Package 'Epi'

January 27, 2015

Version 1.1.67

Date 2014-07-25

Title A package for statistical analysis in epidemiology.

Depends R (>= 3.0.0), utils

Suggests splines, nlme, survival, mstate, etm, MASS, cmprsk

Description Functions for demographic and epidemiological analysis in
the Lexis diagram, i.e. register and cohort follow-up data, in
particular representation and manipulation of multistate data - the
Lexis suite of functions.
Includes interfaces to mstate, etm and cmprsk packages.
Also contains functions for Age-Period-Cohort modeling and a
function for interval censored data and some useful functions for
tabulation and plotting, as well some epidemiological datasets.

License GPL-2

URL http://BendixCarstensen.com/Epi/

Author Bendix Carstensen [aut, cre],
Martyn Plummer [aut],
Esa Laara [ctb],
Michael Hills [ctb]

Maintainer Bendix Carstensen <bxc@steno.dk>

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-07-23 12:12:27

R topics documented:

1

---

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,
           dist = c("poisson","binomial"),
          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 thsi 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. Also used as denominator for binomial data, see the dist argument.
ref.c	Reference cohort, numerical. Defaults to median date of birth among cases. If used with parm="AdCP" or parm="AdPC", the resdiual cohort effects will be 1 at ref.c
ref.p	Reference period, numerical. Defaults to median date of diagnosis among cases.
dist	Distribution (or more precisely: Likelihood) used for modelling. if a binomial model us ised, Y is assuemd to be the denominator; "binomial" gives a binomial model with logit link.
model	Type of model fitted:

- ns fits a model with natural splines for each of the terms, with npar parameters for the terms.
- bs fits a model with B-splines for each of the terms, with npar 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. npar is ignored in this case.

dr.extr   Character. 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   Character. Indicates the parametrization of the effects. The first four 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, the argument dr.extr is ignored. Possible values for parm are:

- "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 log(fitted values) from the Age-drift model as offset. Period effects are from the model with period alone using log(fitted values) 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 log(fitted values) from the Age-drift model as offset. Cohort effects are from the model with cohort alone using log(fitted values) 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 log(fitted values) 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 log(fitted values) from the Age-Period model as offset.

npar   The number of parameters/knots to use for each of the terms in the model. If it is vector of length 3, the numbers are taken as the no. of knots for Age, Period and Cohort, respctively. Unless it has a names attribute with vales "A", "P" and "C" in which case these will be used. The knots chosen are the quantiles (1:nk+0.1)/(nk+0.2) of the events (i.e. of rep(A,D))

> npar may also be a named list of three numerical vectors with names "A", "P" and "C", 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 (1-alpha) confidence limits.

scale   numeric(1), 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.plot] and [apc.lines]) — a list with components:

Type   Text describing the model and parametrization returned

Model   The model object(s) on which the parametrization is based.

Age   Matrix with 4 colums: A.pt with the ages (equals unique(A)) and three columns giving the estimated rates with c.i.s.

Per   Matrix with 4 colums: P.pt with the dates of diagnosis (equals unique(P)) and three columns giving the estimated RRs with c.i.s.

Coh   Matrix with 4 colums: C.pt with the dates of birth (equals unique(P-A)) 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 drift), 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 ref.p or ref.c 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 model 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://BendixCarstensen.com>

References

The considerations behind the parametrizations used in this function are given in details in a preprint from Department of Biostatistics in Copenhagen: "Demography and epidemiology: Age-Period-Cohort models in the computer age", http://biostat.ku.dk/reports/2006/ResearchReport06-1.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",
             ref.line = TRUE,
                  gap = diff(range(c(a.lab, a.tic)))/10,
             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)
`ref.line`	Logical. Should a reference line at RR=1 be drawn at the calendar time part of the plot?
`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 form 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. Moreover, the option apc.frame.par is given the value c("cp.offset","RR.fac"), which is recognized by apc.plot and apc.lines.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://BendixCarstensen.com

References

B. Carstensen: Age-Period-Cohort models for the Lexis diagram. Statistics in Medicine, 26: 3018-3045, 2007.

See Also

apc.lines,apc.fit

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

```
## S3 method for class 'apc'
lines( x, P, C,
         scale = c("log","ln","rates","inc","RR"),
     frame.par = options()[["apc.frame.par"]],
         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,
           ... )
 apc.lines( x, P, C,
         scale = c("log","ln","rates","inc","RR"),
     frame.par = options()[["apc.frame.par"]],
         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

x	If an apc-object, (see `apc.fit`), then the arguments P, C, c0, a0 and p0 are ignored, and the estimates from x plotted.
	Can also be a 4-column matrix with columns age, age-specific rates, lower and upper c.i., in which case period and cohort effects are taken from the arguments P and C.
P	Period effects. Rate-ratios. Same form as for the age-effects.
C	Cohort effects. Rate-ratios. Same form as for the age-effects.
scale	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 apc, it is assumed to be "rates".
frame.par	2-element vector with the cohort-period offset and RR multiplicator. This will typically be the result from the call of `apc.frame`. See this for details.

drift	The drift parameter to be added to the period effect. If scale="log" 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
c0	The cohort where the drift is assumed to be 0; the subtracted drift effect is drift*(C[,1]-c0).
a0	The age where the drift is assumed to be 0.
p0	The period where the drift is assumed to be 0.
ci	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.
lwd	Line widths for estimates, lower and upper confidence limits.
lty	Linetypes for the three effects.
col	Colours for the three effects.
type	What type of lines / points should be used.
knots	Should knots from the model be shown?
...	Further parameters to be transmitted to points lines, matpoints or matlines 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

APC.lines returns (invisibly) a list of three matrices of the effects plotted.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://BendixCarstensen.com>

See Also

[apc.frame](), [pc.lines](), [apc.fit](), [apc.plot]()

B.dk	*Births in Denmark by year and month of birth and sex*

Description

The number of live births as entered from printed publications from Statistics Denmark.

Usage

```
data(B.dk)
```

Format

A data frame with 1248 observations on the following 4 variables.

year Year of birth

month Month of birth

m Number of male births

f Number of female births

Details

Division of births by month and sex is only avaialble for the years 1957–69 and 2002ff. For the
remaining period, the total no. births in each month is divided between the sexes so that the fraction
of boys is equal to the overall fraction for the years where the sex information is available.

There is a break in the series at 1920, when Sonderjylland was joined to Denmark.

Source

Statistiske Undersogelser nr. 19: Befolkningsudvikling og sundhedsforhold 1901-60, Copenhagen
1966. Befolkningens bevaegelser 1957. Befolkningens bevaegelser 1958. ... Befolkningens be-
vaegelser 2003. Befolkningens bevaegelser 2004. Vital Statistics 2005. Vital Statistics 2006.

Examples

```
data( B.dk )
str( B.dk )
attach( B.dk )
# Plot the no of births and the M/F-ratio
par( las=1, mar=c(4,4,2,4) )
matplot( year+(month-0.5)/12,
         cbind( m, f ),
         bty="n", col=c("blue","red"), lty=1, lwd=1, type="l",
         ylim=c(0,5000),
         xlab="Date of birth", ylab="" )
usr <- par()$usr
mtext( "Monthly no. births in Denmark", side=3, adj=0, at=usr[1], line=1/1.6 )
text( usr[1:2] %*% cbind(c(19,1),c(19,1))/20,
    usr[3:4] %*% cbind(c(1,19),c(2,18))/20, c("Boys","Girls"), col=c("blue","red"), adj=0 )
lines( year+(month-0.5)/12, (m/(m+f)-0.5)*30000, lwd=1 )
axis( side=4, at=(seq(0.505,0.525,0.005)-0.5)*30000, labels=c("","","","",""), tcl=-0.3 )
axis( side=4, at=(50:53/100-0.5)*30000, labels=50:53, tcl=-0.5 )
axis( side=4, at=(0.54-0.5)*30000, labels="% boys", tick=FALSE, mgp=c(3,0.1,0) )
abline( v=1920, col=gray(0.8) )
```

---

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   *A 1:1 subset of the endometrial cancer case-control study*

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   *Births in a London Hospital*

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)
```

---

boxes.MS	*Draw boxes and arrows for illustration of multistate models.*

Description

Boxes can be drawn with text (tbox) or a cross (dbox), and arrows pointing between the boxes (boxarr) can be drawn automatically not overlapping the boxes. The boxes method for Lexis objects generates displays of states with person-years and transitions with events or rates.

Usage

```
tbox( txt, x, y, wd, ht,
      font=2, lwd=2,
      col.txt=par("fg"),
      col.border=par("fg"),
      col.bg="transparent" )
dbox( x, y, wd, ht=wd,
      font=2, lwd=2, cwd=5,
      col.cross=par("fg"),
      col.border=par("fg"),
      col.bg="transparent"  )
boxarr( b1, b2, offset=FALSE, pos=0.45, ... )
## S3 method for class 'Lexis'
boxes( obj,
```

```
                               boxpos = FALSE,
                                wmult = 1.15,
                                hmult = 1.15,
                                  cex = 1.45,
                                 show  = inherits( obj, "Lexis" ),
                               show.Y = show,
                              scale.Y = 1,
                             digits.Y = 1,
                              show.BE = FALSE,
                               BE.sep = c("","","             ",""),
                               show.D = show,
                              scale.D = FALSE,
                             digits.D = as.numeric(as.logical(scale.D)),
                               show.R = is.numeric(scale.R),
                              scale.R = 1,
                             digits.R = as.numeric(as.logical(scale.R)),
                               DR.sep = if( show.D ) c("\n(",")") else c("",""),
                                eq.wd = TRUE,
                                eq.ht = TRUE,
                                   wd,
                                   ht,
                               subset = NULL,
                              exclude = NULL,
                                 font = 2,
                                  lwd = 2,
                              col.txt = par("fg"),
                           col.border = col.txt,
                               col.bg = "transparent",
                              col.arr = par("fg"),
                              lwd.arr = 2,
                             font.arr = 2,
                              pos.arr = 0.45,
                              txt.arr = NULL,
                          col.txt.arr = col.arr,
                           offset.arr = 2,
                                      ... )
## S3 method for class 'matrix'
boxes( obj, ... )
## S3 method for class 'MS'
boxes( obj, sub.st, sub.tr, cex=1.5, ... )
   fillarr( x1, y1, x2, y2, gap=2, fr=0.8,
            angle=17, lwd=2, length=par("pin")[1]/30, ... )
```

Arguments

txt	Text to be placed inside the box.
x	x-coordinate of center of box.

y	y-coordinate of center of box.
wd	width of boxes in percentage of the plot width.
ht	height of boxes in percentage of the plot height.
font	Font for the text. Defaults to 2 (=bold).
lwd	Line width of the boxborders.
col.txt	Color for the text in boxes.
col.border	Color of the box border.
col.bg	Background color for the interior of the box.
...	Arguments to be passed on to the call of other functions.
cwd	Width of the lines in the cross.
col.cross	Color of the cross.
b1	Coordinates of the "from" box. A vector with 4 components, x, y, w, h.
b2	Coordinates of the "to" box; like b1.
offset	Logical. Should the arrow be offset a bit to the left.
pos	Numerical between 0 and 1, determines the position of the point on the arrow which is returned.
obj	A [Lexis](#) object or a transition matrix; that is a square matrix indexed by state in both dimensions, and the (i, j)th entry different from NA if a transition i to j can occur. If show.D=TRUE, the arrows between states are annotated by these numbers. If show.Y=TRUE, the boxes representing states are annotated by the numbers in the diagonal of obj.
	For boxes.matrix obj is a matrix and for boxes.MS, obj is an MS.boxes object (see below).
boxpos	If TRUE the boxes are positioned equidistantly on a circle, if FALSE (the default) you are queried to click on the screen for the positions. This argument can also be a named list with elements x and y, both numerical vectors, giving the centers of the boxes.
wmult	Multiplier for the width of the box relative to the width of the text in the box.
hmult	Multiplier for the height of the box relative to the height of the text in the box.
cex	Character expansion for text in the box.
show	Should person-years and transitions be put in the plot. Ignored if obj is not a Lexis object.
show.Y	If logical: Should person-years be put in the boxes. If numeric: Numbers to put in boxes.
scale.Y	What scale should be used for annotation of person-years.
digits.Y	How many digits after the decimal point should be used for the person-years.
show.BE	Logical. Should number of persons beginning resp. ending follow up in each state be shown?
BE.sep	Character vector of length 4, used for annotation of the number of persons beginning and ending in each state: 1st elemet precedes no. beginning, 2nd trails it, 3rd precedes the no. ending (defaults to 8 spaces), and the 4th trails the no. ending.

