame with Covariates, coef, se(coef), exp(coef), lower.95 upper.95 p.value

Author(s)

Dirk Hasenclever 2017-12-20, 2019-02-05

Examples

```
## Not run:
mod<-lm(WBC~HB+SEX+AGE,data=HLProgFactors)
Glm2Df(mod,DIGITS=3)
mod<-glm(SEX~HB+WBC+AGE,data=HLProgFactors,family = "binomial")
Glm2Df(mod,DIGITS=3)

## End(Not run)
```

HLProgFactors *Data form the International Prognostic Factor Project in advanced Hodgkin Lymphoma.*

Description

A dataset with patient characteristics and time to Event data form N= 5141 HL patients.

Usage

```
HLProgFactors
```

Format

A data frame with 5141 rows and 19 variables:

PATNO Patient Number

STUDY Short name of study group providing data

SEX Sex of patient

AGE Age at start of treatment

HISTO Histological subtype

LAP Laparotomy: CS = no; PS = yes

STAGE Stage I, II, III, IV

SSYM Systemic Symptoms: A=no B=yes

OSVtime Overall Survival Time [months]

OSVstatus Vital status at SV: 0=alive; 1=dead

FFTFtime Freedom from treatment failure time [months]

FFTFstatus Failure of therapy: 0=no; 1=yes

HB Haemoglobin [g/dl]

HCT Haematocrit %

ESR Erythrocyte sedimentation rate mm/h

WBC WBC [G/l]

LYMPHO Lymphocyte Count [G/l]

LYMPHPR Lymphocyte Count in % WBC

ALBUMIN Albumin

Source

https://www.nejm.org/doi/10.1056/NEJM199811193392104

`Level2NA` *Level2NA*

Description

Level2NA recodes one selected LEVEL of a factor X to NA and eliminates it from levels(X). Level2NA can be applied to a factor or to all factors in a data.frame. NOTE: Level2NA has no effect on character vectors.

Usage

```
Level2NA(X, LEVEL, EXCEPT = NULL)
```

Arguments

`X`	is either a factor or a data.frame
`LEVEL`	which is to be recoded to NA and has to be specified
`EXCEPT`	=NULL list colnames of a df that should not be transformed

Value

the transformed factor or data.frame

Author(s)

Dirk Hasenclever 2017-03-22, 2018-01-09

Examples

```
## Not run:
# Applied to a vector
Y<-factor(c("BALI","BALI","SUMATRA","LOMBOK","n.n.", "JAVA"))
Y
Level2NA(Y, LEVEL="n.n.")

# Applied to a data.frame
DFR<-data.frame(X=1:6,
                Y=c("BALI","BALI","SUMATRA","LOMBOK","n.n.", "JAVA"),
                Z=factor(c(LETTERS[1:5],"n.n.")))
DFR
DFR<-Level2NA(DFR, LEVEL="BALI")
DFR
DFR<-Level2NA(DFR, LEVEL="n.n.",EXCEPT="Z")
DFR

## End(Not run)
```

LhrPlot *LhrPlot*

Description

Produces a plot of log hazard ratios (lhr) for varying cut-values on a metric variabele, smoothed with lowess and a confidence band based on visreg asymptotics.

Usage

```
LhrPlot(DFR, TIMES, STATUS, METRIC, COLUMNS = NULL, SPANLOESS = 1/2,
  XLAB = NULL, YLAB = "log hasard ratio", MAIN = "Lhr plot",
  SUB = "Note: Lhr depends strongly on distribution")
```

Arguments

DFR	data.frame containing the time to event and metric variable
TIMES	name of variable in DFR: EITHER Time to event variable of class "Surv" OR time component of time to event
STATUS	name of variable in DFR: status component von time to event (needed only if TIMES is not of class "Surv")
METRIC	name of metric variable in DFR to be investigated
COLUMNS	= NULL vector of further covariate names in DFR != METRIC to adjust for in calculating the residuals.

SPANLOESS	=2/3 smoothing parameter for lowess smoother
XLAB	=NULL set range on x-axis label manually instead of automatically
YLAB	="log hasard ratio" standard y-axis label
MAIN	=NULL for no main title set MAIN=""
SUB	="Note: Lhr depends strongly on distribution", subtitle below XLAB

Details

LhrPlot

Value

produces a plot and returns a matrix of cuts and lhrs

Author(s)

Dirk Hasenclever 2017-11-07, 2018-01-09, 2018-02-14

Examples

```
## Not run:
require(MASS)
data(VA)
LhrPlot(VA,"stime","status","age")
LhrPlot(VA,"stime","status","age",COLUMNS="Karn")

## End(Not run)
```

LocationScaleEstimates

LocationScaleEstimates

Description

LocationScaleEstimates estimates mean, pseudomedian, sd, mad and mode (of density) of a numeric variable. Estimates and CIs are based on t.test and wilcox.test for mean and pseudomedian. For other parameters the CI is based on bca-bootstrap relying on package boot. CAUTION: Bootstrap may take a while with large data.

Usage

```
LocationScaleEstimates(METRIC, PARAM = c("Mean", "Pseudomedian", "Sd"),
  CONF.LEVEL = 0.95, DIGITS = 3, MODEADJ = 1, NBOOT = 999)
```

Arguments

METRIC	is a numeric variable
PARAM	=c("Mean","Pseudomedian","Sd") vector of parameter names. Choose from PARAM=c("Mean","Pseudomedian","Sd","Mad","Mode")
CONF.LEVEL	= 0.95 Confidence Level for confidence interval
DIGITS	=3 Number of digits for estimates
MODEADJ	adjust, default=1 for density for mode estimation
NBOOT	=999 number of bootstrap replicates

Value

data.frame with parameters as rows and estimate, lower and upper 95

Author(s)

Dirk Hasenclever 2017-12-14

Examples

```
METRIC<-c(rnorm(200),rnorm(100,2,1),10:17)
## Not run:
LocationScaleEstimates(METRIC)
LocationScaleEstimates(METRIC,PARAM=c("Mean","Pseudomedian","Mode"),MODEADJ = .7)

## End(Not run)
```

`MartingalePlot` *MartingalePlot*

Description

Produces a plot of the martingale or deviance residuals, smoothed with lowess and a confidence band based on visreg asymptotics. Additionally, a bootstrap confidence band can be obtained.

Usage

```
MartingalePlot(DFR, TIMES, STATUS, METRIC, COLUMNS = NULL,
  TYPE = "martingale", SPANLOESS = 2/3, NBOOT = 0, XLIM = NULL,
  YLIM = NULL, XLAB = NULL, YLAB = "Martingale residuals",
  MAIN = NULL, SUB = NULL, PCH = 1, PCEX = 1)
```

Arguments

DFR	data.frame containing the time to event and metric variable
TIMES	name of variable in DFR: EITHER Time to event variable of class "Surv" OR time component of time to event
STATUS	name of variable in DFR: status component von time to event (needed only if TIMES is not of class "Surv")
METRIC	name of metric variable in DFR to be investigated
COLUMNS	= NULL, vector of further covariate names in DFR != METRIC to adjust for in calculating the residuals.
TYPE	="martingale", specifies type of residuum, "martingale" or the more symmetric "deviance"
SPANLOESS	=2/3, smoothing parameter for lowess smoother
NBOOT	=0, number of Bootstrap simulations. Set to > 0 (e.g. 1000), if additional bootstrap confidence band desired
XLIM	=NULL, set range on x-axis manually instead of automatically
YLIM	=NULL, set range on y-axis manually instead of automatically

XLAB	=NULL, set x-axis label manually instead of automatically
YLAB	="Martingale residuals", standard y-axis label
MAIN	=NULL, for no main title set MAIN=""
SUB	=NULL, subtitle below XLAB, if unspecified adjusted for: COLUMNS
PCH	=1 pointtype for martingalresidues
PCEX	=1 pont size (cex)

Details

MartingalePlot

Value

produces a plot

Author(s)

Dirk Hasenclever 2017-09-20

Examples

```
## Not run:
require(MASS)
data(VA)
# The Martingal plot shows the functional form of the dependence of
#              a time to event and a metric variable.
MartingalePlot(VA,"stime","status","age")
MartingalePlot(VA,"stime","status","age",SPANLOESS = .9)
MartingalePlot(VA,"stime","status","age", XLIM=c(40,70))

MartingalePlot(VA,"stime","status","Karn")
MartingalePlot(VA,"stime","status","Karn",SPANLOESS = .9)

# Martingal residues of age after adjusting linearly for karno
MartingalePlot(VA,"stime","status","age",COLUMNS ="Karn",
            SPANLOESS = .9, XLIM=c(40,70))

## End(Not run)
```

Matrix2Df *Matrix2Df*

Description

Matrix2Df converts a matrix (e.g. result of base::table) into a dataframe with the rownames as an additional first column ROW. Matrix2Df(M) is ready to be pandered.

Usage

```
Matrix2Df(M)
```

Arguments

M	as a matrix

Value

dataframe

Author(s)

Dirk Hasenclever 2018-01-12

Examples

```
## Not run:
x<-rep(LETTERS[1:3],each=20)
y<-rep(letters[1:5],12)

# Generate a matrix as it is
tt<-Table(x,y,USENA=T)

tt
is.matrix(tt)

data.frame(tt)  # Not what we want!

Matrix2Df(tt)   # is a Data.frame that can be pandered

M<-diag(10)
M
Matrix2Df(M)

## End(Not run)
```

MergeWithMaster *MergeWithMaster - Safe and error controlled merging of dataframes*

Description

MergeWithMaster combines two dataframes using a single key variable and prints a Match-Report: MASTER keys not in SLAVE, SLAVE keys not in MASTER, and presence of duplicates in MASTER, SLAVE and MASTESLAVE.

Usage

```
MergeWithMaster(MASTER, SLAVE, BY = "PATNO", SUB = NULL,
  DROP_DOUBLE = c("PATSTUID", "CNO", "PATNO"), REPORT = T)
```

Arguments

MASTER	as 1st dataframe
SLAVE	as 2nd dataframe
BY	="PATNO key name both in MASTER and SLAVE

SUB	=NULL is the extention the second colname of a doublette gets as ".SUB"
DROP_DOUBLE	=c("PATSTUID","CNO","PATNO") redundant colnames to be droped from SLAVE
REPORT	= T In general we definitely want a report, but not always.

