Package ‘kyotil’

August 1, 2024

LazyLoad yes
LazyData yes
Version 2024.7-31
Title Utility Functions for Statistical Analysis Report Generation and Monte Carlo Studies
Depends R (>= 3.6)
Imports methods
Suggests RUnit, R.rsp, lme4, nlme, xtable, MASS, splines, survival, abind, pracma, VGAM, copula, mvtnorm, Hmisc, RColorBrewer, zoo, doParallel, Exact, survey, magick
Description Helper functions for creating formatted summary of regression models, writing publication-ready tables to latex files, and running Monte Carlo experiments.
VignetteBuilder R.rsp
License GPL (>= 2)
NeedsCompilation yes
Author Youyi Fong [cre], Krisztian Sebestyen [aut], Han Sunwoo [aut], Jason Becker [ctb], Bendix Carstensen [ctb], Daryl Morris [ctb], Josh Pasek [ctb], Dennis Chao [ctb], Andri Signorell [ctb], Sue Li [ctb], Jonathan Bartlett [ctb], Christophe Dutang [ctb]
Maintainer Youyi Fong <youyifong@gmail.com>
Repository CRAN
Date/Publication 2024-08-01 10:40:02 UTC
Description

Calculate age, by Jason P Becker, modified very slightly in how arguments are passed to the function.

Usage

age_calc(dob, enddate = Sys.Date(), units = c("days", "months", "years"), precise = TRUE)
Arguments

dob POSIXlt or Date. Birthday
enddate POSIXlt or Date. Date to compute age
units string. Choose a unit.
precise Boolean.

Author(s)
Jason P Becker

References

Examples

age_calc (dob=strptime("29OCT2002", format="%d%b%Y"),
enddate=strptime("30OCT2003", format="%d%b%Y"), units='years', precise=TRUE)
age_calc (dob=strptime("29OCT2002", format="%d%b%Y"),
enddate=strptime("30DEC2003", format="%d%b%Y"), units='years', precise=FALSE)

Description
AUC methods.

Usage

## S3 method for class 'auc'
coef(object, ...)
## S3 method for class 'auc'
predict(object, newdata, case.percentage = NULL, ...)
## S3 method for class 'auc'
print(x, ...)
## S3 method for class 'auc'
summary(object, ...)
## S3 method for class 'auc'
trainauc(fit, training.data = NULL, ...)
## S3 method for class 'auc'
ratio(fit)

## S3 method for class 'glm'
trainauc(fit, training.data = NULL, ...)
## S3 method for class 'glm'
ratio(fit)
 Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fit</code></td>
<td>an object that inherits from class 'auc' such as 'rauc' or 'sauc'</td>
</tr>
<tr>
<td><code>object</code></td>
<td>an object that inherits from class 'auc' such as 'rauc' or 'sauc'</td>
</tr>
<tr>
<td><code>x</code></td>
<td>an object that inherits from class 'auc' such as <code>rauc</code>, <code>sauc</code> or <code>sauc.dca</code>.</td>
</tr>
<tr>
<td><code>newdata</code></td>
<td>data at which to predict</td>
</tr>
<tr>
<td><code>case.percentage</code></td>
<td>used for class prediction, defaults to NULL</td>
</tr>
<tr>
<td><code>training.data</code></td>
<td>data frame used to compute auc based on a fit obtained by a call to <code>auc</code>, <code>sauc</code> or <code>sauc.dca</code></td>
</tr>
<tr>
<td><code>...</code></td>
<td>arguments passed to or from methods</td>
</tr>
</tbody>
</table>

Author(s)

Youyi Fong <youyifong@gmail.com>
Krisztian Sebestyen <>

---

Description

cbinduneven binds together a list of matrixes/dataframes of different lengths, rows are matched by names. `binary` returns binary representation of an integer. `binary2` returns binary representation of an integer with leading 0, the length of string is n. `mysystem` can call any exe file that is in the PATH.

