

Package ‘OurTools’

August 14, 2025

Title OurTools for Structured Statistical Analysis

Version 0.61

Date 2025-08-14

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 knitted in analysis reports.

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

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

Depends R (>= 3.5.0)

Imports

graphics, stats, utils, boot, visreg, survival, robust, cmprsk, gplots, glue, gtsummary, nlme, metafor

License MIT + file LICENSE

EnableCodeIndexing Yes

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Contents

AlluvialPlot	3
BinaryAssociation	4
Colnames	5
CompareMetricBySplit	6
CompareSuccessRatesBySplit	7
ConfusionTableStatistics	8
CorDiffCI	9
CountBarplot	10
CountValue	11
Cox2Df	12
CrossTabs	12

CumIncPlot	14
DataMaturityPlot	17
Dates2Character	18
DescribeMetricBySplit-deprecated	19
Df2Csv	20
Df2Rdata	21
FacTable	22
FactorsBySplit	23
findmode-deprecated	24
FollowUpPlot	25
Glm2Df	26
HLProgFactors	27
Is.Outlier	28
Level2NA	29
Levels	30
LhrPlot	30
LocationScaleEstimates	32
MakeCIstring	33
MakeInteractionFactor	33
MartingalePlot	34
Matrix2Df	36
MergeWithMaster	37
MetricBySplit	38
MetricPlotBySplit-deprecated	39
MetricsBySplit	40
MissingRateTable	42
ModalValue	42
MoveAfter	43
MultiParameterTimeCourses	44
MultipleEndpointComparisonCox	46
MultipleEndpointComparisonLogist	47
NAasLevel	48
nice.hist-deprecated	49
NiceCumIncPlot	50
NiceHist	53
NicePvalue	55
NiceSurvPlot	56
NiceVennDiagram	58
Nmissing	59
NoUmlaute	60
Nvalid	61
OneGroupTimeCourse	61
OurTools-deprecated	63
PlotAxis	66
PlotBinaryByMetric	67
PlotDepMetrics	69
PlotEndpointForestCox	70
PlotEndpointForestLogist	71
PlotMetricBySplit	72
PlotOrdinalCourse	74
PlotSubgroupForestCox	76
PlotSubgroupForestLogist	77

Positive_Comparators	78
Positive_IsInInterval_Checkfunctions	79
ProversionProb	79
PruneVarName	80
PseudoMedian	81
QuantilesSurvivalCurves	82
RandomDate	83
ReconcileLists	84
ScaleTransformations	84
ScatterPlot	85
SetUpAnalysisDirs	88
SetUpDir	89
SlidingWindow	89
Source	90
SubgroupAnalysisCox	91
SubgroupAnalysisLogist	92
SuccessRate	93
SurvivalAtT	94
SurvivalRateVersusTargetPvalue	95
SurvPlot	96
SymLine	98
Table	99
ToInitialUpper	100
TrimStr	101
UnivariateCoxTable	102
VariablesBySplit	103
VisualiseCoxModel	104

Index**107**

AlluvialPlot

*AlluvialPlot: Illustrate change between to factors***Description**

AlluvialPlot: Illustrate change between to factors

Usage

```

AlluvialPlot(
  FAC1,
  FAC2,
  FAC1NAME = NULL,
  FAC2NAME = NULL,
  ADD_N = TRUE,
  COL = NULL,
  BORDER = "white",
  CEX = 1.5,
  CEX.AXIS = NULL,
  XW = 0.2,
  ALPHA = 0.8
)

```

Arguments

FAC1	Factor variable initial states
FAC2	Factor variable final states
FAC1NAME	= NULL specify, else derived from variable name
FAC2NAME	= NULL specify, else derived from variable name
ADD_N	= TRUE Should the marginal distribution counts be shown?
COL	= NULL colour specification. If NULL set to default.
BORDER	= "white" set other colour e.g. "black" or NA to suppress
CEX	= 1.5 Size of category labels
CEX.AXIS	= NULL Size of variable labels if NULL = CEX
XW	= .2 numeric, the distance from the set axis to the control points of the xspline
ALPHA	= .8 numeric, vector of transparency of the stripes

Value

Table of FAC1 versus FAC2 after generating an AlluvialPlot

Examples

```
N=100
s1 = letters[sample(size = N, x = 1:3, replace = T)]
s2 = LETTERS[sample(size = N, x = 1:4, replace = T)]
AlluvialPlot(s1, s2)
```

BinaryAssociation

BinaryAssociation

Description

BinaryAssociation produces a data.frame 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 discarded)
Y	=NULL only used if 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)
```

Colnames	<i>Colnames</i>
----------	-----------------

Description

Colnames can be used as base::colnames, but displays the colnames in the console with a comma after the name such that parts of can be directly copied for subscripting on variables in matrices or data.frames.

Usage

```
Colnames(TT, SORT = F)
```

Arguments

TT	is a any object to which colnames() can be applied
SORT	= F, set to T to get the colnames sorted alphabetically

Value

data.frame of one row with colnames in hyphens with attached comma

Author(s)

Dirk Hasenclever 2023-04-21, 2023-08-17

See Also

[Levels](#)

Examples

```
## Not run:
TT <- matrix(1:100, ncol=10)
colnames(TT)<-LETTERS[1:10]
colnames(TT)
Colnames(TT)

## 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: probability difference, odds ratio and relative risk with CI; more than two groups: `chisq.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, Concordances 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	<i>CorDiffCI</i>
-----------	------------------

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,
  LAS = 1
)
```

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)
LAS	= 1 orientation of numbers on y-axis.

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, 2023-11-24

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	<i>CountValue</i>
------------	-------------------

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, 2024-09-26, 2025-04-15

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,
  TEST = "chisq",
```

```

    TESTUSED = NULL ,
    VARNAME = F ,
    LONGNAME = NULL ,
    ALL = NULL ,
    NTOTAL = T
  )

```

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 TEST
TEST	= "chisq", alternatives: "fisher" or "auto" = Fisher only of chisqr gives warning
TESTUSED	= NULL set to TRUE or FALSE to get OR get rid of the column indicating the TestUsed. If NULL, the TestUsed column will only be added to the result if not only a chisq Test was used in order to preserve downward compatibility
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
NTOTAL	= T, set to FALSE to surpress Ntotal in colnames(res)[1]

Value

CrossTable as data.frame with Counts and Percentages intercalated.

Author(s)

Dirk Hasenclever 2017-03-08; 2017-03-14, 2018-01-05, 2024-05-22, 2024-05-30

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)

```

CumIncPlot

CumIncPlot 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

```
CumIncPlot(
  TIMES,
  STATUS,
  SPLIT = NULL,
  CENSCODE = 0,
  TARGETCODE = 1,
  EVENTTABLE = TRUE,
  RATETABLE = T,
  COMPARE = T,
  TIMEPOINT = NULL,
  ORDER = T,
  MAIN = "Cumulative Incidence Function",
  XLAB = NULL,
  YLAB = "cumulative incidence",
  MAXTIME = NULL,
  XLIM = NULL,
  UNIT = "???",
  GRANULARITY = NULL,
  YLIM = c(0, 1),
  PROP = TRUE,
  COL = NULL,
  EVENTNAME = "Event",
  COMPETENAME = "Competing",
  CEX = 1.3,
  CEX.LEGEND = 0.8,
  DIGITS_RATES = 3,
  DIGITS_HR = 2,
  LAS = 1,
  LWD = 3,
  EVENTPOX = 0,
  EVENTPOY = 1,
  RATEPOX = 0,
  RATEPOY = 0.6,
  HRPOX = 1,
  HRPOY = 1,
  LABELPOX = NULL,
  LABELPOY = NULL,
  ...
)
```

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
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.
COMPARE	=T, show table with comparisons based on proportional sub-distribution hazards regression model. If model does not converge, a warning is given and no table plotted.
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
MAXTIME	= NULL, if specified data will be administratively censored at MAXTIME and later events ignored.
XLIM	= NULL, range of x-axis - NOTE that for the comparison also Events beyond XLIM are used, unless MAXTIME is used.
UNIT	= "???", specify time unit to be included in XLAB, e.g. "days", "Months", "years"
GRANULARITY	= NULL, granularity unit for tick distance on time axis
YLIM	= c(0,1), range of y-axis, always between 0 and 1
PROP	= TRUE set to FALSE, if you want the y-axis in percent
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	= 1.3 letter size for axis, labels and title
CEX.LEGEND	= .8 letter size for legends and tables
DIGITS_RATES	= 2 Display DIGITS_RATES ciphers in total in Rate estimates
DIGITS_HR	= 2 Round to DIGITS_HR in Hazard ratio table
LAS	= 1 orientation for y-axis
LWD	= 3 line widths for curves
EVENTPOX	= 0, X Position of Eventtable relative to XLIM: default is top left
EVENTPOY	= 1, Y Position of Eventtable relative to YLIM: default is top left
RATEPOX	= 0, X Position of Ratetable relative to XLIM: default is left below Eventtable
RATEPOY	= .6, Y Position of Ratetable relative to XLIM: default is left below Eventtable

HRPOSX	= 1, X Position of COMPARE HRtable relative to XLIM: default is top right
HRPOSY	= 1, Y Position of COMPARE HRtable relative to YLIM: default is top right
LABELPOSX	= NULL, X Position of legend with Labels relative to XLIM: default is bottom right
LABELPOSY	= NULL, Y Position of legend with Labels relative to YLIM: default is bottom right
...	- do not use: only to deal with obsolete parameters from older versions

Details

CumIncPlot

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, 2020-07-16, 2022-07-19, 2022-07-26, 2023-05-16, 2024-07-19, 2025-07-27

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<-CumIncPlot(
  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),
  GRANULARITY=NULL,
  UNIT="years",
  COL=NULL,
  EVENTNAME ="Event",
  COMPETENAME ="Competing",
  CEX=1.3,
)

## End(Not run)
```