Details

MergeWithMaster

Value

a merged dataframe

Author(s)

Dirk Hasenclever 2017-03-10; 2017-12-12; 2019-02-08

Examples

```
## Not run:
df1 = data.frame(CustomerId=c(1:6),Product=c(rep("Toaster",3),rep("Radio",3)))
df2 = data.frame(CustomerId=c(2,4,4,7),State=c(rep("Alabama",3),rep("Ohio",1)))
MergeWithMaster(df1,df2,"CustomerId")
MergeWithMaster(df1,df2,"CustomerId",REPORT = F)

## End(Not run)
```

MetricBySplit *MetricBySplit produces a data.frame describing one metric variable by an optional SPLIT grouping*

Description

The function produces a data.frame describing one metric variable METRIC with various statistics SPLIT is a group variable which may be left out. Tests may be preformed on Mean (t.test, oneway.test) or Median (wilcox.test,kruskal.test)

Usage

```
MetricBySplit(METRIC, SPLIT = NULL, VARNAME = T, LONGNAME = NULL,
  PVALUE = F, TEST = NULL, ALL = T, SEL = NULL, SHOW_N = T,
  DIGITS = 2)
```

Arguments

METRIC	is a numeric variable
SPLIT	=NULL split factor (group variable), if NULL an ungrouped description is returned
VARNAME	=T indicate the VARNAME name of METRICS in the first column
LONGNAME	=NULL optional long name for METRICS if NULL a LONGNAME is derived from the call
PVALUE	=F If TRUE, tests on mean and Median are performed and added as last column

TEST	=NULL Choose TEST from c("Mean","Median"). If not specified set to both i.e. c("Mean","Median")
ALL	=T If TRUE, also overall statistics
SEL	=NULL if specified select statistics from c("Mean", "Sd", "Median", "Mad", "Perc25", "Perc75", "Min", "Max")
SHOW_N	=T Show N=length(METRIC) top left
DIGITS	=2 Number of digits for statistics

Details

MetricBySplit

Value

a data.frame OF CHARACTERS with statistics by the group variable SPLIT

Author(s)

Dirk Hasenclever 2017-03-16, 2017-12-21

Examples

```
## Not run:
Metric<-c(rnorm(57),rnorm(43,2,2))
Split<-c(rep("Arm A",57),rep("Arm B",43))
#
MetricBySplit(Metric,Split)
MetricBySplit(Metric,Split,PVALUE=T,TEST=c("Mean","Median"),
                           SEL=c("Mean","Median"),DIGITS=3)

## End(Not run)
```

MetricPlotBySplit-deprecated
MetricPlotBySplit

Description

Produces plots of a METRIC variables by a SPLIT factor. Available TYPEs are Ecdf, Dens, Boxplot, Stripchart.

Usage

```
MetricPlotBySplit(METRIC, SPLIT = NULL, TYPE = "Ecdf", XLAB = NULL,
  XLIM = NULL, MAIN = NULL, COL = NULL, ADJ = 1, METHOD = "jitter")
```

Arguments

METRIC	is the metric variable to be plotted
SPLIT	=NULL is a factor or convertable to a factor. If NULL a single group ALL is assumed.
TYPE	="Ecdf chooses between "Ecdf", "Dens", "Boxplot", "Stripchart"
XLAB	=NULL optional, otherwise derived from call METRIC
XLIM	=NULL optional, otherwise derived from pretty
MAIN	=NULL plot title
COL	=NULL colours for the levels of Split - otherwise 2:(G+1)
ADJ	=1 Density smoothing parameter relative to density standard
METHOD	="jitter" METHOD for stripchart

Value

Generates a plot.

See Also

OurTools-deprecated

MetricsBySplit — *MetricsBySplit produces a data.frame describing one or many metric variables by a grouping factor SPLIT*

Description

MetricsBySplit produces a data.frame describing one or several metric variables with various statistics SPLIT is a group VARNAME which may be left out. Tests may be preformed on Mean (t.test, oneway.test) or Median (wilcox.test,kruskal.test) MetricsBySplit is a wrapper function calling MetricBySplit for each COLUMN, rbinding the results.

Usage

```
MetricsBySplit(DFR, COLUMNS, SPLIT = NULL, LONGNAMES = NULL,
  PVALUE = F, TEST = NULL, ALL = T, SEL = NULL, DIGITS = 2,
  SHOW_N = F)
```

Arguments

DFR	is a dataframe
COLUMNS	colnames in DFR to be described (at least one)
SPLIT	=NULL split factor specified by a colname in DFR or as vector of fitting length. If NULL an ungrouped description is returned.
LONGNAMES	=NULL alternative names for the COLUMNS
PVALUE	=F If TRUE, tests on mean and Median are performed and added as last column
TEST	=NULL Choose TEST from c("Mean","Median") oce for all or for each COLUMN. If not specified set to both i.e. c("Mean","Median")

ALL	=T If TRUE, also overall statistics
SEL	=NULL if specified selects statistics from c("Nvalid", "Mean", "Sd", "Median", "Mad", "Perc25", "Perc75", "Min", "Max") [once for all columns]
DIGITS	=2 Number of digits for statistics
SHOW_N	=T Show N=length(METRIC) top left

Details

MetricsBySplit

Value

a data.frame OF CHARACTERS with statistics for each COLUMN by the group variableSPLIT

Author(s)

Dirk Hasenclever 2017-03-16, 2017-12-21

See Also

CrossTabs

Examples

```
## Not run:
DFR<-data.frame( Metric1=c(rnorm(57),rnorm(43,2,2)),
                 Metric2=c(rnorm(57,1,2),rnorm(43,20,5)),
                 Split=c(rep("Arm A",57),rep("Arm B",43)) )
MetricsBySplit(DFR,COLUMNS=c("Metric1","Metric2"),SPLIT="Split")
MetricsBySplit(DFR,COLUMNS=c("Metric1","Metric2"),SPLIT=DFR$Split,ALL=F)
MetricsBySplit(DFR,COLUMNS=c("Metric1","Metric2"),SEL=c("Mean","Sd"))

## End(Not run)
```

MissingRateTable *MissingRateTable*

Description

MissingRateTable produces table describing missingness status of variables in a data.frame.

Usage

```
MissingRateTable(DFR, DIGITS = 1, SORT = T, CLASS = F)
```

Arguments

DFR	data.frame to be described
DIGITS	number of digits for percentages
SORT	= T sort table by decreasing missing rate
CLASS	= F show class of variables

Details

MissingRateTable

Value

data.frame showing number of missings and PercentMissing for variables in DFR

Author(s)

Dirk Hasenclever 2017-09-20; 2017-12-05; 2019-02-03

`ModalValue` *ModalValue*

Description

ModalValue identifies the mode of a numeric variable from a density estimate..

Usage

```
ModalValue(METRIC, ADJ = 1)
```

Arguments

`METRIC`	is a numeric variable
`ADJ`	adjust, default=1

Value

Modal value of a numeric variable

Author(s)

Dirk Hasenclever 2019-02-06

Examples

```
METRIC<-c(rt(200,df=12),rnorm(15,4,1),rnorm(200,5,1),11*runif(100))
findmode(METRIC)
findmode(METRIC,ADJ=.2)
```

MoveAfter *MoveAfter*

Description

Reorders columns of a data.frame by moving MOVECOLS in the sequence listed to a position after AFTER

Usage

```
MoveAfter(DFR, MOVECOLS, AFTER)
```

Arguments

DFR	data.frame to be reordered
MOVECOLS	colnames or colnumbers of the block to be moved
AFTER	=NULL colname or colnumber after which the block MOVECOLS should be placed. If NULL MOVECOLS are put in front.

Value

reordered data.frame

Author(s)

Dirk Hasenclever 2017-03-15

Examples

```
## Not run:
a<-c(rep("M",240),rep("F",170))
b<-sample(c(rep("Stage1",110),rep("Stage2",180),rep("Stage3",120)),410)
DFR<-CrossTabs(a,b,TYPE="COL",PVALUE=F,VARNAME=T,LONGNAME="EGON",NOZEROS=T,ALL=NULL)
colnames(DFR)
DFR
MoveAfter(DFR,c("%All","  All"),NULL)
MoveAfter(DFR,c(10,9),1)
MoveAfter(DFR,c("  All","%All"),"  Stage1")

## End(Not run)
```

MultiParameterTimeCourses
MultiParameterTimeCourses

Description

MultiParameterTimeCourses plots courses of multiple metric parameters over a common x-axis, with multiple y-axis.

Usage

```
MultiParameterTimeCourses(DFR, TIME, COURSES, LEFTRIGHT = F,
  SCALES = NULL, XLIM = NULL, YLIMMIN = NULL, YLIMMAX = NULL,
  HLINES = NULL, MAIN = "", XLAB = "Time Axis", YLABS = NULL,
  LWD = 2, LTY = NULL, COLS = 1, PCH = NULL, ...)
```

Arguments

DFR	is a dataframe
TIME	colname in DFR of available time points to be plotted on the x-axis
COURSES	vector of colnames in DFR for metric parameters to be plotted
LEFTRIGHT	= F default is all y-axes on the left, if set to TRUE y-axis are placed altervatingly left and right.
XLIM	= NULL, specify limits of the time axis
YLIMMIN	= NULL, specify lower limits of axis of the the metric parameters, a vector of length COURSES
YLIMMAX	= NULL, specify upper limits of axis of the the metric parameters, a vector of length COURSES
HLINES	= NULL, plot horizontal lines in the respective scales, a vector of length COURSES with NA to leave out.
MAIN	= "" Plot Title
XLAB	= "Time Axis" x-axis label
YLABS	= NULL optional long y-labels for each parameter in COURSES.
LWD	= 2 line width, one of all or a vector of length COURSES
LTY	= NULL line type, defaults to 1:K, else one of all or a vector of length COURSES
COLS	= 1 colours for parameters, one of all or a vector of length COURSES
PCH	= NULL Point type for measured values, one of all or a vector of length COURSES

Value

lower and upper limit of the last plotted y-axis, useful to place points afterwarts.