Usage

```r
mytable (..., exclude = if (useNA == "no") c(NA, NaN), useNA = "ifany",
          dnn = list.names(...), deparse.level = 1)

cbinduneven(li)

binary(i)

multi.outer (f, ...)

myreshapelong(dat, cols.to.be.stacked, label.cols.to.be.stacked, new.col.name)

binary2(i, n)

f2c(f)

ftoi(f)
```
keepWarnings(expr)

meanmed(x, na.rm = FALSE)

myaggregate(x, by, FUN, new.col.name = "aggregate.value", ...)

myreshapewide(formula, dat, idvar, keep.extra.col=FALSE)

mysapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE, ret.mat = TRUE)

myscale(x)

mysystem(cmd, ...)

mytapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)

read.sv(file, header = TRUE, ...)

read.tsv(file, header = TRUE, sep = "\t", ...)

table.prop(x,y=NULL,digit=1,style=2,whole.table.add.to.1=FALSE,useNA="ifany", 
   add.perc=FALSE, add.total.column = FALSE)

table.cases (case,group,include.all=TRUE,desc="cases")

table.cases.3(case,group1,group2)

unix()

mycor (x, use = "everything", method = c("pearson", "kendall", "spearman"),
   alternative = c("two.sided", "less", "greater"), exact = NULL,
   conf.level = 0.95, continuity = FALSE,
   digits.coef=2, digits.pval=3, 
   ...)

Arguments

   exclude                 exclude
   dnn                     dnn
   deparse.level           deparse.level
   add.total.column  
   use                     tbdi
   method                  tbdi
   alternative             tbdi
exact tbdi
conf.level tbdi
continuity tbdi
digits.coef tbdi
digits.pval tbdi
cols.to.be.stacked tbdi
label.cols.to.be.stacked tbdi
li a list
i tbdi
n tbdn
f In multi.out, f is a function.
case vector of 0/1
group vector of multi-group indicators
formula a formula object.
expr tbdeexpr
x tbdx
na.rm tbdna.rm
desc tbdby
by tbdby
whole.table.add.to.1 Boolean
new.col.name tbdnew.col.name
... tbd...
dat tbddat
idvar tbdidvar
X tbdX
simplify tbdsimplify
USE.NAMES tbdUSE.NAMES
ret.mat tbdret.mat
cmd tbdcmd
INDEX tbdINDEX
file tbdfile
header tbdheader
sep tbdsep
y tbdy
digit tbdigit
Examples

```r
binary(5) ### 101
binary2(5, 4)
```

```r
a=data.frame("x"=1:2)
b=data.frame("y"=3:5);#rownames(b)[3]="
cbinduneven(list(a,b))
```

```r
## Not run:
# the formula in myreshapewide can only have one variable in the right hand side
myreshapewide(fi~week, dat, c("ptid","stim"))

myreshapelong(dat.201.neut, cols.to.be.stacked=c("MN.3","SF162","SVA.MLV"),
label.cols.to.be.stacked="antigen", new.col.name="y")

myaggregate(subset(dat.poc, select=c(HIV, trt)), list(dat.poc$f), function(x)
  with(x, c(fisher.test(HIV, trt)$estimate, fisher.test(HIV, trt)$p.value))

## End(Not run)
```

---

**binaryloess**

*Using loess to Check Functional Form for Logistic Regression*

**Description**

This function plots a smoothed line of how the average value of Y changes with X in order to check functional form for logistic regression.

**Usage**

```r
binaryloess(x, y, scale = c("logit", "linear"), span = 0.7, weights = NULL, ...)
```
Arguments

- `x` `tbdx`
- `y` `tbdy`
- `scale` `tbdscale`
- `span` smoothing parameter, passed to loess. If less than 1, the neighbourhood includes proportion `a` of the points. If greater than 1, all points are used, with the maximum distance assumed to be `a^(1/p)` times the actual maximum distance for `p` explanatory variables.
  Missing records are removed first.
- `weights` sampling weights, passed to loess
- `...` passed to plotting function

Details


Examples

```r
set.seed(1234)
n <- 1000
x <- rnorm(n)
xb <- -2+x
pr <- exp(xb)/(1+exp(xb))
y=rbern(n, pr)

par(mfrow=c(1,2))
binaryloess(x, y, scale = "logit", span = 0.7, weights = NULL, ylab="logit(p)")
binaryloess(x, y, scale = "linear", span = 0.7, weights = NULL, ylab="prob")
```

`cox.zph.2` Test the Proportional Hazards Assumption of a Cox Regression (a slightly modified version)

Description

A slightly modified test of the proportional hazards assumption for a Cox regression model fit (coxph). This version corrects some conservativeness of the test.