show.D	Should no. transitions be put alongside the arrows. Ignored if `obj` is not a `Lexis` object.
scale.D	Synonumous with `scale.R`, retained for compatability.
digits.D	Synonumous with `digits.R`, retained for compatability.
show.R	Should the transition rates be shown on the arrows?
scale.R	If this a scalar, rates instead of no. transitions are printed at the arrows, scaled by `scale.R`.
digits.R	How many digits after the decimal point should be used for the rates.
DR.sep	Character vector of length 2. If rates are shown, the first element is inserted before and the second after the rate.
eq.wd	Should boxes all have the same width?
eq.ht	Should boxes all have the same height?
subset	Draw only boxes and arrows for a subset of the states. Can be given either as a numerical vector or character vector state names.
exclude	Exclude states from the plot. The complementary of `subset`. Ignored if `subset` is given.
col.arr	Color of the arrows between boxes. A vector of character strings, the arrows are referred to as the row-wise sequence of non-NA elements of the transition matrix. Thus the first ones refer to the transitions out of state 1, in order of states.
lwd.arr	Line withs of the arrows.
font.arr	Font of the text annotation the arrows.
pos.arr	Numerical between 0 and 1, determines the position on the arrows where the text is written.
txt.arr	Text put on the arrows.
col.txt.arr	Colors for text on the arrows.
offset.arr	The amount offset between arrows representing two-way transitions, that is where there are arrows both ways between two boxes.
sub.st	Subset of the states to be drawn.
sub.tr	Subset of the transitions to be drawn.
x1	x-coordinate of the starting point.
y1	y-coordinate of the starting point.
x2	x-coordinate of the end point.
y2	y-coordinate of the end point.
gap	Length of the gap between the box and the ends of the arrows.
fr	Length of the arrow as the fraction of the distance between the boxes. Ignored unless given explicitly, in which case any value given for `gap` is ignored.
angle	What angle should the arrow-head have?
length	Length of the arrow head in inches. Defaults to 1/30 of the physical width of the plot.

Details

These functions are designed to facilitate the drawing of multistate models, mainly by automatic calculation of the arrows between boxes.

tbox draws a box with centered text, and returns a vector of location, height and width of the box. This is used when drawing arrows between boxes. dbox draws a box with a cross, symbolizing a death state. boxarr draws an arrow between two boxes, making sure it does not intersect the boxes. Only straight lines are drawn.

boxes.Lexis takes as input a Lexis object sets up an empty plot area (with axes 0 to 100 in both directions) and if boxpos=FALSE (the default) prompts you to click on the locations for the state boxes, and then draws arrows implied by the actual transitions in the Lexis object. The default is to annotate the transitions with the number of transitions.

A transition matrix can also be supplied, in which case the row/column names are used as state names, diagnonal elements taken as person-years, and off-diagnonal elements as number of transitions. This also works for boxes.matrix.

Optionally returns the R-code reproducing the plot in a file, which can be useful if you want to produce exactly the same plot with differing arrow colors etc.

boxarr draws an arrow between two boxes, on the line connecting the two box centers. The offset argument is used to offset the arrow a bit to the left (as seen in the direction of the arrow) on order to accommodate arrows both ways between boxes. boxarr returns a named list with elements x, y and d, where the two former give the location of a point on the arrow used for printing (see argument pos) and the latter is a unit vector in the direction of the arrow, which is used by boxes.Lexis to position the annotation of arrows with the number of transitions.

boxes.MS re-draws what boxes.Lexis has done based on the object of class MS produced by boxes.Lexis. The point being that the MS object is easily modifiable, and thus it is a machinery to make variations of the plot with different color annotations etc.

fill.arr is just a utility drawing nicer arrows than the default arrows command, basically by using filled arrow-heads; called by boxarr.

Value

The functions tbox and dbox return the location and dimension of the boxes, c(x,y,w,h), which are designed to be used as input to the boxarr function.

The boxarr function returns the coordinates (as a named list with names x and y) of a point on the arrow, designated to be used for annotation of the arrow.

The function boxes.Lexis returns an MS object, a list with five elements: 1) Boxes - a dataframe with one row per box and columns xx, yy, wd, ht, font, lwd, col.txt, col.border and col.bg, 2) an object State.names with names of states (possibly an expression, hence not possible to include as a column in Boxes), 3) a matrix Tmat, the transition matrix, 4) a data frame, Arrows with one row per transition and columns: lwd.arr, col.arr, pos.arr, col.txt.arr, font.arr and offset.arr and 5) an object Arrowtext with names of states (possibly an expression, hence not possible to include as a column in Arrows)

An MS object is used as input to boxes.MS, the primary use is to modify selected entries in the MS object first, e.g. colors, or supply subsetting arguments in order to produce displays that have the same structure, but with different colors etc.

Author(s)

Bendix Carstensen

See Also

tmat.Lexis

Examples

```
par( mar=c(0,0,0,0), cex=1.5 )
plot( NA,
      bty="n",
      xlim=0:1*100, ylim=0:1*100, xaxt="n", yaxt="n", xlab="", ylab="" )
bw  <- tbox( "Well"    , 10, 60, 22, 10, col.txt="blue" )
bo  <- tbox( "other Ca", 45, 80, 22, 10, col.txt="gray" )
bc  <- tbox( "Ca"      , 45, 60, 22, 10, col.txt="red" )
bd  <- tbox( "DM"      , 45, 40, 22, 10, col.txt="blue" )
bcd <- tbox( "Ca + DM" , 80, 60, 22, 10, col.txt="gray" )
bdc <- tbox( "DM + Ca" , 80, 40, 22, 10, col.txt="red" )
      boxarr( bw, bo , col=gray(0.7), lwd=3 )
# Note the argument adj= can takes values outside (0,1)
text( boxarr( bw, bc , col="blue", lwd=3 ),
      expression( lambda[Well] ), col="blue", adj=c(1,-0.2), cex=0.8 )
      boxarr( bw, bd , col=gray(0.7) , lwd=3 )
      boxarr( bc, bcd, col=gray(0.7) , lwd=3 )
text( boxarr( bd, bdc, col="blue", lwd=3 ),
      expression( lambda[DM] ), col="blue", adj=c(1.1,-0.2), cex=0.8 )

# Set up a transition matrix allowing recovery
tm <- rbind( c(NA,1,1), c(1,NA,1), c(NA,NA,NA) )
rownames(tm) <- colnames(tm) <- c("Cancer","Recurrence","Dead")
tm
boxes.matrix( tm, boxpos=TRUE )

# Illustrate texting of arrows
boxes.Lexis( tm, boxpos=TRUE, txt.arr=c("en","to","tre","fire") )
zz <- boxes( tm, boxpos=TRUE, txt.arr=c(expression(lambda[C]),
                                        expression(mu[C]),
                                        "recovery",
                                        expression(mu[R]) ) )

# Change color of a box
zz$Boxes[3,c("col.bg","col.border")] <- "green"
boxes( zz )

# Set up a Lexis object
data(DMlate)
str(DMlate)
dml <- Lexis( entry=list(Per=dodm, Age=dodm-dobth, DMdur=0 ),
               exit=list(Per=dox),
        exit.status=factor(!is.na(dodth),labels=c("DM","Dead")),
               data=DMlate[1:1000,] )
```

```
# Cut follow-up at Insulin
dmi <- cutLexis( dml, cut=dml$doins, new.state="Ins", pre="DM" )
summary( dmi )
boxes( dmi, boxpos=TRUE )
boxes( dmi, boxpos=TRUE, show.BE=TRUE )
boxes( dmi, boxpos=TRUE, show.BE="nz" )
boxes( dmi, boxpos=TRUE, show.BE="nz", BE.sep=c("In:","      Out:","") )

# Set up a bogus recovery date just to illustrate two-way transitions
dmi$dorec <- dmi$doins + runif(nrow(dmi),0.5,10)
dmi$dorec[dmi$dorec>dmi$dox] <- NA
dmR <- cutLexis( dmi, cut=dmi$dorec, new.state="DM", pre="Ins" )
summary( dmR )
boxes( dmR, boxpos=TRUE )
boxes( dmR, boxpos=TRUE, show.D=FALSE )
boxes( dmR, boxpos=TRUE, show.D=FALSE, show.Y=FALSE )
boxes( dmR, boxpos=TRUE, scale.R=1000 )
MSobj <- boxes( dmR, boxpos=TRUE, scale.R=1000, show.D=FALSE )
MSobj <- boxes( dmR, boxpos=TRUE, scale.R=1000, DR.sep=c(" (",")") )
class( MSobj )
boxes( MSobj )
MSobj$Boxes[1,c("col.txt","col.border")] <- "red"
MSobj$Arrows[1:2,"col.arr"] <- "red"
boxes( MSobj )
```

---

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 and 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)
```

---

cal.yr	*Functions to convert character, factor and various date objects into a number, and vice versa.*

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="%Y-%m-%d", wh=NULL )
  ## S3 method for class 'cal.yr'
as.Date( x, ... )
```

Arguments

x   A factor or character vector, representing a date in format format, or an object of class Date, POSIXlt, POSIXct, date, dates or chron (the latter two requires the chron package). If x is a data frame, all variables in the data-frame which are of one the classes mentioned are converted to class cal.yr. See arguemt wh, though.

format   Format of the date values if x is factor or character. If this argument is supplied and x is a datafame, all character variables are converted to class cal.yr. Factors in the dataframe will be ignored.

wh	Indices of the variables to convert if x is a data frame. Can be either a numerical or character vector.
...	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"). If x is a data frame a dataframe with some of the columns converted to class "cal.yr" is returned.

as.Date.cal.yr returns a [Date](#) object.

Author(s)

Bendix Carstensen, Steno Diabetes Center \& Dept. of Biostatistics, University of Copenhagen, <bxc@steno.dk>, http://BendixCarstensen.com

See Also

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

Examples

```
# Character vector of dates:
birth <- c("14/07/1852","01/04/1954","10/06/1987","16/05/1990",
          "12/11/1980","01/01/1997","01/01/1998","01/01/1999")
# Proper conversion to class "Date":
birth.dat <- as.Date( birth, format="%d/%m/%Y" )
# Converson 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=FALSE )
```

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, echoes 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)
```

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,
     sample = FALSE )
```

Arguments

obj	A model object (of class `lm`, `glm`, `coxph`, `survreg`, `lme`,`mer`,`nls`,`gnlm`, `MIresult` or `polr`).
ctr.mat	Contrast matrix defining the parameter functions from the parameters of the model.
subset	Subset of the parameters of the model to which `ctr.mat` should be applied.
intl	Interval length for the cumulation. Either a constant or a numerical vector of length `nrow(ctr.mat)`.
alpha	Significance level used when computing confidence limits.
Exp	Should the parameter function be exponentiated before it is cumulated?
sample	Should a sample of the original parameters be used to compute a cumulative rate?

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 from the model, the 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. If `sample` is TRUE, a sampled vector is reurned, if `sample` is numeric a matrix with `sample` columns is returned, each column a cumulative rate based on a random sample from the distribution of the parameter estimates.