DataMaturityPlot	<i>DataMaturityPlot</i>
------------------	-------------------------

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("darkblue", "red"),
  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("darkblue","red"), 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, 2024-02-14, 2025-06-27

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)
```

Dates2Character	<i>Dates2Character</i>
-----------------	------------------------

Description

Dates2Character transforms all dates in DFR to characters. This is useful when exporting a data.frame to excel in order to avoid excel internal seemingly intelligent transformations.

Usage

Dates2Character(DFR)

Arguments

DFR	data.frame (if DFR has no dimensions, it is temporarily transformed into a data.frame and unlisted before returning)
-----	--

Value

DFR with all columns with class "Date" or "POSIXct" transformed to character

Author(s)

Dirk Hasenclever 2024-12-14, 2024-12-17

See Also[Colnames](#)**Examples**

```
## Not run:
Date1<-as.Date("2024-12-14")
class(Date1)
Date2<-as.POSIXct("2024-12-14")
class(Date2)

Dfr<-data.frame(Date1, Date2,Z=123)
class(Dfr$Date2)

Dfr
Dfr_C<-Dates2Character(Dfr)
class(Dfr_C$Date2)

## 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,
  FORTRANSPPOSITION = 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

[ourPkg-deprecated](#)

Df2Csv	<i>Df2Csv writes a data.frame to a csv file</i>
--------	---

Description

Df2Csv writes a data.frame to a csv file such that it can be opened 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	<i>Df2Rdata</i>
----------	-----------------

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, DFR is saved there!

Usage

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

Arguments

DFR	a dataframe or table generated with gtsummary
DFNAME	is Filename (without the .Rdata !!)
ROW.NAMES	= TRUE is default
RESULTDIR	= T Save in ResultDir if character-variable ResultDir defined in global environment

Value

nothing

Author(s)

Dirk Hasenclever 2017-03-22, 2018-01-18, 2021-07-12, 2024-12-13

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	<i>produces a data.frame describing one or many factor variables by a grouping factor SPLIT</i>
----------------	---

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,
  TEST = "chisq",
  TESTUSED = NULL,
  NOZEROS = T,
  ALL = NULL,
  NTOTAL = T,
  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	= F p-value from TEST, can be a logical vector of length COLUMNS
TEST	= "chisq" alternatives: "fisher" or "auto" = Fisher only of chisqr gives warning can be a vector of length COLUMNS
TESTUSED	= NULL set to TRUE or FALSE to get OR get rid of the column indicating the TestUsed. If NULL, the TestUsed column will only be added to the result if not only a chisq Test was used in order to preserve downward compatibility
NOZEROS	eliminates factor levels that were not observed, TRUE is default
ALL	= NULL If ALL=NULL 'All' columns on the right, if ALL=T on the left and if ALL=F omitted.
NTOTAL	= T, set to FALSE to suppress Ntotal in colnames(res)[1]
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, 2024-05-23, 2024-05-29

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[ourPkg-deprecated](#)

FollowUpPlot	<i>FollowUpPlot</i>
--------------	---------------------

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("darkblue", "red"),
  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 appended at XLAB if not NULL or blank. In addition, 'months', 'Months', 'days', 'Days' are recognised and used to find appropriate ticks.
COLS	=c("darkblue", "red"), 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, 2023-02-23, 2024-02-14, 2025-06-27

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	<i>Glm2Df</i>
--------	---------------

Description

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

Usage

```
Glm2Df(MOD, DIGITS = 3, SCALE = "ID", ...)
```

Arguments

MOD	fit of a glm or lm regression Model (without random effects)
DIGITS	=3 round to DIGITS
SCALE	= "ID", if SCALE != "ID" the table will additionally have back-transformed columns.
...	further parameters for scale transformations can be provided via ... Z specifies the power to use with POWER_Z when using POWER_Z PERCENT = F Set to T to use percent instead of proportions in LOGIT when using LOGIT

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, 2022-08-09

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	<i>Data form the International Prognostic Factor Project in advanced Hodgkin Lymphoma.</i>
---------------	--

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>

Is.Outlier	<i>Is.Outlier</i>
------------	-------------------

Description

Is.Outlier is a function to identify outliers in a metric variable. Outliers are identified using the median +/- factor* MADFILTER approach on a specified SCALE. The resulting logical vector can be used to eliminate or to describe outliers in a vector. NOTE: A positive logic is applied: NAs are no outliers per definition with FALSE returned

Usage

Is.Outlier(METRIC, MADFILTER = 4, SCALE = "ID", ...)

Arguments

METRIC	metric variable
MADFILTER	= 4, (typically 3,4, or 5) filters out outliers beyond median +/- MADFILTER* mad If MADFILTER <= 0 or if the mad estimate == 0 no outlier detection is performed.
SCALE	="ID" chooses a scale transformation 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.
...	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

logical vector for outlier status TRUE or FALSE. FALSE for NAs or if MADFILTER <=0 or Mad estimate == 0

Author(s)

Dirk Hasenclever 2023-08-17, 2024-11-22, 2024-11-25

Examples

```
## Not run:
Demo ID scale
TestX<-c(rnorm(100,1),5,7,10,100,1000, NA)
Is.Outlier(TestX)
Is.Outlier(TestX,SCALE = "LOG10")
Is.Outlier(abs(TestX),SCALE = "LOG10")
Is.Outlier(abs(TestX),SCALE = "LOG10")
TestX<-c(runif(100,.2,.5),.7,.8,.9,.999, NA)
Is.Outlier(TestX,SCALE = "LOGIT")
Is.Outlier(100*TestX,SCALE = "LOGIT", PERCENT = T)

## End(Not run)
```

Level2NA	<i>Level2NA</i>
----------	-----------------

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)
```

Levels	<i>Levels</i>
--------	---------------

Description

Levels can be used as base::levels, but displays the levels in the console with a comma after the level such that parts of it can be directly copied.

Usage

```
Levels(FAC, REV = F)
```

Arguments

- FAC is a factor or any object to which levels() can be applied
- REV = F, set to T to get the levels in reverse order

Value

data.frame of one row with levels in hyphens with attached comma

Author(s)

Dirk Hasenclever 2024-02-03

See Also

[Colnames](#)

Examples

```
## Not run:
FAC <- factor(sample(LETTERS[1:10], 100, replace = T))
levels(FAC)
Levels(FAC)
Levels(FAC, REV = T)

## End(Not run)
```

LhrPlot	<i>LhrPlot</i>
---------	----------------

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, 2025-06-27

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, 2025-06-27

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)
```

MakeCIstring	<i>MakeCIstring</i>
--------------	---------------------

Description

MakeCIstring combines a numeric vector of length 3 that represents an estimate and its lower and upper CI limits into a character string of type XX [YY ; ZZ] for use in a condensed table MakeCIstring can be applied row-wise to a matrix or data.frame

Usage

```
MakeCIstring(EST, DIGITS = 1, UNIT = "")
```

Arguments

- EST a numeric vector of length >= 3, the first three components are used
- DIGITS = 1 number of digits desired
- UNIT = ""

Value

a character string of form XX [YY ; ZZ]

Author(s)

Dirk Hasenclever 2023, 2024-07-19

Examples

```
## Not run:
MakeCIstring(c(17, 14.1234, 18.8))
MakeCIstring(c(17, 14.1234, 18.8),2, "%")
MakeCIstring(rbind( c(17, 14, 18.8), c(2, 1.4, 3.8))) # Error
apply(rbind( c(17, 14, 18.8), c(2, 1.4, 3.8)) ,1,MakeCIstring,2)

## End(Not run)
```

MakeInteractionFactor	<i>MakeInteractionFactor produces an InteractionFactor combining two factor variables</i>
-----------------------	---

Description

MakeInteractionFactor produces an InteractionFactor variable combining two factor variables. This is useful when doing subgroup analyses or analyzing if the effects of two factors are additive.

Usage

```
MakeInteractionFactor(FAC1, FAC2, BOTH_VALID = TRUE)
```

Arguments

FAC1	factor or variable convertible to factor
FAC2	factor or variable convertible to factor
BOTH_VALID	= TRUE per default the interaction factor is only defined when both factors are valid (= not NA) if FALSE NA will be treated as a factor level

Value

a factor combining FAC1 and FAC2 with original level order respected and empty levels dropped.