Usage

```r
cox.zph.2(fit, transform = "km", global = TRUE, exact=TRUE)
```
Cross Validation Functions

Description

Cross validation utility functions.

Usage

    sample.for.cv (dat, v, seed)
    get.kfold.splits (dat, k, seed)
    kfold.split (k, n1, n0)
    ran.kfold.split(k, n1, n0, replicates)
    lpo.split(n1, n0)
    get.splits (dat, cv.scheme=c("LPO","5fold","50xrandom4:1"), seed)
Arguments
dat  a data frame. One of the columns must be named y and y should be 0/1 with 1 for case and 0 for control
v    v-fold cross validation
seed seed for random number generators
k    var.equal
n1   var.equal
n0   var.equal
replicates var.equal
cv.scheme var.equal

Details
sample.for.cv: case and controls are sampled separately.

Value
sample.for.cv returns a list of two vector of integers: train and test, which refer to the rows of dat

Deming  Fit Deming regression.

Description
Deming regression fit. Assume x and y variances are the same. Slightly modified from MethComp R package.

Usage
Deming(x, y, vr = sdr^2, sdr = sqrt(vr), boot = TRUE, keep.boot = FALSE, alpha = 0.05)

Arguments
x     tbd
y     tbdy
vr    tbdvr
sdr   tbdstdr
boot  tbdboot
keep.boot tbdkeep.boot
alpha tbdalpha
Examples

```r
## Not run:
set.seed(1)
x=rnorm(100,0,1)
y=x+rnorm(100,0,.5)
x=x+rnorm(100,0,.5)
fit=Deming(x,y, boot=TRUE)
summary(fit)
plot(x,y)
abline(fit)
# compare with lm fit
fit.1=lm(y~x, data.frame(x,y))
summary(fit.1)
abline(fit.1, col=2)
## End(Not run)
```

### DMHeatMap

**Better Heatmap Function**

**Description**

Makes a heatmap representation of correlation coefficients easier.