Author(s)

Bendix Carstensen, <http://BendixCarstensen.com>

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 ) ~ 1, data=lung ) )

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

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

---

ci.lin | *Compute linear functions of parameters with standard erros and con-fidence limits*

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,
     subint = NULL,
      diffs = FALSE,
       fnam = !diffs,
       vcov = FALSE,
      alpha = 0.05,
         df = Inf,
        Exp = FALSE,
     sample = FALSE )
ci.exp( ..., Exp = TRUE )
Wald( obj, H0=0, ... )
ci.mat( alpha = 0.05, df = Inf )
ci.pred( obj, newdata,
         Exp = TRUE,
       alpha = 0.05,
          df = Inf )
```

Arguments

obj
: A model object (of class `lm`, `glm`, `coxph`, `survreg`, `clogistic`, `cch`, `lme`, `mer`, `lmerMod`, `gls`, `nls`, `gnlm`, `MIresult`, `mipo`, `polr`, or `rq`).

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 `grep`) to find the subset. Repeat parameters may result from using a character vector. This is considered a facility.

subint
: SUBset selection like for `subset`, except that elements of a character vector given as argument will be used to select a number of subsets of parameters and only the INTersection of these is returned.

diffs
: If TRUE, all differences between parameters in the subset are computed. `ctr.mat` is ignored. If `obj` inherits from `lm`, and `subset` is given as a string `subset` is used to search among the factors in the model and differences of all factor levels

	for the first match are shown. If subset 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 diffs==T. 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, Exp is ignored. See details.
alpha	Significance level for the confidence intervals.
df	Integer. Number of degrees of freedom in the t-distribution used to compute the quantiles used to construct the confidence intervals.
Exp	If TRUE columns 5:6 are replaced with exp(columns 1,5,6).
sample	Logical or numerical. If TRUE or numerical a sample of size as.numeric(sample) of the linear parameter function as defined by subset and ctr.mat is returned.
H0	The null values for the selected/transformed parameters to be tested by a Wald test. Must have the same length as the selected parameter vector.
...	Parameters passed on to ci.lin.
newdata	Data frame of covariates where prediction is made.

Value

ci.lin returns a matrix with number of rows and row names 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 Exp==TRUE the confidence intervals for the parameters are replaced with three columns: exp(estimate,c.i.).

ci.exp returns only the exponentiated parameter estimates with confidence intervals. It is merely a wrapper for ci.lin, fishing out the last 3 columns from ci.lin(...,Exp=TRUE). If you just want the estimates and confidnece limits, but not exponentiated, use ci.exp(...,Exp=FALSE).

Wald computes a Wald test for a subset of (possibly linearly transformed) parameters being equal to the null value. The selection of the subset of parameters is the same as for ci.lin. Using the ctr.mat argument makes it possible to do a Wald test for equality of parameters. Wald returns a named numerical vector of length 3, with names Chisq, d.f. and P.

ci.mat returns a 2 by 3 matrix with rows c(1,0,0) and c(0,-1,1)*1.96, devised to post-multiply to a p by 2 matrix with columns of estimates and standard errors, so as to produce a p by 3 matrix of estimates and confidence limits. Used internally in ci.lin and ci.cum. The 1.96 is replaced by the appropriate quantile from the normal or t-distribution when arguments alpha and/or df are given.

ci.pred returns a 3-column matrix with estimates and upper and lower confidence intervals as columns. This is just a convenient wrapper for predict(obj,se.fit=TRUE) which returns a rather unhandy structure. The prediction is made in the link scale, and by default transformed by the exponential, since the most common use for this is for multiplicative Poisson models.

Author(s)

Bendix Carstensen, BendixCarstensen.com & Michael Hills http://www.mhills.pwp.blueyonder.co.uk/

See Also

See also `ci.cum`

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 )

# A Wald test of wheter the g-parameters are 0
Wald( mm, subset="g" )
# Wald test of whether the three first f-parameters are equal:
( CM <- rbind( c(1,-1,0,0), c(1,0,-1,0)) )
Wald( mm, subset="f", ctr.mat=CM )
# or alternatively
( CM <- rbind( c(1,-1,0,0), c(0,1,-1,0)) )
Wald( mm, subset="f", ctr.mat=CM )
```

---

ci.pd	*Compute confidence limits for a difference of two independent proportions.*

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://BendixCarstensen.com>

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 )
```

---

clogistic	*Conditional logistic regression*

Description

Estimates a logistic regression model by maximizing the conditional likelihood. The conditional likelihood calculations are exact, and scale efficiently to strata with large numbers of cases.

Usage

```
clogistic(formula, strata, data, subset, na.action, init,
model = TRUE, x = FALSE, y = TRUE, contrasts = NULL,
iter.max=20, eps=1e-6, toler.chol = sqrt(.Machine$double.eps))
```

Arguments

formula	Model formula
strata	Factor describing membership of strata for conditioning
data	data frame containing the variables in the formula and strata arguments
subset	subset of records to use
na.action	missing value handling
init	initial values
model	a logical value indicating whether *model frame* should be included as a component of the returned value
x,y	logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.
contrasts	an optional list. See the `contrasts.arg` of `model.matrix.default`
iter.max	maximum number of iterations
eps	Convergence tolerance. Iteration continues until the relative change in the conditional log likelihood is less than eps. Must be positive.
toler.chol	Tolerance used for detection of a singularity during a Cholesky decomposition of the variance matrix. This is used to detect redundant predictor variables. Must be less than eps.

Value

An object of class `"clogistic"`. This is a list containing the following components:

coefficients	the estimates of the log-odds ratio parameters. If the model is over-determined there will be missing values in the vector corresponding to the redundant columns in the model matrix.
var	the variance matrix of the coefficients. Rows and columns corresponding to any missing coefficients are set to zero.
loglik	a vector of length 2 containing the log-likelihood with the initial values and with the final values of the coefficients.
iter	number of iterations used.
n	number of observations used. Observations may be dropped either because they are missing, or because they belong to a homogeneous stratum. For more details on which observations were used, see `informative` below.
informative	if model=TRUE, a logical vector of length equal to the number of rows in the model frame. This indicates whether an observation is informative. Strata that are homogeneous with respect to either the outcome variable or the predictor variables are uninformative, in the sense that they can be removed without modifying the estimates or standard errors. If model=FALSE, this is NULL.

The output will also contain the following, for documentation see the `glm` object: `terms`, `formula`, `call`, `contrasts`, `xlevels`, and, optionally, `x`, `y`, and/or `frame`.

Author(s)

Martyn Plummer

See Also

[glm](glm)

Examples

```
data(bdendo)
clogistic(d ~ cest + dur, strata=set, data=bdendo)
```

---

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 ia 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)
```

---

crr.Lexis   *Fit a competing risks regression model (Fine-Gray model) using a Lexis object)*

Description

Fits a competing risks regression model using a [Lexis](#) object assuming that every person enters at time 0 and exits at time lex.dur. Thus is only meaningful for Lexis objects with one record per person, (so far).

Usage

```
crr.Lexis( obj, mod, quiet=FALSE, ...)
```

Arguments

obj   A Lexis object; variables in mod are taken from this.

mod   Formula, with the l.h.s. a character constant equal to a level of obj$lex.Xst, and the r.h.s. a model formula interpreted in obj.

quiet   Logical indicating whether a brief summary should be printed.

...   Further arguments passed on to [crr](#).

Details

This function is a simple wrapper for crr, allowing a formula-specification of the model (which allows specifications of covariates on the fly), and utilizing the structure of Lexis objects to simplify specification of the outcome. Prints a summary of the levels used as event, competing events and censoring.

By the structure of the [Lexis](#) object it is not necessary to indicate what the censoring code or competing events are, that is automatically derived from the Lexis object.

Currently only one state is allowed as l.h.s. (response) in mod.

Value

A [crr](#) object (which is a list), with two extra elements in the list, model.Lexis - the model formula supplied, and transitions - a table of transitions and censorings showing which transition was analysed and which were taken as competing events.

Author(s)

Bendix Carstensen, BendixCarstensen.com

See Also

[crr](#), [Lexis](#)

Examples

```
# Thorotrats patients, different histological types of liver cancer
# Load thorotrast data, and restrict to exposed
data(thoro)
tht <- thoro[thoro$contrast==1,]
# Define exitdate as the date of livercancer
tht$dox <- pmin( tht$liverdat, tht$exitdat, na.rm=TRUE )
tht <- subset( tht, dox > injecdat )
# Convert to calendar years in dates
tht <- cal.yr( tht )

# Set up a Lexis object with three subtypes of liver cancer and death
tht.L <- Lexis( entry = list( per = injecdat,
                              tfi = 0 ),
                 exit = list( per = dox ),
          exit.status = factor( 1*hepcc+2*chola+3*hmang+
                                 4*(hepcc+chola+hmang==0 & exitstat==1),
                                 labels=c("No cancer","hepcc","chola","hmang","Dead") ),
                 data = tht )
summary( tht.L )

# Show the transitions
boxes( tht.L, boxpos=list(x=c(20,rep(80,3),30),
                          y=c(60,90,60,30,10) ),
              show.BE=TRUE, scale.R=1000 )

# Fit a model for the Hepatocellular Carcinoma as outcome
# - note that you can create a variable on the fly:
library( cmprsk )
hepcc <- crr.Lexis( tht.L, "hepcc" ~ volume + I(injecdat-1940) )
hepcc$model.Lexis
hepcc$transitions

# Models for the three other outcomes:
chola <- crr.Lexis( tht.L, "chola" ~ volume + I(injecdat-1940) )
hmang <- crr.Lexis( tht.L, "hmang" ~ volume + I(injecdat-1940) )
dead  <- crr.Lexis( tht.L, "Dead"  ~ volume + I(injecdat-1940) )

# Compare the effects
# NOTE: This is not necessarily a joint model for all transitions.
zz <- rbind( ci.exp(hepcc),
             ci.exp(chola),
             ci.exp(hmang),
             ci.exp(dead) )
zz <- cbind( zz[c(1,3,5,7)  ,],
             zz[c(1,3,5,7)+1,] )
rownames( zz ) <- c("hepcc","chola","hmang","dead")
colnames( zz )[c(1,4)] <- rownames( ci.exp(chola) )
round( zz, 3 )
```

---

cutLexis   *Cut follow-up at a specified date for each person.*

Description

Follow-up intervals in a Lexis object are divided into two sub-intervals: one before and one after an
intermediate event. The intermediate event may denote a change of state, in which case the entry
and exit status variables in the split Lexis object are modified.

Usage

```
cutLexis( data, cut, timescale = 1,
                     new.state = nlevels(data$lex.Cst)+1,
                     new.scale = FALSE,
                     split.states = FALSE,
                     progressive = FALSE,
                     precursor.states = NULL,
                     count = FALSE)
countLexis( data, cut, timescale = 1 )
```

Arguments

data	A Lexis object.
cut	A numeric vector with the times of the intermediate event. If a time is missing (NA) then the event is assumed to occur at time Inf. cut can also be a dataframe, see details.
timescale	The timescale that cut refers to. Numeric or character.
new.state	The state to which a transition occur at time cut. It may be a single value, which is then applied to all rows of data, or a vector with a separate value for each row
new.scale	Name of the timescale defined as "time since entry to new.state". If TRUE a name for the new scale is constructed. See details.
split.states	Should states that are not precursor states be split according to whether the intermediate event has occurred.
progressive	a logical flag that determines the changes to exit status. See details.
precursor.states	an optional vector of states to be considered as "less severe" than new.state. See Details below
count	logical indicating whether the countLexis options should be used. Specifying count=TRUE amounts to calling countLexis, in which case the arguments new.state, progressive and precursor.states will be ignored.

Details

The cutLexis function allows a number of different ways of specifying the cutpoints and of modifying the status variable.

If the cut argument is a dataframe it must have columns lex.id, cut and new.state. The values of lex.id must be unique. In this case it is assumed that each row represents a cutpoint (on the timescale indicated in the argument timescale). This cutpoint will be applied to all records in data with the corresponding lex.id. This makes it possible to apply cutLexis to a split Lexis object.

If a new.state argument is supplied, the status variable is only modified at the time of the cut point. However, it is often useful to modify the status variable after the cutpoint when an important event occurs. There are three distinct ways of doing this.

If the progressive=TRUE argument is given, then a "progressive" model is assumed, in which the status can either remain the same or increase during follow-up, but never decrease. This assumes that the state variables lex.Cst and lex.Xst are either numeric or ordered factors. In this case, if new.state=X, then any exit status with a value less than X is replaced with X. The Lexis object must already be progressive, so that there are no rows for which the exit status is less than the entry status. If lex.Cst and lex.Xst are factors they must be ordered factors if progressive=TRUE is given.

As an alternative to the progressive argument, an explicit vector of precursor states, that are considered less severe than the new state, may be given. If new.state=X and precursor.states=c(Y,Z) then any exit status of Y or Z in the second interval is replaced with X and all other values for the exit status are retained.

The countLexis function is a variant of cutLexis when the cutpoint marks a recurrent event, and the status variable is used to count the number of events that have occurred. Times given in cut represent times of new events. Splitting with countLexis increases the status variable by 1. If the current status is X and the exit status is Y before cutting, then after cutting the entry status is X, X+1 for the first and second intervals, respectively, and the exit status is X+1, Y+1 respectively. Moreover the values of the status is increased by 1 for all intervals for all intervals after the cut for the person in question. Hence, a call to countLexis is needed for as many times as the person with most events. But also it is immaterial in what order the cutpoints are entered.