Author(s)

Dirk Hasenclever 2020-02-26 Thanks to #https://www.r-bloggers.com/multiple-y-axis-in-a-r-plot/

See Also

PlotAxis

Examples

```
## Not run:
TT<-1:10
X1<-1:10 + rnorm(10)
X2<- 10^(1:10 + rnorm(10))
X3<- (1:10 + rnorm(10))^3
X4<- T_Logit((1:10 + rnorm(10))/3,INV=T)
X5<- T_Logit((1:10 + rnorm(10))/3,INV=T)*100

X2[4]<-NA

DFR<-data.frame(TT,X1,X2,X3,X4,X5)

MultiParameterTimeCourses(DFR, TIME="TT", COURSES=c("X1","X2","X3","X4","X5"), LEFTRIGHT = F,
                          YLIMMIN=NULL, YLIMMAX=NULL,
                          SCALES = NULL,
                          HLINES = NULL,
                          YLABS = NULL, LWD = 2, LTY = NULL, COLS = 1:5, PCH = NULL,
                          XLAB = "Time Axis",
                          XLIM = NULL,MAIN = "")

Lim<-MultiParameterTimeCourses(DFR, TIME="TT", COURSES=c("X1","X2","X3","X4","X5"), LEFTRIGHT = T,
                               YLIMMIN=c(0,.1,0,.3,40), YLIMMAX=c(12,1e12,2000,1,100),
                               SCALES = c("ID","LOG10","POWER_Z","LOGIT","LOGIT"),
                               HLINES = c(5,1000,NA,.75,80),
                     YLABS = c("Linear","log10","to the power of 1/3","logit","logit in percent"), LWD = 3, LTY
                               XLAB = "Time Axis [months]",
                               XLIM = NULL, MAIN = "Demo MultiParameterTimeCourses",
                               Z=1/3,PERCENT=c(F,F,F,F,T))
# The y-coordinates of theplots remain defined by the last axis plotted.
# This scale is invisably returned. It can be used to place points in the plot afterwarts.
points(5,Lim[1],bg=2,cex=1.7,pch=21)

## End(Not run)
```

NAasLevel *NAasLevel*

Description

Introduces NA as additional level for a factor or all factors in a data.frame. Missing values are recoded to the new level.

Usage

```
NAasLevel(X, except = NULL, NA_LEVEL = "NA")
```

Arguments

X	is either factorial variable or a data.frame
except	=NULL colnames of X (if data.frame) that should not be transformed
NA_LEVEL	="NA" defines the name of the NA level

Value

the transformed factor or data.frame

Author(s)

Dirk Hasenclever 2017-03-10

See Also

droplevels

Examples

```
## Not run:
# test NAasLevel
df <- data.frame(
  x = factor(c("alpha","beta","alpha",NA), levels=c("alpha","beta","gamma")),
  y = c(5,8,2,1),
  z = factor(c("red","green","green",NA), levels=c("red","green","blue")),
  w = c("EGON","DETLEF","DETLEF","DETLEF")
)
df
NAasLevel(df$x)
NAasLevel(df$y)
NAasLevel(df$w)
NAasLevel(df)
NAasLevel(df,NA_LEVEL="missing") # other NA_LEVEL name
NAasLevel(df,except="z")         # z remains unchanged

## End(Not run)
```

---

nice.hist-deprecated *Histogram with Density*

Description

nice.hist is a tool to examine a metric variable in data inspection, cleaning and description. A histogram with density curve is produced. Location and scale parameter as well as information on valid values can be displayed as legend right beside the plot. If the variable was already log10 transformed a log scale can be plotted. Outliers can be identified and filtered out using a +- factor* mad filer.

Usage

```
nice.hist(x, HISTCOL = 7, DENSCOL = 4, XLAB = NULL, YLAB = "Density",
  NCLASS = 40, ADJ = 1, MAIN = "Histogram with Density", XLIM = NULL,
  XAXP = NULL, DIGITS = 3, LEGEND = F, RUG = T, LOG10 = F,
  MADFILTER = 0, ANALYSISSTRING = "", CUTLINE = NULL)
```

Arguments

x	metric variable
HISTCOL	defines the colour of the histogram, default=7 (yellow)
DENSCOL	defines the colour of the density curve, default=4 (blue). DENSCOL=NA suppresses the density curve,
XLAB	specifies the x-axis-Label, if NULL the variable name of x is extracted from the call using PruneVarName
YLAB	="Density" specifies the y-axis-Label
NCLASS	defines the number of classes (=bins)in the histogram
ADJ	controls the smoothing of the density curve relative to default=1
MAIN	title of plot
XLIM	= NULL sets the x-axis range to be plotted manually - generally not necessary
XAXP	= NULL specifices ticks on x-axis - generally not necessary
DIGITS	controls the numbers of digits shown for location/scale parameters
LEGEND	set to TRUE to get extensive legend at right border instead a simple legend topright
RUG	=T logical whether to plot individual values above x-axis
LOG10	=F If true, x values are on the log10 scale and logarithmic x-axis is plotted
MADFILTER	=0, if >0 (typically 3,4, or 5) filter out Outliers
ANALYSISSTRING	="", optional string to indicate data source and/or date of analysis
CUTLINE	=NULL Draws vertical lines from cut values to the density curve (CAUTION: Later abline may use wrong coordinates)

Value

produces a histogram with density, if MADFILTER > 0 a logical selector vector is returned invisibly useful to examine outliers

See Also

OurTools-deprecated

NiceCumIncPlot *NiceCumIncPlot produces a cumulative incidence curve plot in competing risk analysis*

Description

Cumulative incidence curve plot in competing risk analysis by split groups, including a table describing competing events and group comparison using the proportional subdistribution hazards regression model described in Fine and Gray (1999).

Usage

```
NiceCumIncPlot(TIMES, STATUS, SPLIT = NULL, CENSCODE = 0,
  TARGETCODE = 1, EVENTTABLE = TRUE, COMPARE = T, RATETABLE = T,
  TIMEPOINT = NULL, ORDER = T,
  MAIN = "Cumulative Incidence Function", XLAB = NULL,
  YLAB = "cumulative incidence", XLIM = NULL, YLIM = c(0, 1),
  SCALE = NULL, UNIT = "???", COL = NULL, EVENTNAME = "Event",
  COMPETENAME = "Competing", CEX = 0.8, ...)
```

Arguments

TIMES	time variable in competing risk setting
STATUS	status variable in competing risk setting default 0 = Censored 1= Events of interest, 2= Competing EVENT(s)
SPLIT	=NULL optional split variable
CENSCODE	=0, non-standard code for censored observations
TARGETCODE	=1, non-standard code for Events of interest
EVENTTABLE	= T, show event table
COMPARE	=T, show table with comparisons based on proportional subdistribution hazards regression model. If model does not converge, a warning is given and no table plotted.
RATETABLE	= T, show table with rate estimates at TIMEPOINT and their 95 percent CI. Set to FALSE if TIMEPOINT not specified! CI borders <= 0 and >= 1 are set to NA with warning.
TIMEPOINT	=NULL, time point at which ordering of curves is read off if desired
ORDER	=T, order legends such that legends matched curves from top to bottom
MAIN	="Cumulative Incidence Function" plot title
XLAB	=NULL, x-axis label, default is "Time" with UNITS added in brackets if UNITS specified
YLAB	="cumulative incidence", y-axis label
XLIM	=NULL, range of x-axis - NOTE that for the comparison also Events beyond XLIM are used.
YLIM	=c(0,1), range of y-axis
SCALE	=NULL, unit for tick distance on time axis
UNIT	="???", specify time unit to be included in XLAB, e.g. "days", "Months", "years"
COL	=NULL, specify colours for curves to overwrite the default
EVENTNAME	="Event" colname for Event in Event table
COMPETENAME	="Competing" colname for competing events in Event table
CEX	= .8 letter size for legends
...	further plot parameter passed to graphics::plot. TIP: las=1 makes numbers on y-axis horizontal.

Details

NiceCumIncPlot

Value

produces a plot and invisibly returns the fitted proportional subdistribution hazard model

Author(s)

Dirk Hasenclever 2018-02-10, 2018-08-31, 2018-09-13, 2019-02-05, 2019-02-14, 2020-02-26

Examples

```
## Not run:
 N<-200
 TimeToEvent<-rweibull(N,1)
 Events<-sample(0:2,N,replace=T,prob=c(.3,.5,.2))
 Arm<-sample(LETTERS[1:2],N,replace=T)
 xx<-NiceCumIncPlot(
  TIMES=TimeToEvent,
  STATUS=Events,
  SPLIT=Arm,
  CENSCODE=0,
  TARGETCODE=1,
  EVENTTABLE = F,
  COMPARE = T,
  RATETABLE = T,
  TIMEPOINT=2,
  ORDER=T,
  MAIN="Cumulative Incidence Function",
  XLAB=NULL,
  YLAB="cumulative incidence",
  XLIM=NULL,
  YLIM=c(0,1),
  SCALE=NULL,
  UNIT="years",
  COL=NULL,
  EVENTNAME ="Event",
  COMPETENAME ="Competing",
  CEX=.8,
  las=1)

## End(Not run)
```

`NiceHist` *NiceHist Histogram with Density*

Description

NiceHist is a tool to examine a metric variable in data inspection, cleaning and description. A histogram with density curve is produced. ScaleTransformations: ID, POWER_Z, LOG10, LOGIT are supported. Location and scale parameter as well as information on valid values can be displayed as legend. Outliers can be identified and filtered out using a +- factor* mad filter.

Usage

```
NiceHist(METRIC, NCLASS = 40, ADJ = 1, SCALE = "ID", MADFILTER = 0,
  MAIN = "Histogram with Density", XLAB = NULL, YLAB = "Density",
  XLIM = NULL, NPRETTY = 6, RUG = T, HISTCOL = 7, DENSCOL = 4,
  LEGEND = T, DIGITS = 3, CUTLINE = NULL, ...)
```

Arguments

METRIC	metric variable
NCLASS	= 40 defines the number of classes (=bins) in the histogram
ADJ	= 1controls the smoothing of the density curve relative to default=1
SCALE	="ID" chooses a ScaleTransformation ID, POWER_Z, LOG10, LOGIT for X: use optional parameter Z to specify the power to use with POWER_Z. use optional parameter PERCENT = T for percent instead of proportions on LOGIT scale.
MADFILTER	=0, if >0 (typically 3,4, or 5) filters out Outliers beyond +- MADFILTER* mad
MAIN	= "Histogram with Density" specifies the title of plot
XLAB	specifies the x-axis-Label, if NULL the variable name of x is extracted from the call using PruneVarName
YLAB	="Density" specifies the y-axis-Label
XLIM	= NULL sets the x-axis range to be plotted manually - generally not necessary
NPRETTY	=6 influences approximate number of labels when using ID or POWER_Z (cf. base pretty)
RUG	=T logical whether to plot individual values above x-axis
HISTCOL	defines the colour of the histogram, default=7 (yellow)
DENSCOL	defines the colour of the density curve, default=4 (blue). DENSCOL=NA suppresses the density curve,
LEGEND	= T show legend, which is smartly placed on the side where there is most space.
DIGITS	controls the numbers of digits shown for location/scale parameters in the LEGEND
CUTLINE	=NULL Draws vertical lines from cut values to the density curve
...	further parameters for scale transformations can be provided via ... Z specifies the power to use with POWER_Z when using POWER_Z PERCENT =F Use percent instead of proportions in LOGIT when using LOGIT

Value

data.frame Irregular with 4 variables: Rownr, and 2 logicals TransUndef and Outlier in order to identify irregular values.