**Usage**

```r
DMHeatMap(x, Rowv = TRUE, Colv = if (symm) "Rowv" else TRUE, 
distfun = dist, hclustfun = hclust, dendrogram = 
c("both", "row", "column", "none"), symm = FALSE, 
scale = c("none", "row", "column"), na.rm = TRUE, revC = 
identical(Colv, "Rowv"), add.expr, breaks, symbreaks = 
min(x < 0, na.rm = TRUE) || scale != "none", col = 
"heat.colors", colsep, rowsep, sepcolor = "white", 
sepwidth = c(0.05, 0.05), cellnote, notecex = 1, 
notecol = "cyan", na.color = par("bg"), trace = 
c("column", "row", "both", "none"), tracecol = "cyan", 
hline = median(breaks), vline = median(breaks), 
linecol = tracecol, margins = c(5, 5), ColSideColors, 
RowSideColors, cexRow = 0.2 + 1/log10(nr), cexCol = 
0.2 + 1/log10(nc), labRow = NULL, labCol = NULL, 
labColor = NULL, axis = TRUE, heatmapOnly = FALSE, key = 
TRUE, keysizew = 1.5, density.info = c("histogram",  
"density", "none"), denscol = tracecol, symkey = min(x 
< 0, na.rm = TRUE) || symbreaks, densadj = 0.25, main = 
NULL, xlab = NULL, ylab = NULL, lmat = NULL, lhei = 
NULL, lwid = NULL, lower.left.only = TRUE, legend = 
TRUE, legend.x = "topright", verbose = FALSE, ...)```
Arguments

- `x`: tbd
- `axis`: tbd
- `heatmapOnly`: tbd
- `verbose`: tbd
- `legend.x`: tbd
- `legend`: tbd
- `Rowv`: tbdRowv
- `Colv`: tbdColv
- `distfun`: tbddistfun
- `hclustfun`: tbdhclustfun
- `dendrogram`: tbddendrogram
- `symm`: tbdsymm
- `scale`: tbdscale
- `na.rm`: tbdna.rm
- `revC`: tbdrevC
- `add.expr`: tbdadd.expr
- `breaks`: tbdbreaks
- `symbreaks`: tbdbreaks
- `col`: tbdcol
- `colsep`: tbdcolsep
- `rowsep`: tbdrowsep
- `sepcolor`: tbdsepcolor
- `sepwidth`: tbdsepwidth
- `cellnote`: tbdcellnote
- `notecex`: tbdnotecex
- `notecol`: tbdnotecol
- `na.color`: tbdna.color
- `trace`: tbdtrace
- `tracecol`: tbdtracecol
- `hline`: tbdhline
- `vline`: tbdvline
- `linecol`: tbdlinecol
- `margins`: tbdmargins
- `ColSideColors`: tbdColSideColors
- `RowSideColors`: tbdRowSideColors
- `cexRow`: tbdcexRow
get.sim.res

Examples

cor = matrix(runif(15), 5, 3)
breaks = c(-1, -0.7, -0.5, -0.3, -0.1, 0.1, 0.3, 0.5, 0.7, 1)
hU = DMMHeatMap(cor, trace = "none", symm = FALSE, dendrogram = "none", col = RColorBrewer::brewer.pal(length(breaks) - 1, "RdYlGn"), distfun = function(c) as.dist(1 - c), cexRow = 1.5, cexCol = 1.5, lmat = rbind(c(2, 1), c(4, 3)), lhei = c(4, 1), breaks = breaks, margins = c(2, 2), key = FALSE, Rowv = NA, lower.left.only = FALSE)

Description

Go through a folder and read all files and combine the results into a multidimensional array.
get.sim.res (dir, res.name="res", verbose=TRUE)
MCsummary (dir, res.name = "res", exclude.some = TRUE,
          exclude.col = 1, verbose = TRUE)
getFormattedMCSummary (path, sim, nn, fit.method, exclude.some = TRUE,
                       exclude.col = 1, verbose = TRUE, coef.0 = NULL, digit1
                       = 2, sum.est = c("mean", "median"), sum.sd =
                       c("median", "mean"), style = 1, keep.intercept =
                       FALSE)

Arguments

dir        directory of MC result files
path       partial path to the directory of MC result files
res.name   name of the R object saved in the files, default is res, but may be others
verbose    Boolean
sim        a string to denote simulation setting
nn         a vector of sample sizes
fit.method a string to denote fitting method. sim, nn and fit.method together forms the name
            of the directory containing MC result files
exclude.col column number
exclude.some whether to exclude MC results that are extreme
coef.0      simulation truth
digit1     digits
sum.est    use mean or median as location estimate summary
sum.sd     use mean or median as sd estimate summary
style      integer
keep.intercept whether to include intercept in the table

Details

Depending on package abind to combine arrays from files.

Value

A multidimensional array.
getK calculates the kernel matrix between X and itself and returns a n by n matrix. Alternatively, it calculates the kernel matrix between X and X2 and returns a n by n2 matrix.

Usage

getK (X,kernel,para=NULL,X2=NULL,C = NULL)

Arguments

X
covariate matrix with dimension n by d. Note this is not the paired difference of covariate matrix.

kernel
string specifying type of kernel: polynomial or p (1 + <x,y>)^para, rbf or r exp(-para*||x-y||^2), linear or l <x,y>, ibs or i 0.5*mean(2.0 - |x-y|) or sum(w*(2.0 - |x-y|))/sum(w), with x[i],y[i] in \{0,1,2\} and weights 'w' given in 'para'. hamming or h for sum(x == y) with x[i],y[i] binary, no default.

para
parameter of the kernel function. for ibs or hamming, para can be a vector of weights.

X2
optional second covariate matrix with dimension n2 by d

C
logical. If TRUE, kernels are computed by custom routines in C, which may be more memory efficient, and faster too for ibs and hamming kernels.

Details

IBS stands for 'Identical By State'. If 'x','y' are in in \{0,1,2\} then
IBS(x,y) = 0 if |x-y|=2, 1 if |x-y|=1, 2 if |x-y|=0, or IBS(x,y) = 2.0 - |x-y|.
K(u,v) = sum(IBS(u[i],v[i])) / 2K where K = length(u).
The 'hamming' kernel is the equivalent of the 'ibs' kernel for binary data. Note that 'hamming' kernel is based on hamming similarity(!), not on dissimilarity distance.
Within in the code, C is default to TRUE for ibs and hamming kernels and FALSE otherwise.

Value

A kernel matrix.

Author(s)

Youyi Fong <youyifong@gmail.com>
Krisztian Sebestyen <ksebestyen@gmail.com>
Shuxin Yin <>
Examples