Value

A Lexis object, for which each follow-up interval containing the cutpoint is split in two: one before and one after the cutpoint. An extra time-scale is added; the time since the event at cut. This is NA for any follow-up prior to the intermediate event.

Note

The cutLexis function superficially resembles the splitLexis function. However, the splitLexis function splits on a vector of common cut-points for all rows of the Lexis object, whereas the cutLexis function splits on a single time point, which may be distinct for each row, modifies the status variables, adds a new timescale and updates the attribute "time.since". This attribute is a character vector of the same length as the "time.scales" attribute, whose value is '""' if the corresponding timescale is defined for any piece of follow-up, and if the corresponding time scale is defined by say cutLexis(obj,new.state="A",new.scale=TRUE), it has the value "A".

Author(s)

Bendix Carstensen, Steno Diabetes Center, <bxc@steno.dk>, Martyn Plummer, IARC, <plummer@iarc.fr>

See Also

splitLexis, Lexis, summary.Lexis, boxes.Lexis

Examples

```
# A small artificial example
xx <- Lexis( entry=list(age=c(17,24,33,29),per=c(1920,1933,1930,1929)),
              duration=c(23,57,12,15), exit.status=c(1,2,1,2) )
xx
cut <- c(33,47,29,50)
cutLexis(xx, cut, new.state=3, precursor=1)
cutLexis(xx, cut, new.state=3, precursor=2)
cutLexis(xx, cut, new.state=3, precursor=1:2)
# The same as the last example
cutLexis(xx, cut, new.state=3)

# The same example with a factor status variable
yy <- Lexis(entry = list(age=c(17,24,33,29),per=c(1920,1933,1930,1929)),
            duration = c(23,57,12,15),
            entry.status = factor(rep("alpha",4),
            levels=c("alpha","beta","gamma")),
            exit.status = factor(c("alpha","beta","alpha","beta"),
            levels=c("alpha","beta","gamma")))

cutLexis(yy,c(33,47,29,50),precursor="alpha",new.state="gamma")
cutLexis(yy,c(33,47,29,50),precursor=c("alpha","beta"),new.state="aleph")

## Using a dataframe as cut argument
rl <- data.frame( lex.id=1:3, cut=c(19,53,26), timescale="age", new.state=3 )
rl
cutLexis( xx, rl )
cutLexis( xx, rl, precursor=1 )
cutLexis( xx, rl, precursor=0:2 )

## It is immaterial in what order splitting and cutting is done
xs <- splitLexis( xx, breaks=seq(0,100,10), time.scale="age" )
xs
xsC <- cutLexis(xs, rl, precursor=0 )

xC <- cutLexis( xx, rl, pre=0 )
xC
xCs <- splitLexis( xC, breaks=seq(0,100,10), time.scale="age" )
xCs
str(xCs)
```

---

detrend	*Projection of a model matrix on to the orthogonal complement of a trend.*

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://BendixCarstensen.com, 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)
```

DMlate   *The Danish National Diabetes Register.*

Description

These two datasets each contain a random sample of 10,000 persons from the Danish National Diabetes Register. `DMrand` is a random sample from the register, whereas `DMlate` is a random sample among those with date of diagnosis after 1.1.1995. All dates are radomly jittered by adding a U(-7,7) (days).

Usage

```
data(DMrand)
        data(DMlate)
```

Format

A data frame with 10000 observations on the following 7 variables.

sex Sex, a factor with levels `M F`

dobth Date of birth

dodm Date of inclusion in the register

dodth Date of death

dooad Date of 2nd prescription of OAD

doins Date of 2nd insulin prescription

dox Date of exit from follow-up.

Details

All dates are given in fractions of years, so 1997.00 corresponds to 1 January 1997 and 1997.997 to 31 December 1997.

Source

Danish National Board of Health.

References

B Carstensen, JK Kristensen, P Ottosen and K Borch-Johnsen: The Danish National Diabetes Register: Trends in incidence, prevalence and mortality, Diabetologia, 51, pp 2187–2196, 2008.

In partucular see the appendix at the end of the paper.

Examples

```
data(DMlate)
str(DMlate)
dml <- Lexis( entry=list(Per=dodm, Age=dodm-dobth, DMdur=0 ),
               exit=list(Per=dox),
        exit.status=factor(!is.na(dodth),labels=c("DM","Dead")),
               data=DMlate )

# Cut the follow-up at insulin start, and introduce a new timescale,
# and split non-precursor states
system.time(
dmi <- cutLexis( dml, cut = dml$doins,
                      pre = "DM",
                new.state = "Ins",
                new.scale = "t.Ins",
             split.states = TRUE ) )
summary( dmi )
```

---

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,
                  eff = NULL,
       alpha = 0.05,
         base = 1,
        digits = 3,
          data = NULL )
```

Arguments

response   The response variable - must be numeric or logical. If logical, TRUE is considered the outcome.

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. This must be numeric.

exposure	The exposure variable can be numeric or a factor
strata	The `strata` stratifying variable - must be a factor
control	The `control` variable(s) (confounders) - these are passed as a list if there are more than one.
weights	Frequency weights for binary response only
eff	How should effects be measured. If `response` is binomial, the default is "OR" (odds-ratio) with "RR" (relative risk) as an option. If `response` is failure, the default is "RR" (rate-ratio) with "RD" (rate difference) as an option.
base	Baseline for the effects of a categorical exposure, either a number or a name of the level. Defaults to 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/relatiev risks for a binary response, and rate ratios/rate differences 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)
effx(lowbw,type="binary",exposure=hyp,eff="RR",data=births)
```

---

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)
```

---

ewrates   *Rates of lung and nasal cancer mortality, and total mortality.*

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)
```

---

expand.data   *Function to expand data for regression analysis of interval censored data.*

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 fromula, 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	Datafrem 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>

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

---

fit.add	*Fit an addive excess risk model to interval censored data.*

Description

Utility function.

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

Usage

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

Arguments

`y`	Binary vector of outcomes
`rates.frame`	Dataframe expanded from the original data by `expand.data`, cooresponding 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.

Value

A list with one component:

`rates`	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 logaritmic 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

y	Binary vector of outcomes
rates.frame	Dataframe expanded from the original data by [expand.data](#)
start	Starting values for the rate parameters. If not supplied, then starting values are generated.

Value

A [glm](#) object, with binomial error and logaritmic link.

Author(s)

Martyn Plummer, <plummer@iarc.fr>

See Also

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

---

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](#), cooresponding 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 )
```

---

float	*Calculate floated variances*

Description

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

Usage

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

Arguments

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

Details

The `float()` function 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, but the idea is widely applicable.

Treatment contrasts are not orthogonal. Consequently, the variances of treatment contrast estimates may be inflated by a poor choice of reference level, and the correlations between them may also be high. The `float()` function associates each level of the factor with a "floating" variance (or quasi-variance), including the reference level. Floating variances are not real variances, but they can be used to calculate the variance error of 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 contrast estimates. This model can be fitted by minimizing the Kullback-Leibler information divergence between the true distribution of the parameter estimates and the simplified 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, the `float()` function 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-based 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 `floated`. This is a list with the following components

`coef`	A vector of coefficients. These are the same as the treatment contrasts but the reference level is present with coefficient 0.
`var`	A vector of floating (or quasi-) variances
`limits`	The 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

---

foreign.Lexis	*Create a data structures suitable for use with packages mstate, etm.*

Description

The mstate package requires input in the form of a stacked dataset with specific variable names. This is provided by msdata.Lexis. The resulting dataframe contains the same information as the result of a call to [stack.Lexis](#).

The etm package requires input (almost) in the form of a Lexis object, but with specific column names etc. This is provided by etm.Lexis.

Usage

```
msdata(obj, ...)
## S3 method for class 'Lexis'
msdata(obj,
                time.scale = timeScales(obj)[1],
                     ... )
## S3 method for class 'Lexis'
etm( obj,
             time.scale = timeScales(obj)[1],
              cens.name = "cens",
                     s = 0,
                     t = "last",
             covariance = TRUE,
               delta.na = TRUE,
                     ... )
```

Arguments

obj	A [Lexis](#) object.
time.scale	Name or number of timescale in the Lexis object.
cens.name	Name of the code for censoring used by etm. It is only necessary to change this if one of the states in the Lexis object has name "cens".
s	Passed on to etm.
t	Passed on to etm.
covariance	Passed on to etm.
delta.na	Passed on to etm.
...	Further arguments.

Value

msdata.Lexis returns a dataframe with the Lexis specific variables stripped, and with the following added: id, Tstart, Tstop, from, to, trans, status, which are used in the mstate package.

etm.Lexis transforms the Lexis object into a dataframe suitable for analysis by the function etm from the etm package, and actually calls this function, so returns an object of class etm.

Author(s)

Bendix Carstensen, <bxc@steno.dk>, <http://BendixCarstensen.com>

See Also

stack.Lexis, msdata, etm

Examples

```
data(DMlate)
str(DMlate)
dml <- Lexis( entry = list(Per=dodm,Age=dodm-dobth,DMdur=0),
               exit = list(Per=dox),
        exit.status = factor(!is.na(dodth),labels=c("DM","Dead")),
               data = DMlate[1:1000,] )
dmi <- cutLexis( dml, cut=dml$doins, new.state="Ins", pre="DM" )
summary( dmi )

# Use the interface to the mstate package
if( require(mstate) )
{
ms.dmi <- msdata.Lexis( dmi )
# Check that all the transitions and person-years got across.
with( ms.dmi, rbind( table(status,trans),
                     tapply(Tstop-Tstart,trans,sum) ) )
}

# Use the etm package directly with a Lexis object
if( require(etm) )
{
dmi <- subset(dmi,lex.id<1000)
etm.D <- etm.Lexis( dmi, time.scale=3 )
plot( etm.D, col=rainbow(5), lwd=2, lty=1, xlab="DM duration" )
}
```

---

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 `lm` or `glm` object. The model must not have an intercept term
`...`	arguments to the `nlm` 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](#)

---

gen.exp   *Generate covariates for drug-exposure follow-up from drug purchase records.*

Description

From records of drug purchase and possibly known treatment intensity, the time since first drug use and cumulative dose at prespecified times is computed. Optionally, lagged exposures are computed too, i.e. cumulative exposure a prespecified time ago.

Usage

```
gen.exp(purchase, id = "id", dop = "dop", amt = "amt", dpt = "dpt",
              fu, doe = "doe", dox = "dox",
         breaks,
        use.dpt = ( dpt %in% names(purchase) ),
            lags = NULL,
      push.max = Inf,
      pred.win = Inf,
       lag.dec = 1 )
```

Arguments

purchase	Data frame with columns `id`-person id, `dop`-date of purchase, `amt`-amount purchased, and optionally `dpt`-defined daily dose, that is how much is assumed to be ingested per unit time. The time unit used here is assumed to be the same as that used in dop, so despite the name it is not necessarily measured per day.
id	Name of the id variable in the data frame.
dop	Name of the date of purchase variable in the data frame.
amt	Name of the amount purchased variable in the data frame.
dpt	Name of the dose-per-time variable in the data frame.
fu	Data frame with follow-up period for each person, the person id variable must have the same name as in the `purchase` data frame.
doe	Name of the date of entry variable.
dox	Name of the date of exit variable.
use.dpt	Logical, should we use information on dose per time.
breaks	Numerical vector of time points where the time since exposure and the cumulative dose are computed.
lags	Numerical vector of lag-times used in computing lagged cumulative doses.
push.max	How much can purchases maximally be pushed forward in time. See details.

pred.win	The length of the window used for constructing the average dose per time used to compute the duration of the last purchase
lag.dec	How many decimals to use in the construction of names for the lagged exposure variables

Details

Each purchase record is converted into a time-interval of exposure.

If use.dpt is TRUE then the dose per time information is used to compute the exposure interval associated with each purchase. Exposure intervals are stacked, that is each interval is put after any previous. This means that the start of exposure to a given purchase can be pushed into the future. The parameter push.max indicates the maximally tolerated push. If this is reached by a person, the assumption is that some of the purchased drug is not counted in the exposure calculations.

The dpt can either be a constant, basically translating the purchased amount into exposure time the same way for all persons, or it can be a vector with different treatment intensities for each purchase. In any case the cumulative dose is computed taking this into account.

If use.dpt is FALSE then the exposure from one purchase is assumed to stretch over the time to the next purchase, so we are effectively assuming different rates of dose per time between any two adjacent purchases. Moreover, with this approach, periods of non-exposure does not exist.