Author(s)

Dirk Hasenclever 2024-12-16

Examples

```
## Not run:
Fac1<-sample(c("Male", "Female"), size = 20, replace = T)
Fac1[2]<-NA
Fac1<- factor(Fac1, levels = c("Male", "Female"))
Fac2<-sample(c("I", "II", "III"), size = 20, replace = T)
Fac2<- factor(Fac2, levels = rev(c("I", "II", "III")))
Fac2[17:20]<-NA
MakeInteractionFactor(Fac1, Fac2)
MakeInteractionFactor(Fac1, Fac2, BOTH_VALID = F)
## End(Not run)
```

MartingalePlot	<i>MartingalePlot</i>
----------------	-----------------------

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, 2025-06-27

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. if `M` is a `data.frame` already, the rownames are added as first column. if `M` is a vector, a `data.frame` with one row is returned.

Usage

```
Matrix2Df(M)
```

Arguments

`M` as a matrix

Value

dataframe

Author(s)

Dirk Hasenclever 2018-01-12, 2022-07-29

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 = "PATIENT_ID",
  SUB = NULL,
  DROP_DOUBLE = c("PATIENT_ID", "CNO"),
  REPORT = T
)
```

Arguments

MASTER	as 1st dataframe
SLAVE	as 2nd dataframe
BY	= "PATIENT_ID" key name both in MASTER and SLAVE
SUB	= NULL is the extension the second colname of a doublette gets as ".SUB"
DROP_DOUBLE	= c("PATNO", "CNO") redundant colnames to be dropped 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, 2024-02-07

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 performed 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,
  SCALE = "ID",
  ...
)
```

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", "TransMean", "TransSd")

SHOW_N =T Show N=length(METRIC) top left

DIGITS =2 Number of digits for statistics

... further parameters for scale transformations can be provided via ... like Z=1/2 and PROCENT=T

SCALE="ID", chooses
a ScaleTransformation ID, POWER_Z, LOG10, LOGIT, if a SCALE transformation is used, also the backtransformed TransMean and TransSd are provided.

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, 2022-07-29

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, Box-plot, 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 convertible 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

[ourPkg-deprecated](#)

MetricsBySplit	<i>MetricsBySplit produces a data.frame describing one or many metric variables by a grouping factor SPLIT</i>
----------------	--

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 performed 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 variable SPLIT

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	<i>MissingRateTable</i>
------------------	-------------------------

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	<i>ModalValue</i>
------------	-------------------

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))
ModalValue(METRIC)
ModalValue(METRIC,ADJ=.2)
```

MoveAfter	<i>MoveAfter</i>
-----------	------------------

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("%A11", " A11"),NULL)
MoveAfter(DFR,c(10,9),1)
MoveAfter(DFR,c(" A11", "%A11"), " 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,
  UNIT = 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 alternatingly left and right.
SCALES	= NULL, specify type of scale for each course out of c("ID", "LOG10", "POWER_Z", "LOGIT")
XLIM	= NULL, specify limits of the time axis
UNIT	= NULL, if diff(XLIM) is divisible by UNIT TIME axis will have UNIT based ticks
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 for 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 for all or a vector of length COURSES
PCH	= NULL Point type for measured values, one of all or a vector of length COURSES
...	specify Z for SCALE = "POWER_Z" and/or "PERCENT" for SCALE = "LOGIT"

Value

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

Author(s)

Dirk Hasenclever 2020-02-26, 2021-07-12, 2024-12-16 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 = 1, COLS = 1:5, PCH = 15,
                               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 invisibly returned. It can be used to place points in the plot afterwards.
graphics::points(5, Lim[1], bg=2, cex=1.7, pch=21)

## End(Not run)
```

MultipleEndpointComparisonCox

Multiple Endpoint Comparison with Cox Proportional Hazards Model

Description

Performs Cox proportional hazards analysis for multiple time-to-event endpoints, comparing two arms in a study. The function can adjust for additional covariates and formats the output for use with the ‘PlotEndpointForestCox’ function.

Usage

```
MultipleEndpointComparisonCox(
  DFR,
  ARM,
  ENDPOINTS,
  COLUMNS = NULL,
  CONTROLARM = 1
)
```

Arguments

DFR	A data frame containing the data for the analysis.
ARM	A character string specifying the name of the column in ‘DFR’ that represents the treatment or arm variable. Must be a factor with exactly two levels.
ENDPOINTS	A character vector specifying the names of the time-to-event outcome columns in ‘DFR’. Each must be a ‘Surv’ object.
COLUMNS	An optional character vector specifying covariates for adjustment. Defaults to ‘NULL’.
CONTROLARM	An integer (1 or 2) indicating which level of the ‘ARM’ factor corresponds to the control/reference group (Arm A). Defaults to 1.

Value

A data frame with one row for each endpoint, containing the columns: ‘Endpoint’, ‘Events_ArmA’, ‘N_ArmA’, ‘Events_ArmB’, ‘N_ArmB’, ‘HR’, ‘lower95CI’, ‘upper95CI’, and ‘P_value’. This format is the required input for the ‘PlotEndpointForestCox’ function.

Author(s)

Dirk Hasenclever 2025-07-08 (adapted from SubgroupAnalysisCox)

MultipleEndpointComparisonLogist

Multiple Endpoint Comparison with a logistic regression Model

Description

Performs logistic regression analysis for multiple binary endpoints, comparing two arms in a study. The function can adjust for additional covariates and formats the output for use with the ‘PlotEndpointForestLogist’ function.

Usage

```
MultipleEndpointComparisonLogist(
  DFR,
  ARM,
  ENDPOINTS,
  COLUMNS = NULL,
  CONTROLARM = 1
)
```

Arguments

DFR	A data frame containing the data for the analysis.
ARM	A character string specifying the name of the column in ‘DFR’ that represents the treatment or arm variable. Must be a factor with exactly two levels.
ENDPOINTS	A character vector specifying the names of the binary outcome columns in ‘DFR’.
COLUMNS	An optional character vector specifying covariates for adjustment. Defaults to ‘NULL’.
CONTROLARM	An integer (1 or 2) indicating which level of the ‘ARM’ factor corresponds to the control/reference group (Arm A). Defaults to 1.

Value

A data frame with one row for each endpoint, containing the columns: ‘Endpoint’, ‘Count_ArmA’, ‘N_ArmA’, ‘Count_ArmB’, ‘N_ArmB’, ‘OddsRatio’, ‘lower95CI’, ‘upper95CI’, and ‘P_value’. This format is the required input for the ‘PlotEndpointForestLogist’ function.

Author(s)

Dirk Hasenclever 2025-08-12

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 filter.

Usage

```
nice.hist(
  x,
  HISTCOL = 7,
  DENS COL = 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)
DENS COL	defines the colour of the density curve, default=4 (blue). DENS COL=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 specifies 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

[ourPkg-deprecated](#)

NiceCumIncPlot	<i>NiceCumIncPlot produces a cumulative incidence curve plot in competing risk analysis</i>
----------------	---

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",  
  PERCENT = F,  
  MAXTIME = NULL,  
  XLIM = NULL,  
  YLIM = c(0, 1),  
  SCALE = NULL,  
  UNIT = "???",
```

```

COL = NULL,
EVENTNAME = "Event",
COMPETENAME = "Competing",
CEX = 0.8,
DIGITS_RATES = 3,
DIGITS_HR = 2,
LAS = 1,
LWD = 3,
...
)

```

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
PERCENT	= FALSE set to TRUE, if you want y-axis in percent
MAXTIME	= NULL, if specified data will be administratively censored at MAXTIME and later events ignored.
XLIM	=NULL, range of x-axis - NOTE that for the comparison also Events beyond XLIM are used, unless MAXTIME is used.
YLIM	=c(0,1), range of y-axis, always between 0 and 1
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
DIGITS_RATES	= 2 Display DIGITS_RATES ciphers in total in Rate estimates

DIGITS_HR	= 2 Round to DIGITS_HR in Hazard ratio table
LAS	= 1 orientation for y-axis
LWD	= 3 line widths
...	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, 2020-07-16, 2022-07-19, 2022-07-26, 2023-05-16, 2025-06-27

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)
```

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 $\pm$ 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,
  CEX = 1.3,
  CEX.LEGEND = 1.3,
  ...
)
```

Arguments

METRIC	metric variable
NCLASS	= 40 defines the number of classes (=bins) in the histogram
ADJ	= 1 controls 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 $\pm$ 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 LEG- END
CUTLINE	=NULL Draws vertical lines from cut values to the density curve
CEX	= 1.3 letter size for axis, labels and title
CEX.LEGEND	= 1.3 letter size for legends and tables
...	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, 2023-11-24, 2024-08-06, 2024-12-17

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)

```

NicePvalue

NicePvalue

Description

NicePvalue transforms numeric p-values to nice p-value-characters

Usage