```r
X = cbind(x1=rnorm(n=5), x2=rnorm(n=5))
dim(X)
X2 = cbind(x1=rnorm(n=3), x2=rnorm(n=3))
dim(X2)

K = getK(X,"linear")
dim(K)

K = getK(X,"linear",X2=X2)
dim(K)
K1 = getK(X2,"1",X2=X)
dim(K1)
all(K==t(K1))
```

```r
# RBF kernel
K = getK(X,"rbf",para=1,X2=X2)
K1 = getK(X2,"r",para=1,X2=X)
all(K==t(K1))
```

```r
# IBS kernel for ternary data
X <- as.matrix(expand.grid(0:2,0:2))
K = getK(X,kernel = 'ibs')

# add weight
w = runif(ncol(X))
K = getK(X,kernel = 'ibs',para = w)
```

```r
# IBS kernel for binary data via option 'h' for 'hamming similarity measure'
X <- as.matrix(expand.grid(0:1,0:1))
K=getK(X,kernel = 'h')
```

---

**get_count_from_xy_coor**

*Imaging analysis for spatial region*

---

**Description**

Counting the number of masks in a rectangular region

**Usage**

```r
get_count_from_xy_coor(file, topleft, bottomright, image, plot)
```
**iorw**

### Arguments

- **file**  
  _sizes_coordinates.txt

- **topleft**  
  topleft (x,y) coordinate for a rectangular box

- **bottomright**  
  bottomright: bottomright (x,y) coordinate for a rectangular box

- **image**  
  image: an image for plotting

- **plot**  
  plot: plot=TRUE shows image with rectangular box

### Details

This function counts cells inside of rectangular box made by the topleft and bottomright xy-coordinates.

### Value

The number of masks inside of the rectangular box

### Author(s)

Sunwoo Han

### Examples

```r
# get_count_from_xy_coor(file='M926910_Position1_CD3-BUV395_sizes_coordinates.txt',
# topleft=c(500,0), bottomright=c(1392,500),
# image='M926910_Position1_CD3-BUV395.tiff', plot=TRUE)
```

---

**iorw**  

*Causal Mediation Analysis of Cowling et al.*

### Description

Estimate the total, direct, and indirect effects using IORW method (inverse odds ratio weighting) and compute 95

### Usage

```r
iorw(formula.effect, formula.mediators, data, family =
    NULL, nboot = 10000, numCores = 1, save.steps = FALSE,
    verbose = FALSE)
```