The intention of this function is to generate covariates for a particular drug for the entire follow-up of each person. The reason that the follow-up prior to drug purchase and post-exposure is included is that the covariates must be defined for these periods too, in order to be useful for analysis of disease outcomes.

Value

A data frame with one record per follow-up interval between breaks, with columns:

id person id.

dof date of follow up, i.e. start of interval. Apart from possibly the first interval for each person, this will assume values in the set of the values in breaks.

Y the length of interval.

tfi time from first initiation of drug.

tfc time from latest cessation of drug.

cdur cumulative time on the drug.

cdos cumulative dose.

ldos suffixed with one value per element in lags, the latter giving the cumulative doses lags before dof.

Author(s)

Bendix Carstensen, <bxc@steno.dk>

See Also

Lexis, splitLexis

Examples

```
# Construct a simple data frame of purchases for 3 persons
# The purchase units (in variable dose) correspond to
n <- c( 10, 17, 8 )
dop <- c( 1995.2+cumsum(sample(1:4/10,n[1],replace=TRUE)),
          1997.3+cumsum(sample(1:4/10,n[2],replace=TRUE)),
          1997.3+cumsum(sample(1:4/10,n[3],replace=TRUE)) )
amt <- sample(   1:3/15, sum(n), replace=TRUE )
dpt <- sample( 15:20/25, sum(n), replace=TRUE )
dfr <- data.frame( id = rep(1:3,n),
                   dop,
                   amt = amt,
                   dpt = dpt )
round( dfr, 3 )
# Construct a simple dataframe for follow-up periods for these 3 persons
fu   <- data.frame( id = 1:3,
                    doe = c(1995,1997,1996)+1:3/4,
                    dox = c(2001,2003,2002)+1:3/5 )
round( fu, 3 )
( dpos <- gen.exp( dfr,
                    fu = fu,
                breaks = seq(1990,2015,0.5),
                  lags = 2:3/5 ) )
( xpos <- gen.exp( dfr,
                    fu = fu,
               use.dpt = FALSE,
                breaks = seq(1990,2015,0.5),
                  lags = 2:3/5 ) )

# How many relevant columns
nvar <- ncol(xpos)-3
clrs <- rainbow(nvar)

# Show how the variables relate to the follow-up time
par( mfrow=c(3,1), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6, bty="n" )
for( i in unique(xpos$id) )
matplot( xpos[xpos$id==i,"dof"],
         xpos[xpos$id==i,-(1:3)],
         xlim=range(xpos$dof), ylim=range(xpos[,-(1:3)]),
         type="l", lwd=2, lty=1, col=clrs,
         ylab="", xlab="Date of follow-up" )
ytxt <- par("usr")[3:4]
ytxt <- ytxt[1] + (nvar:1)*diff(ytxt)/(nvar+2)
xtxt <- rep( sum(par("usr")[1:2]*c(0.98,0.02)), nvar )
text( xtxt, ytxt, colnames(xpos)[-(1:3)], font=2,
                  col=clrs, cex=1.5, adj=0 )
```

---

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 bootsrap 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 specfied.

cov	A glm object from a binomial model with complementary log-log link, estimating the log-rate-ratios. Only if "MRR" is specfied.
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 <- Icens( 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 <- Icens( entry, well, ill,
               model="AER", formula=~pyr+us, breaks=seq(1980,1990,5),
               data=hivDK)
# There is actually a print method:
print( m.ER )
```

---

lep   *An unmatched case-control study of leprosy incidence*

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   *Create a Lexis object*

Description

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

Usage

```
Lexis( entry, exit, duration, entry.status = 0, exit.status = 0, id, data,
       merge=TRUE, states, tol=.Machine$double.eps^0.5, keep.dropped=FALSE )
```

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). The names of the timescales must be different from any column name in `data`.
`exit`	a named list of exit times.
`duration`	a numeric vector giving the duration of follow-up.
`entry.status`	a vector or a factor giving the status at entry
`exit.status`	a vector or factor 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 person represented in the Lexis object. Defaults to `1:nrow(data)`
`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 equal to `states`.
`tol`	Numerical tolerance for follow-up time. Rows with duration less than this value are automatically dropped.
`keep.dropped`	Logical. Should dropped rows from `data` be saved as an attribute with the object for inspection?

Details

The analysis of long-term population-based follow-up studies typically requires multiple time scales to be taken into account, such as age, calender 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 several timescales.

The `Lexis` function can create 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. The latter is the default.

There are also `merge`, `subset` and `transform` methods for `Lexis` objects. They work as the corresponding methods for data-frames but ensures that the result is a `Lexis` object.

Value

An object of class `Lexis`. This is represented as a data frame with a column for each time scale (with neaes equal to the union of the names of `entry` and `exit`), and additional columns with the following names:

`lex.id`	Identification of the persons.
`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 (the default) 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.

`entry`, `exit` must be numeric, using [Date](#) variables will cause some of the utilites to crash. Transformation by [cal.yr](#) is recommended.

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.

If any of `entry.status` or `exit.status` are of mode character, they will both be converted to factors.

If `entry.status` is not given, then its class is automatically set to that of `exit.status`. If `exit.status` is a character or factor, the value of `entry.status` is set to the first level. This may be highly undesirable, and therefore noted. For example, if `exit.status` is character the first level will be the first in the alphabetical ordering; slightly unfortunate if values are c("Well","Diseased"). If `exit.status` is logical, the value of `entry.status` set to FALSE. If `exit.status` is numeric, the value of `entry.status` set to 0.

If `entry.status` or `exit.status` are factors or character, the corresponding state variables in the returned `Lexis` object, `lex.Cst` and `lex.Xst` will be (unordered) factors with identical set of levels, namely the union of the levels of `entry.status` and `exit.status`.

Author(s)

Martyn Plummer with contributions from Bendix Carstensen

See Also

[plot.Lexis](#), [splitLexis](#), [cutLexis](#), [merge.Lexis](#), [subset.Lexis](#), [transform.Lexis](#), [summary.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 <- cal.yr( xcoh, format="%d/%m/%Y", wh=2:4 )
# See how it looks
xcoh
str( xcoh )

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

# Using character states may have undesired effects:
xcoh$Fail <- c("Dead","Well","Dead")
Lexis( entry = list( per=entry ),
        exit = list( per=exit, age=exit-birth ),
 exit.status = Fail,
        data = xcoh )

# ...unless you order the levels correctly
( xcoh$Fail <- factor( xcoh$Fail, levels=c("Well","Dead") ) )
Lexis( entry = list( per=entry ),
        exit = list( per=exit, age=exit-birth ),
 exit.status = Fail,
        data = xcoh )
```

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. Intended for presentation purposes.

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.
int	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.
lab.int	The interval between labelling of the grids.
col.life	Colour of the life lines.
lwd.life	Width of the life lines.
age.grid	Should grid lines be drawn for age?
date.grid	Should grid lines be drawn for date?
coh.grid	Should grid lines be drawn for birth cohorts (diagonals)?
col.grid	Colour of the grid lines.
lwd.grid	Width of the grid lines.
las	How are the axis labels plotted?
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.
fail	Logical of event status at exit for the persons whose life lines are plotted.
pch.fail	Symbols at the end of the life lines for censorings (`fail==0`) and failures (`fail != 0`).
cex.fail	Expansion of the status marks at the end of life lines.
col.fail	Character vector of length 2 giving the colour of the failure marks for censorings and failures respectively.
data	Dataframe in which to interpret the arguments.
...	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 colums 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. If the primary aim is to illustrate follow-up of a cohort, then it is better to represent the follow-up in a Lexis object, and use the generic plot.Lexis function.

Author(s)

Bendix Carstensen, http://BendixCarstensen.com

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   *Draw life lines in a Lexis diagram.*

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://BendixCarstensen.com>

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 )
```

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 ) )
```

---

lls *Functions to manage and explore the workspace*

Description

These functions help you to find out what has gone wrong and to start afresh if needed.

Usage

```
lls(pos = 1, pat = "", all=FALSE, print=TRUE )
clear()
```

Arguments

pos	Numeric. What position in the search path do you want listed.
pat	Character. List only objects that have this string in their name.
all	Logical. Should invisible objects be printed too - see [ls](#) to which this argument is passed.
print	Logical. Should the result be printed?

Details

lls is designed to give a quick overview of the name, mode, class and dimension of the object in your workspace. They may not always be what you think they are.

clear clears all your objects from workspace, and all attached objects too — it only leaves the loaded packages in the search path; thus allowing a fresh start without closing and restarting R.

Value

lls returns a data frame with four character variables: codename, codemode, codeclass and code-size and one row per object in the workspace (if pos=1). size is either the length or the dimension of the object. The data frame is by default printed with left-justified columns.

Author(s)

lls: Unknown. Modified by Bendix Carstensen from a long forgotten snatch.

clear: Michael Hills / David Clayton.

Examples

```
x <- 1:10
y <- rbinom(10, 1, 0.5)
m1 <- glm( y ~ x, family=binomial )
M <- matrix( 1:20, 4, 5 )
.M <- M
lls()
clear()
lls()
```

---

lungDK   *Male lung cancer incidence in Denmark*

Description

Male lung cancer cases and population riks 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 enpoint of the period interval, a numeric vector.
- C5: Left enpoint of the birth cohort interval, a numeric vector.
- up: Indicator of upper trianges of each age by period rectangle in the Lexis diagram. (up=(P5−A5−C5)/5).
- Ax: The mean age of diagnois (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 correponds 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: B. Carstensen: Age-Period-Cohort models for the Lexis diagram. Statistics in Medicine, 26: 3018-3045, 2007.

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 )
```

M.dk   *Mortality in Denmark 1974 ff.*

Description

Mortality in one-year classes of age (0-98,99+) and period (1974 ff.) in Denmark.

Usage

```
data(M.dk)
```

Format

A data frame with 6400 observations on the following 6 variables.

A Age-class, 0-98, 99:99+

sex Sex. 1:males, 2:females

P Period (year) of death

D Number of deaths

Y Number of person-years

rate Mortality rate per 1000 person-years

Details

Deaths in ages over 100 are in the class labelled 99. Risk time is computed by tabulation of the risk
time in Y.dk, except for the class 99+ where the average of the population size in ages 99+ at the
first and last date of the year is used.

Source

http://www.statistikbanken.dk/statbank5a/SelectTable/omrade0.asp?SubjectCode=02&PLanguage=
1&ShowNews=OFF

Examples

```
data(M.dk)
str(M.dk)

zz <- xtabs( rate ~ sex+A+P, data=M.dk )
zz[zz==0] <- NA # 0s makes log-scale plots crash
par(mfrow=c(1,2), mar=c(0,0,0,0), oma=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
for( i in 1:2 )
{
matplot( dimnames(zz)[[2]], zz[i,,],
         lty=1, lwd=1, col=rev(heat.colors(37)),
         log="y", type="l", ylim=range(zz,na.rm=TRUE),
         ylab="", xlab="", yaxt="n" )
text( 0, max(zz,na.rm=TRUE), c("M","F")[i], font=2, adj=0:1, cex=2, col="gray" )
if( i==1 ) axis( side=2, las=1 )
}
mtext( side=1, "Age", line=2, outer=TRUE )
mtext( side=2, "Mortality rate", line=2, outer=TRUE )
```

---

merge.data.frame   *Merge data frame with a Lexis object*

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](#)

---

merge.Lexis   *Merge a Lexis object with a data frame*

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.

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

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.

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:

age:	Age class, 0–89, 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, gmortDK

Examples

```
data(mortDK)
```

N.dk   *Population size in Denmark*

Description

The population size at 1st January in ages 0-99.

Usage

```
data(N.dk)
```

Format

A data frame with 7200 observations on the following 4 variables.

sex Sex, 1:males, 2:females

A Age. 0:0, 1:1, ..., 98:98, 99:99+

P Year

N Number of persons alive at 1st January year P

Source

Examples

```
data(N.dk)
str(N.dk)
with(N.dk,addmargins(tapply(N,list(P,sex),sum),2))
with(subset(N.dk,P==max(P)),addmargins(tapply(N,list(A,sex),sum)))
```

N2Y	*Create risk time (Person-Years) in Lexis triangles from population data.*

Description

Data on population size at equidistant dates and age-classes are used to estimate person-time at risk in Lexis-triangles, i.e. classes classified by age, period AND cohort.

Usage

