

The Epi package, version 0.9.5

The following is a typeset version of the manual pages for the commands available in the **Epi** package. This document is available with the package as a vignette, along with other documents describing special features in more detail.

Contents

apc.fit	3
apc.frame	5
apc.lines	7
apc.plot	8
bdendo	9
bdendo11	10
births	10
blcaIT	11
brv	11
cal.yr	12
ccwc	13
ci.cum	14
ci.lin	15
ci.pd	17
contr.cum	18
cutLexis	19
detrend	20
diet	21
DMconv	21
effx.match	22
effx	23
ewrates	25
ex1	25
expand.data	26
fcut	27
fcut1	28
fit.add	29
fit.baseline	30
fit.mult	31
float	32
ftrend	33
gmortDK	34
hivDK	35
Icens	36
icut	37
isec	38

lep	39
Lexis.diagram	40
Lexis.lines	42
Lexis	43
Life.lines	45
lungDK	46
merge.data.frame	46
merge.Lexis	47
mh	48
mortDK	49
ncut	50
nice	51
nickel	52
pctab	52
plot.Lexis	53
plotEst	54
plotevent	56
projection.ip	57
rateplot	57
Relevel	61
ROC	61
S.typh	63
splitLexis	63
start.Lexis	65
stattable.funs	66
stat.table	67
subset.Lexis	69
tab	69
tabplot	70
thoro	71
timeBand	72
timeScales	73
twoby2	74
W.Lexis	75

apc.fit

Fit an Age-Period-Cohort model to tabular data.

Description

Fits the classical five models to tabulated rate data (cases, person-years) classified by two of age, period, cohort: Age, Age-drift, Age-Period, Age-Cohort and Age-period. There are no assumptions about the age, period or cohort classes being of the same length, or that tabulation should be only by two of the variables. Only requires that mean age and period for each tabulation unit is given.

Usage

```
apc.fit( data,
         A,
         P,
         D,
         Y,
         ref.c,
         ref.p,
         model = c("ns", "bs", "ls", "factor"),
         dr.extr = c("weighted", "Holford"),
         parm = c("ACP", "APC", "AdCP", "AdPC", "Ad-P-C", "Ad-C-P", "AC-P", "AP-C"),
         npar = c( A=5, P=5, C=5 ),
         scale = 1,
         alpha = 0.05,
         print.AOV = TRUE )
```

Arguments

data	Data frame with (at least) variables, A (age), P (period), D (cases, deaths) and Y (person-years). Cohort (date of birth) is computed as P-A . If this argument is given the arguments A , P , D and Y are ignored.
A	Age; numerical vector with mean age at diagnosis for each unit.
P	Period; numerical vector with mean date of diagnosis for each unit.
D	Cases, deaths; numerical vector.
Y	Person-years; numerical vector.
ref.c	Reference cohort, numerical. Defaults to median date of birth among cases. If used with parm ="AdCP" or parm ="AdPC", the residual cohort effects will be 1 at ref.c
ref.p	Reference period, numerical. Defaults to median date of diagnosis among cases.
model	Type of model fitted: <ul style="list-style-type: none"> • bs fits a model with B-splines for each of the terms, with np parameters for the terms. • ns fits a model with natural splines for each of the terms, with np parameters for the terms. • ls fits a model with linear splines. • factor fits a factor model with one parameter per value of A, P and C. np is ignored in this case.
dr.extr	How the drift parameter should be extracted from the age-period-cohort model. weighted (default) lets the weighted average (by marginal no. cases, D) of the estimated period and cohort effects have 0 slope. Holford uses the naive average over all values for the estimated effects, disregarding the no. cases.
parm	The parametrization of the effects. The first four all refer to the ML-fit of the Age-Period-Cohort model, the last four give Age-effects from a smaller model and residuals relative to this. If one of the latter is chosen, drift is ignored. Possible values are:

	<ul style="list-style-type: none"> • ACP: ML-estimates. Age-effects as rates for the reference cohort. Cohort effects as RR relative to the reference cohort. Period effects constrained to be 0 on average with 0 slope. • APC: ML-estimates. Age-effects as rates for the reference period. Period effects as RR relative to the reference period. Cohort effects constrained to be 0 on average with 0 slope. • AdCP: ML-estimates. Age-effects as rates for the reference cohort. Cohort and period effects constrained to be 0 on average with 0 slope. These effects do not multiply to the fitted rates, the drift is missing and needs to be included to produce the fitted values. • AdPC: ML-estimates. Age-effects as rates for the reference period. Cohort and period effects constrained to be 0 on average with 0 slope. These effects do not multiply to the fitted rates, the drift is missing and needs to be included to produce the fitted values. • Ad-C-P: Age effects are rates for the reference cohort in the Age-drift model (cohort drift). Cohort effects are from the model with cohort alone, using <code>log(fitted values)</code> from the Age-drift model as offset. Period effects are from the model with period alone using <code>log(fitted values)</code> from the cohort model as offset. • Ad-P-C: Age effects are rates for the reference period in the Age-drift model (period drift). Period effects are from the model with period alone, using <code>log(fitted values)</code> from the Age-drift model as offset. Cohort effects are from the model with cohort alone using <code>log(fitted values)</code> from the period model as offset. • AC-P: Age effects are rates for the reference cohort in the Age-Cohort model, cohort effects are RR relative to the reference cohort. Period effects are from the model with period alone, using <code>log(fitted values)</code> from the Age-Cohort model as offset. • AP-C: Age effects are rates for the reference period in the Age-Period model, period effects are RR relative to the reference period. Cohort effects are from the model with cohort alone, using <code>log(fitted values)</code> from the Age-Period model as offset.
npar	The number of parameters to use for each of the terms in the model. It can be a list of three numerical vectors, in which case these taken as the knots for the age, period and cohort effect, the first and last element in each vector are used as the boundary knots.
alpha	The significance level. Estimates are given with <code>(1-alpha)</code> confidence limits.
scale	<code>numeric(1)</code> , factor multiplied to the rate estimates before output.
print.AOV	Should the analysis of deviance table for the models be printed?

Value

An object of class "apc" (recognized by `apc.lines` and `apc.plot`) — a list with components:

Age	Matrix with 4 cols: A.pt with the ages (equals <code>unique(A)</code>) and three columns giving the estimated rates with c.i.s.
Per	Matrix with 4 cols: P.pt with the dates of diagnosis (equals <code>unique(P)</code>) and three columns giving the estimated RRs with c.i.s.
Coh	Matrix with 4 cols: C.pt with the dates of birth (equals <code>unique(P-A)</code>) and three columns giving the estimated RRs with c.i.s.
Drift	A 3 column matrix with drift-estimates and c.i.s: The first row is the ML-estimate of the drift (as defined by <code>drift</code>), the second row is the estimate from the Age-drift model. For the sequential parametrizations, only the latter is given.
Ref	Numerical vector of length 2 with reference period and cohort. If <code>ref.p</code> or <code>ref.c</code> was not supplied the corresponding element is NA.
AOV	Analysis of deviance table comparing the five classical models.
Type	Character string explaining the model and the parametrization.
Knots	If <code>model</code> is one of "ns" or "bs", a list with three components: Age , Per , Coh , each one a vector of knots. The max and the min are the boundary knots.

Author(s)

Bendix Carstensen, <http://www.biostat.ku.dk/~bxc>

References

The considerations behind the parametrizations used in this function are given in details in a preprint from Department of Biostatistics in Copenhagen:

<http://www.pubhealth.ku.dk/bs/publikationer/rr-06-1.pdf>, later published as: B. Carstensen: Age-period-cohort models for the Lexis diagram. Statistics in Medicine, 10; 26(15):3018-45, 2007.

See Also

[apc.frame](#), [apc.lines](#), [apc.plot](#).

Examples

```
library( Epi )
data(lungDK)

# Taylor a dataframe that meets the requirements
exd <- lungDK[,c("Ax","Px","D","Y")]
names(exd)[1:2] <- c("A","P")

# Two different ways of parametrizing the APC-model, ML
ex.H <- apc.fit( exd, npar=7, model="ns", dr.extr="Holford", parm="ACP", scale=10^5 )
ex.W <- apc.fit( exd, npar=7, model="ns", dr.extr="weighted", parm="ACP", scale=10^5 )

# Sequential fit, first AC, then P given AC.
ex.S <- apc.fit( exd, npar=7, model="ns", parm="AC-P", scale=10^5 )

# Show the estimated drifts
ex.H[["Drift"]]
ex.W[["Drift"]]
ex.S[["Drift"]]

# Plot the effects
fp <- apc.plot( ex.H )
apc.lines( ex.W, frame.par=fp, col="red" )
apc.lines( ex.S, frame.par=fp, col="blue" )
```

apc.frame

Produce an empty frame for display of parameter-estimates from Age-Period-Cohort-models.

Description

A plot is generated where both the age-scale and the cohort/period scale is on the x-axis. The left vertical axis will be a logarithmic rate scale referring to age-effects and the right a logarithmic rate-ratio scale of the same relative extent as the left referring to the cohort and period effects (rate ratios).

Only an empty plot frame is generated. Curves or points must be added with **points**, **lines** or the special utility function [apc.lines](#).

Usage

```
apc.frame( a.lab,
           cp.lab,
           r.lab,
           rr.lab = r.lab / rr.ref,
           rr.ref = r.lab[length(r.lab)/2],
```

```

    a.tic = a.lab,
    cp.tic = cp.lab,
    r.tic = r.lab,
    rr.tic = r.tic / rr.ref,
    tic.fac = 1.3,
    a.txt = "Age",
    cp.txt = "Calendar time",
    r.txt = "Rate per 100,000 person-years",
    rr.txt = "Rate ratio",
    gap = diff(range(c(a.lab, a.tic)))/3,
    col.grid = gray(0.85),
    sides = c(1,2,4) )

```

Arguments

a.lab	Numerical vector of labels for the age-axis.
cp.lab	Numerical vector of labels for the cohort-period axis.
r.lab	Numerical vector of labels for the rate-axis (left vertical)
rr.lab	Numerical vector of labels for the RR-axis (right vertical)
rr.ref	At what level of the rate scale is the RR=1 to be.
a.tic	Location of additional tick marks on the age-scale
cp.tic	Location of additional tick marks on the cohort-period-scale
r.tic	Location of additional tick marks on the rate-scale
rr.tic	Location of additional tick marks on the RR-axis.
tic.fac	Factor with which to diminish intermediate tick marks
a.txt	Text for the age-axis (left part of horizontal axis).
cp.txt	Text for the cohort/period axis (right part of horizontal axis).
r.txt	Text for the rate axis (left vertical axis).
rr.txt	Text for the rate-ratio axis (right vertical axis)
gap	Gap between the age-scale and the cohort-period scale
col.grid	Colour of the grid put in the plot.
sides	Numerical vector indicating on which sides axes should be drawn and annotated. This option is aimed for multi-panel displays where axes only are put on the outer plots.

Details

The function produces an empty plot frame for display of results from an age-period-cohort model, with age-specific rates in the left side of the frame and cohort and period rate-ratio parameters in the right side of the frame. There is a gap of **gap** between the age-axis and the calendar time axis, vertical grid lines at `c(a.lab,a.tic,cp.lab,cp.tic)`, and horizontal grid lines at `c(r.lab,r.tic)`.

The function returns a numerical vector of length 2, with names `c("cp.offset", "RR.fac")`. The y-axis for the plot will be a rate scale for the age-effects, and the x-axis will be the age-scale. The cohort and period effects are plotted by subtracting the first element (named `"cp.offset"`) of the returned result from the cohort/period, and multiplying the rate-ratios by the second element of the returned result (named `"RR.fac"`).

Value

A numerical vector of length two, with names `c("cp.offset", "RR.fac")`. The first is the offset for the cohort period-axis, the second the multiplication factor for the rate-ratio scale.

Side-effect: A plot with axes and grid lines but no points or curves.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.pubhealth.ku.dk/~bxc/>

References

<http://www.pubhealth.ku.dk/~bxc/APC/notes.pdf>

See Also

[apc.lines](#), [apc.lines](#)

Examples

```
par( mar=c(4,4,1,4) )
res <-
apc.frame( a.lab=seq(30,90,20), cp.lab=seq(1880,2000,30), r.lab=c(1,2,5,10,20,50),
           a.tic=seq(30,90,10), cp.tic=seq(1880,2000,10), r.tic=c(1:10,1:5*10),
           gap=27 )

res
# What are the axes actually?
par(c("usr","xlog","ylog"))
# How to plot in the age-part: a point at (50,10)
points( 50, 10, pch=16, cex=2, col="blue" )
# How to plot in the cohort-period-part: a point at (1960,0.3)
points( 1960-res[1], 0.3*res[2], pch=16, cex=2, col="red" )
```

apc.lines

Plot APC-estimates in an APC-frame.

Description

When an APC-frame has been produced by [apc.frame](#), this function draws a set of estimates from an APC-fit in the frame. An optional drift parameter can be added to the period parameters and subtracted from the cohort and age parameters.

Usage

```
apc.lines( A, P, C,
           scale = c("log","ln","rates","inc","RR"),
           frame.par = NULL,
           drift = 0,
           c0 = median( C[,1] ),
           a0 = median( A[,1] ),
           p0 = c0 + a0,
           ci = rep( FALSE, 3 ),
           lwd = c(3,1,1),
           lty = 1,
           col = "black",
           type = "l",
           knots = FALSE,
           ... )
```

Arguments

- | | |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A | Age effects. A 4-column matrix with columns age, age-specific rates, lower and upper c.i. If A is of class <code>apc</code> (see apc.fit , P, C, c0, a0 and p0 are ignored, and the estimates from there plotted. |
| P | Period effects. Rate-ratios. Same form as for the age-effects. |
| C | Cohort effects. Rate-ratios. Same form as for the age-effects. |

<code>scale</code>	Are effects given on a log-scale? Character variable, one of "log", "ln", "rates", "inc", "RR". If "log" or "ln" it is assumed that effects are log(rates) and log(RRs) otherwise the actual effects are assumed given in A, P and C. If A is of class <code>apc</code> , it is assumed to be "rates".
<code>frame.par</code>	2-element vector with the cohort-period offset and RR multiplier. This will typically be the result from the call of <code>apc.frame</code> . See this for details.
<code>drift</code>	The drift parameter to be added to the period effect. If <code>scale="log"</code> this is assumed to be on the log-scale, otherwise it is assumed to be a multiplicative factor per unit of the first columns of A, P and C
<code>c0</code>	The cohort where the drift is assumed to be 0; the subtracted drift effect is <code>drift*(C[,1]-c0)</code> .
<code>a0</code>	The age where the drift is assumed to be 0.
<code>p0</code>	The period where the drift is assumed to be 0.
<code>ci</code>	Should confidence interval be drawn. Logical or character. If character, any occurrence of "a" or "A" produces confidence intervals for the age-effect. Similarly for period and cohort.
<code>lwd</code>	Line widths for estimates, lower and upper confidence limits.
<code>lty</code>	Linetypes for the three effects.
<code>col</code>	Colours for the three effects.
<code>type</code>	What type of lines / points should be used.
<code>knots</code>	Should knots from the model be shown?
<code>...</code>	Further parameters to be transmitted to <code>matlines</code> used for plotting the three sets of curves.

Details

The drawing of three effects in an APC-frame is a rather trivial task, and the main purpose of the utility is to provide a function that easily adds the functionality of adding a drift so that several sets of lines can be easily produced in the same frame.

Value

A list of three matrices with the effects plotted is returned invisibly.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.pubhealth.ku.dk/~bxc>

See Also

`\code{apc.frame}`, `\code{apc.frame}`, `\code{apc.frame}`

`apc.plot`

Plot the estimates from a fitted Age-Period-Cohort model

Description

This function plots the estimates created by `apc.fit` in a single graph. It just calls `apc.frame` after computing some sensible values of the parameters, and subsequently plots the estimates using `apc.lines`.

Usage

```
apc.plot(obj, r.txt = "Rate", ...)
```


Arguments

<code>obj</code>	An object of class <code>apc</code> .
<code>r.txt</code>	The text to put on the vertical rate axis.
<code>...</code>	Additional arguments passed on to <code>apc.lines</code> .

Value

A numerical vector of length two, with names `c("cp.offset", "RR.fac")`. The first is the offset for the cohort period-axis, the second the multiplication factor for the rate-ratio scale. Therefore, if you want to plot at `(x,y)` in the right panel, use `(x-res["cp.offset"], y/res["RR.fac"])=(x-res[1], y/res[2])`. This vector should be supplied for the parameter `frame.par` to `apc.lines` if more sets of estimates is plotted in the same graph.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.pubhealth.ku.dk/~bxc>

See Also

`\code{apc.lines}` , `\code{apc.lines}`, `\code{apc.lines}`

Examples

```
data( lungDK )
attach( lungDK )
apc1 <- apc.fit( A=Ax, P=Px, D=D, Y=Y/10^5 )
fp <- apc.plot( apc1 )
apc.lines( apc1, frame.par=fp, drift=1.01, col="red" )
for( i in 1:11 )
  apc.lines( apc1, frame.par=fp, drift=1+(i-6)/100, col=rainbow(12)[i] )
```

bdendo
A case-control study of endometrial cancer

Description

The **bdendo** data frame has 315 rows and 13 columns. These data concern a study in which each case of endometrial cancer was matched with 4 controls. Matching was by date of birth (within one year), marital status, and residence.