```

NicePvalue(
  PVAL,
  LOWEST_P = 0.001,
  SIGDIGITS = 2,
  TAILING_ZEROS = T,
  PRUNE = T,
  LEADING_ZERO = T
)

```

Arguments

PVAL	a numeric (vector of) p-values or characters or factor levels that can be interpreted as numeric after removing blanks, p, P, = or <. Else an empty character string is returned.
LOWEST_P	= 0.001 cut values for label <LOWEST_P
SIGDIGITS	= 2, number of significant digits required
TAILING_ZEROS	= T, fill with tailing zeros, e.g. 0.9 to 0.90
PRUNE	= T, cut number to have not more digits than LOWEST_P
LEADING_ZERO	= T, set to F to have e.g. .049 instead of 0.049

Value

character (vector of) formatted p-value characters

Author(s)

Dirk Hasenclever 2021-10-21, 2021-10-22, 2022-03-31

Examples

```
## Not run:
# Example
PVAL<-(c(0.001,0.9, 0.901, 0.906, 0.4723, 0.04763, 0.003034, 0.003054,
         0.004723,0.0004723,0.00009723, 0.0000012))

data.frame(PVAL, PVAL_C=as.character(PVAL), P_Value=
           NicePvalue(PVAL, LOWEST_P = 0.001, SIGDIGITS = 2,
                     TAILING_ZEROS = T, LEADING_ZERO = T, PRUNE = T))

## 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 confidence 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", Specify 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 viy post hoc administrative censoring at c, i.e. ignoring events beyond c and restricting time to c. 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, 2022-07-26, 2025-06-27

Examples

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

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

VA$Karnvsky<-factor(VA$Karn)
levels(VA$Karnvsky)<-c("10-40","10-40","10-40","10-40","50-74","50-74",
                      "50-74","50-74","75-100","75-100","75-100","75-100")
NiceSurvPlot(Times=VA$time,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$time,Status=VA$status,Groups=VA$Karnvsky,
              Medians=T,Scale=180,Unit = "Days",Timepoint = 180,OrderByPlateau = T)

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

NiceSurvPlot(Times=VA$time,Status=VA$status,Groups=VA$treat,
              Medians=T,Scale=180,Unit = "Days",Xlab="OSV [Days]",Timepoint = 180,
              OrderByPlateau = T, AnalysisString="NiceSurvPlot Demo")
NiceSurvPlot(Times=VA$time,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$time,Status=VA$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, 2025-04-10, 2025-06-27

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 ASCII if of class character or factor

Author(s)

Dirk Hasenclever 2019-02-15, 2019-10-28, 2022-07-20

Nvalid	<i>Nvalid</i>
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	
<pre>## Not run: Nvalid(c(1,2,NA,NA,5)) ## End(Not run)</pre>	
OneGroupTimeCourse	<i>OneGroupTimeCourse analyses the time course (or gradual change with dose or intensity) of a metric endpoint in one group. Comparison different groups is not yet supported.</i>

Description

OneGroupTimeCourse fits a linear mixed model with PATIENT as random effect modelling the mean structure. The dependence taken into the account my estimating a variance covariance matrix. This also allows dealing with missing values as long as at least one measurement is available for a given patient or unit of observation. The main result is a plot showing all individual measurments, the naive raw mean structure in yellow and the model estimated mean structure in darkblue. In addition, it is color coded whether a mean differs significantly from baseline (the frist mean) and from the preceding mean (black= not significant, red = significant at SIG.LEVEL. This plot is saved and displayed. If DESCRIBE several tables in particular the means and the contrasts are saved.

Usage

```

OneGroupTimeCourse(
  DFR,
  ENDP,
  LONGNAME = NULL,
  TIME,
  VISITS = NULL,
  VISIT_TIMES = NULL,
  PATIENT = "PATIENT_ID",
  SCALE = "ID",
  YLIM = NULL,
  MADFILTER = 4,
  MAIN = "",
  XLAB = "time points",
  SHOW_SCALE = T,
  SIG_LEVEL = 0.05,
  JITTER = 0.15,
  RESULTDIR = T,
  FLAG = "",
  PRINT = T,
  DESCRIBE = F,
  RAWMEANS = T,
  TO_BASELINE = T,
  TO_PREDECESSOR = T,
  ...
)

```

Arguments

DFR	is a deep dataframe that contains the relevant variables
ENDP	Name of metric variable in DFR to be analysed
LONGNAME	=NULL longname for ENDP to be used in plot
TIME	Name of factor (or character) variable identifying distinct time points (or ordinal conditions)
VISITS	= NULL, if NULL all levels of TIME are used, else only those named in vector VISITS
VISIT_TIMES	= NULL, spacing of Visits on the time axis. Set to 1:length(VISITS) if NULL
PATIENT	= "PATIENT_ID", name or the variable identifying the units of observation on which multiple measurments were taken.
SCALE	= "ID" chooses a ScaleTransformation ID, POWER_Z, LOG10, LOGIT
YLIM	= NULL optional ON THE ORIGINAL SCALE, otherwise chosen from SCALE and range
MADFILTER	= 4 , if >0 (typically 3,4, or 5) filters out Outliers beyond +- MADFILTER* mad
MAIN	= "" Main title for plot
XLAB	= "time points", time axis label
SHOW_SCALE	= T, set to F of you do not want to indicate the scale on the y-axis label
SIG_LEVEL	= 0.05, control significance levels
JITTER	= .15, control the degree of jitter for the individual points

RESULTDIR	= T, results to be saved in the result directory indicated in the string ResultDir in the global environment, else in working directory
FLAG	= "", optional Prefix for file names to be saved.
PRINT	= T, If TRUE the tables generated to define the plot are printed for inspection
DESCRIBE	= F, If TRUE several tables and possibly a Vennplot are saved in the ResultDir
RAWMEANS	= T Set to FALSE to omit plotting the yellow raw means
TO_BASELINE	= T Set to FALSE to omit plotting vertical differences to the horizonatal line indicating the mean at baseline (first visit)
TO_PREDECESSOR	= T Set to FALSE to omit plotting differences to the predecessor mean at the visit before
...	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

Details

OneGroupTimeCourse

Value

a generic plot showing all individual data points, the mean value structure and

Author(s)

Dirk Hasenclever 2022-07-28, 2024-02-15, 2024-11-23, 2025-01-08, 2025-06-27, 2025-08-12

OurTools-deprecated	<i>Deprecated functions in package OurTools.</i>
---------------------	---

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,  
  FORTRANSPPOSITION = 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,
  DENS COL = 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)
## 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,
  CEX.AXIS = 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) divisible 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
CEX.AXIS	= 1 will be passed to control the size of axis characters
...	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), 2022-07-29, 2022-10-18

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	<i>Plot Binary versus Metric</i>
--------------------	----------------------------------

Description

Basic plot to look at the dependence of a binary variable from a metric one. Scatterplot BINARY versus 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 regression.

Usage

```
PlotBinaryByMetric(
  BINARY,
  METRIC,
  BNAME = NULL,
  MNAME = NULL,
  MAIN = "",
  YLAB = NULL,
  SCALE = "ID",
  XLIM = NULL,
  NPRETTY = 6,
  SPAN = 0.75,
```

```

LOWESSCOL = 8,
CURVECOL = 1,
POINTCOLS = c("darkblue", "red"),
LEGEND = T,
PVALUE = T,
CEX = 1.3,
CEX.LEGEND = NULL,
PROP = T,
...
)

```

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 METRIC
MAIN	= "" is an optional title, if = "", "probability(BNAME MNAME)"
YLAB	= NULL Define your own y-axis label. if NULL YLAB = paste("Probability of", BNAME) or paste("Probability of", BNAME, "[%]") depending on PROP
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
CURVECOL	= 1 colour of the logistic regression curve
POINTCOLS	= c("darkblue", "red") colours for points
LEGEND	= T show legend,
PVALUE	= T show p-value within LEGEND
CEX	= 1.3 adjust size of all symbols
CEX.LEGEND	= NULL, option to choose a letter size different from CEX for the legend
PROP	= T logical switch: set to FALSE in order to get the probability on the y-axis in percent
...	Specify Z or PERCENT for SCALE = "POWER_Z" or "LOGIT". Specify XAXT = "n" or YAXT = "n" to suppress drawing of respective axes

Value

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

Author(s)

Dirk Hasenclever old; new version 2019-08-09, 2023-02-13, 2023-08-23, 2024-07-18

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) divisible 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)
```

PlotEndpointForestCox *Plot Endpoint Forest Plot Cox*

Description

Creates a comprehensive forest plot for multiple time-to-event endpoints (e.g., OS, PFS, EFS). This version is designed to display the endpoint name, event/sample size for two arms, the hazard ratio with its confidence interval, and a p-value for each endpoint.

Usage