```
N2Y( A, P, N,
     data = NULL,
     return.dfr = TRUE)
```

Arguments

A	Name of the age-variable, which should be numeric, corresponding to the left endpoints of the age intervals.
P	Name of the period-variable, which should be numeric, corresponding to the date of population count.
N	The population size at date P in age class A.
data	A data frame in which arguments are interpreted.
return.dfr	Logical. Should the results be returned as a data frame (default TRUE) or as a table.

Details

The calculation of the risk time from the population figures is done as described in: B. Carstensen: Age-Period-Cohort models for the Lexis diagram. Statistics in Medicine, 26: 3018-3045, 2007.

Value

A data frame with variables A, P and Y, representing the mean age and period in the Lexis triangles and the person-time in them.

If res.dfr=FALSE a three-way table classified by the left end point of the age-classes and the periods and a factor wh taking the values up and lo corresponding to upper (early cohort) and lower (late cohort) Lexis triangles.

Author(s)

Bendix Carstensen, BendixCarstensen.com

References

B. Carstensen: Age-Period-Cohort models for the Lexis diagram. Statistics in Medicine, 26: 3018-3045, 2007.

See Also

splitLexis, apc.fit

Examples

```
# Danish population at 1 Jan each year by sex and age
data( N.dk )
# An illustrative subset
( Nx <- subset( N.dk, sex==1 & A<5 & P<1975 ) )
# Show the data in tabular form
xtabs( N ~ A + P, data=Nx )
# Lexis triangles as data frame
Nt <- N2Y( data=Nx, return.dfr=TRUE )
xtabs( Y ~ round(A,2) + round(P,2), data=Nt )
# Lexis triangles as a 3-dim array
ftable( N2Y( data=Nx, return.dfr=FALSE ) )
```

NArray   *Set up an array of NAs, solely from the list of dimnames*

Description

Defines an array of NAs, solely from the list of dimnames

Usage

```
NArray( x, cells=NA )
ZArray( x, cells=0 )
```

Arguments

x	A (possibly named) list to be used as dimnames for the resulting array
cells	Value(s) to fill the array

Details

This is a simple useful way of defining arrays to be used for collection of results. The point is that everything is defined from the named list, so in the process of defining what you want to collect, there is only one place in the program to edit. It's just a wrapper for array. ZArray is just a wrapper for NArray with a different default.

Value

An array with dimnames attribute x, and all values equal to cells.

Author(s)

Bendix Carstensen

Examples

```
ftable(
NArray( list(Aye = c("Yes", "Si", "Oui"),
             Bee = c("Hum", "Buzz"),
             Sea = c("White", "Black", "Red", "Dead") ) ) )
```

---

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(breaks) if type="left", for values above max(breaks) if type="right" and for values outside range(breaks) 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, <bxc@steno.dk>, http://BendixCarstensen.com, 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://BendixCarstensen.com>

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)
```

Ns *Natural splines - cubic splines, made linear beyond outermost knots.*

Description

This function is just for convenient specification of natural splines in practical modelling. The convention used is to take the smallest and the largest of the supplied knots as boundary knots.

Usage

```
Ns(x, df = NULL, knots = NULL, intercept = FALSE, Boundary.knots = NULL)
```

Arguments

x	A variable.
df	degrees of freedom.
knots	knots to be used both as boundary and internal knots. If Boundary.knots are given, this will be taken as the set of internal knots.
intercept	Should the intercept be included in the resulting basis?
Boundary.knots	The boundary knots beyond which the spline is linear.

Details

The function is a convenience wrapper, which gives the same result as ns, but allows omission of Boundary.knots. It has the same arguments as ns, and in the same order.

Value

A matrix of dimension c(length(x),df) where either df was supplied or if knots were supplied, df = length(knots) - intercept.

Note

The need for this function is from analysis of rates in epidemiology and demography, where the dataset are time-split records of follow-up, and the range of data therefore rarely is of any interest (let alone is meaningful). In Poisson modelling of rates based on time-split records one should aim at having the same number of *events* between knots, rather than the same number of observations.

Author(s)

Bendix Carstensen

Examples

```
require(splines); require(stats); require(graphics)
ns( women$height, df = 3)
Ns( women$height, knots=c(63,59,71,67) )
# Gives the same results as ns:
summary( lm(weight ~ ns(height, df = 3), data = women) )
summary( lm(weight ~ Ns(height, df = 3), data = women) )

# Get the diabetes data and set up as Lexis object
data(DMlate)
DMlate <- DMlate[sample(1:nrow(DMlate),500),]
dml <- Lexis( entry = list(Per=dodm, Age=dodm-dobth, DMdur=0 ),
                exit = list(Per=dox),
          exit.status = factor(!is.na(dodth),labels=c("DM","Dead")),
                 data = DMlate )

# Split in 1-year age intervals
dms <- splitLexis( dml, time.scale="Age", breaks=0:100 )
summary( dms )

# Model age-specific rates using Ns with 7 knots
n.kn <- 7
( a.kn <- with( subset(dms,lex.Xst=="Dead"),
                 quantile( Age+lex.dur, probs=(1:n.kn-0.5)/n.kn ) ) )
m0 <- glm( lex.Xst=="Dead" ~ Ns( Age, kn=a.kn ),
            offset = log( lex.dur ), family=poisson, data=dms )

# Plot estimated curve:
matplot( 50:100, ci.exp( m0, ctr.mat=cbind(1,Ns(50:100,knots=a.kn)) ),
         type="l", lwd=c(3,1,1), lty=1, col="black", log="y",
         ylab="Mortality rates per 1000 PY", xlab="Age (years)", las=1 )
rug( a.kn, lwd=2 )
```

---

occup   *A small occupational cohort*

Description

This is the data that is behind the illustrative Lexis diagram in Breslow & Day's book on case-control studies.

Usage

```
data(occup)
```

Format

A data frame with 13 observations on the following 4 variables.

AoE a numeric vector, Age at Entry

DoE a numeric vector, Date of entry

DoX a numeric vector, Date of eXit

Xst eXit status D-event, W-withdrawal, X-censoring

References

Breslow & Day: Statistical Methods in Cancer Research, vol 1: The analysis of case-control studies, figure 2.2, p. 48.

Examples

```
data(occup)
lx <- Lexis( entry = list( per=DoE, age=AoE ),
              exit = list( per=DoX ),
      entry.status = "W",
       exit.status = Xst,
              data = occup )
plot( lx )
# Split follow-up in 5-year classes
sx <- splitLexis( lx, seq(1940,1960,5), "per" )
sx <- splitLexis( sx, seq(  40,  60,5), "age" )
plot( sx )

# Plot with a bit more paraphernalia and a device to get
# the years on the same physical scale on both axes
ypi <- 2.5 # Years per inch
dev.new( height=15/ypi+1, width=20/ypi+1 ) # add an inch in each direction for
par( mai=c(3,3,1,1)/4, mgp=c(3,1,0)/1.6 )  # the margins set in inches by mai=
plot(sx,las=1,col="black",lty.grid=1,lwd=2,type="l",
     xlim=c(1940,1960),ylim=c(40,55),xaxs="i",yaxs="i",yaxt="n",
     xlab="Calendar year", ylab="Age (years)")
axis( side=2, at=seq(40,55,5), las=1 )
points(sx,pch=c(NA,16)[(sx$lex.Xst=="D")+1] )
box()
# Annotation with the person-years
PY.ann.Lexis( sx, cex=0.8 )
```

---

pc.lines   *Plot period and cohort effects in an APC-frame.*

Description

When an APC-frame has been produced by apc.frame, this function draws curves in the period/cohort part of the frame.

Usage

```
pc.points( x, y, ... )
pc.lines( x, y, ... )
pc.matpoints( x, y, ... )
pc.matlines( x, y, ... )
cp.points( x, y, ... )
cp.lines( x, y, ... )
cp.matpoints( x, y, ... )
cp.matlines( x, y, ... )
```

Arguments

x	vector of x-coordinates.
y	vector of y-coordinates.
...	Further parameters to be transmitted to points, lines, matpoints or matlines used for plotting the three sets of curves.

Details

Since the Age-part of the frame is referred to by its real coordinates plotting in the calendar time part requires translation and scaling to put things correctly there, that is done by the functions `pc.points` etc.

The functions `cp.points` etc. are just synonyms for these, in recognition of the fact that you can never remember whether it is "pc" pr "cp".

Value

The functions return nothing.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://BendixCarstensen.com>

See Also

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

---

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? If 0 or FALSE nothing is printed

Value

A table of percentages, 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://BendixCarstensen.com>.

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( pctab( A ) )
ftable( pctab( addmargins( A, 1 ), 3 ) )
round( ftable( pctab( addmargins( A, 1 ), 3 ), row.vars=3 ), 1)
```

---

plot.apc   *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

```
## S3 method for class 'apc'
plot( x, r.txt="Rate", ...)
          apc.plot( x, r.txt="Rate", ...)
```

Arguments

x	An object of class apc.
r.txt	The text to put on the vertical rate axis.
...	Additional arguments passed on to `apc.lines`.

Details

`plot.apc` is just a wrapper for `apc.plot`.

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, however see `cp.points`.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://BendixCarstensen.com>

See Also

`apc.lines`, `lines.apc`, `apc.frame`, `apc.fit`

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] )
```

---

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=Lexis( entry=list(Date=1900,Age=0), exit=list(Age=0) ),
                    time.scale = NULL, type="l", breaks="lightgray", ...)
## S3 method for class 'Lexis'
points(x, time.scale = options()[["Lexis.time.scale"]] , ...)
## S3 method for class 'Lexis'
lines(x, time.scale = options()[["Lexis.time.scale"]], ...)
## S3 method for class 'Lexis'
PY.ann(x, time.scale = options()[["Lexis.time.scale"]], digits=1, ...)
```

Arguments

x	An object of class `Lexis`. The default is a bogus `Lexis` object, so that `plot.Lexis` can be called without the first argument and still produce a(n empty) Lexis diagram. Unless arguments `xlim` and `ylim` are given in this case the diagram is looking pretty daft.
time.scale	A vector of length 1 or 2 giving the time scales to be plotted either by name or numerical order
type	Character indication what to draw: "n" nothing (just set up the diagram), "l" - liefelines, "p" - endpoints of follow-up, "b" - both lifelines and endpoints.
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
digits	Numerical. How many digits after the demimal points should be when plotting the person-years.
...	Further graphical parameters to be passed to the plotting methods. Grids can be drawn (behind the life lines) using the following parameters in `plot`:

- `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 nunerical 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.

PY.ann writes the length of each (segment of) life line at the middle of the line. Not advisable to use with large cohorts. Another example is in the example file for occup.

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", "01/04/1957", "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 )
# Show follow-up time
PY.ann( Lcoh )

# Show exit status
plot( Lcoh, type="b" )
# Same but failures only
plot( Lcoh, type="b", pch=c(NA,16)[Lcoh$fail+1] )

# 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 (forest plot)*

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.txt = "black",
       font.txt = 1,
      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. Either a character vector or an expression vector.
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.txt	Colour of the text annotating the estimates.
font.txt	Font for the text annotating the estimates.
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.
...	Arguments passed on to ci.lin when a model object is supplied as ests.

Details

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

If a model object of class "glm", "coxph", "clogistic" or "gnlm" is supplied the argument xlog defaults to TRUE, and exponentiated estimates are extracted by default.

Value

NULL

Author(s)

Bendix Carstensen, <bxc@steno.dk>, <http://BendixCarstensen.com>

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 <- ci.lin( m1, subset=-1 )[,-(2:4)] )

# Plots with increasing amounts of bells and whistles
par( mfcol=c(3,2), mar=c(3,3,2,1) )
plotEst( cf )
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 fb"
plotEst( cf, grid=TRUE, cex=2, col.points=rainbow(4),
                                col.lines=rainbow(4), vref=1 )
#
etxt <- expression("Plain text, qouted",
                    "combined with maths:"*sqrt(a)*phi[c],
                    f^d*"  Hb"*A[1][c],
                    eff^e*"  kg/"*m^2)
plotEst( cf, txt=etxt, 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: <x|y>=sum(x*w*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 span(X)?
weight	Weights defining the inner product. Numerical vector of length nrow(X).

Value

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

Author(s)

Bendix Carstensen, Steno Diabetes Center, <http://BendixCarstensen.com>, 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 c("ap","ac","apc","pa","ca"), 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?
a.ann	Logical indicating whether age-curves should be annotated.
p.ann	do. for period-curves.
c.ann	do. for cohort-curves.
xannx	The fraction that the x-axis is expanded when curves are annotated.
cex.ann	Expansion factor for characters annotating curves.
a.thin	Vector of integers indicating which of the age-classes should be labelled.

p.thin	do. for the periods.
c.thin	do. for the cohorts.
col	Colours for the curves.
a.col	Colours for the age-curves.
p.col	do. for the period-curves.
c.col	do. for the cohort-curves.
p.lines	Should rates from the same period be connected?
c.lines	Should rates from the same cohort be connected?
...	Additional arguments pssed on to `matlines` when plotting the curves.

Details

Zero values of the rates are ignored. They are neiter 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://BendixCarstensen.com>

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 an 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