Format

This data frame contains the following columns:

set:	Case-control set: a numeric vector
d:	Case or control: a numeric vector (1=case, 0=control)
gall:	Gall bladder disease: a factor with levels No Yes .
hyp:	Hypertension: a factor with levels No Yes .
ob:	Obesity: a factor with levels No Yes .
est:	A factor with levels No Yes .
dur:	Duration of conjugated oestrogen therapy: an ordered factor with levels 0 < 1 < 2 < 3 < 4 .
non:	Use of non oestrogen drugs: a factor with levels No Yes .
duration:	Months of oestrogen therapy: a numeric vector.
age:	A numeric vector.
cest:	Conjugated oestrogen dose: an ordered factor with levels 0 < 1 < 2 < 3 .
agegrp:	A factor with levels 55-59 60-64 65-69 70-74 75-79 80-84
age3:	a factor with levels <64 65-74 75+

Source

Breslow NE, and Day N, Statistical Methods in Cancer Research. Volume I: The Analysis of Case-Control Studies. IARC Scientific Publications, IARC:Lyon, 1980.

Examples

```
data(bdendo)
```

bdendo11	<i>A 1:1 subset of the endometrial cancer case-control study</i>
-----------------	------------------------------------------------------------------

Description

The **bdendo11** data frame has 126 rows and 13 columns. This is a subset of the dataset **bdendo** in which each case was matched with a single control.

Source

Breslow NE, and Day N, Statistical Methods in Cancer Research. Volume I: The Analysis of Case-Control Studies. IARC Scientific Publications, IARC:Lyon, 1980.

Examples

```
data(bdendo11)
```

births	<i>Births in a London Hospital</i>
---------------	------------------------------------

Description

Data from 500 singleton births in a London Hospital

Usage

```
data(births)
```

Format

A data frame with 500 observations on the following 8 variables.

id:	Identity number for mother and baby.
bweight:	Birth weight of baby.
lowbw:	Indicator for birth weight less than 2500 g.
gestwks:	Gestation period.
preterm:	Indicator for gestation period less than 37 weeks.
matage:	Maternal age.
hyp:	Indicator for maternal hypertension.
sex:	Sex of baby: 1:Male, 2:Female.

Source

Anonymous

References

Michael Hills and Bianca De Stavola (2002). A Short Introduction to Stata 8 for Biostatistics, Timberlake Consultants Ltd <http://www.timberlake.co.uk>

Examples

```
data(births)
```

```
blcaIT
```

```
Bladder cancer mortality in Italian males
```

Description

Number of deaths from bladder cancer and person-years in the Italian male population 1955–1979, in ages 25–79.

Format

A data frame with 55 observations on the following 4 variables:

```
age:  Age at death. Left endpoint of age class
period:  Period of death. Left endpoint of period
D:  Number of deaths
Y:  Number of person-years.
```

Examples

```
data(blcaIT)
```

```
brv
```

```
Bereavement in an elderly cohort
```

Description

The **brv** data frame has 399 rows and 11 columns. The data concern the possible effect of marital bereavement on subsequent mortality. They arose from a survey of the physical and mental health of a cohort of 75-year-olds in one large general practice. These data concern mortality up to 1 January, 1990 (although further follow-up has now taken place).

Subjects included all lived with a living spouse when they entered the study. There are three distinct groups of such subjects: (1) those in which both members of the couple were over 75 and therefore included in the cohort, (2) those whose spouse was below 75 (and was not, therefore, part of the main cohort study), and (3) those living in larger households (that is, not just with their spouse).

Format

This data frame contains the following columns:

```
id:  subject identifier, a numeric vector
couple:  couple identifier, a numeric vector
dob:  date of birth, a date
doe:  date of entry into follow-up study, a date
dox:  date of exit from follow-up study, a date
dosp:  date of death of spouse, a date (if the spouse was still alive at the end of follow-up, this was coded to January 1, 2000)
fail:  status at end of follow-up, a numeric vector (0=alive,1=dead)
group:  see Description, a numeric vector
disab:  disability score, a numeric vector
health:  perceived health status score, a numeric vector
sex:  a factor with levels Male Female
```

Source

Jagger C, and Sutton CJ, Death after Marital Bereavement. Statistics in Medicine, 10:395-404, 1991. (Data supplied by Carol Jagger).

Examples

```
data(brv)
```

<code>cal.yr</code>	<i>Functions to convert character, factor and various date objects into a number, and vice versa.</i>
---------------------	-------------------------------------------------------------------------------------------------------

Description

Dates are converted to a numerical value, giving the calendar year as a fractional number. 1 January 1970 is converted to 1970.0, and other dates are converted by assuming that years are all 365.25 days long, so inaccuracies may arise, for example, 1 Jan 2000 is converted to 1999.999. Differences between converted values will be 1/365.25 of the difference between corresponding `Date` objects.

Usage

```
cal.yr(x, format)
as.Date.cal.yr( x, ... )
as.Date.numeric( x, ..., unit="d" )
```

Arguments

<code>x</code>	A factor or character vector, representing a date in format <code>format</code> , or an object of class <code>Date</code> , <code>POSIXlt</code> , <code>POSIXct</code> , <code>date</code> , <code>dates</code> or <code>chron</code> (the latter two requires the <code>chron</code> package). If <code>x</code> is a data frame, all variables in the data-frame which are of the classes mentioned are converted to class <code>cal.yr</code> .
<code>format</code>	Format of the date values if <code>x</code> is factor or character
<code>unit</code>	Which units are the date measured in, "y" for years, "d" for days.
<code>...</code>	Arguments passed on from other methods.

Value

`cal.yr` returns a numerical vector of the same length as `x`, of class `c("cal.yr", "numeric")`.
`as.Date.cal.yr` and `as.Date.numeric` return `Date` objects.

Author(s)

Bendix Carstensen, Steno Diabetes Center & Dept. of Biostatistics, University of Copenhagen, (bx@steno.dk), <http://www.pubhealth.ku.dk/~bx>

See Also

[DateTimeClasses](#), [Date](#)

Examples

```
# Character vector of dates:
birth <- c("14/07/1852", "01/04/1954", "10/06/1987", "16/05/1990",
          "01/01/1996", "01/01/1997", "01/01/1998", "01/01/1999")
# Proper conversion to class "Date":
birth.dat <- as.Date( birth, format="%d/%m/%Y" )
# Conversion of character to class "cal.yr"
bt.yr <- cal.yr( birth, format="%d/%m/%Y" )
```

```

# Back to class "Date":
bt.dat <- as.Date( bt.yr )
# Numerical calculation of days since 1.1.1970:
days <- Days <- (bt.yr-1970)*365.25
# Blunt assignment of class:
class( Days ) <- "Date"
# Then data.frame() to get readable output of results:
data.frame( birth, birth.dat, bt.yr, bt.dat, days, Days, round(Days) )

```

ccwc

Generate a nested case-control study

Description

Given the basic outcome variables for a cohort study: the time of entry to the cohort, the time of exit and the reason for exit ("failure" or "censoring"), this function computes risk sets and generates a matched case-control study in which each case is compared with a set of controls randomly sampled from the appropriate risk set. Other variables may be matched when selecting controls.

Usage

```
ccwc(entry=0, exit, fail, origin=0, controls=1, match=list(), include=list(), data=NULL, silent=F)
```

Arguments

entry	Time of entry to follow-up
exit	Time of exit from follow-up
fail	Status on exit (1=Fail, 0=Censored)
origin	Origin of analysis time scale
controls	The number of controls to be selected for each case
match	List of categorical variables on which to match cases and controls
include	List of other variables to be carried across into the case-control study
data	Data frame in which to look for input variables
silent	If False, echos a . to the screen for each case-control set created; otherwise produces no output.

Value

The case-control study, as a dataframe containing:

Set	case-control set number
Map	row number of record in input dataframe
Time	failure time of the case in this set
Fail	failure status (1=case, 0=control)

These are followed by the matching variables, and finally by the variables in the **include** list

Author(s)

David Clayton

References

Clayton and Hills, Statistical Models in Epidemiology, Oxford University Press, Oxford:1993.

See Also

[Lexis](#)

Examples

```
#
# For the diet and heart dataset, create a nested case-control study
# using the age scale and matching on job
#
data(diet)
dietcc <- ccwc(doe, dox, chd, origin=dob, controls=2, data=diet,
               include=energy, match=job)
```

<code>ci.cum</code>	<i>Compute cumulative sum of estimates.</i>
---------------------	---------------------------------------------

Description

Computes the cumulative sum of parameter functions and the standard error of it. Optionally the exponential is applied to the parameter functions before it is cumulated.

Usage

```
ci.cum( obj,
        ctr.mat = NULL,
        subset = NULL,
        intl = 1,
        alpha = 0.05,
        Exp = TRUE )
```

Arguments

<code>obj</code>	A model object.
<code>ctr.mat</code>	Contrast matrix defining the parameter functions from the parameters of the model.
<code>subset</code>	Subset of the parameters of the model to which <code>ctr.mat</code> should be applied.
<code>intl</code>	Interval length for the cumulation. Either a constant or a numerical vector of length <code>nrow(ctr.mat)</code> .
<code>alpha</code>	Significance level used when computing confidence limits.
<code>Exp</code>	Should the parameter function be exponentiated before it is cumulated?

Details

The purpose of this function is to compute cumulative rate based on a model for the rates. If the model is a multiplicative model for the rates, the purpose of `ctr.mat` is to return a vector of rates or log-rates when applied to the coefficients of the model. If log-rates are returned, they should be exponentiated before cumulated, and the variances computed accordingly. Since log-linear models are the most common the `Exp` parameter defaults to `TRUE`.

Value

A matrix with 4 columns: Estimate, lower and upper c.i. and standard error.

Author(s)

Bendix Carstensen, <http://www.pubhealth.ku.dk/~bxc>

See Also

See also [ci.lin](#)

Examples

```
# Packages required for this example
library( splines )
library( survival )
data( lung )
par( mfrow=c(1,2) )

# Plot the Kaplan-meier-estimator
plot( survfit( Surv( time, status==2 ), data=lung ) )

# Declare data as Lexis
lungL <- Lexis( exit=list("tfd"=time),
               exit.status=(status==2)*1, data=lung )
tab.Lexis( lungL )

# Cut the follow-up every 10 days
sL <- splitLexis( lungL, "tfd", breaks=seq(0,1100,10) )
str( sL )
tab.Lexis( sL )

# Fit a Poisson model with a natural spline for the effect of time.
# Extract the variables needed
D <- status(sL)[,"exit"]
Y <- dur(sL)
tB <- timeBand( sL, "tfd", "left" )
MM <- ns( tB, knots=c(50,100,200,400,700), intercept=TRUE )
mp <- glm( D ~ MM - 1 + offset(log(Y)),
          family=poisson, eps=10^-8, maxit=25 )

# Contrast matrix to extract effects, i.e. matrix to multiply with the
# coefficients to produce the log-rates: unique rows of MM, in time order.
T.pt <- sort( unique( tB ) )
T.wh <- match( T.pt, tB )
Lambda <- ci.cum( mp, ctr.mat=MM[T.wh,], intl=diff(c(0,T.pt)) )

# Put the estimated survival function on top of the KM-estimator
matlines( c(0,T.pt[-1]), exp(-Lambda[,1:3]), lwd=c(3,1,1), lty=1, col="Red" )

# Extract and plot the fitted intensity function
lambda <- ci.lin( mp, ctr.mat=MM[T.wh,], Exp=TRUE )
matplot( T.pt, lambda[,5:7]*10^3, type="l", lwd=c(3,1,1), col="black", lty=1,
        log="y", ylim=c(0.2,20) )
```

Description

For a given model object the function computes a linear function of the parameters and the corresponding standard errors, p-values and confidence intervals.

Usage

```
ci.lin( obj,
        ctr.mat = NULL,
        subset = NULL,
        diffs = FALSE,
        fnam = !diffs,
        vcov = FALSE,
        alpha = 0.05,
        Exp = FALSE )
```

Arguments

obj	A model object (of class <code>lm</code> , <code>glm</code> , <code>lme</code> , <code>coxph</code> or <code>polr</code>).
ctr.mat	Contrast matrix to be multiplied to the parameter vector, i.e. the desired linear function of the parameters.
subset	The subset of the parameters to be used. If given as a character vector, the elements are in turn matched against the parameter names (using <code>grep</code>) to find the subset. Repeat parameters may result from using a character vector. This is considered a facility.
diffs	If TRUE, all differences between parameters in the subset are computed. <code>ctr.mat</code> is ignored. If <code>obj</code> inherits from <code>lm</code> , and <code>subset</code> is given as a string <code>subset</code> is used to search among the factors in the model and differences of all factor levels for the first match are shown. If <code>subset</code> does not match any of the factors in the model, all pairwise differences between parameters matching are returned.
fnam	Should the common part of the parameter names be included with the annotation of contrasts? Ignored if <code>diffs==T</code> . If a sting is supplied this will be prefixed to the labels.
vcov	Should the covariance matrix of the set of parameters be returned? If this is set, <code>Exp</code> is ignored.
alpha	Significance level for the confidence intervals.
Exp	If TRUE columns 5:6 are replaced with <code>exp(columns 1,5,6)</code> .

Value

A matrix with number of rows and rownames as `ctr.mat`. The columns are Estimate, Std.Err, z, P, 2.5% and 97.5%. If `vcov=TRUE` a list with components `est`, the desired functional of the parameters and `vcov`, the variance covariance matrix of this, is returned but not printed. If `codeExp==TRUE` the confidence intervals for the parameters are replaced with three columns: `exp(estimate,c.i.)`.

Author(s)

Bendix Carstensen, <http://www.pubhealth.ku.dk/~bxc>

Examples

```
# Bogus data:
f <- factor( sample( letters[1:5], 200, replace=TRUE ) )
g <- factor( sample( letters[1:3], 200, replace=TRUE ) )
x <- rnorm( 200 )
y <- 7 + as.integer( f ) * 3 + 2 * x + 1.7 * rnorm( 200 )

# Fit a simple model:
mm <- lm( y ~ x + f + g )
ci.lin( mm )
ci.lin( mm, subset=3:6, diff=TRUE, fnam=FALSE )
ci.lin( mm, subset=3:6, diff=TRUE, fnam=TRUE )
ci.lin( mm, subset="f", diff=TRUE, fnam="f levels:" )
print( ci.lin( mm, subset="g", diff=TRUE, fnam="gee!", vcov=TRUE ) )
```



```
# Use character defined subset to get ALL contrasts:
ci.lin( mm, subset="f", diff=TRUE )
```

ci.pd	<i>Compute confidence limits for a difference of two independent proportions.</i>
-------	-----------------------------------------------------------------------------------

Description

The usual formula for the c.i. of at difference of proportions is inaccurate. Newcombe has compared 11 methods and method 10 in his paper looks like a winner. It is implemented here.

Usage

```
ci.pd(aa, bb=NULL, cc=NULL, dd=NULL,
      method = "Nc",
      alpha = 0.05, conf.level=0.95,
      digits = 3,
      print = TRUE,
      detail.labs = FALSE )
```

Arguments

aa	Numeric vector of successes in sample 1. Can also be a matrix or array (see details).
bb	Successes in sample 2.
cc	Failures in sample 1.
dd	Failures in sample 2.
method	Method to use for calculation of confidence interval, see "Details".
alpha	Significance level
conf.level	Confidence level
print	Should an account of the two by two table be printed.
digits	How many digits should the result be rounded to if printed.
detail.labs	Should the computing of probability differences be reported in the labels.

Details

Implements method 10 from Newcombe(1998) (method="Nc") or from Agresti & Caffo(2000) (method="AC").

aa, **bb**, **cc** and **dd** can be vectors. If **aa** is a matrix, the elements [1:2,1:2] are used, with successes **aa[,1:2]**. If **aa** is a three-way table or array, the elements **aa[1:2,1:2,]** are used.

Value

A matrix with three columns: probability difference, lower and upper limit. The number of rows equals the length of the vectors **aa**, **bb**, **cc** and **dd** or, if **aa** is a 3-way matrix, **dim(aa)[3]**.

Author(s)

Bendix Carstensen, Esa Laara. <http://www.biostat.ku.dk/~bxc>

References

RG Newcombe: Interval estimation for the difference between independent proportions. Comparison of eleven methods. *Statistics in Medicine*, 17, pp. 873-890, 1998.

A Agresti & B Caffo: Simple and effective confidence intervals for proportions and differences of proportions result from adding two successes and two failures. *The American Statistician*, 54(4), pp. 280-288, 2000.

See Also

[twoby2](#), [binom.test](#)

Examples

```
( a <- matrix( sample( 10:40, 4 ), 2, 2 ) )
ci.pd( a )
twoby2( t(a) )
prop.test( t(a) )
( A <- array( sample( 10:40, 20 ), dim=c(2,2,5) ) )
ci.pd( A )
ci.pd( A, detail.labs=TRUE, digits=3 )
```

`contr.cum`

Contrast matrices

Description

Return a matrix of contrasts for factor coding.

Usage

```
contr.cum(n)
contr.diff(n)
contr.2nd(n)
contr.orth(n)
```

Arguments

n A vector of levels for a factor, or the number of levels.

Details

These functions are used for creating contrast matrices for use in fitting regression models. The columns of the resulting matrices contain contrasts which can be used for coding a factor with **n** levels.

`contr.cum` gives a coding corresponding to successive differences between factor levels.

`contr.diff` gives a coding that correspond to the cumulative sum of the value for each level. This is not meaningful in a model where the intercept is included, therefore **n** columns are always returned.

`contr.2nd` gives contrasts corresponding to 2nd order differences between factor levels. Returns a matrix with **n-2** columns.