```
PlotEndpointForestCox(
  ENDPOINT_DATA,
  MAIN = "",
  LONGNAMES = NULL,
  AT = NULL,
  LEFT_ADJ = 1,
  RIGHT_ADJ = 1,
  XLAB = "Hazard Ratio",
  ARM_NAMES = c("Arm A", "Arm B")
)
```

Arguments

ENDPOINT_DATA	A data frame containing the columns: 'Endpoint', 'Events_ArmA', 'N_ArmA', 'Events_ArmB', 'N_ArmB', 'HR', 'lower95CI', 'upper95CI', and 'P_value', as produced by MultipleEndpointComparisonCox.
MAIN	An optional title for the plot.
LONGNAMES	A named list or vector for custom display names of the endpoints.
AT	An optional vector specifying tick locations on the HR scale. Will be automatically generated if NULL.
LEFT_ADJ	Numeric. Increases space on the left for text columns.
RIGHT_ADJ	Numeric. Increases space on the right for text columns.
XLAB	The label for the x-axis.
ARM_NAMES	A character vector of length 2 with the names of the two arms being compared. Arm A should be the reference arm (e.g., c("Control", "Experiment")).

Value

A forest plot is drawn. Returns NULL invisibly.

Author(s)

Dirk Hasenclever 2025-07-08 (adapted from PlotSubgroupForestCox)

PlotEndpointForestLogist

Plot Endpoint Forest Plot Logist

Description

Creates a comprehensive forest plot for multiple binary endpoints (e.g., PE90, PE1yr). This version is designed to display the endpoint name, count/sample size two arms, the hazard ratio with its confidence interval, and a p-value for each endpoint.

Usage

```
PlotEndpointForestLogist(
  ENDPOINT_DATA,
  MAIN = "",
  LONGNAMES = NULL,
  AT = NULL,
  LEFT_ADJ = 1,
  RIGHT_ADJ = 1,
  XLAB = "Odds Ratio",
  ARM_NAMES = c("Arm A", "Arm B")
)
```

Arguments

ENDPOINT_DATA	A data frame containing the columns: 'Endpoint', 'Count_ArmA', 'N_ArmA', 'Count_ArmB', 'N_ArmB', 'OddsRatio', 'lower95CI', 'upper95CI', and 'P_value', as produced by MultipleEndpointComparisonLogist.
MAIN	An optional title for the plot.
LONGNAMES	A named list or vector for custom display names of the endpoints.
AT	An optional vector specifying tick locations on the OddsRatio scale. Will be automatically generated if NULL.
LEFT_ADJ	Numeric. Increases space on the left for text columns.
RIGHT_ADJ	Numeric. Increases space on the right for text columns.
XLAB	The label for the x-axis.
ARM_NAMES	A character vector of length 2 with the names of the two arms being compared. Arm A should be the reference arm (e.g., c("Control", "Experiment")).

Value

A forest plot is drawn. Returns NULL invisibly.

Author(s)

Dirk Hasenclever 2025-08-12

PlotMetricBySplit	<i>PlotMetricBySplit</i>
-------------------	--------------------------

Description

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

Usage

```
PlotMetricBySplit(  
  METRIC,  
  SPLIT = NULL,  
  TYPE = "Ecdf",  
  PROP = T,  
  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,
```

```

PCH = 19,
LWD = 2,
CEX = 1.3,
CEX.LEGEND = NULL,
TEST = NULL,
...
)

```

Arguments

METRIC	is the metric variable to be plotted
SPLIT	= NULL is a factor or convertible to a factor. If NULL a single group ALL is assumed.
TYPE	= "Ecdf" chooses between "Ecdf", "Dens", "Boxplot", "Stripchart"
PROP	= T logical switch: set to FALSE in order to get the probability on the y-axis of the ecdf plot in percent
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 optional if SCALE = "ID" AND diff(LIM) divisible by UNIT plot interval will have UNIT based ticks
MAIN	= NULL, specify title, if MAIN = NULL type of plot
COL	= NULL colours for the levels of Split - otherwise standard palette
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)
PCH	= 19 Point type used in Stripchart - possibly a vector specifying different point-types by group.
LWD	= 2 Set line width in Ecdf and Density
CEX	= 1.3 adjust size of all symbols
CEX.LEGEND	= NULL, option to choose a letter size different from CEX for the legend
TEST	= NULL if TEST = "Means" plot p-value of t-test or oneway-test on transformed scale, if TEST = "Medians" plot p-value of MW-U-test or Kruskal-Wallis-Test
...	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, 2020-07-16, 2020-10-01, 2021-07-12, 2022-07-21; 2024-07-18; 2024-08-26(AS), 2025-03-14

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 COURSE-LABEL 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,
  CEX = 1.3,
  CEX.LEGEND = 1
)
```

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 suppress the table
CEX	= 1.3 letter size for axis, labels and title
CEX.LEGEND	= 1.0 letter size for legends and tables

Value

Generates a stacked barplot with a frequency table.

Author(s)

Dirk Hasenclever Update 2019-04.11, 2024-07-23

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)
```

Description

Creates a refined forest plot with correct layout, dynamic scale selection, and directional labels below the axis. Layout is controlled by the LEFT_ADJ and RIGHT_ADJ parameters.

Usage

```
PlotSubgroupForestCox(  
  SUBGROUP_CIS,  
  MAIN = "",  
  LONGNAMES = NULL,  
  INTERACTION_PVALUE = TRUE,  
  AT = NULL,  
  LEFT_ADJ = 1,  
  RIGHT_ADJ = 1,  
  XLAB = "Hazard Ratio"  
)
```

Arguments

- SUBGROUP_CIS A data frame created by SubgroupAnalysisCox.
- MAIN = "" An optional title for the plot.
- LONGNAMES A named list or vector for display names.
- INTERACTION_PVALUE
 = TRUE Logical. Display interaction p-value column?
- AT = NULL Optional vector specifying tick locations on the HR scale. Will be automatically generated if NULL.
- LEFT_ADJ Numeric. Increases space on the left for text columns.
- RIGHT_ADJ Numeric. Increases space on the right for text columns.
- XLAB = "Hazard Ratio" The label for the x-axis.

Value

A forest plot is drawn. Returns NULL invisibly.

Author(s)

Dirk Hasenclever 2025-06-27 (thanks to GeminiPRO), 2025-07-02

PlotSubgroupForestLogist

Logistic Regression Subgroup Analysis Forest Plot

Description

Creates a refined forest plot with correct layout, dynamic scale selection, and directional labels below the axis. Layout is controlled by the LEFT_ADJ and RIGHT_ADJ parameters.

Usage

```
PlotSubgroupForestLogist(
  SUBGROUP_CIS,
  MAIN = "",
  LONGNAMES = NULL,
  INTERACTION_PVALUE = TRUE,
  AT = NULL,
  LEFT_ADJ = 1,
  RIGHT_ADJ = 1,
  XLAB = "Odds Ratio"
)
```

Arguments

SUBGROUP_CIS	A data frame created by SubgroupAnalysisLogist.
MAIN	= "" An optional title for the plot.
LONGNAMES	A named list or vector for display names.
INTERACTION_PVALUE	= TRUE Logical. Display interaction p-value column?
AT	= NULL Optional vector specifying tick locations on the OR scale. Will be automatically generated if NULL.
LEFT_ADJ	Numeric. Increases space on the left for text columns.
RIGHT_ADJ	Numeric. Increases space on the right for text columns.
XLAB	= "Odds Ratio" The label for the x-axis.

Value

A forest plot is drawn. Returns NULL invisibly.

Author(s)

Dirk Hasenclever 2025-08-14

Positive_Comparators *Collection of positive ordinal comparators*

Description

TRUE if condition positively fulfilled, 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))
```

Positive_IsInInterval_Checkfunctions

Collection of Positive_IsInInterval_Checkfunctions

Description

TRUE if condition positively fulfilled, FALSE if not met or if any NA!

- GE_LE $A \leq X \leq B$
- GE_LT $A \leq X < B$
- GT_LE $A < X \leq B$
- GT_LT $A < X < B$

Usage

GE_LE(X, A, B)

GT_LE(X, A, B)

GE_LT(X, A, B)

GT_LT(X, A, B)

Arguments

X	vector of ordinal values
A	ordinal lower limit on same ordinal scale, length = 1
B	ordinal upper limit on same ordinal scale, length =

Value

TRUE or FALSE

ProversionProb

ProversionProb

Description

ProversionProb calculates a confidence interval for the proversion probability $\text{pr}(X > Y) + 0.5 \cdot \text{pr}(X = Y)$ 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, 2021-05-19

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	<i>Prune a deparse(substitute(X)) type VarName</i>
--------------	--

Description

PruneVarName postprocesses a deparse(substitute(X)) type VarName assigned to an argument variable in a function. 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 2018, 2025-01-03

Examples

```

VarName<- "DIRK"
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)
# DF exists in the global environment colnames(DF)[2] is returned
VarName<-deparse(substitute( log10(DF[, "a"]) ))
PruneVarName(VarName)

```

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 2018-01-19, 2025-06-27

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)
```

RandomDate	<i>RandomDate</i>
------------	-------------------

Description

RandomDate generates N random dates of class Date in the interval STARTDAT to ENDDAT inclusive.

Usage

RandomDate(N, STARTDAT = "2000-01-01", ENDDAT = "2000-12-31", REPLACE = T)

Arguments

- N number of random dates desired
- STARTDAT = "2000-01-01" USE YYYY-MM-DD as standard!
- ENDDAT = "2000-12-31", must be later than STARTDAT USE YYYY-MM-DD as standard!
- REPLACE = T are duplicate dates allowed

Details

RandomDate

Value

a vector of random Dates between STARTDAT and ENDDAT

Author(s)

Dirk Hasenclever 2021-05-18, 2024-02-07

Examples