Author(s)

Dirk Hasenclever 2017-01-22, 2017-03-22, 2019-02-04, 2019-02-08, 2019-02-12

See Also

hist

Examples

```
## Not run:
Metric<-c(rnorm(200),rnorm(100,3,1))
NiceHist(Metric)
NiceHist(Metric,RUG=F,ADJ=.7)
NiceHist(Metric,LEGEND = F)

# Demo LOG10 scale
NiceHist(exp(Metric),LEGEND = T,SCALE="ID",NCLASS=100)
NiceHist(exp(Metric),LEGEND = T,SCALE="ID",NCLASS=1000,XLIM=c(0,50))
NiceHist(exp(Metric),LEGEND = T,SCALE="LOG10")

# Demo POWER_Z scale
Metric<-rnorm(300,5,1)^3
NiceHist(Metric)
NiceHist(Metric,SCALE="POWER_Z",Z=1/3,ADJ=.8)

# Demo LOGIT scale
Metric<-runif(300)^3
NiceHist(Metric)
NiceHist(Metric,SCALE="LOGIT")
NiceHist(Metric,SCALE="LOGIT",LEFT=T)
NiceHist(100*Metric,SCALE="LOGIT",LEFT=T,PERCENT=T)
NiceHist(100*Metric,SCALE="LOGIT",LEFT=T,PERCENT=T,XLAB="Percent values")

# Dealing with outliers and missing values
Metric<-c(rnorm(200),rnorm(100,3,1),runif(10,5,100),rep(NA,5))
NiceHist(Metric)
Irregular<-NiceHist(Metric,MADFILTER=4)
Metric[Irregular$Outlier]

## End(Not run)
```

NiceSurvPlot *NiceSurvPlot*

Description

Produces a time to event "survival plot" with many optional information outside the plot margins

Usage

```
NiceSurvPlot(Times, Status = NULL, Groups = NULL, ArmLabels = NULL,
  ArmCols = NULL, ByLineType = FALSE, CensTicks = TRUE, Main = "",
  AnalysisString = "", Xlab = NULL, Ylab = "proportion event free",
  Scale = 12, Unit = "months", Ylim = c(0, 1), Timepoint = 12,
  MaxTime = NULL, OrderByPlateau = TRUE, NriskBelow = TRUE,
  Estimates = TRUE, CIplot = FALSE, Medians = FALSE,
  CompareLogrank = TRUE, LHRestimate = NULL, YreferenceLine = NULL,
  XreferenceLine = NULL)
```

Arguments

Times	EITHER Time to event variable of class "Surv" OR time component of time to event
Status	=NULL status component von time to event (needed only if Times is not of class "Surv")
Groups	=NULL grouping variable; if NULL a single curve is plotted with confdence band (set ArmCols=c(1,0,0) to suppress it)
ArmLabels	=NULL, Specify Arm labels - if NULL: levels(Groups) ATTENTION: up to 10 characters!
ArmCols	=NULL, Specify colour code - if NULL a standard is used
ByLineType	=FALSE, black curves distinguished by LineType - ignore ArmCols
CensTicks	=TRUE, show censoring ticks - better set to FALSE to show LineTypes
Main	="", plot title
AnalysisString	="", CharacterString to be plotted bottomright
Xlab	=NULL, set x-axis label manually instead of automatically
Ylab	="proportion event free", Specifiy y axis label
Scale	=12, Time Unit for time axis
Unit	="months", Time Unit name
Ylim	=c(0,1), specify range y-axis, if lower > 0 this is marked on the plot
Timepoint	=12, Time point to sort curves, provide rate estimates, and plot CIs
MaxTime	=NULL, Restrict time axis to MaxTime, later events still count in tests etc
OrderByPlateau	=TRUE, Order legend by decreasing order of curves at timepoint
NriskBelow	=TRUE, Show Nrisk numbers below plot at ticks
Estimates	=TRUE Show legend with rate estimates at timepoint with 95 percent CI
CIplot	= FALSE plot with 95 percent CIs at Timepoint
Medians	=FALSE, show legend with median estimates with 95 percent CI - if(Medians) Estimates set to FALSE
CompareLogrank	=TRUE, compare groups with logrank test and show p.value
LHRestimate	=NULL, Estimate of log hasard ratio Group first mentioned in legend as reference
YreferenceLine	=NULL, Horizontal reference line at (possibly a vector)
XreferenceLine	=NULL, Vertical reference line at (possibly avector)

Value

produces a plot (choose: width:height 3:2) and returns an invisible list with components: missing = selector for missing observations, KMtable Kaplan-Meier table, model summary of Coxmodelfit, Rates Rate estimates at Timepoint, Medians with Median estimates

Author(s)

Dirk Hasenclever 2011-08-30, 2012-10-26, 2017-11-22, 2018-01-18, 2018-02-09

Examples

```
## Not run:
require(MASS)

## Set wide Window to about 3 x 2
NiceSurvPlot(Times=VA$stime,Status=VA$status,Groups=VA$celltype,
            Medians=T,Scale=180,Unit = "Days",Xlab="OSV",Timepoint = 360)

VA$Karnvsky<-factor(VA$Karn)
levels(VA$Karnvsky)<-c("10-40","10-40","10-40","10-40","50-74","50-74",
                       "50-74","50-74","75-100","75-100","75-100","75-100")
NiceSurvPlot(Times=VA$stime,Status=VA$status,Groups=VA$Karnvsky,
             Medians=T,Scale=180,Unit = "Days",Xlab="OSV",Timepoint = 60,
           OrderByPlateau = F, AnalysisString="NiceSurvPlot Demo",Main="OSV by Karnvsky")

NiceSurvPlot(Times=VA$stime,Status=VA$status,Groups=VA$Karnvsky,
             Medians=T,Scale=180,Unit = "Days",Timepoint = 180,OrderByPlateau = T)

NiceSurvPlot(Times=VA$stime,Status=VA$status,Groups=VA$Karnvsky,
             Medians=F,Scale=180,Unit = "Days",Xlab="OSV",Timepoint = 180,
             OrderByPlateau = T, AnalysisString="NiceSurvPlot Demo",ByLineType = T)

NiceSurvPlot(Times=VA$stime,Status=VA$status,Groups=VA$treat,
             Medians=T,Scale=180,Unit = "Days",Xlab="OSV [Days]",Timepoint = 180,
             OrderByPlateau = T, AnalysisString="NiceSurvPlot Demo")
NiceSurvPlot(Times=VA$stime,Status=VA$status,Groups=VA$treat,
             Medians=T,Scale=180,Unit = "Days",Xlab="OSV [Days]",Timepoint = 180,
             OrderByPlateau = T,
             AnalysisString="NiceSurvPlot Demo",ArmLabels = c("standard","test"))

NiceSurvPlot(Times=VA$stime,Status=VA$status,Groups=VA$treat,
             Medians=F,Scale=60,MaxTime = 540,Unit = "Days",Xlab="OSV [Days]",
             Timepoint = 180,OrderByPlateau = T,AnalysisString ="NiceSurvPlot Demo",
             ArmLabels = c("standard","test"),ArmCols = c(4,6))

## End(Not run)
```

`NiceVennDiagram` *NiceVennDiagram draws a Venn diagram for up to five sets and returns a data.frame with pattern frequencies and proportions.*

Description

NiceVennDiagram is a wrapper function for gplots::venn. It draws a Venn diagram and returns a data.frame with pattern frequencies and proportions. The patterns are sorted by combinatorically. Cases with any missing set indicators are discarded with an appropriate warning.

Usage

```
NiceVennDiagram(BINARY_SET_INDICATORS, PLOT = T, SORT_BY_FREQ = F)
```

Arguments

BINARY_SET_INDICATORS	data.frame of binary indicators of class logical or binary numeric 0 / 1
PLOT	= T generate a plot
SORT_BY_FREQ	= F sort patterns by frequency, not combinatorically

Details

NiceVennDiagram

Value

a data.frame with pattern frequencies and proportions

Author(s)

Dirk Hasenclever 2019-02-21

Examples

```
## Not run:
 input.df<-data.frame(fever=sample(0:1,200,replace = T, prob = c(.8,.2)),
                      sweat=sample(0:1,200,replace = T, prob = c(.7,.3)),
                  weightloss=sample(0:1,200,replace = T, prob = c(.75,.25)),
                   nightmare=sample(0:1,200,replace = T, prob = c(.6,.4)))
 input.df[4,4]<-NA    # generate warning
 NiceVennDiagram(input.df, PLOT = F)
 NiceVennDiagram(input.df, SORT_BY_FREQ = T)

## End(Not run)
```

Nmissing *Nmissing*

Description

Nmissing counts the number of missing values in a vector X.

Usage

```
Nmissing(X)
```

Arguments

X	vector possibly containing NAs

Details

Nmissing

Value

number of NA components

Author(s)

Dirk Hasenclever 2019-02-03

Examples

```
## Not run:
Nmissing(c(1,2,NA,NA,5))

## End(Not run)
```

`NoUmlaute` *NoUmlaute eliminates all German Umlaute and sz*

Description

NoUmlaute eliminates all German language specific characters (Umlaute and sz replaced ASCII character combinations) in character and factor vectors preserving the class. The ordering of factor levels is not changed. Applying NoUmlaute to other classes triggers a warning and returns the argument identically. NoUmlaute can be applied to a data.frame! ATTENTION: Do not use apply(DFR,2, NoUmlaute) as it changes all classes to factor.

Usage

```
NoUmlaute(X)
```

Arguments

`X`	character or factor variable or data.frame

Details

NoUmlaute

Value

X with all German language specific characters replaced by ASCI if of class character or factor

Author(s)

Dirk Hasenclever 2019-02-15, 2019-10-28

Nvalid *Nvalid*

Description

Nvalid counts the number of valid values in a vector X.

Usage

```
Nvalid(X)
```

Arguments

X	vector possibly containing NAs

Details

Nvalid

Value

number of nonNA components

Author(s)

Dirk Hasenclever 2017-11-07

Examples

```
## Not run:
Nvalid(c(1,2,NA,NA,5))

## End(Not run)
```

OurTools-deprecated *Deprecated functions in package* **OurTools**.