`contr.orth` gives a matrix with **n-2** columns, which are mutually orthogonal and orthogonal to the matrix `cbind(1,1:n)`

Value

A matrix with **n** rows and **k** columns, with **k=n** for `contr.diff` **k=n-1** for `contr.cum` **k=n-2** for `contr.2nd` and `contr.orth`.

Author(s)

Bendix Carstensen

See Also

[contr.treatment](#)

Examples

```
contr.cum(6)
contr.2nd(6)
contr.diff(6)
contr.orth(6)
```

cutLexis

Cuts follow-up at a specified date for each person.

Description

The follow-up of a person is divided into the period before and after an intermediate event, typically disease recurrence or something like that. This event is assigned a new state.

Usage

```
cutLexis( data,
          cut,
          timescale = timeScales(data)[1],
          new.state = max( as.integer( c(data$lex.Cst,data$lex.Xst) )+1 ),
          na.cut = Inf,
          cens = ifelse( is.character(new.state), "0", 0 ) )
```

Arguments

data	A Lexis object.
cut	Numerical vector with the times of intermediate event. NAs are allowed, in which cases the original record is retained.
timescale	The timescale that cut refers to.
new.state	What value should the new state have?
na.cut	What value should be substituted for the NAs in cut . The default is Inf which results in the original records being kept.
cens	Censoring values for the states. Persons in any of these states will have their entry and exit status (lex.Cst , lex.Xst , updated if the follow up is after cut .

Details

The **cens** parameter is used to decide whether an existing status (**lex.Cst** or **lex.Xst**) should be updated to **new.state** or not, if the corresponding entry or exit time is after **cut**.

Value

A [Lexis](#) object, where the follow-up of each person is cut in two at the time as given in **cut**.

Author(s)

Bendix Carstensen, Steno Diabetes Center, (bxc@steno.dk)

See Also

[splitLexis](#)

Examples

```
data( nickel )
nic <- Lexis( data=nickel,
              entry=list(age=agein),
              exit=list(age=ageout,cal=ageout+dob,tfh=ageout-age1st),
              # Lung cancer deaths and other deaths are coded 1 and 2
              exit.status=( icd > 0) + (icd %in% c(162,163)) ) )
str( nic )
tab.Lexis( nic )
nicc <- cutLexis( nic, 50, "age" )
tab.Lexis( nicc )
```

detrend	<i>Projection of a model matrix on to the orthogonal complement of a trend.</i>
---------	---------------------------------------------------------------------------------

Description

The columns of the model matrix **M** is projected on the orthogonal complement to the matrix **(1,t)**. Orthogonality is defined w.r.t. an inner product defined by the weights **weight**.

Usage

```
detrend( M, t, weight = rep(1, nrow(M)) )
```

Arguments

M	A model matrix.
t	The trend defining a subspace. A numerical vector of length nrow(M)
weight	Weights defining the inner product of vectors x and y as sum(x*w*y) . A numerical vector of length nrow(M) , defaults to a vector of 1s.

Details

The functions is intended to be used in parametrization of age-period-cohort models.

Value

A full-rank matrix with columns orthogonal to **(1,t)**.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.pubhealth.ku.dk/~bxc>, with help from Peter Dalgaard.

See Also

[projection.ip](#)

diet*Diet and heart data*

Description

The **diet** data frame has 337 rows and 14 columns. The data concern a subsample of subjects drawn from larger cohort studies of the incidence of coronary heart disease (CHD). These subjects had all completed a 7-day weighed dietary survey while taking part in validation studies of dietary questionnaire methods. Upon the closure of the MRC Social Medicine Unit, from where these studies were directed, it was found that 46 CHD events had occurred in this group, thus allowing a serendipitous study of the relationship between diet and the incidence of CHD.

Format

This data frame contains the following columns:

- id:** subject identifier, a numeric vector.
- doe:** date of entry into follow-up study, a [Date](#) variable.
- dox:** date of exit from the follow-up study, a [Date](#) variable.
- dob:** date of birth, a [Date](#) variable.
- y:** - number of years at risk, a numeric vector.
- fail:** status on exit, a numeric vector (codes 1, 3, 11, and 13 represent CHD events)
- job:** occupation, a factor with levels **Driver Conductor Bank worker**
- month:** month of dietary survey, a numeric vector
- energy:** total energy intake (KCal per day/100), a numeric vector
- height:** (cm), a numeric vector
- weight:** (kg), a numeric vector
- fat:** fat intake (g/day), a numeric vector
- fibre:** dietary fibre intake (g/day), a numeric vector
- energy.grp:** high daily energy intake, a factor with levels **<=2750 KCal >2750 KCal**
- chd:** CHD event, a numeric vector (1=CHD event, 0=no event)

Source

The data are described and used extensively by Clayton and Hills, Statistical Models in Epidemiology, Oxford University Press, Oxford:1993. They were rescued from destruction by David Clayton and reentered from paper printouts.

Examples

```
data(diet)
# Illustrate the follow-up in a Lexis diagram
Lexis.diagram( age=c(30,75), date=c(1965,1990),
               entry.date=cal.yr(doe), exit.date=cal.yr(dox), birth.date=cal.yr(dob),
               fail=(fail>0), pch.fail=c(NA,16), col.fail=c(NA,"red"), cex.fail=1.0,
               data=diet )
```

DMconv*Conversion to diabetes*

Description

Data from a randomized intervention study ("Addition") where persons with prediabetic conditions are followed up for conversion to diabetes (DM). Conversion dates are interval censored. Original data are not published yet, so id-numbers have been changed and all dates have been randomly perturbed.

Usage

```
data(DMconv)
```

Format

A data frame with 1519 observations on the following 6 variables.

id Person identifier

doe Date of entry, i.e. first visit.

dlw Date last seen well, i.e. last visit without DM.

dfi Date first seen ill, i.e. first visit with DM.

gtol Glucose tolerance. Factor with levels: 1="IFG" (impaired fasting glucose), 2="IGT" (impaired glucose tolerance).

grp Randomization. Factor with levels: 1="Intervention", 2="Control".

Source

Signe Saetre Rasmussen, Steno Diabetes Center. The Addition Study.

Examples

```
data(DMconv)
str(DMconv)
head(DMconv)
```

```
effx.match
```

Function to calculate effects for individually matched case-control studies

Description

The function calculates the effects of an exposure on a response, possibly stratified by a stratifying variable, and/or controlled for one or more confounding variables.

Usage

```
effx.match(response,
exposure,
match,
strata=NULL,
control=NULL,
base=1,
digits=3,
alpha=0.05,
data=NULL)
```

Arguments

response	The response variable - must be numeric
exposure	The exposure variable can be numeric or a factor
match	The variable which identifies the matched sets
strata	The strata stratifying variable - must be a factor
control	The control variable(s). These are passed as a list if there are more than one of them.
base	Baseline for the effects of a categorical exposure, default 1
digits	Number of significant digits for the effects, default 3
alpha	1 - confidence level
data	data refers to the data used to evaluate the function

Details

Effects are calculated odds ratios. The function is a wrapper for clogit, from the survival package. The k-1 effects for a categorical exposure with k levels are relative to a baseline which, by default, is the first level. The effect of a metric (quantitative) exposure is calculated per unit of exposure. The exposure variable can be numeric or a factor, but if it is an ordered factor the order will be ignored.

Value

comp1	Effects of exposure
comp2	Tests of significance

Author(s)

Michael Hills

References

www.mhills.pwp.blueyonder.co.uk

Examples

```
library(Epi)
library(survival)
data(bdendo)

# d is the case-control variable, set is the matching variable.
# The variable est is a factor and refers to estrogen use (no,yes)
# The variable hyp is a factor with 2 levels and refers to hypertension (no, yes)
# effect of est on the odds of being a case
effx.match(d,exposure=est,match=set,data=bdendo)
# effect of est on the odds of being a case, stratified by hyp
effx.match(d,exposure=est,match=set,strata=hyp,data=bdendo)
# effect of est on the odds of being a case, controlled for hyp
effx.match(d,exposure=est,match=set,control=hyp,data=bdendo)
```

effx

Function to calculate effects

Description

The function calculates the effects of an exposure on a response, possibly stratified by a stratifying variable, and/or controlled for one or more confounding variables.

Usage

```
effx( response, type = "metric",
      fup = NULL,
      exposure,
      strata = NULL,
      control = NULL,
      weights = NULL,
      alpha = 0.05,
      base = 1,
      digits = 3,
      data = NULL )
```

Arguments

response	The response variable - must be numeric
type	The type of responsetype - must be one of "metric", "binary", "failure", or "count"
fup	The fup variable contains the follow-up time for a failure response
exposure	The exposure variable can be numeric or a factor
strata	The strata stratifying variable - must be a factor
control	The control variable(s) - these are passed as a list if there are more than one.
weights	Frequency weights for binary response only
base	Baseline for the effects of a categorical exposure, default 1
digits	Number of significant digits for the effects, default 3
alpha	1 - confidence level
data	data refers to the data used to evaluate the function

Details

The function is a wrapper for glm. Effects are calculated as differences in means for a metric response, odds ratios for a binary response, and rate ratios for a failure or count response.

The k-1 effects for a categorical exposure with k levels are relative to a baseline which, by default, is the first level. The effect of a metric (quantitative) exposure is calculated per unit of exposure.

The exposure variable can be numeric or a factor, but if it is an ordered factor the order will be ignored.

Value

comp1	Effects of exposure
comp2	Tests of significance

Author(s)

Michael Hills

References

www.mhills.pwp.blueyonder.co.uk

Examples

```
library(Epi)
data(births)
births$hyp <- factor(births$hyp,labels=c("normal","hyper"))
births$sex <- factor(births$sex,labels=c("M","F"))

# bweight is the birth weight of the baby in gms, and is a metric
# response (the default)

# effect of hypertension on birth weight
effx(bweight,exposure=hyp,data=births)
# effect of hypertension on birth weight stratified by sex
effx(bweight,exposure=hyp,strata=sex,data=births)
# effect of hypertension on birth weight controlled for sex
effx(bweight,exposure=hyp,control=sex,data=births)
# effect of gestation time on birth weight
effx(bweight,exposure=gestwks,data=births)
# effect of gestation time on birth weight stratified by sex
effx(bweight,exposure=gestwks,strata=sex,data=births)
# effect of gestation time on birth weight controlled for sex
effx(bweight,exposure=gestwks,control=sex,data=births)
```



```
# lowbw is a binary response coded 1 for low birth weight and 0 otherwise
# effect of hypertension on low birth weight
effx(lowbw,type="binary",exposure=hyp,data=births)
# etc.
```

ewrates	<i>Rates of lung and nasal cancer mortality, and total mortality.</i>
----------------	-----------------------------------------------------------------------

Description

England and Wales mortality rates from lung cancer, nasal cancer, and all causes 1936 - 1980. The 1936 rates are repeated as 1931 rates in order to accomodate follow up for the [nickel](#) study.

Usage

```
data(ewrates)
```

Format

A data frame with 150 observations on the following 5 variables:

id	Subject identifier (numeric)
year	Calendar period, 1931: 1931–35, 1936: 1936–40, ...
age	Age class: 10: 10–14, 15:15–19, ...
lung	Lung cancer mortality rate per 1,000,000 py.
nasal	Nasal cancer mortality rate per 1,000,000 py.
other	All cause mortality rate per 1,000,000 py.

Source

From Breslow and Day, Vol II, Appendix IX.

Examples

```
data(ewrates)
str(ewrates)
```

ex1	<i>Split follow-up time along a timescale</i>
------------	-----------------------------------------------

Description

Splits follow-up time at prespecified points of follow-up.

Usage

```
ex1( enter, exit, fail, origin = 0, scale = 1, breaks,
    data = data.frame(enter, exit, fail),
    Expand = 1:nrow(data) )
```

Arguments

enter	Date of entry into the study (start of follow-up). Numeric.
exit	Date of termination of follow-up. Numeric.
fail	Status at exit from the study.
origin	Origin of the timescale to split on. Specified on the input timescale, i.e. that of enter and exit .

<code>scale</code>	Scaling between input and analysis timescale.
<code>breaks</code>	Breakpoints on the analysis timescale. Follow-up before <code>min(breaks)</code> and after <code>max(breaks)</code> is discarded.
<code>data</code>	Dataframe of variables to carry over to the output.
<code>Expand</code>	Variable identifying original records.

Details

If `entry` and `exit` are given in days (for example as `Date` variables, and we want follow-up cut at 5-year age intervals, then we should choose `origin` equal to birthdate, `scale` equal to 365.25 and `breaks` as `seq(0,100,5)`. Thus the input timescale is calendar time measured in days, and output timescale is age measured in years.

Value

A dataframe with one row per follow-up interval, and variables as in `data`, preceded by the variables:

<code>Expand</code>	Identification of the rows from the input dataframe.
<code>Enter</code>	Entry date for the interval.
<code>Exit</code>	Exit date for the interval.
<code>Fail</code>	Failure indicator for end of the current interval.

Author(s)

Bendix Carstensen, Steno Diabetes Center, (bxc@steno.dk), www.biostat.ku.dk/~bxc

See Also

[Lexis](#), [isec](#), [icut](#), [fcut1](#), [ex1](#)

Examples

```
one <- round( runif( 15, 0, 10 ), 1 )
two <- round( runif( 15, 0, 10 ), 1 )
doe <- pmin( one, two )
dox <- pmax( one, two )
# Goofy data rows to test possibly odd behaviour
doe[1:3] <- dox[1:3] <- 8
dox[2] <- 6
dox[3] <- 7.5
# Some failure indicators
fail <- sample( 0:1, 15, replace=TRUE, prob=c(0.7,0.3) )
# Split follow-up:
ex1( doe, dox, fail, breaks=0:10 )
```

<code>expand.data</code>	<i>Function to expand data for regression analysis of interval censored data.</i>
--------------------------	-----------------------------------------------------------------------------------

Description

This is a utility function.

The original records with `first.well`, `last.well` and `first.ill` are expanded to multiple records; several for each interval where the person is known to be well and one where the person is known to fail. At the same time columns for the covariates needed to estimate rates and the response variable are generated.

Usage

```
expand.data(fu, formula, breaks, data)
```

Arguments

fu	A 3-column matrix with first.well , last.well and first.ill in each row.
formula	Model formula, used to derive the model matrix.
breaks	Defines the intervals in which the baseline rate is assumed constant. All follow-up before the first and after the last break is discarded.
data	Dataframe in which fu and formula is interpreted.

Value

	Returns a list with three components
rates.frame	Dataframe of covariates for estimation of the baseline rates — one per interval defined by breaks .
cov.frame	Dataframe for estimation of the covariate effects. A data-framed version of the designmatrix from formula .
y	Response vector.

Author(s)

Martyn Plummer, [⟨plummer@iarc.fr⟩](mailto:plummer@iarc.fr)

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. *Statistics in Medicine*, 15(20):2177-2189, 1996.

See Also

[Icens fit.mult fit.add](#)

Examples

fcut	<i>Cuts follow-up time at multiple failure times.</i>
-------------	-------------------------------------------------------

Description

This function cuts the follow-up time at multiple failure times allowing a person to stay at risk between and after the last failure. It is aimed at processing of recurrent events. Failure times outside the interval **(enter,exit)** are ignored.

Usage

```
fcut( enter, exit, dof, fail = 0,
      data = data.frame(enter, exit),
      Expand = 1:nrow( data ))
```

Arguments

enter	Date of entry into the study. Numerical vector.
exit	Date of exit from the study. Numerical vector.
fail	Failure indicator for the exit date.
dof	Failure time(s). For multiple failures per individual, dof must be a list.
data	Dataframe of variables to be carried over to the output.
Expand	Variable identifying original records.

Value

A dataframe with the same variables as in `data` preceded by the variables:

<code>Expand</code>	Identification of the rows from the input dataframe.
<code>Enter</code>	Entry date for the interval.
<code>Exit</code>	Exit date for the interval.
<code>Fail</code>	Failure indicator for end of the current interval.
<code>n.Fail</code>	Number of failures prior to the start of the current interval. Counts all failures given in the list <code>dof</code> , including those prior to <code>enter</code> .

Author(s)

Bendix Carstensen, Steno Diabetes Center, (bxc@steno.dk), www.biostat.ku.dk/~bxc

See Also

[Lexis](#), [isec](#), [icut](#), [fcut1](#), [ex1](#)

Examples

```
one <- round( runif( 15, 0, 10 ), 1 )
two <- round( runif( 15, 0, 10 ), 1 )
doe <- pmin( one, two )
dox <- pmax( one, two )
# Goofy data rows to test possibly odd behaviour
doe[1:3] <- dox[1:3] <- 8
dox[2] <- 6
dox[3] <- 7.5
# Some failure indicators
fail <- sample( 0:1, 15, replace=TRUE, prob=c(0.7,0.3) )
# Failure times in a list
dof <- sample( c(one,two), 15 )
l.dof <- list( f1=sample( c(one,two), 15 ),
              f2=sample( c(one,two), 15 ),
              f3=sample( c(one,two), 15 ) )
# The same, but with events prior to entry removed
lx.dof <- lapply( l.dof, FUN=function(x){ x[x<doe] <- NA ; x } )
# So what have we got
data.frame( doe, dox, fail, l.dof, lx.dof )
# Cut follow-up at event times
fcut( doe, dox, lx.dof, fail, data=data.frame( doe, dox, lx.dof ) )
```

`fcut1`

Cut follow-up time at a failure time.

Description

This function cuts the follow-up time at a failure allowing a person to stay at risk after the failure. It is aimed at processing of recurrent events. Failure times outside the interval (`enter`,`exit`) are ignored.