```
## Not run:
RandomDate(12, "2000-02-28", "2000-11-28")
RandomDate(1200, REPLACE = F)
# Use standard "YYYY-MM-DD" format!
RandomDate(12, STARTDAT = "2000-02-28", ENDDAT = "30-12-2000", REPLACE = T)

## End(Not run)
```

ReconcileLists	<i>ReconcileLists</i>
----------------	-----------------------

Description

used to compare and reconcile 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	<i>Collection of scale Transformations</i>
----------------------	--

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 $Y \sim X$ and $X \sim Y$ 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,
PLOT = 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, specify COL as vector with length = nlevel(SPLIT), else 1:nlevel(SPLIT)
PCH	= 19 point type(s). For subgroups, specify 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 specify title, if MAIN = NULL no title
OUTLIER	= 4 outlier are identified using lmRob from package robust a points with $ residual > 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 coefficient and its CI
DIGITS	= 3 Number of digits for correlation coefficients
LINE_THRESHOLD	= .3 only show lines iff $ cor \geq LINE_THRESHOLD$
XY_LINE	= T As a default the SymLine, i.e. the principal component 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 $Y \sim X$. 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 $X \sim Y$. 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 $\text{pcor}(x,y \text{SPLIT})$ is given in the legend reporting correlation.
PLOT	= T Option to suppress plot - only to pick the outliers
...	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, 2020-04-02, 2020-10-08.

See Also

[SymLine](#) for adding a symmetric 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)
```

SetupAnalysisDirs

Set Up a standard SubDirectory structure for an analysis

Description

SetupAnalysisDir uses SetupDir to create standard subdirectories in the current working directory and places respective DIRPATHNAME characters in the global environment: _Results, Collect all results here; _Reports, Collect all reports here; _RawData, Collect copy of raw data used here (-> AccessSecutrial::GetAllTables); _Rsources, Collect all study specific functions here; _Prepared-Data Collect all prepared or curated analysis sets here

Usage

```
SetupAnalysisDirs()
```

Value

nothing

Author(s)

Dirk Hasenclever 2024-06-26

See Also

[SetupDir](#)

Examples

```
## Not run:
SetupAnalysisDirs()

## End(Not run)
```

SetUpDir	<i>Set Up a SubDirectory</i>
----------	------------------------------

Description

SetUpDir creates a subdirectory DIRNAME in the current working directory, if it does not yet exists, and places a character DIRPATHNAME in the global environment.

Usage

SetUpDir(DIRNAME, DIRPATHNAME = NULL)

Arguments

DIRNAME name of subdirectory as character
DIRPATHNAME = NULL name of path to dir character, standard paste0(DIRNAME, "Dir")

Value

nothing

Author(s)

Dirk Hasenclever 2024-06-26, 2024-10-07

SlidingWindow	<i>SlidingWindow</i>
---------------	----------------------

Description

SlidingWindow time trends of arbitrary statistic.

Usage

```
SlidingWindow(  
  DFR,  
  DATE,  
  ARGUMENTS,  
  STATISTIC,  
  WINDOW = 30,  
  BY = 7,  
  STARTDATE = NULL,  
  ENDDATE = NULL,  
  PLOT = T,  
  MAIN = "Sliding Window Plot",  
  XLAB = NULL,  
  YLAB = NULL,  
  UNIT = 30,  
  YLIM = NULL,  
  COL = 4,  
  ...  
)
```

Arguments

DFR	a data.frame containing a DATE variable and ARGUMENTS for the STATISTIC
DATE	name of a date variable in DFR of class POSIXct or Date, or character of form "YYYY-MM-DD" or "YYYY/MM/DD".
ARGUMENTS	vector of names of those variables in DFR that are used by STATISTIC function; can be set to NULL.
STATISTIC	user supplied function evaluating a data.frame restricting DFR to ARGUMENTS
WINDOW	= 30 width of the sliding window in days
BY	= 7 shift to the next window in days
STARTDATE	= NULL. If not specified, set to the minimum date encountered, ignoring time part. Use "YYYY-MM-DD" format to specify.
ENDDATE	= NULL. If not specified, set to the maximum date encountered, ignoring time part. Use "YYYY-MM-DD" format to specify.
PLOT	= T logical whether a plot is desired
MAIN	= "Sliding Window Plot" is an optional title, to suppress set MAIN = ""
XLAB	= NULL if left NULL, set to Days or years since earliest DATE
YLAB	= NULL if left NULL, set to the name of the supplied STATISTIC function.
UNIT	= 30 days between ticks on time axis. NOTE: UNIT Will be doubled while more than 12 ticks result. Set UNIT = NULL to switch scale to years of 365 days.
YLIM	= NULL optional control over limits of yaxis
COL	= 4 set colour of sliding curve
...	further arguments passed to STATISTIC, e.g. na.rm = T

Value

Returns a data.frame with statistics at sliding windows, and generates a Plot if PLOT = T, plotting values of the statistic at window mids.

Author(s)

Dirk Hasenclever 2020-06-05, Norbert Koehler 2020-06-08, DH 2020-06-08, DH 2020-06-23

Source	Source
--------	--------

Description

Source is a safer wrapper for base::source with changed default parameters. encoding = "UTF-8" ensures that comparisons of characters work when text data is read from a DB. print = T suppresses the source code, but lets evaluations, messages and lattice plots through to the session.

Usage

Source(SOURCE)

Arguments

SOURCE the filename to be sourced

Value

executes Sourced R-Script, with increased probability that the effect is the same as copying it into the console.

Author(s)

Dirk Hasenclever 2024-07-12

SubgroupAnalysisCox	<i>Subgroup Analysis with Cox Proportional Hazards Model</i>
---------------------	--

Description

Performs subgroup analyses using a Cox proportional hazards interaction model for one or more splitting variables with possibly additional adjustment for covariates. Results from multiple splitting variables are combined into a single table.

Usage

```
SubgroupAnalysisCox(  
  DFR,  
  ARM,  
  SPLIT,  
  RESPONSE,  
  COLUMNS = NULL,  
  CONTROLARM = 1,  
  DYNAMIC = TRUE  
)
```

Arguments

DFR	A data frame containing the data for the analysis.
ARM	A character string specifying the name of the column in ‘DFR’ that represents the treatment or arm variable. Must be a factor with exactly two levels.
SPLIT	A character vector specifying the names of the columns in ‘DFR’ that define the subgroups. The function will iterate through each variable in this vector. Each variable must be suitable for conversion to a factor with at least two levels.
RESPONSE	A character string specifying the name of the column in ‘DFR’ that represents the time-to-event outcome. This should typically be a ‘Surv’ object created using the ‘survival’ package.
COLUMNS	An optional character vector specifying the names of additional columns in ‘DFR’ to be included as covariates in the Cox model. Defaults to ‘NULL’.
CONTROLARM	An integer (1 or 2) indicating which level of the ‘ARM’ factor corresponds to the control group. Used to potentially reverse factor levels so the control arm is the reference. Defaults to 1.
DYNAMIC	= TRUE, If TRUE the ‘N_*’ and ‘E_*’ colnames are changed replacing ‘*’ by the respective levels of ARM.

Details

The function iterates through each variable name provided in the ‘SPLIT’ vector. For each variable, it performs a subgroup analysis. The results for each variable are formatted into rows indicating the SPLIT variable name and level, and then all results are combined using ‘rbind’.

The overall interaction test for each SPLIT variable is performed using a likelihood ratio test comparing a global model with interaction to a global model with only main effects.

Value

A data frame containing the combined subgroup analysis results for all specified SPLIT variables. Includes columns: ‘Split Variable’ (name of the SPLIT variable), ‘Split Level’ (the subgroup level), ‘N_Control’, ‘E_Control’, ‘N_Experiment’, ‘E_Experiment’, ‘HR’, ‘lower .95 CI’, ‘upper .95 CI’, ‘Within-Subgroup Pvalue’, ‘Overall Interaction Pvalue’. If DYNAMIC = TRUE, the ‘N_*’ and ‘E_*’ colnames are changed replacing ‘*’ by the respective levels of ARM.

Author(s)

Dirk Hasenclever 2025-06-27 (thanks to GeminiPRO), 2025-07-02

SubgroupAnalysisLogist

Subgroup Analysis with a Logistic Regression Model

Description

Performs subgroup analyses using a logistic regression interaction model for one or more splitting variables with possibly additional adjustment for covariates. Results from multiple splitting variables are combined into a single table.

Usage

```
SubgroupAnalysisLogist(
  DFR,
  ARM,
  SPLIT,
  RESPONSE,
  COLUMNS = NULL,
  CONTROLARM = 1,
  DYNAMIC = TRUE
)
```

Arguments

DFR	A data frame containing the data for the analysis.
ARM	A character string specifying the name of the column in ‘DFR’ that represents the treatment or arm variable. Must be a factor with exactly two levels.
SPLIT	A character vector specifying the names of the columns in ‘DFR’ that define the subgroups. The function will iterate through each variable in this vector. Each variable must be suitable for conversion to a factor with at least two levels.