Description

The functions listed below are deprecated and should no longer be used. When possible, alternative functions with similar functionality are mentioned. These functions still work in order to protect old R-scripts, but will not be developed further. Help pages for deprecated functions are available at `help("<function>-deprecated")`.

Usage

```
DescribeMetricBySplit(DFR, COLUMNS = NULL, SPLIT = NULL,
  LONGNAMES = NULL, SEL = "All", DIGITS = 3, ALL = T,
  FORTRANSPOSITION = F)

MetricPlotBySplit(METRIC, SPLIT = NULL, TYPE = "Ecdf", XLAB = NULL,
  XLIM = NULL, MAIN = NULL, COL = NULL, ADJ = 1,
  METHOD = "jitter")

findmode(x, ADJ = 1)

nice.hist(x, HISTCOL = 7, DENSCOL = 4, XLAB = NULL,
  YLAB = "Density", NCLASS = 40, ADJ = 1,
  MAIN = "Histogram with Density", XLIM = NULL, XAXP = NULL,
  DIGITS = 3, LEGEND = F, RUG = T, LOG10 = F, MADFILTER = 0,
  ANALYSISSTRING = "", CUTLINE = NULL)
```

`DescribeMetricBySplit`

For `DescribeMetricBySplit`, use `MetricsBySplit`.

`MetricPlotBySplit`

For `MetricPlotBySplit`, use `PlotMetricBySplit`.

`findmode`

For `findmode`, use `ModalValue`.

`nice.hist`

For `nice.hist`, use `NiceHist`.

Author(s)

Dirk Hasenclever 2017, 2018-01-31, 2018-09-13, 2019-02-06

Dirk Hasenclever 2017-03-09, 2019-02-06

Dirk Hasenclever 2019-02-06

Dirk Hasenclever 2017-01-22, 2017-03-22, 2019-02-06

See Also

`MetricsBySplit`

`hist`

Examples

```
## Not run:
x<-c(rnorm(100,2,1),rnorm(200,3,1))
y<-c(rnorm(100,5,1),rnorm(200,7,1))
x[sample(1:300,13)]<-NA
gr<-factor(c(rep("Gr1",100),rep("Gr2",200)))
```

```
s<-rep(c(1,2,3),100)
s1<-factor(rep(1,300))
dfr<-data.frame(X=x,Y=y,GR=gr,S=s,S1=s1)

DescribeMetricBySplit(dfr,COLUMNS=c("X","Y"),
              LONGNAMES=c("Variable1","Variable2"),dfr$S1)
DescribeMetricBySplit(dfr,COLUMNS=c("X","Y"),
              LONGNAMES=c("Variable1","Variable2"),DIGITS=1)
DescribeMetricBySplit(dfr,COLUMNS=c("X","Y"),dfr[,"S"])

DescribeMetricBySplit(dfr,COLUMNS=c("X","Y"),dfr$GR,
              SEL=c("MEDIAN","SD","MEAN","NVALID"))

DescribeMetricBySplit(dfr$X,SPLIT=gr,LONGNAMES="Variable1")
DescribeMetricBySplit(dfr$X,COLUMNS=c("X"),gr,LONGNAMES="Variable1")
DescribeMetricBySplit(dfr,COLUMNS=c("X","Y"),SPLIT=gr,
                      LONGNAMES=c("Variable1","Variable2"),SEL="MEDIAN")

## End(Not run)
## Not run:
METRIC<-c(rnorm(100,10,2),rnorm(200,12,4))
SPLIT<-c(rep("Good",100),rep("Bad",100),rep("DEVELISH",100))

MetricPlotBySplit(METRIC,SPLIT,XLAB="test variable",COL=c(2,3,4),TYPE="Ecdf")
MetricPlotBySplit(METRIC,SPLIT,XLAB="test variable",COL=c(2,3,4),TYPE="Dens",
                  ADJ=c(1,.5,2))
MetricPlotBySplit(METRIC,SPLIT,XLAB="test variable",COL=c(2,3,4),TYPE="Boxplot")
MetricPlotBySplit(METRIC,SPLIT,XLAB="test variable",COL=c(2,3,4),
                  TYPE="Stripchart",METHOD="jitter")

## End(Not run)
x<-c(rt(200,df=12),rnorm(15,4,1),rnorm(200,5,1),11*runif(100))
findmode(x)
findmode(x,ADJ=.2)
## Not run:
Metric<-c(rnorm(200),rnorm(100,3,1))
nice.hist(Metric)
nice.hist(Metric,RUG=F)
nice.hist(Metric,LEGEND = T)
nice.hist(Metric,LEGEND = T,LOG10=T)
# Dealing with outliers and missing values
Metric<-c(rnorm(200),rnorm(100,3,1),runif(10,5,100),rep(NA,5))
nice.hist(Metric)
SEL<-nice.hist(Metric,MADFILTER=5)
Metric[SEL]
SEL<-nice.hist(Metric,MADFILTER=5,LEGEND=T,LOG10=F,ADJ=.7)

## End(Not run)
```

PlotAxis *PlotAxis*

Description

Function to plot a transformed axis. Available TYPEs are ID, POWER_Z, LOG10, LOGIT.

Usage

```
PlotAxis(x, AXIS = 1, SCALE = "ID", LIM = NULL, NPRETTY = 8,
  UNIT = NULL, LINE = NA, COL = 1, ...)
```

Arguments

x	is the untransformed metric variable to be plotted
AXIS	=1 Choose which axis 1=x-axis, 2=y-axis
SCALE	="ID" chooses between ID, POWER_Z, LOG10, LOGIT
LIM	=NULL plot interval for axis i.e. xlim or ylim
NPRETTY	= 8 influences approximate number of labels when using ID or POWER_Z (cf. base pretty)
UNIT	=NULL if SCALE ="ID" AND diff(LIM) divisable by UNIT plot interval will have UNIT based ticks
LINE	= NA will be passed to axis as line=LINE
COL	= 1 will be passed to axis as col = COL to colour the axis
...	further parameters for scale transformations can be provided via ... Z specifies the power to use with POWER_Z when using POWER_Z PERCENT =F Use percent instead of proportions in LOGIT when using LOGIT

Value

plots the chosen axis in the currently open plot.

Author(s)

Dirk Hasenclever 2017-03-29, 2018-09-12, 2020-02-15 (LINE parameter)

Examples

```
## Not run:
x<-rnorm(300,10,3)
hist(x,xaxt="n")
PlotAxis(x,AXIS=1,SCALE="ID",NPRETTY=12)
hist(x,xaxt="n")
PlotAxis(x,AXIS=1,SCALE="ID",NPRETTY=3)

x<-rnorm(300,10,6)^(3/2)
hist(x^(2/3),xaxt="n")
PlotAxis(x,AXIS=1,SCALE="POWER_Z", Z=2/3)

x<-exp(rnorm(700,1,1))
hist(log10(x),xaxt="n")
PlotAxis(x,AXIS=1,SCALE="LOG",LIM=NULL,NPRETTY=2)

x<-rbeta(300,12,1)
hist(T_Logit(x),xaxt="n",nclass=40)
PlotAxis(x,AXIS=1,SCALE="LOGIT",NPRETTY=8)
x<-rbeta(300,3,1)*100
hist(T_Logit(x,PERCENT=T),xaxt="n",nclass=40)
PlotAxis(x,AXIS=1,SCALE="LOGIT",NPRETTY=8,PERCENT=T)

## End(Not run)
```

PlotBinaryByMetric *Plot Binary versus Metric*

Description

Basic plot to look at the dependence of a binary variable from a metric one. Scatterplot BINARY verus METRIC. BINARY is jittered to avoid overplotting. Blue line is logistic regression curve. Grey line is lowess smoother. Legend details Nvalid, Baserate and Wald pValue of METRIC in logistic regession.

Usage

```
PlotBinaryByMetric(BINARY, METRIC, BNAME = NULL, MNAME = NULL,
  MAIN = "", SCALE = "ID", XLIM = NULL, NPRETTY = 6, SPAN = 0.75,
  LOWESSCOL = 8, ...)
```

Arguments

BINARY	is a variable of any type with only two values
METRIC	is a metric variable
BNAME	= NULL is an optional label for BINARY
MNAME	= NULL is an optional label for M
MAIN	= "" is an optional title
SCALE	="ID" chooses a ScaleTransformation ID, POWER_Z, LOG10, LOGIT for X: use optional parameter Z to specify the power to use with POWER_Z. use optional parameter PERCENT = T for percent instead of proportions on LOGIT scale.
XLIM	= NULL sets the x-axis range to be plotted manually - generally not necessary
NPRETTY	= 6 influences approximate number of labels when using ID or POWER_Z (cf. base pretty)
SPAN	= 0.75 is the smoothing parameter for lowess
LOWESSCOL	= 8 Colour of Lowess-Smoother. Set to 0 to eliminate the curve

Value

Generates a plot and returns the summary of the logistic regression.

Author(s)

Dirk Hasenclever old; new version 2019-08-09

Examples

```
METRIC<-rnorm(200); BINARY<-(METRIC+rnorm(200))>0; PlotBinaryByMetric(BINARY,METRIC)
```

PlotDepMetrics *PlotDepMetrics*

Description

Produces plots of several dependent METRIC variables on the same scale. The metric variables are specified as COLUMNS of a data.frame DFR Available TYPEs are Ecdf, Dens, Boxplot, Stripchart as in PlotMetricBySplit.

Usage

```
PlotDepMetrics(DFR, COLUMNS, TYPE = "Ecdf", SCALE = "ID",
  MLAB = NULL, SPLITNAMES = NULL, MLIM = NULL, UNIT = NULL,
  MAIN = NULL, NVALID = T, COL = NULL, DRAW_STEPS = F, ADJ = 1,
  METHOD = "jitter", NPRETTY = 8, ...)
```

Arguments

DFR	data frame
COLUMNS	colnames of dependent metric variables to be plotted
TYPE	="Ecdf chooses between "Ecdf", "Dens", "Boxplot", "Stripchart"
SCALE	="ID" chooses a ScaleTransformation ID, POWER_Z, LOG10, LOGIT
MLAB	=NULL optional, otherwise derived from call METRIC
SPLITNAMES	=NULL optional names instead of COLUMNS for metric variables;
MLIM	=NULL optional ON THE ORIGINAL SCALE, otherwise derived from pretty
UNIT	=NULL if SCALE ="ID" AND diff(MLIM) divisable by UNIT plot interval will have UNIT based ticks
MAIN	= NULL plot title via main=
NVALID	=T logical switch whether to show number of valid observations per SPLIT group
COL	=NULL colours for the levels of Split - otherwise 2:(G+1)
DRAW_STEPS	=F Set to TRUE for drawing the ecdf steps in full
ADJ	=1 Density smoothing parameter relative to density standard
METHOD	="jitter" METHOD for stripchart
NPRETTY	=8 influences approximate number of labels when using ID or POWER_Z (cf. base pretty)
...	further parameters for scale transformations can be provided via ... Z specifies the power to use with POWER_Z when using POWER_Z PERCENT =F Use percent instead of proportions in LOGIT when using LOGIT

Value

Generates a plot.