Usage

```
fcut1( enter, exit, fail, dof, fail.value = 1,
       data = data.frame(enter, exit, fail, dof),
       Expand = 1:nrow(data) )
```

Arguments

<code>enter</code>	Date of entry into the study. Numerical vector.
<code>exit</code>	Date of exit from the study. Numerical vector.
<code>fail</code>	Failure indicator for the exit date.
<code>dof</code>	Date of failure. Numerical vector. Missing for persons who do not have an event.
<code>fail.value</code>	Value for failure indicator <code>Fail</code> at the date <code>dof</code> .
<code>data</code>	Dataframe of variables to be carried unchanged to the output.
<code>Expand</code>	Variable identifying original records.

Value

A dataframe with the same columns as in `data`, preceded by the columns:

<code>Expand</code>	Identification of the rows from the input dataframe.
<code>Enter</code>	Entry date for the interval.
<code>Exit</code>	Exit date for the interval.
<code>Fail</code>	Failure indicator for the interval.

Author(s)

Bendix Carstensen, Steno Diabetes Center, {bxc@steno.dk}, www.biostat.ku.dk/~bxc

See Also

[Lexis](#), [isec](#), [fcut1](#), [fcut](#), [ex1](#)

Examples

```
one <- round( runif( 15, 0, 10 ), 1 )
two <- round( runif( 15, 0, 10 ), 1 )
doe <- pmin( one, two )
dox <- pmax( one, two )
# Goofy data rows to test possibly odd behaviour
doe[1:3] <- dox[1:3] <- 8
dox[2] <- 6
dox[3] <- 7.5
# Some failure indicators and failure times
fail <- sample( 0:1, 15, replace=TRUE, prob=c(0.7,0.3) )
dof <- sample( c(one,two), 15 )
# So what have we got
data.frame( doe, dox, fail, dof )
# Cut follow-up at dof
fcut1( doe, dox, fail, dof )
```

`fit.add`

Fit an additive excess risk model to interval censored data.

Description

Utility function.

The model fitted assumes a piecewise constant inensity for the baseline, and that the covariates act additively on the rate scale.

Usage

```
fit.add( y, rates.frame, cov.frame, start )
```

Arguments

<code>y</code>	Binary vector of outcomes
<code>rates.frame</code>	Dataframe expanded from the original data by <code>expand.data</code> , cooresponding to covariates for the rate parameters.
<code>cov.frame</code>	do., but covariates corresponding to the <code>formula</code> argument of <code>Icens</code>
<code>start</code>	Starting values for the rate parameters. If not supplied, then starting values are generated.

Value

A `glm` object from a binomial model with log-link function.

Author(s)

Martyn Plummer, (plummer@iarc.fr)

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. *Statistics in Medicine*, 15(20):2177-2189, 1996.

CP Farrington: Interval censored survival data: a generalized linear modelling approach. *Statistics in Medicine*, 15(3):283-292, 1996.

See Also

`Icens fit.mult`

Examples

```
data( HIV.dk )
```

`fit.baseline`

Fit a piecewise contsnt intesity model for interval censored data.

Description

Utility function

Fits a binomial model with logarithmic link, with `y` as outcome and covariates in `rates.frame` to estimate rates in the inttervals between `breaks`.

Usage

```
fit.baseline( y, rates.frame, start )
```

Arguments

<code>y</code>	Binary vector of outcomes
<code>rates.frame</code>	Dataframe expanded from the original data by <code>expand.data</code>
<code>start</code>	Starting values for the rate parameters. If not supplied, then starting values are generated.

Value

A `glm` object, with binomial error and logarithmic link.

Author(s)

Martyn Plummer, <plummer@iarc.fr>

See Also

[fit.add](#) [fit.mult](#)

Examples

```
fit.mult
```

Fits a multiplicative relative risk model to interval censored data.

Description

Utility function.

The model fitted assumes a piecewise constant baseline rate in intervals specified by the argument **breaks**, and a multiplicative relative risk function.

Usage

```
fit.mult( y, rates.frame, cov.frame, start )
```

Arguments

y	Binary vector of outcomes
rates.frame	Dataframe expanded from the original data by expand.data , coresponding to covariates for the rate parameters.
cov.frame	do., but covariates corresponding to the formula argument of Icens
start	Starting values for the rate parameters. If not supplied, then starting values are generated.

Details

The model is fitted by alternating between two generalized linear models where one estimates the underlying rates in the intervals, and the other estimates the log-relative risks.

Value

A list with three components:

rates	A glm object from a binomial model with log-link, estimating the baseline rates.
cov	A glm object from a binomial model with complementary log-log link, estimating the log-rate-ratios
niter	Nuber of iterations, a scalar

Author(s)

Martyn Plummer, <plummer@iarc.fr>, Bendix Carstensen, <bxc@steno.dk>

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. *Statistics in Medicine*, 15(20):2177-2189, 1996.

CP Farrington: Interval censored survival data: a generalized linear modelling approach. *Statistics in Medicine*, 15(3):283-292, 1996.

See Also

[Icens fit.add](#)

Examples

```
data( HIV.dk )
```

<code>float</code>	<i>Calculate floated variances</i>
--------------------	------------------------------------

Description

Given a fitted model object, the `float()` function calculates floating variances (aka quasi-variances) for a given factor in the model.

Usage

```
float(object, factor, iter.max=50)
```

Arguments

<code>object</code>	a fitted model object
<code>factor</code>	character string giving the name of the factor of interest. If this is not given, the first factor in the model is used.
<code>iter.max</code>	Maximum number of iterations for EM algorithm

Details

`float()` implements the "floating absolute risk" proposal of Easton, Peto and Babiker(1992). This is an alternative way of presenting parameter estimates for factors in regression models, which avoids some of the difficulties of treatment contrasts. It was originally designed for epidemiological studies of relative risk (hence the name) but the idea is widely applicable.

A problem with treatment contrasts is that they are not orthogonal. The variances of the treatment contrasts may be inflated by a poor choice of reference level, and the correlations between them may be very high. `float()` associates each level of the factor, including the reference level, with a "floating" variance (or quasi-variance). Floating variances are not real variances, but they can be used to calculate the variance of any contrast by treating each level as independent.

Plummer (2003) showed that floating variances can be derived from a covariance structure model applied to the variance-covariance matrix of the parameter estimates. This model can be fitted by minimizing the Kullback-Leibler information divergence between the true and distributions for the parameter estimates and the distribution given by the covariance structure model. Fitting is done using the EM algorithm.

In order to check the goodness-of-fit of the floating variance model, `float()` compares the standard errors predicted by the model with the standard errors derived from the true variance-covariance matrix of the parameter contrasts. The maximum and minimum ratios between true and model standard errors are calculated over all possible contrasts. These should be within 5 percent, or the use of the floating variances may lead to invalid confidence intervals.

Value

An object of class `float`. This is a list with the following components

<code>coef</code>	vector of coefficients. These are the same as the treatment contrasts but the reference level is present with coefficient 0.
<code>var</code>	vector of floating (or quasi-) variances
<code>limits</code>	Bounds on the accuracy of standard errors over all possible contrasts

Note

Menezes(1999) and Firth and Menezes (2004) take a slightly different approach to this problem, using a pseudo-likelihood approach to fit the quasi-variance model. Their work is implemented in the package `qvcalc`.

Author(s)

Martyn Plummer

References

- Easton DF, Peto J and Babiker GAG (1991) Floating absolute risk: An alternative to relative risk in survival and case control analysis avoiding an arbitrary reference group. *Statistics in Medicine*, **10**, 1025-1035.
- Firth D and Mezezes RX (2004) Quasi-variances. *Biometrika* **91**, 65-80.
- Menezes RX(1999) More useful standard errors for group and factor effects in generalized linear models. *D.Phil. Thesis*, Department of Statistics, University of Oxford.
- Plummer M (2003) Improved estimates of floating absolute risk, *Statistics in Medicine*, **23**, 93-104.

See Also

[ftrend](#), `qvcalc`

ftrend

Fit a floating trend to a factor in generalized linear model

Description

Fits a "floating trend" model to the given factor in a glm in a generalized linear model by centering covariates.

Usage

```
ftrend(object, ...)
```

Arguments

object	fitted <code>lm</code> or <code>glm</code> object. The model must not have an intercept term
...	arguments to the <code>nlm</code> function

Details

ftrend() calculates "floating trend" estimates for factors in generalized linear models. This is an alternative to treatment contrasts suggested by Greenland et al. (1999). If a regression model is fitted with no intercept term, then contrasts are not used for the first factor in the model. Instead, there is one parameter for each level of this factor. However, the interpretation of these parameters, and their variance-covariance matrix, depends on the numerical coding used for the covariates. If an arbitrary constant is added to the covariate values, then the variance matrix is changed.

The **ftrend()** function takes the fitted model and works out an optimal constant to add to the covariate values so that the covariance matrix is approximately diagonal. The parameter estimates can then be treated as approximately independent, thus simplifying their presentation. This is particularly useful for graphical display of dose-response relationships (hence the name).

Greenland et al. (1999) originally suggested centring the covariates so that their weighted mean, using the fitted weights from the model, is zero. This heuristic criterion is improved upon by **ftrend()** which uses the same minimum information divergence criterion as used by Plummer (2003) for floating variance calculations. **ftrend()** calls **nlm()** to do the minimization and will pass optional arguments to control it.

Value

A list with the following components

coef coefficients for model with adjusted covariates.
vcov Variance-covariance matrix of adjusted coefficients.

Note

The "floating trend" method is an alternative to the "floating absolute risk" method, which is implemented in the function `float()`.

Author(s)

Martyn Plummer

References

Greenland S, Michels KB, Robins JM, Poole C and Willet WC (1999) Presenting statistical uncertainty in trends and dose-response relations, *American Journal of Epidemiology*, **149**, 1077-1086.

See Also

[float](#)

gmortDK

Population mortality rates for Denmark in 5-years age groups.

Description

The `gmortDK` data frame has 418 rows and 21 columns.

Format

This data frame contains the following columns:

agr: Age group, 0:0–4, 5:5–9,..., 90:90+.
per: Calendar period, 38: 1938–42, 43: 1943–47, ..., 88:1988–92.
sex: Sex, 1: male, 2: female.
risk: Number of person-years in the Danish population.
dt: Number of deaths.
rt: Overall mortality rate in cases per 1000 person-years, i.e. $rt=1000*dt/risk$
Cause-specific mortality rates in cases per 1000 person-years:
r1: Infections
r2: Cancer.
r3: Tumors, benign, unspecific nature.
r4: Endocrine, metabolic.
r5: Blood.
r6: Nervous system, psychiatric.
r7: Cerebrovascular.
r8: Cardiac.
r9: Respiratory diseases, excl. cancer.
r10: Liver, excl. cancer.
r11: Digestive, other.
r12: Genitourinary.
r13: Ill-defined symptoms.
r14: All other, natural.
r15: Violent.

Source

Statistics Denmark, National board of health provided original data. Michael Andersson grouped the causes of death.

See Also

[thoro](#), [mortDK](#)

Examples

```
data(gmortDK)
```

hivDK

hivDK: seroconversion in a cohort of Danish men

Description

Data from a survey of HIV-positivity of a cohort of Danish men followed by regular tests from 1983 to 1989.

Usage

```
data(hivDK)
```

Format

A data frame with 297 observations on the following 7 variables.

id ID of the person

entry Date of entry to the study. Date variable.

well Date last seen seronegative. Date variable.

ill Date first seen seroconverted. Date variable.

bth Year of birth minus 1950.

pyr Annual number of sexual partners.

us Indicator of wheter the person has visited the USA.

Source

Mads Melbye, Statens Seruminstitut.

References

Becker N.G. and Melbye M.: Use of a log-linear model to compute the empirical survival curve from interval-censored data, with application to data on tests for HIV-positivity, Australian Journal of Statistics, 33, 125–133, 1990.

Melbye M., Biggar R.J., Ebbesen P., Sarngadharan M.G., Weiss S.H., Gallo R.C. and Blattner W.A.: Seroepidemiology of HTLV-III antibody in Danish homosexual men: prevalence, transmission and disease outcome. British Medical Journal, 289, 573–575, 1984.

Examples

```
data(hivDK)
str(hivDK)
```

Icens*Fits a regression model to interval censored data.*

Description

The models fitted assumes a piecewise constant baseline rate in intervals specified by the argument **breaks**, and for the covariates either a multiplicative relative risk function (default) or an additive excess risk function.

Usage

```
Icens( first.well, last.well, first.ill,
       formula, model.type=c("MRR","AER"), breaks,
       boot=FALSE, alpha=0.05, keep.sample=FALSE,
       data )
```

Arguments

first.well	Time of entry to the study, i.e. the time first seen without event. Numerical vector.
last.well	Time last seen without event. Numerical vector.
first.ill	Time first seen with event. Numerical vector.
formula	Model formula for the log relative risk.
model.type	Which model should be fitted.
breaks	Breakpoints between intervals in which the underlying timescale is assumed constant. Any observation outside the range of breaks is discarded.
boot	Should bootstrap be performed to produce confidence intervals for parameters. If a number is given this will be the number of bootstrap samples. The default is 1000.
alpha	1 minus the confidence level.
keep.sample	Should the bootstrap sample of the parameter values be returned?
data	Data frame in which the times and formula are interpreted.

Details

The model is fitted by calling either `fit.mult` or `fit.add`.

Value

An object of class "**Icens**": a list with three components:

rates	A glm object from a binomial model with log-link, estimating the baseline rates, and the excess risk if " AER " is specified.
cov	A glm object from a binomial model with complementary log-log link, estimating the log-rate-ratios. Only if " MRR " is specified.
niter	Nuber of iterations, a scalar
boot.ci	If boot=TRUE , a 3-column matrix with estimates and 1- alpha confidence intervals for the parameters in the model.
sample	A matrix of the parameterestimates from the bootstrapping. Rows refer to parameters, columns to bootstrap samples.

Author(s)

Martyn Plummer, (plummer@iarc.fr), Bendix Carstensen, (bxc@steno.dk)

References

B Carstensen: Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. *Statistics in Medicine*, 15(20):2177-2189, 1996.

CP Farrington: Interval censored survival data: a generalized linear modelling approach. *Statistics in Medicine*, 15(3):283-292, 1996.

See Also

[fit.add fit.mult](#)

Examples

```
data( hivDK )
# Convert the dates to fractional years so that rates are
# expressed in cases per year
for( i in 2:4 ) hivDK[,i] <- cal.yr( hivDK[,i] )

m.RR <- Izens( entry, well, ill,
               model="MRR", formula=~pyr+us, breaks=seq(1980,1990,5),
               data=hivDK)
# Currently the MRR model returns a list with 2 glm objects.
round( ci.lin( m.RR$rates ), 4 )
round( ci.lin( m.RR$cov, Exp=TRUE ), 4 )
# There is actually a print method:
print( m.RR )

m.ER <- Izens( entry, well, ill,
               model="AER", formula=~pyr+us, breaks=seq(1980,1990,5),
               data=hivDK)
# There is actually a print method:
print( m.ER )
```

icut

Function to cut the follow-up in cohort at a point in time.

Description

The follow up time from **enter** to **exit** is classified as to wheter it is before **cut** (**Time=0**) or after **Time=1**). If **cut** is between **enter** and **exit**, the follow-up is split in two intervals, the first gets the value **cens** for the status.

Usage

```
icut( enter, exit, cut, fail = 0, cens.value = 0,
      data = data.frame(enter, exit, fail, cut),
      Expand = 1:nrow( data ),
      na.cut = Inf )
```

Arguments

enter	Date of entry. Numerical vector.
exit	Date of exit. Numerical vector.
fail	Indicator if exit status.
cens.value	Value to use for censoring status.
cut	Date where to cut follow-up. Numerical vector.
data	Dataframe of variables to carry over to the output dataframe.

Expand	Variable identifying original records.
na.cut	What value should be assigned to missing values of the cutpoint. Defaults to Inf , so the intermediate event is considered not to have occurred. If set to -Inf , all persons with missing cut are considered to have had an intermediate event. If set to NA records with missing cut are omitted from the result.

Details

The purpose of this function is to divide follow-up into pre- and post some intermediate event like recurrence of disease, thus enabling Follow-up for persons with a recurrence date (**cut**) will be split in two, with indication (in **Time**) of what is pre and what is post recurrence. This is typically what precedes a survival analysis where recurrence is modelled as a time-dependent variable.

Value

A data frame with one row per interval of follow up and columns given in the **data** argument, preceded by the columns:

Expand	Identification of the rows from the input dataframe.
Enter	Entry date for the interval.
Exit	Exit date for the interval.
Fail	Failure indicator for the interval.
Time	Indicator variable for intervals after cut .

Author(s)

Bendix Carstensen, Steno Diabetes Center, (bxc@steno.dk), www.biostat.ku.dk/~bxc

See Also

[Lexis](#), [isec](#), [fcut1](#), [fcut](#), [ex1](#)

Examples

```
one <- round( runif( 15, 0, 15 ), 1 )
two <- round( runif( 15, 0, 15 ), 1 )
doe <- pmin( one, two )
dox <- pmax( one, two )
# Goofy data rows to test possibly odd behaviour
doe[1:3] <- dox[1:3] <- 8
dox[2] <- 6
dox[3] <- 7.5
# Some failure indicators
fail <- sample( 0:1, 15, replace=TRUE, prob=c(0.7,0.3) )
# So what have we got
data.frame( doe, dox, fail )
# Cut follow-up at 5
icut( doe, dox, fail, cut=5 )
```

isec	<i>Determine the intersection between follow-up intervals and a fixed interval.</i>
-------------	-------------------------------------------------------------------------------------

Description

For a given piece of follow-up for each person, this function determines the part of the follow-up which is inside a pre-specified interval. The function may occasionally be useful but is mainly included as a tool to be used in the function [Lexis](#).