```
## S3 method for class 'factor'
Relevel( x, ref, first = TRUE, collapse="+", ... )
```

Arguments

x   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.

...   Arguments passed on to other methods.

Value

An unordered factor, where levels of x have been reordered and/or collapsed.

See Also

[Relevel.Lexis](#)

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`. 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

Details

As an alternative to a test and a status variable, 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. The test used to derive sensitivity, specificity, PV+ and PV- as a function of x is test$\geq x$ as a predictor of stat=TRUE.

Value

A list with two components:

| res | dataframe with variables sens, spec, pvp, pvn and name of the test variable. The latter is the unique values of test or linear predictor from the logistic regression in ascending order with -Inf prepended. Since the sensitivity is defined as $P(test > x)|status = TRUE$, the first row has sens equal to 1 and spec equal to 0, corresponding to drawing the ROC curve from the upper right to the lower left corner. |
| 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://BendixCarstensen.com>

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)
```

---

simLexis   *Simulate a Lexis object representing follow-up in a multistate model.*

Description

Based on a (pre-)Lexis object representing persons at given states and times, and full specification of transition intensities between states in the form of models for the transition rates, this function simulates transition times and -types for persons and returns a Lexis object representing the simulated cohort. The simulation scheme accommodates multiple timescales, including time since entry into an intermediate state, and accepts fitted Poisson models, Cox-models or just a function as specification of rates.

Usage

```
simLexis( Tr, init,
            N = 1,
       lex.id,
      t.range = 20,
        n.int = 101,
     time.pts = seq(0,t.range,length.out=n.int) )
nState( obj, at, from, time.scale = 1 )
pState( nSt, perm = 1:ncol(nSt) )
## S3 method for class 'pState'
plot( x,
                      col = rainbow(ncol(x)),
                   border = "transparent",
                     xlab = "Time",
                     ylim = 0:1,
                     ylab = "Probability", ... )
## S3 method for class 'pState'
lines( x,
                      col = rainbow(ncol(x)),
                   border = "transparent", ... )
```

Arguments

Tr   A named list of named lists. The names of the list are names of the transient
                states in the model. Each list element is again a named list. The names of the
                elements of this inner list are the names of the states reachable from the state
                with name equal to the list. Elements of the intter lists represent transitions. See
                details.

init   A (pre-)[Lexis] object representing the initial state of the persons whose trajec-
                 tories through the multiple states we want to simulate. Must have an attribute
                 "time.since" — see details. Duplicate values of lex.id are not sensible and not
                 accepted.

N   Numeric. How many persons should be simulated. N persons with covariate
                 configuration of each row in init will be simulated. Either a scalar or a vector
                 of length nrow(init).

lex.id   Vector of ids of the simulated persons. Useful when simulating in chunks.

t.range   Numerical scalar. The range of time over which to compute the cumulative
                 rates when simulating. Simulted times beyond this will result in an obervation
                 censored at t.range after entry.

n.int   Number of intervals to use when computing (cumulative) rates.

time.pts   Numerical vector of times since start. Cumulative rates for transitions are com-
                 puted at these times after stater and entry state. Simulation is only done till time
                 max(time.pts) after start, where persons are censored. Must start with 0.

obj   A Lexis object.

from   The point on the time scale time.scale from which we start counting.

time.scale   The timescale to which from refer.

at   Time points (after from) where the number of persons in each state is to be
                 computed.

nSt   A table obtained by nState.

perm   A permutation of columns used before cumulating row-wise and taking percent-
                 ages.

x   An object of class pState, e.g. created by pState.

col   Colors for filling the areas between curves.

border   Colors for outline of the areas between curves.

xlab   Label on x-axis

ylim   Limits on y-axis

ylab   Label on y-axis

...   Further arguments passed on to plot.

Details

The simulation command simLexis is not defined as a method for Lexis objects, because the
input is not a Lexis object, the Lexis-like object is merely representing a prevalent population
and a specification of which variables that are timescales. The variables lex.dur and lex.Xst are
ignored (and overwritten) if present. The core input is the list Tr giving the transitions.

The components of Tr represents the transition intensities between states. The transition from state
A to B, say, is assumed stored in TrAB. Thus names of the elements of Tr are names of transient
states, and the names of the elements of each these are the names of states reachable from the
corresponding transient state.

The transition intensities are assumed modelled by either a glm with Poisson family or a Cox-
model. In both cases the timescale(s) in the model must be using the names fo the timescales in a

Lexis object representng the follow-up in a cohort, and the risk time must be taken from the variable `lex.dur` — see the example.

Alternatively, an element in `Tr` could be a function that from a data frame produces transition rates, or specifically cumulative transition rates over intervals of length `lex.dur`.

The pre-Lexis object `init` must contain values of all variables used in any of the objects in `Tr`. Moreover the attributes `time.scales` and `time.since` must be present. The attribute `time.since` is a character vector of the same length as `time.scales` and an element has value `"A"` if the corresponding time scale is defined as "time since entry into state A", otherwise the value is `""`. If not present it will be set to a vector of `""`s, which is only OK if no time scales are defined as time since entry to a state.

The function [Lexis](#) automatically generates an attribute `time.since`, and [cutLexis](#) updates it when new time scales are defined. Hence, the simplest way of defining a initial pre-Lexis object representing a current state of a (set of) persons to be followed through a multistate model is to take NULL rows of an existing Lexis object (normally the one used for estimation), and so ensuring that all relevant attributes and state levels are properly defined. See the example code.

The prevalence function `nState` computes the distribution of individuals in different states at pre-specified times. Only sensible for a simulated Lexis object. The function `pState` takes a matrix as output by `nState` and computes the row-wise cumulative probabilities across states, and leaves an object of class `pState`, suitable for plotting.

Value

`simLexis` returns a [Lexis](#) object representing the experience of a population starting as `init` followed through the states according to the transitions in `Tr`.

The function `nState` returns a table of persons classified by states at each of the times in `at`. Note that this function can easily produce meaningless results, for example if applied to a `Lexis` object not created by simulation. If you apply it to a `Lexis` object generated by `simLexis`, you must make sure that you start (`from`) the point where you started the simulation on the correct timescale, and you will get funny results if you try to tabulate beyond the censoring time for the simulation. The resulting object has class `"table"`.

The result from using `pState` on the result from `nState` has class `c("pState","matrix")`.

Author(s)

Bendix Carstensen, [BendixCarstensen.com](#).

See Also

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

Examples

```
data(DMlate)
dml <- Lexis( entry = list(Per=dodm, Age=dodm-dobth, DMdur=0 ),
               exit = list(Per=dox),
        exit.status = factor(!is.na(dodth),labels=c("DM","Dead")),
               data = DMlate[runif(nrow(DMlate))<0.1,] )
# Split follow-up at insulin, introduce a new timescale,
```

```
# and split non-precursor states
dmi <- cutLexis( dml, cut = dml$doins,
                           pre = "DM",
                     new.state = "Ins",
                     new.scale = "t.Ins",
                  split.states = TRUE )
# Split the follow in 1-year intervals for modelling
Si <- splitLexis( dmi, 0:30/2, "DMdur" )
# Define knots
nk <- 4
( ai.kn <- with( subset(Si,lex.Xst=="Ins"),
                   quantile( Age+lex.dur, probs=(1:nk-0.5)/nk ) ) )
( ad.kn <- with( subset(Si,lex.Xst=="Dead"),
                   quantile( Age+lex.dur, probs=(1:nk-0.5)/nk ) ) )
( di.kn <- with( subset(Si,lex.Xst=="Ins"),
                   quantile( DMdur+lex.dur, probs=(1:nk-0.5)/nk ) ) )
( dd.kn <- with( subset(Si,lex.Xst=="Dead"),
                   quantile( DMdur+lex.dur, probs=(1:nk-0.5)/nk ) ) )
( td.kn <- with( subset(Si,lex.Xst=="Dead(Ins)"),
                   quantile( t.Ins+lex.dur, probs=(1:nk-0.5)/nk ) ) )

# Fit Poisson models to transition rates
library( splines )
DM.Ins <- glm ( (lex.Xst=="Ins") ~ Ns( Age  , knots=ai.kn ) +
                                   Ns( DMdur, knots=di.kn ) +
                                   I(Per-2000) + sex,
               family=poisson, offset=log(lex.dur),
               data = subset(Si,lex.Cst=="DM") )
DM.Dead <- glm ( (lex.Xst=="Dead") ~ Ns( Age  , knots=ad.kn ) +
                                     Ns( DMdur, knots=dd.kn ) +
                                     I(Per-2000) + sex,
               family=poisson, offset=log(lex.dur),
               data = subset(Si,lex.Cst=="DM") )
Ins.Dead <- glm ( (lex.Xst=="Dead(Ins)") ~ Ns( Age  , knots=ad.kn ) +
                                           Ns( DMdur, knots=dd.kn ) +
                                           Ns( t.Ins, knots=td.kn ) +
                                           I(Per-2000) + sex,
               family=poisson, offset=log(lex.dur),
               data = subset(Si,lex.Cst=="Ins") )

# Stuff the models into an object representing the transitions
Tr <- list( "DM" = list( "Ins"        = DM.Ins,
                         "Dead"       = DM.Dead  ),
            "Ins" = list( "Dead(Ins)" = Ins.Dead ) )
lapply( Tr, names )

# Define an initial object - note the combination of "select=" and NULL
# which ensures that the relevant attributes from the Lexis object 'Si'
# are carried over to 'ini' ( Si[NULL,1:9] will lose essential
# attributes ):
ini <- subset(Si,select=1:9)[NULL,]
ini[1:2,"lex.Cst"] <- "DM"
ini[1:2,"Per"] <- 1995
```

```
ini[1:2,"Age"] <- 60
ini[1:2,"DMdur"] <- 5
ini[1:2,"sex"] <- c("M","F")
str(ini)

# Simulate 200 of each sex using the estimated models in Tr
simL <- simLexis( Tr, ini, time.pts=seq(0,11,0.5), N=200 )
summary( simL )

# Find the number of persons in each state at a set of times.
# Note that the times are shirter than the time-span simulated.
nSt <- nState( subset(simL,sex=="M"),
               at=seq(0,10,0.1), from=1995, time.scale="Per" )
nSt

# Show the cumulative prevalences in a different order than that of the
# state-level ordering and plot them using all defaults
pp <- pState( nSt, perm=c(1,2,4,3) )
head( pp )
plot( pp )

# A more useful set-up of the graph
clr <- c("orange2","forestgreen")
par( las=1 )
plot( pp, col=clr[c(2,1,1,2)] )
lines( as.numeric(rownames(pp)), pp[,2], lwd=2 )
mtext( "60 year old male, diagnosed 1995", side=3, line=2.5, adj=0 )
mtext( "Survival curve", side=3, line=1.5, adj=0 )
mtext( "DM, no insulin   DM, Insulin", side=3, line=0.5, adj=0, col=clr[1] )
mtext( "DM, no insulin", side=3, line=0.5, adj=0, col=clr[2] )
axis( side=4 )

# Using a Cox-model for the mortality rates assuming the two mortality
# rates to be proportional:
# When we fit a Cox-model, lex.dur must be used in the Surv() function,
# and the I() constrction must be used when specifying intermediate
# states as covariates, since factors with levels not present in the
# data will create NAs in the parameter vector returned by coxph, which
# in return will crash the simulation machinery.
library( survival )
Cox.Dead <- coxph( Surv( DMdur, DMdur+lex.dur,
                         lex.Xst %in% c("Dead(Ins)","Dead")) ~
                   Ns( Age-DMdur, knots=ad.kn ) +
                   I(lex.Cst=="Ins") +
                   I(Per-2000) + sex,
               data = Si )
Cr <- list( "DM" = list( "Ins"       = DM.Ins,
                         "Dead"      = Cox.Dead  ),
            "Ins" = list( "Dead(Ins)" = Cox.Dead ) )
simL <- simLexis( Cr, ini, time.pts=seq(0,11,0.2), N=200 )
summary( simL )
nSt <- nState( subset(simL,sex=="M"),
               at=seq(0,10,0.2), from=1995, time.scale="Per" )
```

```
pp <- pState( nSt, perm=c(1,2,4,3) )
plot( pp )
```

---

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, tol=.Machine$double.eps^0.5)
```

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
tol	numeric value >= 0. Intervals shorter than this value are dropped

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

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" ) )
str( x2 )

# Tabulate the cases and the person-years
summary( 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 )
```

---

  stack.Lexis   *Functions to facilitate analysis of multistate models.*

Description

    stack.Lexis produces a stacked object suited for analysis of several transition intensities simultaneously.