Author(s)

Dirk Hasenclever 2017-03-09; 2017-03-20 with scale options, 2017-04-02, 2018-09-12

Examples

```
## Not run:
x<-rnorm(100)
y<-rnorm(100,1,1)
z<-rnorm(100,2,3)
dfr<-data.frame(x,y,z)
PlotDepMetrics(DFR=dfr,COLUMNS=colnames(dfr),TYPE="Ecdf",SCALE="ID",
      MLAB=NULL,MLIM=NULL,
      MAIN=NULL,COL=NULL,NVALID=T,ADJ=1,METHOD="jitter",NPRETTY=8)
PlotDepMetrics(DFR=dfr,COLUMNS=colnames(dfr),TYPE="Ecdf",SCALE="ID",
      MLAB=NULL,MLIM=NULL,DRAW_STEPS =  T,
      MAIN=NULL,COL=NULL,NVALID=T,ADJ=1,METHOD="jitter",NPRETTY=8)

## End(Not run)
```

PlotMetricBySplit *PlotMetricBySplit*

Description

Produces plots of a METRIC variables by a SPLIT factor. Available TYPEs are Ecdf, Dens, Boxplot, Stripchart.

Usage

```
PlotMetricBySplit(METRIC, SPLIT = NULL, TYPE = "Ecdf", SCALE = "ID",
  MLAB = NULL, MLIM = NULL, UNIT = NULL, MAIN = NULL, COL = NULL,
  DRAW_STEPS = F, ECDF_POINTS = T, NVALID = T, ADJ = 1,
  METHOD = "jitter", NPRETTY = 8, ...)
```

Arguments

METRIC	is the metric variable to be plotted
SPLIT	=NULL is a factor or convertable to a factor. If NULL a single group ALL is assumed.
TYPE	="Ecdf chooses between "Ecdf", "Dens", "Boxplot", "Stripchart"
SCALE	="ID" chooses a ScaleTransformation ID, POWER_Z, LOG10, LOGIT
MLAB	=NULL optional label for METRIC, otherwise derived from call METRIC
MLIM	=NULL optional ON THE ORIGINAL SCALE, otherwise derived from pretty
UNIT	=NULL if SCALE ="ID" AND diff(LIM) divisable by UNIT plot interval will have UNIT based ticks
MAIN	=NULL, specifiy title, if MAIN = NULL type of plot
COL	=NULL colours for the levels of Split - otherwise 2:(G+1)
DRAW_STEPS	=F Set to TRUE for drawing the ecdf steps in full
ECDF_POINTS	=T Set to FALSE to suppress plotting points in ecdf
NVALID	=T logical switch whether to show number of valid observations per SPLIT group
ADJ	=1 Density smoothing parameter relative to density standard

METHOD	="jitter" METHOD for stripchart. Options: "jitter", "stack", "overplot"
NPRETTY	=6 influences approximate number of labels when using ID or POWER_Z (cf. base pretty)
...	further parameters for scale transformations can be provided via ... Z specifies the power to use with POWER_Z when using POWER_Z PERCENT =F Use percent instead of proportions in LOGIT when using LOGIT

Value

Generates a plot.

Author(s)

Dirk Hasenclever 2017-03-09; 2017-03-20 with scale options; 2017-04-02; 2017-12-05; 2018-02-05; 2018-08-13, 2018-10-23, 2019-09-02

Examples

```
## Not run:
METRIC<-c(rnorm(100,10,2),rnorm(200,12,4))
SPLIT<-c(rep("Good",100),rep("Bad",100),rep("DEVELISH",100))

PlotMetricBySplit(METRIC,SPLIT,MLAB="test variable",COL=c(2,3,4),TYPE="Ecdf")
PlotMetricBySplit(METRIC,SPLIT,
COL=c(2,3,4),TYPE="Ecdf",NPRETTY=4,SCALE="ID",DRAW_STEPS = T)
PlotMetricBySplit(METRIC,SPLIT,
            COL=c(2,3,4),TYPE="Ecdf",NPRETTY=4,SCALE="ID",DRAW_STEPS = T,ECDF_POINTS = F)
PlotMetricBySplit(METRIC,SPLIT,MLAB="test variable",COL=c(2,3,4),
                TYPE="Dens",ADJ=c(1,.5,2))
PlotMetricBySplit(METRIC,SPLIT,MLAB="test variable",COL=c(2,3,4),TYPE="Boxplot")
PlotMetricBySplit(METRIC,SPLIT,MLAB="test variable",COL=c(2,3,4),
                TYPE="Stripchart",METHOD="jitter")

## End(Not run)
```

PlotOrdinalCourse *Plot Ordinal Course*

Description

Creates a bar graph of the cumulative maximum CTC GRADE 0:5 in the course of chemotherapy cycles. The ORDLEVELS parameter can be used to select another ordinal variable. The COURSELABEL parameter defines the course variable. The maximum number of bars of the course variable is 16. A frequency table is displayed on the right.

Usage

```
PlotOrdinalCourse(proztab, COL = c(0, 7, 3, 4, 6, 2),
  XLAB = "cumulative maximum CTC grade",
  MAIN = "Barplot course of ordinal", ORDLEVELS = "CTC",
  COURSELABEL = "CY", SHOW_TABLE = TRUE)
```

Arguments

proztab	is a prepared data matrix with cumulative frequencies of ORDLEVELS and COURSLABEL
COL	Colour of Subgroups, default = c(0,7, 3, 4, 6, 2) works for CTC0:4 and CTC 0:5, but can be adjusted
XLAB	="cumulative maximum CTC grade" is default and can be adjusted
MAIN	="Barplot course of ordinal" is default and can be adjusted
ORDLEVELS	="CTC" is default and can be adjusted
COURSELABEL	="CY" is default and can be adjusted
SHOW_TABLE	= TRUE logical switch to supress the table

Value

Generates a stacked barplot with a frequency table.

Author(s)

Dirk Hasenclever Update 2019-04.11

Examples

```
## Not run:
# Simulate CTC 0:4 data
tt<-list( factor(sample(0:4,101,replace=T, prob=c(.2, .3, .4, .08, .02)),levels=paste(0:4)),
        factor(sample(0:4,101,replace=T, prob=c(.1, .2, .5, .1, .1)),levels=paste(0:4)),
       factor(sample(0:4,100,replace=T, prob=c(.1, .15, .4, .2, .15)),levels=paste(0:4)),
      factor(sample(0:4,99,replace=T,  prob=c(.05, .12, .38, .25, .2)),levels=paste(0:4)),
        factor(sample(0:4,100,replace=T, prob=c(.1, .15,.4,.2,.15)),levels=paste(0:4)),
       factor(sample(0:4,99,replace=T,  prob=c(.05, .12,.38,.25,.2)),levels=paste(0:4)))

# Percent Table

ttt<-prop.table(sapply(tt,function(x){table(x)}),2)*100
colnames(ttt)<-paste("CY",1:length(tt),sep="-")
PlotOrdinalCourse(ttt)
PlotOrdinalCourse(ttt,SHOW_TABLE = F)

# Simulate CTC 0:5 data
tt<-list( factor(sample(0:5,101,replace=T, prob=c(.2, .3, .4, .08, .02, .01)),levels=paste(0:5)),
      factor(sample(0:5,101,replace=T, prob=c(.1, .2, .5, .1, .1, .01)),levels=paste(0:5)),
      factor(sample(0:5,100,replace=T, prob=c(.1, .15, .4, .2, .15, .01)),levels=paste(0:5)),
      factor(sample(0:5,99,replace=T,  prob=c(.05, .12, .38, .25, .2, .01)),levels=paste(0:5)),
      factor(sample(0:5,100,replace=T, prob=c(.1, .15,.4,.2,.15,.01)),levels=paste(0:5)),
      factor(sample(0:5,99,replace=T,  prob=c(.05, .12,.38,.25,.2,.05)),levels=paste(0:5)))

# Percent Table

ttt<-prop.table(sapply(tt,function(x){table(x)}),2)*100
colnames(ttt)<-paste("CY",1:length(tt),sep="-")
PlotOrdinalCourse(ttt)

## End(Not run)
```

`Positive_Comparators` *Collection of positive ordinal comparators*

Description

TRUE if condition positively fullfilled, FALSE if not met or if NA!

- EQ equal
- NE unequal
- LT less than
- LE less equal
- GT greater than
- GE greater equal

When x and/or y is a vector, comparison is component-wise. If x and y differ in length the shorter will be recycled.

Usage

```
EQ(x, y)

NE(x, y)

LT(x, y)

LE(x, y)

GT(x, y)

GE(x, y)

LT(x, y)
```

Arguments

x	ordinal
y	ordinal

Value

TRUE or FALSE

Examples

```
EQ(1,1)
EQ(1,NA)
EQ(2,c(1, 2, 3, NA))
EQ(c(2,4,3,1),c(1, 2, 3, NA))
EQ(1:2,c(1, 2, 1, NA))
```

`ProversionProb` *ProversionProb*

Description

ProversionProb calculates a confidence interval for the proversion probability pr(X>Y)+0.5*pr(X=Y) using method 5 of Newcombe RG (2006). Confidence intervals for an effect size measure based on the Mann-Whitney statistic. Part 1: General issues and tail-area based methods. Stat Med 25(4):543-557 Newcombe RG (2006); Confidence intervals for an effect size measure based on the Mann-Whitney statistic. Part 2: Asymptotic methods and evaluation. Stat Med 25(4):559-573

Usage

```
ProversionProb(X, Y, CI_LEVEL = 0.95, DIGITS = 7)
```

Arguments

`X, Y`	metric or at least ordinal numeric
`CI_LEVEL`	=0.95 desired confidence level
`DIGITS`	=7 number of digits

Value

Vector of estimated ProversionProbability with Lower, Upper of CI_LEVEL confidence interval

Author(s)

Dirk Hasenclever 2010-04-28, 2017-12-12, 2018-08-23

Examples

```
## Not run:
X<-rnorm(30,10,1)
Y<-rnorm(30,9.5,1)
PlotMetricBySplit(c(X,Y),c(rep("X",length(X)),rep("Y",length(Y))))
ProversionProb(X,Y,DIGITS=4)

## End(Not run)
```

`PruneVarName` *Prune a deparse(substitute(X)) type VarName*

Description

PruneVarName postprocesses a deparse(substitute(X)) type VarName assigned to an argument variable in a funcion. In a subscripted vector "V[1:3]" the brackets are deleted. In a subscripted data.frame "DF[, \"b\"]" b is extracted. From DF$b b is extracted. A function call around is preserved.