Usage

```
isec( enter, exit, fail = 0, int, cens.value = 0,
      Expand = 1:length(enter))
```

Arguments

enter	Numerical vector of entry times.
exit	Numerical vector of exit times.
fail	Vector of exit status, i.e. status at time exit
int	The fixed interval. Numerical vector of length 2.
cens.value	Censoring value. Numerical. If the person survives the fixed interval, this will be the value of the status indicator, Fail
Expand	Person id to be carried to the output

Value

A matrix with columns **Expand** - person identification, **Enter** - date of entry into the interval, **Exit** - date of exit from the interval and **Fail**.

Author(s)

Bendix Carstensen, Steno Diabetes Center, (bxc@steno.dk), www.biostat.ku.dk/~bxc

See Also

[Lexis](#), [icut](#), [fcut1](#), [fcut](#), [ex1](#)

Examples

```
one <- round( runif( 15, 0, 15 ), 1 )
two <- round( runif( 15, 0, 15 ), 1 )
doe <- pmin( one, two )
dox <- pmax( one, two )
# Goofy data rows to test possibly odd behaviour
doe[1:3] <- dox[1:3] <- 8
dox[2] <- 6
dox[3] <- 7.5
# Some failure indicators
fail <- sample( 0:1, 15, replace=TRUE, prob=c(0.7,0.3) )
# So what have we got?
data.frame( doe, dox, fail )
# Find intersection with interval (4,8)
isec( doe, dox, fail, int=c(4,8) )
# See how it compares to original data
merge( data.frame( Expand=1:15, doe, dox, fail ),
       data.frame( isec( doe, dox, fail, int=c(4,8) ) ), all=TRUE )
```

Description

The **lep** data frame has 1370 rows and 7 columns. This was an unmatched case-control study in which incident cases of leprosy in a region of N. Malawi were compared with population controls.

Format

This data frame contains the following columns:

```

id:    subject identifier: a numeric vector
d:     case/control status: a numeric vector (1=case, 0=control)
age:   a factor with levels 5-9 10-14 15-19 20-24 25-29 30-44 45+
sex:   a factor with levels male, female
bcg:   presence of vaccine scar, a factor with levels no yes
school: schooling, a factor with levels none 1-5yrs 6-8yrs sec/tert
house: housing, a factor with levels brick sunbrick wattle temp

```

Source

The study is described in more detail in Clayton and Hills, Statistical Models in Epidemiology, Oxford University Press, Oxford:1993.

Examples

```
data(lep)
```

```
Lexis.diagram
```

```
Plot a Lexis diagram
```

Description

Draws a Lexis diagram, optionally with life lines from a cohort, and with lifelines of a cohort if supplied.

Usage

```

Lexis.diagram( age = c( 0, 60),
               alab = "Age",
               date = c( 1940, 2000 ),
               dlab = "Calendar time",
               int = 5,
               lab.int = 2*int,
               col.life = "black",
               lwd.life = 2,
               age.grid = TRUE,
               date.grid = TRUE,
               coh.grid = FALSE,
               col.grid = gray(0.7),
               lwd.grid = 1,
               las = 1,
               entry.date = NA,
               entry.age = NA,
               exit.date = NA,
               exit.age = NA,
               risk.time = NA,
               birth.date = NA,
               fail = NA,
               cex.fail = 1.1,
               pch.fail = c(NA,16),
               col.fail = rep( col.life, 2 ),
               data = NULL, ... )

```

Arguments

age	Numerical vector of length 2, giving the age-range for the diagram
alab	Label on the age-axis.
date	Numerical vector of length 2, giving the calendar time-range for the diagram
dlab	label on the calendar time axis.

<code>int</code>	The interval between grid lines in the diagram. If a vector of length two is given, the first value will be used for spacing of age-grid and the second for spacing of the date grid.
<code>lab.int</code>	The interval between labelling of the grids.
<code>col.life</code>	Colour of the life lines.
<code>lwd.life</code>	Width of the life lines.
<code>age.grid</code>	Should grid lines be drawn for age?
<code>date.grid</code>	Should grid lines be drawn for date?
<code>coh.grid</code>	Should grid lines be drawn for birth cohorts (diagonals)?
<code>col.grid</code>	Colour of the grid lines.
<code>lwd.grid</code>	Width of the grid lines.
<code>las</code>	How are the axis labels plotted?
<code>entry.date, entry.age, exit.date, exit.age, risk.time, birth.date</code>	Numerical vectors defining lifelines to be plotted in the diagram. At least three must be given to produce lines. Not all subsets of three will suffice, the given subset has to define life lines. If insufficient data is given, no life lines are produced.
<code>fail</code>	Logical of event status at exit for the persons whose life lines are plotted.
<code>pch.fail</code>	Symbols at the end of the life lines for censorings (<code>fail=0</code>) and failures (<code>fail≠0</code>).
<code>cex.fail</code>	Expansion of the status marks at the end of life lines.
<code>col.fail</code>	Character vector of length 2 giving the colour of the failure marks for censorings and failures respectively.
<code>data</code>	Dataframe in which to interpret the arguments.
<code>...</code>	Arguments to be passed on to the initial call to plot.

Details

The default unit for supplied variables are (calendar) years. If any of the variables `entry.date`, `exit.date` or `birth.date` are of class "Date" or if any of the variables `entry.age`, `exit.age` or `risk.time` are of class "difftime", they will be converted to calendar years, and plotted correctly in the diagram. The returned dataframe will then have columns of classes "Date" and "difftime".

Value

If sufficient information on lifelines is given, a data frame with one row per person and columns with entry ages and dates, birth date, risk time and status filled in.

Side effect: a plot of a Lexis diagram is produced with the life lines in it is produced. This will be the main reason for using the function.

Author(s)

Bendix Carstensen, <http://www.biostat.ku.dk/~bxc>

See Also

[Life.lines](#), [Lexis.lines](#)

Examples

```
Lexis.diagram( entry.age = c(3,30,45),
               risk.time = c(25,5,14),
               birth.date = c(1970,1931,1925.7),
               fail = c(TRUE,TRUE,FALSE) )
LL <- Lexis.diagram( entry.age = sample( 0:50, 17, replace=TRUE ),
                   risk.time = sample( 5:40, 17, r=TRUE),
                   birth.date = sample( 1910:1980, 17, r=TRUE ),
                   fail = sample( 0:1, 17, r=TRUE ),
```

```

        cex.fail = 1.1,
        lwd.life = 2 )
# Identify the persons' entry and exits
text( LL$exit.date, LL$exit.age, paste(1:nrow(LL)), col="red", font=2, adj=c(0,1) )
text( LL$entry.date, LL$entry.age, paste(1:nrow(LL)), col="blue", font=2, adj=c(1,0) )
data( nickel )
attach( nickel )
LL <- Lexis.diagram( age=c(10,100), date=c(1900,1990),
                    entry.age=age1st, exit.age=ageout, birth.date=dob,
                    fail=(icd %in% c(162,163)), lwd.life=1,
                    cex.fail=0.8, col.fail=c("green","red") )
abline( v=1934, col="blue" )
nickel[1:10,]
LL[1:10,]

```

Lexis.lines	<i>Draw life lines in a Lexis diagram.</i>
--------------------	--------------------------------------------

Description

Add life lines to a Lexis diagram.

Usage

```

Lexis.lines( entry.date = NA,
             exit.date = NA,
             birth.date = NA,
             entry.age = NA,
             exit.age = NA,
             risk.time = NA,
             col.life = "black",
             lwd.life = 2,
             fail = NA,
             cex.fail = 1,
             pch.fail = c(NA, 16),
             col.fail = col.life,
             data = NULL )

```

Arguments

entry.date, entry.age, exit.date, exit.age, risk.time, birth.date	Numerical vectors defining lifelines to be plotted in the diagram. At least three must be given to produce lines. Not all subsets of three will suffice, the given subset has to define life lines. If insufficient data is given, no life lines are produced.
col.life	Colour of the life lines.
lwd.life	Width of the life lines.
fail	Logical of event status at exit for the persons whose life lines are plotted.
cex.fail	The size of the status marks at the end of life lines.
pch.fail	The status marks at the end of the life lines.
col.fail	Colour of the marks for censorings and failures respectively.
data	Data frame in which to interpret values.

Value

If sufficient information on lifelines is given, a data frame with one row per person and columns with entry ages and dates, birth date, risk time and status filled in.

Side effect: Life lines are added to an existing Lexis diagram. Lexis.lines adds life lines to an existing plot.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.biostat.ku.dk/~bxc>

See Also

[Lexis.diagram](#), [Life.lines](#)

Examples

```
Lexis.diagram( entry.age = c(3,30,45),
               risk.time = c(25,5,14),
               birth.date = c(1970,1931,1925.7),
               fail = c(TRUE,TRUE,FALSE) ),
Lexis.lines( entry.age = sample( 0:50, 100, replace=TRUE ),
             risk.time = sample( 5:40, 100, r=TRUE),
             birth.date = sample( 1910:1980, 100, r=TRUE ),
             fail = sample(0:1,100,r=TRUE),
             cex.fail = 0.5,
             lwd.life = 1 )
```

Lexis

Create a Lexis object

Description

Create an object of class **Lexis** to represent follow-up on multiple time scales.

Usage

```
Lexis(entry, exit, duration, entry.status = 0, exit.status = 0, id, data,
      merge=TRUE, states )
```

Arguments

entry	a named list of entry times. Each element of the list is a numeric variable representing the entry time on the named time scale. All time scales must have the same units (e.g. years).
exit	a named list of exit times.
duration	a numeric vector giving the duration of follow-up.
entry.status	a vector giving the status at entry.
exit.status	a vector giving status at exit. Any change in status during follow-up is assumed to take place exactly at the exit time.
id	a vector giving a unique identity value for each row of the Lexis object.
data	an optional data frame, list, or environment containing the variables. If not found in data , the variables are taken from the environment from which Lexis was called.
merge	a logical flag. If TRUE then the data argument will be coerced to a data frame and then merged with the resulting Lexis object.
states	A vector of labels for the states. If given, the state variables lex.Cst and lex.Xst are returned as factors with identical levels attributes.

Details

The analysis of long-term population-based follow-up studies typically requires multiple time scales to be taken into account, such as age, calendar time, or time since an event. A **Lexis** object is a data frame with additional attributes that allows these multiple time dimensions of follow-up to be managed.

Separate variables for current end exit state allows representation of multistate data.

Lexis objects are named after the German demographer Wilhelm Lexis (1837-1914), who is credited with the invention of the "Lexis diagram" for representing population dynamics simultaneously by age, period and cohort.

The **Lexis** function creates a minimal **Lexis** object with only those variables required to define the follow-up history in each row. Additional variables can be merged into the **Lexis** object using the **merge** method for **Lexis** objects.

Value

An object of class **Lexis**. This is represented as a data frame with a column for each time scale, and additional columns with the following names:

lex.id	Identification of the individual
lex.dur	Duration of follow-up
lex.Cst	Entry status (Current state), i.e. the state in which the follow up takes place.
lex.Xst	Exit status (eXit state), i.e. that state taken up after dur in lex.Cst .

If **merge=TRUE** then the **Lexis** object will also contain all variables from the **data** argument.

Note

Only two of the three arguments **entry**, **exit** and **duration** need to be given. If the third parameter is missing, it is imputed. If duration is imputed, it must be the same on all time scales.

If only either **exit** or **duration** are supplied it is assumed that **entry** is 0. This is only meaningful (and therefore checked) if there is only one timescale.

Author(s)

Martyn Plummer

See Also

[plot.Lexis](#), [splitLexis](#), [cutLexis](#), [merge.Lexis](#), [timeScales](#), [timeBand](#), [entry](#), [exit](#), [dur](#)

Examples

```
# A small bogus cohort
xcoh <- structure( list( id = c("A", "B", "C"),
                        birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                        entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                        exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                        fail = c(1, 0, 1) ),
                  .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                  class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
xcoh
```

```
# Define as Lexis object with timescales calendar time and age
Lcoh <- Lexis( entry = list( per=en ),
              exit = list( per=ex, age=ex-bt ),
              exit.status = fail,
              data = xcoh )

Lcoh
```

Life.lines
Compute dates/ages for life lines in a Lexis diagram

Description

Fills out the missing information for follow up of persons in a Lexis diagram if sufficient information is given.

Usage

```
Life.lines( entry.date = NA,
            exit.date = NA,
            birth.date = NA,
            entry.age = NA,
            exit.age = NA,
            risk.time = NA )
```

Arguments

`entry.date`, `exit.date`, `birth.date`, `entry.age`, `exit.age`, `risk.time`

Vectors defining lifelines to be plotted in the diagram. At least three must be given to produce a result. Not all subsets of three will suffice, the given subset has to define life lines. If insufficient data is given, nothing is returned and a warning is given.

Value

Data frame with variables `entry.date`, `entry.age`, `exit.date`, `exit.age`, `risk.time`, `birth.date`, with all entries computed for each person. If any of `entry.date`, `exit.date` or `birth.date` are of class `Date` or if any of `entry.age`, `exit.age` or `risk.time` are of class `difftime` the date variables will be of class `Date` and the other three of class `difftime`.

See Also

[Lexis.diagram](#), [Lexis.lines](#)

Examples

```
( Life.lines( entry.age = c(3,30,45),
              risk.time = c(25,5,14),
              birth.date = c(1970,1931,1925.7) ) )

# Draw a Lexis diagram
Lexis.diagram()

# Compute entry and exit age and date.
( LL <- Life.lines( entry.age = c(3,30,45),
                  risk.time = c(25,5,14),
                  birth.date = c(1970,1931,1925.7) ) )
segments( LL[,1], LL[,2], LL[,3], LL[,4] ) # Plot the life lines.

# Compute entry and exit age and date, supplying a date variable
bd <- ( c(1970,1931,1925.7) - 1970 ) * 365.25
class( bd ) <- "Date"
```

```
( Life.lines( entry.age = c(3,30,45),
              risk.time = c(25,5,14),
              birth.date = bd ) )
```

lungDK	<i>Male lung cancer incidence in Denmark</i>
--------	----------------------------------------------

Description

Male lung cancer cases and population risk time in Denmark, for the period 1943–1992 in ages 40–89.

Usage

```
data(lungDK)
```

Format

A data frame with 220 observations on the following 9 variables.

A5: Left end point of the age interval, a numeric vector.
P5: Left endpoint of the period interval, a numeric vector.
C5: Left endpoint of the birth cohort interval, a numeric vector.
up: Indicator of upper triangles of each age by period rectangle in the Lexis diagram. ($up = (P5 - A5 - C5) / 5$).
Ax: The mean age of diagnosis (at risk) in the triangle.
Px: The mean date of diagnosis (at risk) in the triangle.
Cx: The mean date of birth in the triangle, a numeric vector.
D: Number of diagnosed cases of male lung cancer.
Y: Risk time in the male population, person-years.

Details

Cases and person-years are tabulated by age and date of diagnosis (period) as well as date of birth (cohort) in 5-year classes. Each observation in the dataframe corresponds to a triangle in a Lexis diagram. Triangles are classified by age and date of diagnosis, period of diagnosis and date of birth, all in 5-year groupings.

Source

The Danish Cancer Registry and Statistics Denmark.

References

For a more thorough exposition of statistical inference in the Lexis diagram, see:
<http://staff.pubhealth.ku.dk/~bxc/APC/notes.pdf>

Examples

```
data( lungDK )
# Draw a Lexis diagram and show the number of cases in it.
attach( lungDK )
Lexis.diagram( age=c(40,90), date=c(1943,1993), coh.grid=TRUE )
text( Px, Ax, paste( D ), cex=0.7 )
```

merge.data.frame	<i>Merge data frame with a Lexis object</i>
------------------	---------------------------------------------

Description

Merge two data frames, or a data frame with a Lexis object.

Usage

```
## S3 method for class 'data.frame':
merge(x, y, ...)
```

Arguments

x, **y** data frames, or objects to be coerced into one
... optional arguments for the merge method

Details

This version of `merge.default` masks the one in the **base**. It ensures that, if either **x** or **y** is a **Lexis** object, then `merge.Lexis` is called.

Value

A merged **Lexis** object or data frame.

Author(s)

Martyn Plummer

See Also

[Lexis](#)

<code>merge.Lexis</code>	<i>Merge a Lexis object with a data frame</i>
--------------------------	-----------------------------------------------

Description

Merge additional variables from a data frame into a **Lexis** object.

Usage

```
## S3 method for class 'Lexis':
merge(x, y, id, by, ...)
```

Arguments

x an object of class **Lexis**
y a data frame
id the name of the variable in **y** to use for matching against the variable `lex.id` in **x**.
by if matching is not done by **id**, a vector of variable names common to both **x** and **y**
... optional arguments to be passed to `merge.data.frame`

Details

A **Lexis** object can be considered as an augmented data frame in which some variables are time-dependent variables representing follow-up. The **Lexis** function produces a minimal object containing only these time-dependent variables. Additional variables may be added to a **Lexis** object using the `merge` method.

Value

A **Lexis** object with additional columns taken from the merged data frame.

Note

The variable given as the `by.y` argument must not contain any duplicate values in the data frame **y**.