Usage

```
## S3 method for class 'Lexis'
stack(x, ...)
tmat( x, ... )
## S3 method for class 'Lexis'
tmat(x, Y=FALSE, mode = "numeric", ...)
```

Arguments

x	A [Lexis](#) object.
Y	Logical. Should the risk time be put in the diagonal? This is a facility which is used by [boxes.Lexis](#).
mode	Should the matrix be returned as a numeric matrix with NAs at unused places or (mode="logical") as a logical matrix with FALSE on the diagonal.
...	Not used.

Value

    tmat.Lexis returns a square transition matrix, classified by the levels of lex.Cst and lex.Xst, for every transition occurring the entry is the number of transitions occurring and NA in all oter entries. If Y=TRUE, the diagonal will contain the risk time in each of the states.

    stack.Lexis returns a dataframe to be used for analysis of multistate data when all transitions are modelled together, for example if some parameters are required to be the same for different transitions. The dataframe has class stacked.Lexis, and inherits the attributes time.scales and breaks from the Lexis object, and so function [timeBand](#) applies to a stacked.Lexis object too.

    The dataframe has same variables as the original Lexis object, but with each record duplicated as many times as there are possible exits from the current state, lex.Cst. Two variables are added: lex.Fail, an indicator of wheter an event for the transition named in the factor lex.Tr has occurred or not. lex.Tr is a factor with levels made up of combinations of the levels of lex.Cst and lex.Xst that do occur together in x, joined by a "->".

Author(s)

    Bendix Carstensen, <bxc@steno.dk>, http://BendixCarstensen.com

See Also

    [splitLexis](#) [cutLexis](#) [Lexis](#)

Examples

```
data(DMlate)
str(DMlate)
dml <- Lexis( entry=list(Per=dodm, Age=dodm-dobth, DMdur=0 ),
               exit=list(Per=dox),
        exit.status=factor(!is.na(dodth),labels=c("DM","Dead")),
               data=DMlate )
dmi <- cutLexis( dml, cut=dml$doins, new.state="Ins", pre="DM" )
summary( dmi )
ls.dmi <- stack( dmi )
str( ls.dmi )
# Check that all the transitions and person-years got across.
with( ls.dmi, rbind( table(lex.Fail,lex.Tr),
                     tapply(lex.dur,lex.Tr,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, by.id=FALSE)
  exit(x, time.scale = NULL, by.id=FALSE)
status(x, at="exit"        , by.id=FALSE)
   dur(x,                     by.id=FALSE)
```

Arguments

x	an object of class `Lexis`.
time.scale	a string or integer indicating the time scale. If omitted, all times scales are used.
by.id	Logical, if `TRUE`, only one record per unique value of `lex.id` is returned; either the first, the last or for `dur`, the sum of `lex.dur`. If `TRUE`, the returned object have the `lex.id` as (row)nmes attribute.
at	string indicating the time point(s) at which status is to be measured.

Value

The `entry` and `exit` functions return a vector of entry times and exit times, respectively, on the requested time scale. If multiple time scales are requested, then a matrix is returned.

The `status` function returns a vector giving the status at entry or exit and `dur` returns a vector with the lengths of the follow-up intervals.

Author(s)

Martyn Plummer

See Also

[Lexis](#)

---

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. It does NOT work inside with()!

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, percent, and sd.

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, sd.

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))
```

---

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:

count	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.
ratio	returns a table of values scale * sum(d)/sum(y)
percent	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.

Author(s)

Martyn Plummer

See Also

stat.table

---

subset.Lexis   *Subsetting Lexis (and stacked.Lexis) objects*

Description

Return subsets of Lexis objects which meet conditions

Usage

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

Arguments

x   an object of class Lexis

...   additional arguments to be passed to subset.data.frame. This will normally
                 be some logical expression selecting a subset of the rows. For details see subset.data.frame.

Details

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

The method for stacked.Lexis objects also shrinks the set of levels for lex.Cst and lex.Xst to
those actually occurring in the data frame.

Value

A Lexis object with selected rows and columns.

Author(s)

Martyn Plummer

See Also

Lexis, merge.Lexis

summary.Lexis	*Summarize 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'
summary( object, simplify=TRUE, scale=1, by=NULL, Rates=FALSE, ... )
  ## S3 method for class 'summary.Lexis'
print( x, ..., digits=2 )
```

Arguments

object	A Lexis object.
simplify	Should rows with 0 follow-up time be dropped?
scale	Scaling factor for the rates. The calculated rates are multiplied by this number.
by	Character vector of name(s) of variable(s) in `object`. Used to give a separate summaries for subsets of `object`. If longer than than 1, the interaction between that variables is used to stratify the summary. It is also possible to supply a vector of length `nrow(object)`, and the distinct values of this will be used to stratify the summary.
Rates	Should a component with transition rates be returned (and printed) too?
x	A `summary.Lexis` object.
digits	How many digits should be used for printing?
...	Other parameters - ignored

Value

An object of class `summary.Lexis`, a list with two components, `Transitions` and `Rates`, each one a matrix with rows classified by states where persons spent time, and columns classified by states to which persons transit. The `Transitions` contains number of transitions and has 4 extra columns with number of records, total number of events, total risk time and number of person contributing attached. The `Rates` contains the transitions rates.

If the argument `Rates` is FALSE (the default), then only the first component of the list is returned.

Author(s)

Bendix Carstensen, <bxc@steno.dk>

Examples

```
data( nickel )
# Lung cancer deaths and other deaths are coded 1 and 2
nic <- Lexis( data=nickel,
               entry=list(age=agein),
                exit=list(age=ageout,cal=ageout+dob,tfh=ageout-age1st),
         exit.status=factor( (icd > 0) + (icd %in% c(162,163)),
                                labels=c("Alive","Other","Lung") ) )
str( nic )
head( nic )
summary( nic )
# More detailed summary, by exposure level
summary( nic, by=nic$exposure>5, Rates=TRUE, scale=100 )
```

---

testisDK   *Testis cancer incidence in Denmark, 1943–1996*

Description

Number of testiscancer cases and male person-years in the Danish population 1943–1996

Usage

```
data(testisDK)
```

Format

A data frame with 4860 observations on the following 4 variables.

A Age class, 0,1,2,...,89

P Year, 1943,...,1996

D Number of testis cancer cases

Y Person years

Source

The Danish Cancer Registry

Examples

```
data(testisDK)
head(testisDK)
```

---

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 + hmang

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](#)

Examples

```
data(diet)
diet <- cal.yr(diet)
diet.lex <- Lexis(entry=list(period=doe),
                   exit=list(period=dox, age=dox-dob),
            exit.status=chd,
                   data=diet)
diet.split <- splitLexis(diet.lex, breaks=seq(40,70,5), "age" )
age.left <- timeBand(diet.split, "age", "left")
table(age.left)
age.fact <- timeBand(diet.split, "age", "factor")
table(age.fact)
age.mid <- timeBand(diet.split, "age", "mid")
table(age.mid)
```

---

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. Extracted from the time.scales attribute of the object.

Author(s)

Martyn Plummer

See Also

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

---

transform.Lexis   *Transform a Lexis (or stacked.Lexis) objects*

Description

Modify a Lexis object.

Usage

```
## S3 method for class 'Lexis'
transform( `_data`, ...)
## S3 method for class 'Lexis'
Relevel( x, states, print = TRUE, ... )
## S3 method for class 'Lexis'
levels( x )
## S3 method for class 'Lexis'
factorize( x, states, print = TRUE, ... )
## S3 method for class 'stacked.Lexis'
transform( `_data`, ...)
```

Arguments

_data	an object of class `Lexis`.
x	an object of class `Lexis`.
states	Names of the factor levels (states) for `lex.Cst` and `lex.Xst`. Can be a list, in which case some levels are collapsed, see the documentation for [Relevel](#). No sanity check for the latter operation is undertaken.
print	Should a conversion between old and new levels be printed?
...	Additional arguments to be passed to [transform.data.frame](#) or [Relevel](#).

Details

The transform method for `Lexis` objects works exactly as the method for data frames. `factorize` transforms the variables `lex.Cst` and `lex.Xst` to factors with identical set of levels, optionally with names given in `states`, and optionally collapsing states. `Relevel` is merely an alias for `factorize`, since the function does the same as [Relevel](#), but for both the factors `lex.Cst` and `lex.Xst`. A default sideeffect is to produce a table of old states versus new states if `states` is a list.

If `states` is `NULL`, as when for example the argument is not passed to the function, the returned object have levels of `lex.Cst`, `lex.Xst` (and for `stacked.Lexis` objects `lex.Tr`) shaved down to the actually occurring values.

Value

A transformed `Lexis` object.

The function `levels` returns the names of the states (levels of the factors `lex.Cst` and `lex.Xst`.

Author(s)

Martyn Plummer, Bendix Carstensen

See Also

[Lexis](), [merge.Lexis](), [subset.Lexis](), [subset.stacked.Lexis](), [Relevel]()

Examples

```
data( nickel )
nic <- Lexis( data = nickel,
                 id = id,
              entry = list(age=agein),
               exit = list(age=ageout,cal=ageout+dob,tfh=ageout-age1st),
    ## Lung cancer deaths are coded 2 and other deaths are coded 1
        exit.status = ( (icd > 0) + (icd %in% c(162,163)) ) )
str( nic )
levels( nic )
nit <- transform( nic, cumex = exposure*(agein-age1st) )
str( nit )
## It is still a Lexis object!
summary( nic )
nix <- factorize.Lexis( nic, c("Alive","Lung","Dead"))
niw <- factorize.Lexis( nix, c("Alive","Pulm","Mort"))
niz <- factorize.Lexis( niw, states=list("Alive",c("Pulm","Mort")), coll=" \n& ")
boxes( niw, boxpos=TRUE )
par( new=TRUE )
boxes( niz, boxpos=TRUE )
siw <- stack( niw )
str( siw )
```

---

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 asymtotic 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
```

Y.dk   *Population risk time in Denmark*

Description

Risk time (person-years) in the Danish population, classified by sex, age, period and date of birth in 1-year classes. This corresponds to triangles in a Lexis diagram.

Usage

```
data(Y.dk)
```

Format

A data frame with 13860 observations on the following 6 variables.

sex Sex. 1:males, 2:females

A One-year age class

P Period

C Birth cohort

Y Person-years

upper Indicator of upper triangle in the Lexis diagram

Details

The risk time is computed from the population size figures in N.dk, using the formulae devised in
B. Carstensen: "Demography and epidemiology: Age-Period-Cohort models in the computer age",
http://biostat.ku.dk/reports/2006/ResearchReport06-1.pdf/, later published as: B. Carstensen:
Age-period-cohort models for the Lexis diagram. Statistics in Medicine, 10; 26(15):3018-45, 2007.

Source

http://www.statistikbanken.dk/statbank5a/SelectTable/omrade0.asp?SubjectCode=02&PLanguage=
1&ShowNews=OFF

Examples

```
data(Y.dk)
str(Y.dk)
# Compute mean age, period for the triangles
attach( Y.dk )
age <- A + (1+upper)/3
per <- P + (2-upper)/3
# Plot a Lexis diagram
library( Epi )
Lexis.diagram( age=c(0,10), date=c(1990,2000), coh.grid=TRUE, int=1 )
box()
# Print the person-years for males there
text( per[sex==1], age[sex==1],
      formatC( Y[sex==1]/1000, format="f", digits=1 ) )
```

Index