Usage

```
PruneVarName(str)
```

Arguments

str	is a string, typically the result of deparse(substitute(X))

Value

pruned string

Author(s)

Dirk Hasenclever

Examples

```
## Not run:
DF<-data.frame(a=rnorm(10),b=rep(1:2,5))
PruneVarName(VarName)
VarName<-deparse(substitute(LETTERS[1:3]))
PruneVarName(VarName)
VarName<-deparse(substitute(DF$a))
PruneVarName(VarName)
VarName<-deparse(substitute(DF[,"b"]))
PruneVarName(VarName)
VarName<-deparse(substitute(DF[1:2,"b"]))
PruneVarName(VarName)
VarName<-deparse(substitute(DF[,2]))
PruneVarName(VarName)
VarName<-deparse(substitute( log10(DF[,"a"]) )
PruneVarName(VarName)

## End(Not run)
```

PseudoMedian *PseudoMedian*

Description

Calculate the PseudoMedian of a metric variable. The PseudoMedian is the median of all means of pairs of data points. Difference in PseudoMedian underlies the Mann-Whitney-U test.

Usage

```
PseudoMedian(METRIC, DIGITS = 7)
```

Arguments

METRIC	is a metric variable
DIGITS	=7 number of digits

Value

PseudoMedian

Author(s)

Dirk Hasenclever

Examples

```
## Not run:
x<-c(rt(200,df=12),rnorm(15,4,1),rnorm(200,5,1),11*runif(100))
PseudoMedian(x,3)
nice.hist(x,LEGEND=T,CUTLINE=PseudoMedian(x),
   XLAB=paste("x ---> PseudoMedian=",PseudoMedian(x,3)))

## End(Not run)
```

QuantilesSurvivalCurves

QuantilesSurvivalCurves

Description

Read off quantiles (default median) and corresponding CIs from a survfit object.

Usage

```
QuantilesSurvivalCurves(SVOBJECT, PROBS = 0.5, DIGITS = 1,
  SMARTLABEL = F)
```

Arguments

SVOBJECT,	Survival Object as produced by survfit()
PROBS	= 0.5, vector of desired quantiles. CAVE: 0.25 corresponds to S(t)= 1-0.25!
DIGITS	= 1, number of digits in output
SMARTLABEL	= F, if TRUE the group label will be shortened to the part after the "="

Details

QuantilesSurvivalCurves

Value

dataframe with Group Quantile Estimate Lower Upper

Author(s)

Dirk Hasenclever

Examples

```
## Not run:
### Test SurvivalAtT
Times<-c(rexp(100,1),rexp(100,.5))*12
Status<-c(rbinom(100,1,.8),rbinom(100,1,.2))
gr<-sample(c("A","B"),200,replace=T)
mod<-survfit(Surv(Times,Status)~gr)
plot(mod)
QuantilesSurvivalCurves(mod,PROBS=c(.4,.6))
QuantilesSurvivalCurves(mod,PROBS=c(.4,.6),SMARTLABEL = T)
QuantilesSurvivalCurves(mod,SMARTLABEL = T)

## End(Not run)
```

ReconcileLists *ReconcileLists*

Description

used to compare and reconsile two lists.

Usage

```
ReconcileLists(OldDFR, NewDFR, KEYS, ONLYNEW = F)
```

Arguments

OldDFR	is an old list possibly amended by data managment
NewDFR	is a newly created, updated list
KEYS	is a vector of variable names both in OLDDFR and NEWDFR which identify the entries.
ONLYNEW	= F logical switch to only filter new entries

Value

returns an updated list either with onlynew entries (ONLYNEW=T)or with new and still unresolved entries.

Author(s)

Annett Schrock & Dirk Hasenclever

`ScaleTransformations` *Collection of scale Transformations*

Description

Scale Transformation functions with INVerse option, safely dealing with NaN and +-Inf

- T_Power_Z
- T_Log_B
- T_Logit

Usage

```
T_Power_Z(x, INV = FALSE, Z)

T_Log_B(x, INV = FALSE, B = 10)

T_Logit(x, INV = FALSE, PERCENT = FALSE)
```

Arguments

`x`	metric variable
`INV`	=FALSE logical switch between transformation and its inverse
`Z`	Exponent of power transformation (T_Power_Z only)
`B`	=10 base of the logarithm (T_Log_B only)
`PERCENT`	=FALSE logical switch proportions as percent (T_Logit only)

Value

transformed metric

Examples

```
T_Power_Z(2,Z=1/2)
T_Power_Z( T_Power_Z(2,Z=1/2), INV=TRUE,Z=1/2)
T_Log_B(100)
T_Log_B(128,2)
T_Log_B( T_Log_B(128,2), INV=TRUE) # = 1e+07
T_Logit(T_Logit(0.2),INV=TRUE)
T_Logit(T_Logit(0.2),INV=TRUE,PERCENT=TRUE)
T_Logit(0) # NA (not -Inf because used for PlotAxis)
```

ScatterPlot *ScatterPlot*

Description

ScatterPlot produces a scatterplot including legend of valid values, pearson correlation coefficient with confidence interval, outlier detection including option to exclude outlier, principal axis and_or regression lines Y~X and X~Y shown, subgroup (SPLIT) shown by colour and_or point type together with optional partial correlation coefficient given the subgroup information.

Usage

```
ScatterPlot(X, Y, SCALE_X = "ID", SCALE_Y = "ID", SPLIT = NULL,
  COL = 4, PCH = 19, LINECOL = 4, CONF.LEVEL = 0.95, XLAB = NULL,
  YLAB = NULL, MAIN = NULL, OUTLIER = 4, NO_OUTLIERS = F,
  SHOW_COR = T, DIGITS = 3, LINE_THRESHOLD = 0.3, XY_LINE = T,
  Y_ON_X = F, X_ON_Y = F, SUBLINES = F, MEANPOINT = F,
  PARCOR = T, ...)
```

Arguments

X	metric variable for x-axis
Y	metric variable for y-axis
SCALE_X	="ID" chooses a ScaleTransformation ID, POWER_Z, LOG10, LOGIT for X: use optional parameter ZX to specify the power to use with POWER_Z. use optional parameter PERCENT_X = T for percent instead of proportions on LOGIT scale.
SCALE_Y	="ID" chooses a ScaleTransformation ID, POWER_Z, LOG10, LOGIT for X: use optional parameter ZY to specify the power to use with POWER_Z. use optional parameter PERCENT_Y = T for percent instead of proportions on LOGIT scale.
SPLIT	=NULL split analysis by subgrouping factor
COL	=4 colour of points. For subgroups, specifiy COL as vector with length=nlevel(SPLIT), else 1:nlevel(SPLIT)
PCH	=19 point type(s). For subgroups, specifiy PCH as vector with length=nlevel(SPLIT), else 1:nlevel(SPLIT)
LINECOL	=4 colour of SymLine or plotted line if only one line is plotted
CONF.LEVEL	=0.95 confidence level for CI of correlation coefficient
XLAB	=NULL optional label for X, otherwise derived from call X
YLAB	=NULL optional label for Y, otherwise derived from call Y
MAIN	=NULL specifiy title, if MAIN = NULL no title
OUTLIER	=4 outlier are identified using lmRob from package robust a points with |residue| > OUTLIER * robustSd. set OUTLIER=Inf if no outliers should be highlighted.
NO_OUTLIERS	=F set to TRUE to eliminate outliers from the plot and the analysis
SHOW_COR	=T show legend with correlation corefficient and its CI
DIGITS	=3 Number of digits for correlation coefficients

LINE_THRESHOLD	=.3 only show lines iff |cor| >= LINE_THRESHOLD
XY_LINE	=T As a default the SymLine, i.e. the principal componant axis (symmetrical in X and Y!) is plotted. This line minimises the sum of the squares of the Euclidian distances between points and line.
Y_ON_X	=F Plot the usual but asymmetric regression line Y~X. This line minimises the sum of the squares of the Euclidian distances between points and line parallel to the Y-axis.
X_ON_Y	=F Plot the regression line X~Y. This line minimises the sum of the squares of the Euclidian distances between points and line parallel to the X-axis.
SUBLINES	=F If TRUE SymLines for subgroups defined by SPLIT are plotted instead of any overall line.
MEANPOINT	=F Option to plot the overall mean within the plot.
PARCOR	=T As a default, the partial correlation pcor(x,y |SPLIT) is given in the legend reporting correlation.
...	further parameters for scale transformations can be provided via ... ZX, ZY specifies the power to use with POWER_Z when using POWER_Z PERCENT =F Use percent instead of proportions in LOGIT when using LOGIT

Value

data.frame with the index and the X,Y values of the identified outliers if SPLIT=NULL or SUBLINES=F. If SUBLINES=T and more than one subgroup a data.frame is returned with all pairwise comparisons of subgroup correlation coefficients.

Author(s)

Dirk Hasenclever 2018-01-03, 2018-08-13, 2019-03-12

See Also

SymLine for adding a symetric line, CorDiffCI for confidence intervals for difference of correlation coefficients.

Examples

```
## Not run:
x <- rnorm(100, 10, 2)
y <- rnorm(100, 100, 4) + 1.7 * x

ScatterPlot(x, y, PCH = 2)

x <- c(x, 10 + 0.2 * rnorm(20))
y <- c(y, 170 + rnorm(20))
x[1] <- NA

# check Outlier detection and listing

ScatterPlot(x, y, NO_OUTLIERS = F)
ScatterPlot(x, y, NO_OUTLIERS = F, OUTLIER = 2)
ScatterPlot(x, y, NO_OUTLIERS = T, MAIN = "Example with outliers removed")

# Check SPLIT
```

```
x <- c(rnorm(50), rnorm(50)+5)
y <- c(rnorm(50), rnorm(50)+5)
Split <- factor(paste("Group", rep(LETTERS[1:2], each = 50), sep = "-"))
ScatterPlot(x, y, SPLIT = Split)
ScatterPlot(x, y, SPLIT = Split, SUBLINES=T)

x <- c(rnorm(50), rnorm(50, 3))
y <- c(rnorm(50), rnorm(50, 3) + 5 * x[51:100]) * 2
Split <- factor(rep(LETTERS[1:2], each = 50))
ScatterPlot(x, y, SPLIT = Split, CONF.LEVEL = 0.99)

ScatterPlot(x, y, SPLIT = Split, CONF.LEVEL = 0.99,
               SUBLINES=T, COL=c(4,6), PCH=c(19,21))

## End(Not run)
```

SuccessRate *SuccessRate*

Description

SuccessRate calculates the Wilson-confidence interval for a sucess-rate (using prop.test).