Author(s)

Martyn Plummer

See Also`merge.data.frame`, `subset.Lexis`

mh

Mantel-Haenszel analyses of cohort and case-control studies

Description

This function carries out Mantel-Haenszel comparisons in tabulated data derived from both cohort and case-control studies. Multiway tables of data are accepted and any two levels of any dimension can be chosen as defining the comparison groups. The rate (odds) ratio estimates and the associated significance tests may be collapsed over all the remaining dimensions of the table, or over selected dimensions only (so that tables of estimates and tests are computed).

Usage

```
mh(cases, denom, compare=1, levels=c(1, 2), by=NULL,
   cohort=!is.integer(denom), confidence=0.9)
```

Arguments

cases	the table of case frequencies (a multiway array).
denom	the denominator table. For cohort studies this should be a table of person-years observation, while for case-control studies it should be a table of control frequencies.
compare	the dimension of the table which defines the comparison groups (can be referred to either by number or by name). The default is the first dimension of the table.
levels	a vector identifying (either by number or by name) the two groups to be compared. The default is the first two levels of the selected dimension.
by	the dimensions not to be collapsed in the Mantel-Haenszel computations. Thus, this argument defines the structure of the resulting tables of estimates and tests.
cohort	an indicator whether the data derive from a cohort or a case-control study. If the denominator table is stored as an integer, a case-control study is assumed.
confidence	the approximate coverage probability for the confidence intervals to be computed.

Details**Value**

A list giving tables of rate (odds) ratio estimates, their standard errors (on a log scale), lower and upper confidence limits, chi-squared tests (1 degree of freedom) and the corresponding p-values. The result list also includes numerator and denominator of the Mantel-Haenszel estimates (q, r), and score test statistics and score variance (u, v).

Side Effects

None

References

Clayton, D. and Hills, M. : Statistical Models in Epidemiology, Oxford University Press (1993).

See Also

[Lexis](#)

Examples

```
# If d and y are 3-way tables of cases and person-years
# observation formed by tabulation by two confounders
# (named "C1" and "C2") an exposure of interest ("E"),
# the following command will calculate an overall
# Mantel-Haenszel comparison of the first two exposure
# groups.
#
# Generate some bogus data
dnam <- list( E=c("low","medium","high"), C1=letters[1:2], C2=LETTERS[1:4] )
d <- array( sample( 2:80, 24),
            dimnames=dnam, dim=sapply( dnam, length ) )
y <- array( abs( rnorm( 24, 227, 50 ) ),
            dimnames=dnam, dim=sapply( dnam, length ) )
mh(d, y, compare="E")
#
# Or, if exposure levels named "low" and "high" are to be
# compared and these are not the first two levels of E :
#
mh(d, y, compare="E", levels=c("low", "high"))
#
# If we wish to carry out an analysis which controls for C1,
# but examines the results at each level of C2:
#
mh(d, y, compare="E", by="C2")
#
# It is also possible to look at rate ratios for every
# combination of C1 and C2 :
#
mh(d, y, compare="E", by=c("C1", "C2"))
#
# If dimensions and levels of the table are unnamed, they must
# be referred to by number.
#
```

mortDK

Population mortality rates for Denmark in 1-year age-classes.

Description

The `mortDK` data frame has 1820 rows and 21 columns.

Format

This data frame contains the following columns:

<code>age</code> :	Age class, 0–89, 90:90+.
<code>per</code> :	Calendar period, 38: 1938–42, 43: 1943–47, ..., 88:1988–92.
<code>sex</code> :	Sex, 1: male, 2: female.
<code>risk</code> :	Number of person-years in the Danish population.
<code>dt</code> :	Number of deaths.
<code>rt</code> :	Overall mortality rate in cases per 1000 person-years, i.e. $rt=1000*dt/risk$
	Cause-specific mortality rates in cases per 1000 person-years:
<code>r1</code> :	Infections

r2: Cancer.
 r3: Tumors, benign, unspecific nature.
 r4: Endocrine, metabolic.
 r5: Blood.
 r6: Nervous system, psychiatric.
 r7: Cerebrovascular.
 r8: Cardiac.
 r9: Respiratory diseases, excl. cancer.
 r10: Liver, excl. cancer.
 r11: Digestive, other.
 r12: Genitourinary.
 r13: Ill-defined symptoms.
 r14: All other, natural.
 r15: Violent.

Source

Statistics Denmark, National board of health provided original data. Michael Andersson grouped the causes of death.

See Also

[thoro](#), [gmortDK](#)

Examples

```
data(mortDK)
```

ncut

Function to group a variable in intervals.

Description

Cuts a continuous variable in intervals. As opposed to `cut` which returns a factor, `ncut` returns a numeric variable.

Usage

```
ncut(x, breaks, type="left" )
```

Arguments

x A numerical vector.
breaks Vector of breakpoints. NA will results for values below `min(x)` if `type="left"`, for values above `max(x)` if `type="right"` and for values outside `range(x)` if `type="mid"`
type Character: one of `c("left", "right", "mid")`, indicating whether the left, right or midpoint of the intervals defined in `breaks` is returned.

Details

The function uses the base function `findInterval`.

Value

A numerical vector of the same length as `x`.

Author(s)

Bendix Carstensen, Steno Diabetes Center, (bx@steno.dk), <http://www.biostat.ku.dk/~bx/>, with essential input from Martyn Plummer, IARC.

See Also

[cut](#), [findInterval](#)

Examples

```
br <- c(-2,0,1,2.5)
x <- c( rnorm( 10 ), br, -3, 3 )
cbind( x, l=ncut( x, breaks=br, type="l" ),
       m=ncut( x, breaks=br, type="m" ),
       r=ncut( x, breaks=br, type="r" ) )[order(x),]
x <- rnorm( 200 )
plot( x, ncut( x, breaks=br, type="l" ), pch=16, col="blue", ylim=range(x) )
abline( 0, 1 )
abline( v=br )
points( x, ncut( x, breaks=br, type="r" ), pch=16, col="red" )
points( x, ncut( x, breaks=br, type="m" ), pch=16, col="green" )
```

nice

Nice breakpoints

Description

The function calls [pretty](#) for linear scale. For a log-scale nice are computed using a set of specified number in a decade.

Usage

```
nice(x, log = F, lpos = c(1, 2, 5), ...)
```

Arguments

x	Numerical vector to
log	Logical. Is the scale logartimic?
lpos	Numeric. Numbers between 1 and 10 giving the desired breakpoints in this interval.
...	Arguments passed on to pretty if log =FALSE

Value

A vector of breakpoints.

Author(s)

Bendix Carstensen, (bxc@steno.dk), <http://www.biostat.ku.dk/~bxc>

See Also

[pretty](#)

Examples

```
nice( exp( rnorm( 100 ) ), log=TRUE )
```

nickel*A Cohort of Nickel Smelters in South Wales*

Description

The **nickel** data frame has 679 rows and 7 columns. The data concern a cohort of nickel smelting workers in South Wales and are taken from Breslow and Day, Volume 2. For comparison purposes, England and Wales mortality rates (per 1,000,000 per annum) from lung cancer (ICDs 162 and 163), nasal cancer (ICD 160), and all causes, by age group and calendar period, are supplied in the dataset [ewrates](#).

Format

This data frame contains the following columns:

id:	Subject identifier (numeric)
icd:	ICD cause of death if dead, 0 otherwise (numeric)
exposure:	Exposure index for workplace (numeric)
dob:	Date of birth (numeric)
age1st:	Age at first exposure (numeric)
agein:	Age at start of follow-up (numeric)
ageout:	Age at end of follow-up (numeric)

Source

Breslow NE, and Day N, Statistical Methods in Cancer Research. Volume II: The Design and Analysis of Cohort Studies. IARC Scientific Publications, IARC:Lyon, 1987.

Examples

```
data(nickel)
str(nickel)
```

pctab*Create percentages in a table*

Description

Computes percentages and a margin of totals along a given margin of a table.

Usage

```
pctab(TT, margin = length(dim(TT)), dec=1)
```

Arguments

TT	A table or array object
margin	Which margin should be the the total?
dec	How many decimals should be printed?

Value

A table, where all dimensions except the one specified **margin** has two extra levels named "All" (where all entries are 100) and "N". The function prints the table with **dec** decimals.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.biostat.ku.dk/~bxc>.

See Also

[addmargins](#)

Examples

```
Aye <- sample( c("Yes","Si","Oui"), 177, replace=TRUE )
Bee <- sample( c("Hum","Buzz"), 177, replace=TRUE )
Sea <- sample( c("White","Black","Red","Dead"), 177, replace=TRUE )
A <- table( Aye, Bee, Sea )
A
ftable( ptab( A ) )
ftable( ptab( addmargins( A, 1 ), 3 ) )
round( ftable( ptab( addmargins( A, 1 ), 3 ), row.vars=3 ), 1)
```

plot.Lexis

Lexis diagrams

Description

The follow-up histories represented by a Lexis object can be plotted using one or two dimensions. The two dimensional plot is a Lexis diagram showing follow-up time simultaneously on two time scales.

Usage

```
## S3 method for class 'Lexis':
plot(x, time.scale = NULL, breaks="lightgray", ...)
## S3 method for class 'Lexis':
points(x, time.scale = NULL, ...)
## S3 method for class 'Lexis':
lines(x, time.scale = NULL, ...)
```

Arguments

x	An object of class Lexis
time.scale	A vector of length 1 or 2 giving the time scales to be plotted either by name or numerical order
breaks	a string giving the colour of grid lines to be drawn when plotting a split Lexis object. Grid lines can be suppressed by supplying the value NULL to the breaks argument
...	Further graphical parameters to be passed to the plotting methods. Grids can be drawn (behind the life lines) using the following parameters in plot : <ul style="list-style-type: none"> • grid If logical, a background grid is set up using the axis ticks. If a list, the first component is used as positions for the vertical lines and the last as positions for the horizontal. If a numerical vector, grids on both axes are set up using the distance between the numbers. • col.grid="lightgray" Color of the background grid. • lty.grid=2 Line type for the grid. • coh.grid=FALSE Should a 45 degree grid be plotted?

Details

The plot method for **Lexis** objects traces “life lines” from the start to the end of follow-up. The **points** method plots points at the end of the life lines.

If **time.scale** is of length 1, the life lines are drawn horizontally, with the time scale on the X axis and the id value on the Y axis. If **time.scale** is of length 2, a Lexis diagram is produced, with diagonal life lines plotted against both time scales simultaneously.

If **lex** has been split along one of the time axes by a call to **splitLexis**, then vertical or horizontal grid lines are plotted (on top of the life lines) at the break points.

Author(s)

Martyn Plummer

See Also[Lexis](#), [splitLexis](#)**Examples**

```
# A small bogus cohort
xcoh <- structure( list( id = c("A", "B", "C"),
                        birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                        entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                        exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                        fail = c(1, 0, 1) ),
                  .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                  class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
xcoh

# Define as Lexis object with timescales calendar time and age
Lcoh <- Lexis( entry = list( per=en ),
              exit = list( per=ex, age=ex-bt ),
              exit.status = fail,
              data = xcoh )

# Default plot of follow-up
plot( Lcoh )
# With a grid and deaths as endpoints
plot( Lcoh, grid=0:10*10, col="black" )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1] )
# With a lot of bells and whistles:
plot( Lcoh, grid=0:20*5, col="black", xaxs="i", yaxs="i",
      xlim=c(1960,2010), ylim=c(0,50), lwd=3, las=1 )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1], col="red", cex=1.5 )
```

plotEst

*Plot estimates with confidence limits***Description**

Plots parameter estimates with confidence intervals, annotated with parameter names. A dot is plotted at the estimate and a horizontal line extending from the lower to the upper limit is superimposed.

Usage

```
plotEst( ests,
        y = dim(ests)[1]:1,
        txt = rownames(ests),
        txtpos = y,
        ylim = range(y)-c(0.5,0),
```

```

      xlab = "",
      xtic = nice(ests[!is.na(ests)], log = xlog),
      xlim = range( xtic ),
      xlog = FALSE,
      pch = 16,
      cex = 1,
      lwd = 2,
      col = "black",
      col.lines = col,
      col.points = col,
      vref = NULL,
      grid = FALSE,
      col.grid = gray(0.9),
      restore.par = TRUE )

linesEst( ests, y = dim(ests)[1]:1, pch = 16, cex = 1, lwd = 2,
          col="black", col.lines=col, col.points=col )

pointsEst( ests, y = dim(ests)[1]:1, pch = 16, cex = 1, lwd = 2,
           col="black", col.lines=col, col.points=col )

```

Arguments

ests	Matrix with three columns: Estimate, lower limit, upper limit. If a model object is supplied, ci.lin is invoked for this object first.
y	Vertical position of the lines.
txt	Annotation of the estimates.
txtpos	Vertical position of the text. Defaults to y .
ylim	Extent of the vertical axis.
xlab	Annotation of the horizontal axis.
xtic	Location of tickmarks on the x-axis.
xlim	Extent of the x-axis.
xlog	Should the x-axis be logarithmic?
pch	What symbol should be used?
cex	Expansion of the symbol.
col	Colour of the points and lines.
col.lines	Colour of the lines.
col.points	Colour of the symbol.
lwd	Thickness of the lines.
vref	Where should vertical reference line(s) be drawn?
grid	If TRUE, vertical gridlines are drawn at the tickmarks. If a numerical vector is given vertical lines are drawn at grid .
col.grid	Colour of the vertical gridlines
restore.par	Should the graphics parameters be restored? If set to FALSE the coordinate system will still be available for additional plotting, and par("mai") will still have the very large value set in order to make room for the labelling of the estimates.

Details

plotEst make a news plot, whereas **linesEst** and **pointsEst** (identical functions) adds to an existing plot.

Value

NULL

Author(s)

Bendix Carstensen, (bxc@steno.dk), <http://www.pubhealth.ku.dk/~bxc>

See Also

ci.lin

Examples

```
# Bogus data and a linear model
f <- factor( sample( letters[1:5], 100, replace=TRUE ) )
x <- rnorm( 100 )
y <- 5 + 2 * as.integer( f ) + 0.8 * x + rnorm(100) * 2
m1 <- lm( y ~ f )

# Produce some confidence intervals for contrast to first level
( cf <- summary( m1 )$coef[2:5,1:2] %*% rbind( c(1,1,1), 1.96*(c(0,-1,1) ) ) )

# Plots with increasing amount of bells and whistles
par( mfc=c(3,2), mar=c(3,3,2,1) )
plotEst( cf )
plotEst( cf, grid=TRUE )
plotEst( cf, grid=TRUE, cex=2, lwd=3 )
plotEst( cf, grid=TRUE, cex=2, col.points="red", col.lines="green" )
plotEst( cf, grid=TRUE, cex=2, col.points="red", col.lines="green",
        xlog=TRUE, xtic=c(1:8), xlim=c(0.8,6) )
rownames( cf )[1] <- "Contrast to fa:\n\n fb"
plotEst( cf, grid=TRUE, cex=2, col.points=rainbow(4), col.lines=rainbow(4), vref=1 )
```

plotevent

Plot Equivalence Classes

Description

For interval censored data, segments of times between last.well and first.ill are plotted for each conversion in the data. It also plots the equivalence classes.

Usage

```
plotevent(last.well, first.ill, data)
```

Arguments

last.well	Time at which the individuals are last seen negative for the event
first.ill	Time at which the individuals are first seen positive for the event
data	Data with a transversal shape

Details

last.well and first.ill should be written as character in the function.

Value

Graph

Author(s)

Delphine Maucort-Boulch, Bendix Carstensen, Martyn Plummer

References

- Carstensen B. Regression models for interval censored survival data: application to HIV infection in Danish homosexual men. Stat Med. 1996 Oct 30;15(20):2177-89.
- Lindsey JC, Ryan LM. Tutorial in biostatistics methods for interval-censored data. Stat Med. 1998 Jan 30;17(2):219-38.

See Also

[Icens](#)

`projection.ip`

Projection of columns of a matrix.

Description

Projects the columns of the matrix **M** on the space spanned by the columns of the matrix **X**, with respect to the inner product defined by **weight**: $\langle \mathbf{x} | \mathbf{y} \rangle = \text{sum}(\mathbf{x} * \mathbf{w} * \mathbf{y})$.

Usage

```
projection.ip(X, M, orth = FALSE, weight = rep(1, nrow(X)))
```

Arguments

X	Matrix defining the space to project onto.
M	Matrix of columns to be projected. Must have the same number of rows as X .
orth	Should the projection be on the orthogonal complement to <code>span(X)</code> ?
weight	Weights defining the inner product. Numerical vector of length <code>nrow(X)</code> .

Value

A matrix of full rank with columns in `span(X)`.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.pubhealth.ku.dk/~bxc>, with help from Peter Dalgaard.

See Also

[detrend](#)

`rateplot`

Functions to plot rates from a table classified by age and calendar time (period)

Description

Produces plots of rates versus age, connected within period or cohort (**Aplot**), rates versus period connected within age-groups (**Pplot**) and rates and rates versus date of birth cohort (**Cplot**). **rateplot** is a wrapper for these, allowing to produce the four classical displays with a single call.