RESPONSE	A character string specifying the name of the column in 'DFR' that represents the binary outcome.
COLUMNS	An optional character vector specifying the names of additional columns in 'DFR' to be included as covariates in the logistic regression model. Defaults to 'NULL'.
CONTROLARM	An integer (1 or 2) indicating which level of the 'ARM' factor corresponds to the control group. Used to potentially reverse factor levels so the control arm is the reference. Defaults to 1.
DYNAMIC	= TRUE, If TRUE the 'N_*' and 'Count_*' colnames are changed replacing '*' by the respective levels of ARM.

Details

The function iterates through each variable name provided in the 'SPLIT' vector. For each variable, it performs a subgroup analysis. The results for each variable are formatted into rows indicating the SPLIT variable name and level, and then all results are combined using 'rbind'.

The overall interaction test for each SPLIT variable is performed using a likelihood ratio test comparing a global model with interaction to a global model with only main effects.

Value

A data frame containing the combined subgroup analysis results for all specified SPLIT variables. Includes columns: 'Split Variable' (name of the SPLIT variable), 'Split Level' (the subgroup level), 'N_Control', 'Count_Control', 'N_Experiment', 'Experiment', 'OR', 'lower .95 CI', 'upper .95 CI', 'Within-Subgroup Pvalue', 'Overall Interaction Pvalue'. If DYNAMIC = TRUE, the 'N_*' and 'Count_*' colnames are changed replacing '*' by the respective levels of ARM.

Author(s)

Dirk Hasenclever 2025-08-14

SuccessRate	<i>SuccessRate</i>
-------------	--------------------

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	<i>SurvivalAtT</i>
-------------	--------------------

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, 2025-06-27

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)
```

SurvivalRateVersusTargetPvalue

Calculates p-value for a survival rate point hypothesis

Description

Calculates a two-sided p-value for the null hypothesis $H_0: S(T_0) = \text{TARGET}$, where $S(T_0)$ is the survival rate at a specific time T_0 . The p-value is found by determining the confidence level (1-alpha) at which the TARGET value equals one of the confidence interval bounds. The CI calculation within this function uses the default "log" transformation (conf.type = "log"). The p-value is then alpha.

Usage

```
SurvivalRateVersusTargetPvalue(SURVOBJ, T0, TARGET, DIGITS = 3)
```

Arguments

SURVOBJ	A 'Surv' object from the 'survival' package.
T0	Numeric. The specific time point of interest for the survival rate. Must be a single non-negative finite value.
TARGET	Numeric. The hypothesized target survival rate under the null hypothesis at T_0 . Must be a single finite value between 0 and 1 (exclusive).
DIGITS	= 3 numbers of digits for output of rate estimates

Value

A summary data.frame including the two-sided 95 percent CI and the dual p-value.

Author(s)

Dirk Hasenclever 2025-06-27

SurvPlot

*SurvPlot***Description**

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

Usage

```
SurvPlot(
  TIMES,
  STATUS = NULL,
  SPLIT = NULL,
  SPLIT.LABELS = NULL,
  COL = NULL,
  BY.LINETYPE = FALSE,
  CENS.TICKS = FALSE,
  MAIN = "",
  XLAB = NULL,
  YLAB = "proportion event free",
  GRANULARITY = NULL,
  UNIT = "????",
  YLIM = c(0, 1),
  TIMEPOINT = NULL,
  STATISTIC = "Rate",
  MAXTIME = NULL,
  ORDER = TRUE,
  NRISK = TRUE,
  EVENTTABLE = TRUE,
  CILOT = NULL,
  LOGRANK = FALSE,
  COMPARE = NULL,
  Y.REFLINE = NULL,
  X.REFLINE = NULL,
  PROP = F,
  CEX = 1.3,
  CEX.LEGEND = 1.3,
  CEX.NRISK = 1.3,
  LWD = 3,
  EVENTPOSX = NULL,
  EVENTPOSY = NULL,
  STATPOSX = NULL,
  STATPOSY = NULL,
  HRPOSX = 1,
  HRPOSY = 1
)
```

Arguments

TIMES	EITHER Time to event variable of class "Surv" OR time component of time to event
-------	--

STATUS	= NULL, status component of time to event (needed only if TIMES is not of class "Surv")
SPLIT	= NULL, grouping variable; if NULL a single curve is plotted with confidence band (set COL=c(1,0,0) to suppress it)
SPLIT.LABELS	= NULL, Specify Arm labels - if NULL: levels(SPLIT) ATTENTION: up to 10 characters!
COL	= NULL, Specify colour code - if NULL a standard palette is used
BY.LINE TYPE	= FALSE, plot black curves distinguished by LineType ignoring COL
CENS.TICKS	= FALSE, show censoring ticks
MAIN	= "", plot title
XLAB	= NULL, set x-axis label manually instead of default. NOTE that "in UNIT" is appended to the XLAB if the UNIT is not part of XLAB!!
YLAB	= "% event free" , or "proportion event free" depending on PROP, Or specify y-axis label yourself
GRANULARITY	= NULL, granularity unit for ticks on time axis (default 12), if NULL is adapted to have legible x-axis
UNIT	= "????", Unit of time (days, months, years) Please specify!
YLIM	= c(0,1) , specify range y-axis, if lower > 0, this is marked in the plot
TIMEPOINT	= NULL, Time point at which to sort curves, at which to provide rate estimates, up to which to provide RMean estimates. If not specified, automatically set based on min(maxtime/2, mean(TIMES)).
STATISTIC	= "Rate", Options for SPLIT-wise statistic table "Rate", "Median", "RMean", "None"
MAXTIME	= NULL, Restrict time axis to MAXTIME, later events are ignored in any statistics!!
ORDER	= TRUE, Order legend by decreasing order of curves at TIMEPOINT
NRISK	= TRUE, Show N at risk (cumulative censored) below plot at x-axis ticks
EVENTTABLE	= TRUE Show EventTable detailing Number of events and N for each curve
CI PLOT	= NULL plot 95 percent CIs at TIMEPOINT if CI PLOT = TRUE, else CI PLOT can be a numeric vector
LOGRANK	= FALSE, compare groups with logrank test and show p.value
Y.REFLINE	= NULL , Horizontal reference line at (possibly a vector)
X.REFLINE	= NULL , Vertical reference line at (possibly a vector)
PROP	= F set to TRUE, if you want the y-axis in probability
CEX	= 1.3 letter size for axis, labels and title
CEX.LEGEND	= 1.3 letter size for legends and tables
CEX.NRISK	= 1.3 letter size for numbers at risk below the plot
LWD	= 3, line width of curves
EVENTPOSX	= NULL, X Position of Eventtable relative to XLIM: default is top left
EVENTPOSY	= NULL, Y Position of Eventtable relative to YLIM: default is top left
STATPOSX	= NULL, X Position of Ratetable relative to XLIM: default is left below Event-table
STATPOSY	= NULL, Y Position of Ratetable relative to XLIM: default is left below Event-table

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, 2012, 2017, 2018, 2022, 2024-07-26, 2025-06-27

Examples

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

## Set wide Window to about 3 x 2
SurvPlot(TIMES=VA$time, STATUS=VA$status, SPLIT=VA$celltype,
          STATISTIC = "Median", GRANULARITY=180, UNIT = "Days", XLAB="OSV", TIMEPOINT = 360)
VA$Karnovsky<-factor(VA$Karn)
levels(VA$Karnovsky)<-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")
SurvPlot(TIMES=VA$time, STATUS=VA$status, SPLIT=VA$Karnovsky,
          STATISTIC = "Median", GRANULARITY=180, UNIT = "Days", XLAB="OSV", TIMEPOINT = 60,
          ORDER = F, MAIN="OSV by Karnovsky")

SurvPlot(TIMES=VA$time, STATUS=VA$status, SPLIT=VA$Karnovsky,
          STATISTIC = "Median", GRANULARITY=180, UNIT = "Days", TIMEPOINT = 180, ORDER = T)

SurvPlot(TIMES=VA$time, STATUS=VA$status, SPLIT=VA$Karnovsky,
          STATISTIC = "Rate", GRANULARITY=180, UNIT = "Days", XLAB="OSV", TIMEPOINT = 180,
          ORDER = T, BY.LINETYPE = T)

SurvPlot(TIMES=VA$time, STATUS=VA$status, SPLIT=VA$treat,
          STATISTIC = "Median", GRANULARITY=180, UNIT = "Days", XLAB="OSV [Days]", TIMEPOINT = 180,
          ORDER = T)
SurvPlot(TIMES=VA$time, STATUS=VA$status, SPLIT=VA$treat,
          STATISTIC = "Median", GRANULARITY=180, UNIT = "Days", XLAB="OSV [Days]", TIMEPOINT = 180,
          ORDER = T,
          SPLIT.LABELS = c("standard", "test"))

SurvPlot(TIMES=VA$time, STATUS=VA$status, SPLIT=VA$treat,
          STATISTIC = "Median", GRANULARITY=60, MAXTIME = 540, UNIT = "Days", XLAB="OSV [Days]",
          TIMEPOINT = 180, ORDER = T,
          SPLIT.LABELS = c("standard", "test"), COL = c(4,6))

## 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	<i>Table</i>
-------	--------------

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)
```

ToInitialUpper

ToInitialUpper

Description

ToInitialUpper is a character string manipulation function to produce strings starting with an upper case and continuing in lower case.