Usage

```
SuccessRate(FAC, SUCCESS = 2, CONF.LEVEL = 0.95, DIGITS = 1)
```

Arguments

FAC	is a factor. Will be converted to a factor if not a factor.
SUCCESS	=2, factor level that defines success. If numeric, the respective level of FAC is assumed.
CONF.LEVEL	=0.95, desired confidence level
DIGITS	=7 number of digits

Value

data.frame with SuccessCategory, Number of Successes, Nvalid, Rate estimate in procent and lower and upper of the CONF.LEVEL confidence interval.

Author(s)

Dirk Hasenclever, 2017-12-12

Examples

```
## Not run:
SuccessRate(rbinom(453,1,0.33))
x<-sample(c("Male", "Female","Inter"),453,replace=T, prob = c(.503,.494,.003))
x<-factor(x,levels = c("Male", "Female","Inter"))
table(x)
SuccessRate(x,DIGITS=2,SUCCESS=3)
SuccessRate(x,DIGITS=3,SUCCESS=3,CONF.LEVEL=.9)
```

```
## End(Not run)
```

SurvivalAtT *SurvivalAtT*

Description

Read off survival rates and corresponding CIs at time T from a survfit object.

Usage

```
SurvivalAtT(TIMEPOINT, SVOBJECT, DIGITS = 7)
```

Arguments

TIMEPOINT	is the time point at which to read off the Kaplan Meier Kurves is a factorial variable
SVOBJECT,	Survival Object as produced by survfit()
DIGITS	=4, number of digits in output

Details

SurvivalAtT

Value

dataframe with SVestimate, SE, lower95.CI, upper95.CI for each arm/strata

Author(s)

Dirk Hasenclever, 2018-02-26

Examples

```
## Not run:
### Test SurvivalAtT
Times<-c(rexp(100,1),rexp(100,.5))*12
Status<-c(rbinom(100,1,.8),rbinom(100,1,.2))
gr<-sample(c("A","B"),200,replace=T)
mod<-survival::survfit(survival::Surv(Times,Status)~gr)
plot(mod)
SurvivalAtT(40,mod)
mod<-survival::survfit(survival::Surv(Times,Status)~1)
SurvivalAtT(40,mod)

## End(Not run)
```

SymLine *SymLine*

Description

SymLine ADDs a symmetric line, i.e. the principal component axis to an open plot. This line minimises the sum of the squares of the Euclidian distances between points and line.

Usage

```
SymLine(X, Y, COL = 4, LWD = 2, LTY = 1, LINE_THRESHOLD = 0.3)
```

Arguments

X	metric variable for x-axis
Y	metric variable for y-axis
COL	=4 line colour
LWD	=2 line width
LTY	=1 line type
LINE_THRESHOLD	=.3 only show lines iff |cor| >= LINE_THRESHOLD

Value

nothing

Author(s)

Dirk Hasenclever 2018-01-03

Examples

```
## Not run:
x <- rnorm(100, 10, 2)
y <- rnorm(100, 100, 4) + 1.7 * x
graphics::plot(x, y)
SymLine(x,y)

## End(Not run)
```

Table *Table*

Description

Shortcut for table with addmargins and default useNA=T. As with base::table more than two factors possible

Usage

```
Table(X, ..., USENA = T)
```

Arguments

X	first variable to table
...	further variables to be tabled
USENA	=T, treat NA as category "NA", must be named to switch it off!!

Details

Table

Value

a frequency array (NOT a data.frame)

Author(s)

Dirk Hasenclever 2017-11-07, 2018-01-31

Examples

```
## Not run:
x<-rep(LETTERS[1:5],each=20)
y<-rep(letters[1:5],20)
y[1:2]<-NA
z<-sample(1:5,100,replace=T)
z[17]<-NA
Table(x,y)
# Table(x,y,F)
# Does not work because of ... higher dimensional arrays are allowed
Table(x,y,USENA=F)
Table(x,y,z,USENA=T) # 3-dimensional array

## End(Not run)
```

TrimStr *TrimStr removes leading or trailing blanks within a string*

Description

Removes spaces within a string variable with leading or trailing blanks. Blanks inbetween are preserved.

Usage

```
TrimStr(str)
```

Arguments

str	is a string variable

Value

string variable without leading or trailing blanks

Author(s)

Dirk Hasenclever 2017-03-22

Examples

```
## Not run:
test TrimStr
str<-"   Apollo 13 "
str<-" up to date         "
TrimStr(str)

## End(Not run)
```

UnivariateCoxTable *UnivariateCoxTable*

Description

Produces a table with univariate results of COX models for various covariates.

Usage

```
UnivariateCoxTable(DFR, COLUMNS, TIMES, STATUS, SORT = T, DIGITS = 3)
```

Arguments

DFR	data.frame containing the time to event and the covariates
COLUMNS	vector of covariate names in DFR
TIMES	EITHER Time to event variable of class "Surv" OR time component of time to event
STATUS	status component von time to event (needed only if TIMES is not of class "Surv")
SORT	=T, sort by p.value?
DIGITS	=3, round to DIGITS

Details

UnivariateCoxTable

Value

data.frame with univariate results

Author(s)

Dirk Hasenclever 2017-09-20, 2017-12-20

Examples

```
## Not run:
require(MASS)
UnivariateCoxTable(VA,"stime","status",
                   COLUMNS=c("treat","age","Karn","prior"))
UnivariateCoxTable(VA,"stime","status",
                   COLUMNS=c("treat","age","Karn","prior"),SORT=F,DIGITS=2)

## End(Not run)
```

VisualiseCoxModel *VisualiseCoxModel*

Description

VisualiseCoxModel produces a plot of time to event curves for specified scenario values of covariates in a COX models.

Usage

```
VisualiseCoxModel(DFR, SURV, SCENARIOS, COMBINE = F, LEGEND = T,
  STAT = "MEDIAN", TIMEPOINT = NULL, SORT_CURVES = T, LINE = T,
  MAIN = NULL, XLAB = NULL, YLAB = NULL, XLIM = NULL,
  XAXP = NULL, YLIM = c(0, 1), YAXP = c(0, 1, 10), LWD = 1,
  UNIT = NULL, SHORTNAMES = NULL, RATIO = 1/3, XPAD = 0.2)
```

Arguments

DFR	data.frame containing the time to event variable (as Surv object) and the covariates named in the columns of the SCENARIOS data frame.
SURV	name of the Surv object of the time to event endpoint in DFR
SCENARIOS	a named list or data.frame specifing values of the covariate for which curves are to be plotted. Names MUST correspond to Colnames of DFR. The Cox model is SURV ~ names(SCENARIOS)
COMBINE	=F, if TRUE curves are plotted for ALL combinations of values specified in columns of SCENARIOS. If FALSE only scenarios corresponding to rows in SCENARIOS are plotted. In this case SCENARIOSmust be a data.frame or a list the same length components.
LEGEND	=T, show various statistics on the right outside the plot.
STAT	="MEDIAN", show a table with MEDIANs or RATEs for all curves plotted. For RATEs TIMEPOINT must be specified.
TIMEPOINT	=NULL, time point at which the rates are read off.
SORT_CURVES	=T, sort the medians respectively rates in the legend table?
LINE	=T, dotted horizontal line at 50 percent for MEDIAN or dotted vertical line to show the RATEs time point
MAIN	=NULL, plot title default is Visualisation COX model
XLAB	=NULL, x-axis label, default is "Time" with UNITS added in brackets if UNITS specified
YLAB	=NULL, y-axis label, default is "Proportion event free"
XLIM	=NULL, range of x-axis
XAXP	=NULL, control on ticks on x-axis
YLIM	=c(0,1), range of y-axis
YAXP	=c(0,1,10), ticks on y-axis
LWD	=1, line width for time to event curves
UNIT	=NULL, specify time unit to beincluded in XLAB, e.g. "days", "Months", "years"
SHORTNAMES	=NULL, provide alternative names for the covariates in SCENARIOS to be used in legend tables.
RATIO	=1/3, control the width ratio bewteen the plot and the legend.
XPAD	=0.2, control distance between columns of the legend table

Value

produces a plot and returns dataframe with MEDIAN respectively RATE estimates

Author(s)

Dirk Hasenclever 2017-12-20, 2017-12-30

Examples

```
## Not run:
DAT<-ovarian
DAT[,"SURV"]<-Surv(DAT[,"futime"], DAT[,"fustat"])
SCENARIOS<-list(age=c(60,70),rx=c(1,2),ecog.ps=c(1.2))
VisualiseCoxModel(DAT,"SURV",SCENARIOS,LEGEND=T)
VisualiseCoxModel(DAT,"SURV",SCENARIOS,COMBINE=T,LEGEND=F)
## make a data.frame with all combinations to be used with COMBINE=F
SCENARIOS<-data.frame(age=c(60,70),rx=c(1,2),ecog.ps=c(1,2))
VisualiseCoxModel(DAT,"SURV",SCENARIOS,COMBINE=F,LEGEND=T)
VisualiseCoxModel(DAT,"SURV",SCENARIOS,COMBINE=T,LEGEND=T,RATIO=1/2,
                  XLIM=c(0,800),XAXP=c(0,1000,10))
SCENARIOS<-list(rx=c(1,2),age=c(50,60,65,70),ecog.ps=c(1,2))
VisualiseCoxModel(DAT,"SURV",SCENARIOS,COMBINE=F,LEGEND=T,UNIT="days")
VisualiseCoxModel(DAT,"SURV",SCENARIOS,COMBINE=T,LEGEND=T,LWD=3,
                  STAT="RATE",TIMEPOINT=400,YLIM=c(0.3,1),YAXP=c(.3,1,7))
SCENARIOS<-list(age=seq(40,80,5))
VisualiseCoxModel(DAT,"SURV",SCENARIOS,COMBINE=F,LEGEND=T)
VisualiseCoxModel(DAT,"SURV",SCENARIOS,COMBINE=F,LEGEND=T,
                  STAT="RATE",TIMEPOINT=400)

## End(Not run)
```

Index