Usage

```

rateplot( rates,
  which = c("ap","ac","pa","ca"),
  age = as.numeric( dimnames( rates )[[1]] ),
  per = as.numeric( dimnames( rates )[[2]] ),
  grid = FALSE,
  a.grid = grid,
  p.grid = grid,
  c.grid = grid,
  ygrid = grid,
  col.grid = gray( 0.9 ),
  a.lim = range( age, na.rm=TRUE ) + c(0,diff( range( age ) )/30),
  p.lim = range( per, na.rm=TRUE ) + c(0,diff( range( age ) )/30),
  c.lim = NULL,
  ylim = range( rates[rates>0], na.rm=TRUE ),
  at = NULL,
  labels = paste( at ),
  a.lab = "Age at diagnosis",
  p.lab = "Date of diagnosis",
  c.lab = "Date of birth",
  ylab = "Rates",
  type = "l",
  lwd = 2,
  lty = 1,
  log.ax = "y",
  las = 1,
  ann = FALSE,
  a.ann = ann,
  p.ann = ann,
  c.ann = ann,
  xannx = 1/20,
  cex.ann = 0.8,
  a.thin = seq( 1, length( age ), 2 ),
  p.thin = seq( 1, length( per ), 2 ),
  c.thin = seq( 2, length( age ) + length( per ) - 1, 2 ),
  col = par( "fg" ),
  a.col = col,
  p.col = col,
  c.col = col,
  ... )

Aplot( rates, age = as.numeric( dimnames( rates )[[1]] ),
  per = as.numeric( dimnames( rates )[[2]] ), grid = FALSE,
  a.grid = grid, ygrid = grid, col.grid = gray( 0.9 ),
  a.lim = range( age, na.rm=TRUE ), ylim = range( rates[rates>0], na.rm=TRUE ),
  at = NULL, labels = paste( at ), a.lab = names( dimnames( rates ) )[[1]],
  ylab = deparse( substitute( rates ) ), type = "l", lwd = 2, lty = 1,
  col = par( "fg" ), log.ax = "y", las = 1, c.col = col, p.col = col,
  c.ann = FALSE, p.ann = FALSE, xannx = 1/20, cex.ann = 0.8,
  c.thin = seq( 2, length( age ) + length( per ) - 1, 2 ),
  p.thin = seq( 1, length( per ), 2 ), p.lines = TRUE,
  c.lines = !p.lines, ... )

Pplot( rates, age = as.numeric( dimnames( rates )[[1]] ),
  per = as.numeric( dimnames( rates )[[2]] ), grid = FALSE,
  p.grid = grid, ygrid = grid, col.grid = gray( 0.9 ),
  p.lim = range( per, na.rm=TRUE ) + c(0,diff(range(per))/30),
  ylim = range( rates[rates>0], na.rm=TRUE ), p.lab = names( dimnames( rates ) )[[2]],
  ylab = deparse( substitute( rates ) ), at = NULL, labels = paste( at ),

```

```

type = "l", lwd = 2, lty = 1, col = par( "fg" ), log.ax = "y",
las = 1, ann = FALSE, cex.ann = 0.8, xannx = 1/20,
a.thin = seq( 1, length( age ), 2 ), ... )

Cplot( rates, age = as.numeric( rownames( rates ) ),
per = as.numeric( colnames( rates ) ), grid = FALSE,
c.grid = grid, ygrid = grid, col.grid = gray( 0.9 ),
c.lim = NULL, ylim = range( rates[rates>0], na.rm=TRUE ),
at = NULL, labels = paste( at ), c.lab = names( dimnames( rates ) )[2],
ylab = deparse( substitute( rates ) ), type = "l", lwd = 2, lty = 1,
col = par( "fg" ), log.ax = "y", las = 1, xannx = 1/20, ann = FALSE,
cex.ann = 0.8, a.thin = seq( 1, length( age ), 2 ), ... )

```

Arguments

rates	A two-dimensional table (or array) with rates to be plotted. It is assumed that the first dimension is age and the second is period.
which	A character vector with elements from <code>c("ap", "ac", "apc", "pa", "ca")</code> , indication which plots should be produced. One plot per element is produced. The first letter indicates the x-axis of the plot, the remaining which groups should be connected, i.e. "pa" will plot rates versus period and connect age-classes, and "apc" will plot rates versus age, and connect both periods and cohorts.
age	Numerical vector giving the means of the age-classes. Defaults to the rownames of rates as numeric.
per	Numerical vector giving the means of the periods. Defaults to the columnnames of rates as numeric.
grid	Logical indicating whether a background grid should be drawn.
a.grid	Logical indicating whether a background grid on the age-axis should be drawn. If numerical it indicates the age-coordinates of the grid.
p.grid	do. for the period.
c.grid	do. for the cohort.
ygrid	do. for the rate-dimension.
col.grid	The colour of the grid.
a.lim	Range for the age-axis.
p.lim	Range for the period-axis.
c.lim	Range for the cohort-axis.
ylim	Range for the y-axis (rates).
at	Position of labels on the y-axis (rates).
labels	Labels to put on the y-axis (rates).
a.lab	Text on the age-axis. Defaults to "Age".
p.lab	Text on the period-axis. Defaults to "Date of diagnosis".
c.lab	Text on the cohort-axis. Defaults to "Date of birth".
ylab	Text on the rate-axis. Defaults to the name of the rate-table.
type	How should the curves be plotted. Defaults to "l".
lwd	Width of the lines. Defaults to 2.
lty	Which type of lines should be used. Defaults to 1, a solid line.
log.ax	Character with letters from "apcyr", indicating which axes should be logarithmic. "y" and "r" both refer to the rate scale. Defaults to "y".
las	see par .
ann	Should the curves be annotated?

<code>a.ann</code>	Logical indicating whether age-curves should be annotated.
<code>p.ann</code>	do. for period-curves.
<code>c.ann</code>	do. for cohort-curves.
<code>xannx</code>	The fraction that the x-axis is expanded when curves are annotated.
<code>cex.ann</code>	Expansion factor for characters annotating curves.
<code>a.thin</code>	Vector of integers indicating which of the age-classes should be labelled.
<code>p.thin</code>	do. for the periods.
<code>c.thin</code>	do. for the cohorts.
<code>col</code>	Colours for the curves.
<code>a.col</code>	Colours for the age-curves.
<code>p.col</code>	do. for the period-curves.
<code>c.col</code>	do. for the cohort-curves.
<code>p.lines</code>	Should rates from the same period be connected?
<code>c.lines</code>	Should rates from the same cohort be connected?
<code>...</code>	Additional arguments passed on to <code>matlines</code> when plotting the curves.

Details

Zero values of the rates are ignored. They are neither in the plot nor in the calculation of the axis ranges.

Value

NULL. The function is used for its side-effect, the plot.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://www.pubhealth.ku.dk/~bxc/>

See Also

[apc.frame](#)

Examples

```
data( blcaIT )
attach(blcaIT)

# Table of rates:
bl.rate <- tapply( D, list(age,period), sum ) /
               tapply( Y, list(age,period), sum )
bl.rate

# The four classical plots:
par( mfrow=c(2,2) )
rateplot( bl.rate*10^6 )

# The labels on the vertical axis could be nicer:
rateplot( bl.rate*10^6, at=10^(-1:3), labels=c(0.1,1,10,100,1000) )

# More bells and whistles
par( mfrow=c(1,3), mar=c(3,3,1,1), oma=c(0,3,0,0), mgp=c(3,1,0)/1.6 )
rateplot( bl.rate*10^6, ylab="", ann=TRUE, which=c("AC","PA","CA"),
          at=10^(-1:3), labels=c(0.1,1,10,100,1000),
          col=topo.colors(11), cex.ann=1.2 )
```

Relevel*Reorder and combine levels of a factor*

Description

The levels of a factor are re-ordered so that the levels specified by **ref** is first and the others are moved down. This is useful for **contr.treatment** contrasts which take the first level as the reference. Levels may also be combined.

Usage

```
Relevel(f, ref, first = TRUE, collapse="+" )
```

Arguments

f	An unordered factor
ref	The names or numbers of levels to be the first. If ref is a list, factor levels mentioned in each list element are combined. If the list is named the names are used as new factor levels.
first	Should the levels mentioned in ref come before those not?
collapse	String used when collapsing factor levels.

Value

An unordered factor.

Examples

```
ff <- factor( sample( letters[1:5], 100, replace=TRUE ) )
table( ff, Relevel( ff, list( AB=1:2, "Dee"=4, c(3,5) ) ) )
table( ff, rr=Relevel( ff, list( 5:4, Z=c("c","a") ), coll="-und-", first=FALSE ) )
```

ROC*Function to compute and draw ROC-curves.*

Description

Computes sensitivity, specificity and positive and negative predictive values for a test based on dichotomizing along the variable **test**, for prediction of **stat**. Alternatively a model formula may given, in which case the the linear predictor is the test variable and the response is taken as the true status variable. Plots curves of these and a ROC-curve.

Usage

```
ROC( test = NULL,
     stat = NULL,
     form = NULL,
     plot = c("sp", "ROC"),
     PS = is.null(test),
     PV = TRUE,
     MX = TRUE,
     MI = TRUE,
     AUC = TRUE,
     grid = seq(0,100,10),
     col.grid = gray( 0.9 ),
     cuts = NULL,
     lwd = 2,
     data = parent.frame(),
     ... )
```

Arguments

test	Numerical variable used for prediction.
stat	Logical variable of true status.
form	Formula used in a logistic regression. If this is given, test and stat are ignored. If not given then both test and stat must be supplied.
plot	Character variable. If "sp", the a plot of sensitivity, specificity and predictive values against test is produced, if "ROC" a ROC-curve is plotted. Both may be given.
PS	logical, if TRUE the x-axis in the plot "ps"-plot is the the predicted probability for stat ==TRUE, otherwise it is the scale of test if this is given otherwise the scale of the linear predictor from the logistic regression.
PV	Should sensitivity, specificity and predictive values at the optimal cutpoint be given on the ROC plot?
MX	Should the "optimal cutpoint" (i.e. where sens+spec is maximal) be indicated on the ROC curve?
MI	Should model summary from the logistic regression model be printed in the plot?
AUC	Should the area under the curve (AUC) be printed in the ROC plot?
grid	Numeric or logical. If FALSE no background grid is drawn. Otherwise a grid is drawn on both axes at grid percent.
col.grid	Colour of the grid lines drawn.
cuts	Points on the test-scale to be annotated on the ROC-curve.
lwd	Thickness of the curves
data	Data frame in which to interpret the variables.
...	Additional arguments for the plotting of the ROC-curve. Passed on to plot

Value

A list with two components:

res	dataframe with variables sn, sp, pvp, pvn and fv. The latter is the unique values of test (for PS==FALSE) or linear predictor from the logistic regression
lr	glm object with the logistic regression result used for construction of the ROC curve

0, 1 or 2 plots are produced according to the setting of **plot**.

Author(s)

Bendix Carstensen, Steno Diabetes Center & University of Copenhagen, <http://www.biostat.ku.dk/~bxc>

References

Examples

```
x <- rnorm( 100 )
z <- rnorm( 100 )
w <- rnorm( 100 )
tigol <- function( x ) 1 - ( 1 + exp( x ) )^(-1)
y <- rbinom( 100, 1, tigol( 0.3 + 3*x + 5*z + 7*w ) )
ROC( form = y ~ x + z, plot="ROC" )
```

S.typh

Salmonella Typhimurium outbreak 1996 in Denmark.

Description

Matched case-control study of food poisoning.

Format

A data frame with 136 observations on the following 15 variables:

id:	Person identification
set:	Matched set indicator
case:	Case-control status (1:case, 0:control)
age:	Age of individual
sex:	Sex of individual (1:male, 2:female)
abroad:	Within the last two weeks visited abroad (1:yes, 0:no)
beef:	Within the last two weeks eaten beef
pork:	Within the last two weeks eaten pork
veal:	Within the last two weeks eaten veal
poultry:	Within the last two weeks eaten poultry
liverp:	Within the last two weeks eaten liverpaste
veg:	Within the last two weeks eaten vegetables
fruit:	Within the last two weeks eaten fruit
egg:	Within the last two weeks eaten eggs
plant7:	Within the last two weeks eaten meat from plant no. 7

Details

In the fall of 1996 an unusually large number of Salmonella Typhimurium cases were recorded in Fyn county in Denmark. The Danish Zoonosis Centre set up a matched case-control study to find the sources. Cases and two age-, sex- and residency-matched controls were telephone interviewed about their food intake during the last two weeks.

The participants were asked at which retailer(s) they had purchased meat. Retailers were independently of this linked to meat processing plants, and thus participants were linked to meat processing plants. This way persons could be linked to (amongst other) plant no 7.

Source

Tine Hald.

References

Molbak K and Hald T: Salmonella Typhimurium outbreak in late summer 1996. A Case-control study. (In Danish: Salmonella typhimurium udbrud paa Fyn sensommeren 1996. En case-kontrol undersogelse.) Ugeskrift for Laeger., 159(36):5372-7, 1997.

Examples

```
data(S.typh)
```

splitLexis

Split follow-up time in a Lexis object

Description

The `splitLexis` function divides each row of a `Lexis` object into disjoint follow-up intervals according to the supplied break points.

Usage

```
splitLexis(lex, breaks, time.scale)
```

Arguments

lex	an object of class Lexis
breaks	a vector of break points
time.scale	the name or number of the time scale to be split

Value

An object of class **Lexis** with multiple rows for each row of the argument **lex**. Each row of the new **Lexis** object contains the part of the follow-up interval that falls inside one of the time bands defined by the break points.

The variables representing the various time scales, are appropriately updated in the new **Lexis** object. The entry and exit status variables are also updated according to the rule that the entry status is retained until the end of follow-up. All other variables are considered to represent variables that are constant in time, and so are replicated across all rows having the same id value.

Note

The `splitLexis()` function divides follow-up time into intervals using breakpoints that are common to all rows of the **Lexis** object. To split a **Lexis** object by break points that are unique to each row, use the `cut.Lexis` function.

Author(s)

Martyn Plummer

See Also

[timeBand](#), [cutLexis](#)

Examples

```
# A small bogus cohort
xcoh <- structure( list( id = c("A", "B", "C"),
                        birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                        entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                        exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                        fail = c(1, 0, 1) ),
                  .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                  class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
xcoh

# Define as Lexis object with timescales calendar time and age
Lcoh <- Lexis( entry = list( per=en ),
              exit = list( per=ex, age=ex-bt ),
              exit.status = fail,
              data = xcoh )
```



```

# Default plot of follow-up
plot( Lcoh )
# With a grid and deaths as endpoints
plot( Lcoh, grid=0:10*10, col="black" )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1] )
# With a lot of bells and whistles:
plot( Lcoh, grid=0:20*5, col="black", xaxs="i", yaxs="i",
      xlim=c(1960,2010), ylim=c(0,50), lwd=3, las=1 )
points( Lcoh, pch=c(NA,16)[Lcoh$lex.Xst+1], col="red", cex=1.5 )

# Split time along two time-axes
( x2 <- splitLexis( Lcoh, breaks = seq(1900,2000,5), time.scale="per") )
( x2 <- splitLexis( x2, breaks = seq(0,80,5), time.scale="age" ) )

# Tabulate the cases and the person-years
tab.Lexis( x2 )
tapply( status(x2,"exit")==1, list( timeBand(x2,"age","left"),
                                   timeBand(x2,"per","left") ), sum )
tapply( dur(x2), list( timeBand(x2,"age","left"),
                      timeBand(x2,"per","left") ), sum )

```

start.Lexis

Time series methods for Lexis objects

Description

Extract the entry time, exit time, status, or duration of follow-up from a **Lexis** object.

Usage

```

entry(x, time.scale = NULL)
exit(x, time.scale = NULL)
status(x, at=c("entry","exit"))
dur(x)

```

Arguments

x	an object of class Lexis .
time.scale	a string or integer indicating the time scale. If omitted, all times scales are used.
at	string indicating the time point(s) at which status is to be measured.

Value

The **entry** and **exit** functions return a matrix containing the entry times and exit times, respectively, on the requested time scales. The **status** function returns a dataframe with the status at entry and/or exit and **dur** returns a vector with the lengths of the follow-up intervals.

Author(s)

Martyn Plummer

See Also

[Lexis](#)

stattable.funs
Special functions for use in stat.table

Description

These functions may be used as **contents** arguments to the function **stat.table**. They are defined internally in **stat.table** and have no independent existence.

Usage

```
count(id)
ratio(d,y,scale=1, na.rm=TRUE)
percent(...)
```

Arguments

id	numeric vector in which identical values identify the same individual.
d, y	numeric vectors of equal length (d for Deaths, y for person-Years)
scale	a scalar giving a value by which the ratio should be multiplied
na.rm	a logical value indicating whether NA values should be stripped before computation proceeds.
...	a list of variables taken from the index argument to stat.table

Value

When used as a **contents** argument to **stat.table**, these functions create the following tables:

this-is-escaped-codenormal-bracket30bracket-normal

If given without argument (**count()**) it returns a contingency table of counts. If given an **id** argument it returns a table of the number of different values of **id** in each cell, i.e. how many persons contribute in each cell.

this-is-escaped-codenormal-bracket36bracket-normal

returns a table of values **scale * sum(d)/sum(y)**

this-is-escaped-codenormal-bracket40bracket-normal

returns a table of percentages of the classifying variables. Variables that are in the **index** argument to **stat.table** but not in the call to **percent** are used to define strata, within which the percentages add up to 100.

normal-bracket46bracket-normal

Author(s)

Martyn Plummer

See Also

stat.table

stat.table

Tables of summary statistics

Description

stat.table creates tabular summaries of the data, using a limited set of functions. A list of index variables is used to cross-classify summary statistics.