Usage

```
ToInitialUpper(STR)
```

Arguments

STR vector typically of character strings

Details

ToInitialUpper

Value

STR transformed to character vector with initial upper cases. NA remain NA

Author(s)

Dirk Hasenclever 2024-02-07

See Also

[base::toupper](#), [base::tolower](#)

Examples

```
## Not run:
ToInitialUpper(c("DIRK",NA,"oana",1))

## 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	<i>UnivariateCoxTable</i>
--------------------	---------------------------

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, 2025-06-27

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)
```

VariablesBySplit	<i>produces a presentation table of mixed numeric or discrete variables by a grouping factor SPLIT</i>
------------------	--

Description

The function produces a presentation table of mixed numeric or discrete 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. VariablesBySplit depends on packages gtsummary and glue. Based on the parameter specification the functions uses glue to build complex gtsummary code, which is than evaluated.

Usage

```
VariablesBySplit(
  DFR,
  COLUMNS,
  LONGNAMES = NULL,
  SPLIT = NULL,
  SPLIT_HEADER = NULL,
  ALL = FALSE,
  NO_ZEROS = TRUE,
  SHOW_N = FALSE,
  STATS_COLUMN = FALSE,
  MAIN = NULL,
  TESTS = TRUE,
  PARAMETRIC = FALSE,
  SHOW = TRUE
)
```

Arguments

DFR	a data.frame containing all the relevant variables
COLUMNS	colnames in DFR to be described (at least one)
LONGNAMES	=NULL alternative names for the COLUMNS
SPLIT	= NULL colname in DFR that names a grouping variable. NOTE: rows with missing SPLIT are eliminated with a warning
SPLIT_HEADER	= NULL, wide header above the SPLIT levels in the table
ALL	= FALSE If TRUE a 'Overall' column is added on the left of the SPLIT group table columns.
SHOW_N	= FALSE, add a table column showing the number of valid observations for each variable
STATS_COLUMN	= FALSE show statistics used in a separate column instead of by footnote
MAIN	= NULL if a character, a main table title is added.
PARAMETRIC	= FALSE, default: non-parametric descriptions and tests
SHOW	= T do you want to see what you prduced in the presentation window?
NOZEROS	= TRUE eliminates factor levels that were not observed, TRUE is default
TEST	= TRUE logical: Tests for all variable or none

Details

VariablesBySplit

Value

returns a presentation table by group variable of class: Large tbl_summary, which can be assigned to a variable. Save with base::saveRDS. In Rmarkdown: load with base:: readRDS, transform with tbl<-gtsummary::as_flex_table(tbl) and just call: tbl to include the table in the knitred document.

Author(s)

Dirk Hasenclever with Annett Schrock 2024-12-13, 2025-03-18, 2025-06-27

See Also

[FactorsBySplit](#), [MetricsBySplit](#)

VisualiseCoxModel	<i>VisualiseCoxModel</i>
-------------------	--------------------------

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 specifying values of the covariate for which curves are to be plotted. Names MUST correspond to Colnames of DFR. The Cox model is <code>SURV ~ names(SCENARIOS)</code>
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 SCENARIOS must 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 be included 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 between the plot and the legend.
XPAD	=0.2, control distance between columns of the legend table

Value

produces a plot and returns data.frame with MEDIAN respectively RATE estimates

Author(s)

Dirk Hasenclever 2017-12-20, 2017-12-30, 2025-06-27

Examples

```
## Not run:
DAT<-ovarian
DAT[, "SURV"]<-Surv(DAT[, "fuptime"], 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

- * **datasets**
 - HLProgFactors, [27](#)
- * **internal**
 - DescribeMetricBySplit-deprecated, [19](#)
 - findmode-deprecated, [24](#)
 - MetricPlotBySplit-deprecated, [39](#)
 - nice.hist-deprecated, [49](#)
 - OurTools-deprecated, [63](#)
- AlluvialPlot, [3](#)
- BinaryAssociation, [4](#)
- Colnames, [5](#), [19](#), [30](#)
- CompareMetricBySplit, [6](#)
- CompareSuccessRatesBySplit, [7](#)
- ConfusionTableStatistics, [8](#)
- CorDiffCI, [9](#), [87](#)
- CountBarplot, [10](#)
- CountValue, [11](#)
- Cox2Df, [12](#)
- CrossTabs, [12](#), [24](#), [41](#)
- CumIncPlot, [14](#)
- DataMaturityPlot, [17](#)
- Dates2Character, [18](#)
- DescribeMetricBySplit
 - (OurTools-deprecated), [63](#)
- DescribeMetricBySplit-deprecated, [19](#)
- Df2Csv, [20](#)
- Df2Rdata, [21](#)
- droplevels, [48](#)
- EQ (Positive_Comparators), [78](#)
- FacTable, [22](#)
- FactorsBySplit, [13](#), [23](#)
- findmode (OurTools-deprecated), [63](#)
- findmode-deprecated, [24](#)
- FollowUpPlot, [25](#)
- GE (Positive_Comparators), [78](#)
- GE_LE
 - (Positive_IsInInterval_Checkfunctions), [79](#)
- GE_LT
 - (Positive_IsInInterval_Checkfunctions), [79](#)
- Glm2Df, [26](#)
- GT (Positive_Comparators), [78](#)
- GT_LE
 - (Positive_IsInInterval_Checkfunctions), [79](#)
- GT_LT
 - (Positive_IsInInterval_Checkfunctions), [79](#)
- hist, [54](#), [65](#)
- HLProgFactors, [27](#)
- Is.Outlier, [28](#)
- LE (Positive_Comparators), [78](#)
- Level2NA, [29](#)
- Levels, [5](#), [30](#)
- LhrPlot, [30](#)
- LocationScaleEstimates, [32](#)
- LT (Positive_Comparators), [78](#)
- MakeCstring, [33](#)
- MakeInteractionFactor, [33](#)
- MartingalePlot, [34](#)
- Matrix2Df, [36](#)
- MergeWithMaster, [37](#)
- MetricBySplit, [38](#)
- MetricPlotBySplit
 - (OurTools-deprecated), [63](#)
- MetricPlotBySplit-deprecated, [39](#)
- MetricsBySplit, [40](#), [64](#), [65](#)
- MissingRateTable, [42](#)
- ModalValue, [42](#), [64](#)
- MoveAfter, [43](#)
- MultiParameterTimeCourses, [44](#)
- MultipleEndpointComparisonCox, [46](#)
- MultipleEndpointComparisonLogist, [47](#)
- NAasLevel, [48](#)
- NE (Positive_Comparators), [78](#)
- nice.hist (OurTools-deprecated), [63](#)
- nice.hist-deprecated, [49](#)

NiceCumIncPlot, [50](#)
NiceHist, [53](#), [64](#)
NicePvalue, [55](#)
NiceSurvPlot, [56](#)
NiceVennDiagram, [58](#)
Nmissing, [59](#)
NoUmlaute, [60](#)
Nvalid, [61](#)

OneGroupTimeCourse, [61](#)
OurTools-deprecated, [63](#)

PlotAxis, [45](#), [66](#)
PlotBinaryByMetric, [67](#)
PlotDepMetrics, [69](#)
PlotEndpointForestCox, [70](#)
PlotEndpointForestLogist, [71](#)
PlotMetricBySplit, [64](#), [72](#)
PlotOrdinalCourse, [74](#)
PlotSubgroupForestCox, [76](#)
PlotSubgroupForestLogist, [77](#)
Positive_Comparators, [78](#)
Positive_IsInInterval_Checkfunctions, [79](#)
ProversionProb, [79](#)
PruneVarName, [80](#)
PseudoMedian, [81](#)

QuantilesSurvivalCurves, [82](#)

RandomDate, [83](#)
ReconcileLists, [84](#)

ScaleTransformations, [84](#)
ScatterPlot, [85](#)
SetUpAnalysisDirs, [88](#)
SetUpDir, [88](#), [89](#)
SlidingWindow, [89](#)
Source, [90](#)
SubgroupAnalysisCox, [91](#)
SubgroupAnalysisLogist, [92](#)
SuccessRate, [93](#)
SurvivalAtT, [94](#)
SurvivalRateVersusTargetPvalue, [95](#)
SurvPlot, [96](#)
SymLine, [87](#), [98](#)

T_Log_B (ScaleTransformations), [84](#)
T_Logit (ScaleTransformations), [84](#)
T_Power_Z (ScaleTransformations), [84](#)
Table, [99](#)
ToInitialUpper, [100](#)
TrimStr, [101](#)

UnivariateCoxTable, [102](#)
VariablesBySplit, [103](#)
VisualiseCoxModel, [104](#)