Usage

```
stat.table(index, contents = count(), data, margins = FALSE)
## S3 method for class 'stat.table':
print(x, width=7, digits,...)
```

Arguments

index	A factor, or list of factors, used for cross-classification. If the list is named, then the names will be used when printing the table. This feature can be used to give informative labels to the variables.
contents	A function call, or list of function calls. Only a limited set of functions may be called (See Details below). If the list is named, then the names will be used when printing the table.
data	an optional data frame containing the variables to be tabulated. If this is omitted, the variables will be searched for in the calling environment.
margins	a logical scalar or vector indicating which marginal tables are to be calculated. If a vector, it should be the same length as the index argument: values corresponding to TRUE will be retained in marginal tables.
x	an object of class stat.table .
width	a scalar giving the minimum column width when printing.
digits	a scalar, or named vector, giving the number of digits to print after the decimal point. If a named vector is used, the names should correspond to one of the permitted functions (See Details below) and all results obtained with that function will be printed with the same precision.
...	further arguments passed to other print methods.

Details

This function is similar to **tapply**, with some enhancements: multiple summaries of multiple variables may be mixed in the same table; marginal tables may be calculated; columns and rows may be given informative labels; pretty printing may be controlled by the associated print method.

This function is not a replacement for **tapply** as it also has some limitations. The only functions that may be used in the **contents** argument are: **count**, **mean**, **weighted.mean**, **sum**, **quantile**, **median**, **IQR**, **max**, **min**, **ratio**, and **percent**.

The **count()** function, which is the default, simply creates a contingency table of counts. The other functions are applied to each cell created by combinations of the **index** variables.

Value

An object of class **stat.table**, which is a multi-dimensional array. A print method is available to create formatted one-way and two-way tables.

Note

The permitted functions in the contents list are defined inside **stat.table**. They have the same interface as the functions callable from the command line, except for two differences. If there is an argument **na.rm** then its default value is always **TRUE**. A second difference is that the **quantile** function can only produce a single quantile in each call.

Author(s)

Martyn Plummer

See Also[table](#), [tapply](#), [mean](#), [weighted.mean](#), [sum](#), [quantile](#), [median](#), [IQR](#), [max](#), [min](#), [ratio](#), [percent](#), [count](#)**Examples**

```

data(warpbreaks)
# A one-way table
stat.table(tension,list(count(),mean(breaks)),data=warpbreaks)
# The same table with informative labels
stat.table(index=list("Tension level"=tension),list(N=count(),
            "mean number of breaks"=mean(breaks)),data=warpbreaks)

# A two-way table
stat.table(index=list(tension,wool),mean(breaks),data=warpbreaks)
# The same table with margins over tension, but not wool
stat.table(index=list(tension,wool),mean(breaks),data=warpbreaks,
            margins=c(TRUE, FALSE))

# A table of column percentages
stat.table(list(tension,wool), percent(tension), data=warpbreaks)
# Cell percentages, with margins
stat.table(list(tension,wool),percent(tension,wool), margin=TRUE,
            data=warpbreaks)

# A table with multiple statistics
# Note how each statistic has its own default precision
a <- stat.table(index=list(wool,tension),
                contents=list(count(),mean(breaks),percent (wool)),
                data=warpbreaks)

print(a)
# Print the percentages rounded to the nearest integer
print(a, digits=c(percent=0))

# An Epidemiological example with follow-up time
data(nickel)
str(nickel)

# Make a grouped version of the exposure variable
nickel$egr <- cut( nickel$exposure, breaks=c(0, 0.5, 5, 10, 100), right=FALSE )
stat.table( egr, list( count(), percent(egr), mean( age1st ) ), data=nickel )

# Split the follow-up time by current age
nickel.ex <-
W.Lexis( entry=agein, exit=ageout, fail=icd %in% c(162,163),
          origin=0, breaks=seq(0,100,20),
          include=list( id, exposure, egr, age1st, icd ), data=nickel )
str( nickel.ex )

# Table of rates
stat.table( Time, list( n=count(), N=count(id), D=sum(Fail),
                        "Rate/1000"=ratio(Fail,Exit-Entry,1000) ),
            margin=1, data=nickel.ex )

# Two-way table of rates and no. persons contributing
stat.table( list(age=Time, Exposure=egr),
            list( N=count(id), D=sum(Fail), Y=sum((Exit-Entry)/1000),

```

```
Rate=ratio(Fail,Exit-Entry,1000) ),
margin=TRUE, data=nickel.ex )
```

subset.Lexis

Subsetting Lexis objects

Description

Return subsets of Lexis objects which meet conditions

Usage

```
## S3 method for class 'Lexis':
subset(x, ...)
```

Arguments

x an object of class **Lexis**
... additional arguments to be passed to **subset.data.frame**

Details

The subset method for **Lexis** objects works exactly as the method for data frames.

Value

A **Lexis** object with selected rows and columns.

Author(s)

Martyn Plummer

See Also

[Lexis](#), [merge.Lexis](#)

tab

Tabulate transitions and risk time from a Lexis object

Description

A two-way table of records and transitions classified by states (**lex.Cst** and **lex.Xst**), as well the risk time in each state.

Usage

```
## S3 method for class 'Lexis':
tab( x, simplify=TRUE, ... )
```

Arguments

x A Lexis object.
simplify Should rows with 0 follow-up time be dropped?
... Other parameters - ignored

Details

The first part of the table gives the number of rows in the Lexis object, classified by states. The D column is the number of transitions i.e. rows where `lex.Cst != lex.Xst`, and the final column Y is the person-years spent in each state.

Value

A matrix with rows classified by states where persons spend time, and columns classified by states to which persons transit. Two additional columns of transitions from and time spent in each state is added.

Author(s)

Bendix Carstensen, (bxc@steno.dk)

Examples

```
data( nickel )
nic <- Lexis( data=nickel,
             entry=list(age=agein),
             exit=list(age=ageout,cal=ageout+dob,tfh=ageout-age1st),
             # Lung cancer deaths and other deaths are coded 1 and 2
             exit.status=(icd > 0) + (icd %in% c(162,163)) )
str( nic )
head( nic )
tab.Lexis( nic )
```

`tabplot`

Graphical display of a 2-way contingency table

Description

Entries in a table are plotted as rectangles proportional to the entry in the table. Width of rectangles are proportional to column totals, height proportional to entries within each column, hence areas are proportional to entries in the table.

Usage

```
tabplot( M,
        col,
        border = "black",
        lwd = 2,
        collabs = TRUE,
        rowlabs = NULL,
        equal = FALSE,
        las = 1,
        main = NULL,
        cex.main = 1.0,
        vaxis = FALSE )
```

Arguments

<code>M</code>	Two-way table
<code>col</code>	colors to use for coloring within each column. Defaults to a grayscale. May also be a function that takes an integer argument, as e.g. <code>rainbow()</code> .
<code>border</code>	color of borders around rectangles.
<code>lwd</code>	width of the lines around rectangles.
<code>collabs</code>	should columns be labelled.

rowlabs	character "r" or "l": rows labelled on left or right side
equal	should columns be plotted of equal width? If yes a plot similar to that obtainable from <code>barplot</code> is the result.
las	how should labelling be rotated.
main	heading for the plot
cex.main	character expansion for the heading.
vaxis	should a vertical axis be drawn. If character it gives the side where it is drawn.

Details

The function offers a few more facilities for two-way tables than `mosaicplot`, but is restricted to two-way tables as input.

Value

NULL. The function is used for its sideeffects.

Author(s)

Bendix Carstensen, Steno Diabetes Center & Dept. of Biostatistics, University of Copenhagen
(bxc@steno.dk), <http://www.pubhealth.ku.dk/~bxc>

See Also

See Also `barplot`, `plot.table`, `mosaicplot`

Examples

```
b <- sample( letters[1:4], 300, replace=TRUE, prob=c(3,1,2,4)/10 )
a <- rnorm( 300 ) - as.integer( factor( b ) ) / 8
tb <- table( cut( a, -3:2 ), b )
tabplot( tb )
tabplot( tb, rowlabs="right", col=heat.colors )

# Very similar plots
ptb <- sweep( tb, 2, apply( tb, 2, sum ), "/" )
par( mfrow=c(2,2) )
barplot( ptb, space=0 )
tabplot( tb, equal=TRUE, lwd=1 )
tabplot( tb, equal=TRUE, lwd=1, rowlabs="l" )
tabplot( tb, equal=FALSE, lwd=1, rowlabs="l" )
```

thoro

Thorotrast Study

Description

The `thoro` data frame has 2470 rows and 14 columns. Each row represents one patient that have had cerebral angiography (X-ray of the brain) with an injected contrast medium, either Thorotrast or another one (the controls).

Format

This data frame contains the following columns:

id: Identification of person.
sex: Sex, 1: male / 2: female.
birthdat: Date of birth, **Date** variable.
contrast: Group, 1: Thorotrast / 2: Control.
injecdat: Date of contrast injection, **Date** variable.
volume: Injected volume of Thorotrast in ml. Control patients have a 0 in this variable.
exitdat: Date of exit from the study, **Date** variable.
exitstat: Status at exit, 1: dead / 2: alive, censored at closing of study, 20 February 1992 / 3: censored alive at some earlier date.
cause: Cause of death. See causes in the helpfile for [gmortDK](#)
liverdat: Date of liver cancer diagnosis, **Date** variable.
liver: Indicator of liver cancer diagnosis. Not all livercancers are histologically verified, hence `liver >= hepcc + chola`
hepcc: Hepatocellular carcinoma at **liverdat**.
chola: Cholangiocellular carcinoma at **liverdat**.
hmang: Haemangisarcoma carcinoma at **liverdat**.

Source

M Andersson, M Vyberg, J Visfeldt, B Carstensen & HH Storm: Primary liver tumours among Danish patients exposed to Thorotrast. Radiation Research, 137, pp. 262–273, 1994.

M Andersson, B Carstensen HH Storm: Mortality and cancer incidence after cerebral angiography. Radiation Research, 142, pp. 305–320, 1995.

See Also

[mortDK](#), [gmortDK](#)

Examples

```
data(thoro)
str(thoro)
```

timeBand

Extract time band data from a split Lexis object

Description

The break points of a **Lexis** object (created by a call to `splitLexis`) divide the follow-up intervals into time bands along a given time scale. The `breaks` function returns the break points, for a given time scale, and the `timeBand` classifies each row (=follow-up interval) into one of the time bands.

Usage

```
timeBand(lex, time.scale, type="integer")
breaks(lex, time.scale)
```

Arguments

lex an object of class **Lexis**
time.scale a character or integer vector of length 1 identifying the time scale of interest
type a string that determines how the time bands are labelled. See Details below

Details

Time bands may be labelled in various ways according to the **type** argument. The permitted values of the **type** argument, and the corresponding return values are:

"integer" a numeric vector with integer codes starting from 0.

"factor" a factor (unordered) with labels "(left,right]"

"left" the left-hand limit of the time band

"middle" the midpoint of the time band

"right" the right-hand limit of the time band

Value

The **breaks** function returns a vector of break points for the **Lexis** object, or NULL if no break points have been defined by a call to **splitLexis**. The **timeBand** function returns a numeric vector or factor, depending on the value of the **type** argument.

Note

A newly created **Lexis** object has no break points defined. In this case, **breaks** will return NULL, and **timeBand** will a vector of zeros.

Author(s)

Martyn Plummer

See Also

[Lexis](#)

timeScales

The time scales of a Lexis object

Description

Function to get the names of the time scales of a **Lexis** object.

Usage

```
timeScales(x)
```

Arguments

x an object of class **Lexis**

Value

A character vector containing the names of the variables in **x** that represent the time scales

Author(s)

Martyn Plummer

See Also

[Lexis](#), [splitLexis](#)

twoby2

Analysis of a two by two table

Description

Computes the usual measures of association in a 2 by 2 table with confidence intervals. Also produces asymptotic and exact tests. Assumes that comparison of probability of the first column level between levels of the row variable is of interest. Output requires that the input matrix has meaningful row and column labels.

Usage

```
twoby2(exposure, outcome,
       alpha = 0.05, print = TRUE, dec = 4,
       conf.level = 1-alpha, F.lim = 10000)
```

Arguments

exposure	If a table the analysis is based on the first two rows and first two columns of this. If a variable, this variable is tabulated against
outcome	as the second variable
alpha	Significance level
print	Should the results be printed?
dec	Number of decimals in the printout.
conf.level	1-alpha
F.lim	If the table total exceeds F.lim, Fisher's exact test is not computed

Value

A list with elements:

table	The analysed 2 x 2 table augmented with probabilities and confidence intervals. The confidence intervals for the probabilities are computed using the normal approximation to the log-odds. Confidence intervals for the difference of proportions are computed using method 10 from Newcombe, Stat.Med. 1998, 17, pp.873 ff.
measures	A table of Odds-ratios and relative risk with confidence intervals.
p.value	Exact p-value for the null hypothesis of OR=1

Author(s)

Mark Myatt. Modified by Bendix Carstensen.

Examples

```
Treat <- sample(c("A","B"), 50, rep=TRUE )
Resp <- c("Yes","No")[1+rbinom(50,1,0.3+0.2*(Treat=="A"))]
twoby2( Treat, Resp )
twoby2( table( Treat, Resp )[,2:1] ) # Comparison the other way round
```

Description

This is an old version of time-splitting. Consult the new version, [Lexis](#).

For cohort input data the follow-up time is chopped in pieces along several time scales, and a dataframe of follow-up intervals is returned. Entry and exit times are assumed to be in the same timescale (the input time scale).

Usage

```
W.Lexis( entry = 0,
         exit,
         fail,
         origin = 0,
         scale = 1,
         breaks,
         include = NULL,
         data = NULL )
```

Arguments

entry	Date of entry on the input timescale. Numerical variable.
exit	Date of exit on the input timescale. Numerical variable.
fail	Failure indicator.
origin	Origin of the output timescale(s) on the input timescale. If for example the input timescale is calendar time and the output timescale is (current) age, the the origin is date of birth. If more than one timescale is used for splitting time this is a list. Elements of the list must be named and must have the same names as those in scale and breaks .
scale	Scale of the output timescale(s) relative to the input timescale. Elements of the list must be named and have the same names as those in origin and breaks .
breaks	Points on the output scale where the follow-up is cut. If more than one timescale is used for splitting time this is a list. Elements of the list must be named and must have the same names as those in origin and scale .
include	List of variables to carry unchanged from the original dataframe to the output dataframe.
data	Dataframe in which to interpret the arguments.

Details

The **data** is assumed to be a dataframe describing the follow-up of a cohort, giving entry and exit time (on the input timescale) for each individual as well as the exit status (failure status, **fail**). The purpose of the function is to split each individual's follow-up time along a number of timescales for example age, calendar time, time since entry etc. Any follow-up time before the first break on any timescale or after the last break on any of these timescales (the output timescales) is discarded.

NOTE: If a person has his/her exit before the first break or his entry after the last break on any of the timescales the function will crash.

Value

A dataframe with one row per follow-up interval, with the following variables:

Expand	A numerical vector with values in <code>1:nrows(data)</code> , pointing at the rows of the input data frame that is expanded to the output intervals.
Entry	Date of entry for each interval. On the input time scale.
Exit	Date of exit for each interval. On the input time scale.

Fail	Exit status for each interval. Coded 0 for censoring, for the last follow-up interval for each person it takes the value of fail .
Time	<p>If origin, scale or breaks were given as vectors this gives the left endpoints of the intervals on the output scale.</p> <p>If origin, scale or breaks, were given as lists, there is no variable Time in the dataframe, instead variables with the same names as the list elements of these will be in the dataframe. The variables have values corresponding to the left endpoints of the intervals on the respective output time scales.</p>

Finally, variables given in the argument **include**, values replicated across all intervals from each individual.

Author(s)

David Clayton, approx. 2000. Small modifications by Bendix Carstensen.

References

This function has approximately the same functionality as **stsplit** in Stata and the SAS-macro **%Lexis** (<http://www.biostat.ku.dk/~bxc/Lexis/Lexis.sas>). It has been attempted to keep argument names similar between the three functions.

See Also

[Lexis](#), [Lexis.diagram](#)

Examples

```
# A small bogus cohort
#
xcoh <- structure( list( id = c("A", "B", "C"),
                        birth = c("14/07/1952", "01/04/1954", "10/06/1987"),
                        entry = c("04/08/1965", "08/09/1972", "23/12/1991"),
                        exit = c("27/06/1997", "23/05/1995", "24/07/1998"),
                        fail = c(1, 0, 1) ),
                  .Names = c("id", "birth", "entry", "exit", "fail"),
                  row.names = c("1", "2", "3"),
                  class = "data.frame" )

# Convert the character dates into numerical variables (fractional years)
#
xcoh$bt <- cal.yr( xcoh$birth, format="%d/%m/%Y" )
xcoh$en <- cal.yr( xcoh$entry, format="%d/%m/%Y" )
xcoh$ex <- cal.yr( xcoh$exit , format="%d/%m/%Y" )

# See how it looks
#
xcoh

# Split time along one time-axis
#
W.Lexis( entry = en,
        exit = ex,
        fail = fail,
        scale = 1,
        origin = bt,
        breaks = seq( 5, 40, 5 ),
        include = list( bt, en, ex, id ),
        data = xcoh )
```

```
# Split time along two time-axes
#
( x2 <-
W.Lexis( entry = en,
        exit = ex,
        fail = fail,
        scale = 1,
        origin = list( per=0, age=bt ),
        breaks = list( per=seq(1900,2000,10), age=seq(0,80,5) ),
        include = list( bt, en, ex, id ),
        data = xcoh ) )

# Tabulate the cases and the person-years
#
tapply( x2$Fail, list( x2$age, x2$per ), sum )
tapply( x2$Exit - x2$Entry, list( x2$age, x2$per ), sum )
```