```r
## S3 method for class 'iorw'
print(x, ...)
```
Arguments

**formula.effect**  
a formula object for the total and direct effect regression. The first term on the right is assumed to be the binary treatment/exposure variable.

**formula.mediators**  
a formula object for logistic regression. It should be of the form: ~ mediation marker1 + mediation marker2.

**data**  
a data frame.

**family**  
if Cox regression, leave as NULL; otherwise, it will be passed to glm().

**nboot**  
an integer. Number of bootstrap replicates.

**numCores**  
an integer. Number of cores to use for parallel processing.

**save.steps**  
boolean. Whether or not to save the fits from the three steps and the weights.

**x**  
Object of type iorw

**verbose**  
boolean.

...  
Additional arguments passed to the print function.

Details

Code by Cowling and Lim was downloaded from https://datadryad.org/stash/dataset/doi:10.5061/dryad.cv37539

If a bootstrap replicate generates warnings during regression, NA will be returned for that replicate.

The number of such occurrences is recorded in an attribute of boot.perc in the return value.

It does not handle sampling weights yet.

Value

Point estimates and percentile bootstrap confidence intervals.

Author(s)

Youyi Fong, based on code by Cowling and Lim

References


Examples

##### Cox regression

# without adjusting for baseline markers
library(survival)
formula.effect=Surv(surv_time, flu)=vaccine+age
formula.mediators=~log2(postvax.B.Brisbane/5)
res.1=iorw(formula.effect, formula.mediators, kid, nboot=10, numCores=1); res.1
stopifnot(max(abs(res.1$boot[,1] - c(0.2029779,0.6070105,0.3039110,0.4283389,0.2124268))<1e-6)

# adjust for baseline markers
formula.effect=Surv(surv_time, flu)=vaccine+log2(prevax.B.Brisbane)+age
formula.mediators=~log2(postvax.B.Brisbane/5)
res.2=iorw(formula.effect, formula.mediators, kid, nboot=10, numCores=1); res.2

##### Logistic regression

# without adjusting for baseline markers
formula.effect=flu=vaccine+age
formula.mediators=~log2(postvax.B.Brisbane/5)
res.3=iorw(formula.effect, formula.mediators, kid, family=binomial(), nboot=10, numCores=1); res.3
stopifnot(max(abs(res.3$boot[,1] - c(0.1960024,0.6154349,0.2937164,0.4145470,0.2168644))<1e-6)

# adjust for baseline markers
formula.effect=flu=vaccine+log2(prevax.B.Brisbane)+age
formula.mediators=~log2(postvax.B.Brisbane/5)
res.4=iorw(formula.effect, formula.mediators, kid, family=binomial(), nboot=10, numCores=1); res.4

---

kid

Dataset from Cowling et al.

---

Description

Influenza immune response biomarkers dataset.

Usage

data("kid")
Format

A data frame with 736 observations on the following 10 variables.

hhID   a numeric vector
age    a numeric vector
intervention a character vector
vaccine a numeric vector
vaccine.date a Date
postvax.date a Date
prevax.B.Brisbane a numeric vector
postvax.B.Brisbane a numeric vector
surv_time a numeric vector
flu     a numeric vector

References


Description

Utility functions by Youyi Fong and Krisz Sebestyen, and some functions copied from other packages for convenience (acknowledged on their manual pages).

Most useful functions: mypostscript/mypdf, mytex,

See the Index link below for a list of available functions.

The package depends on Hmisc. The main reason for that, besides the usefulness of the package, is Hmisc depends on ggplot2, which also define
make.timedep.dataset

Create Dataset for Time-dependent Covariate Proportional Hazard Model Analysis

Description

Returns a data frame that is suitable for time-dependent covariate Cox model fit.

Usage

make.timedep.dataset(dat, X, d, baseline.ageyrs, t.1, t.2 = NULL)

Arguments

- **dat**: data frame
- **X**: string. Name of the followup time column in dat. Unit needs to be years.
- **d**: string. Name of the followup time column in dat.
- **baseline.ageyrs**: string. Name of the followup time column in dat.
- **t.1**: numerical. Cutoff for age group
- **t.2**: numerical. Second cutoff for age group

Details

The function assumes that the followup length is such that only one change of age group is possible.

Value

Returns a data frame with the following columns added: tstart, tstop, .timedep.agegrp, .baseline.agegrp

- **tstart**: left bound of time interval
- **tstop**: right bound of time interval
- **.timedep.agegrp**: time-dependent age group
- **.baseline.agegrp**: baseline age group

Author(s)

Youyi Fong

References

Therneau, T. and Crowson, C. Using Time Dependent Covariates and Time Dependent Coefficients in the Cox Model. A vignette from the R package survival.
**Examples**

```r
library(survival)

n=3000; followup.length=5; incidence.density=0.015; age.sim="continuous"

dat.0=sim.dat.tvarying.two(n, followup.length, incidence.density, age.sim, seed=1)
dat=subset(dat.0, for.non.tvarying.ana, select=c(ptid, X, d, baseline.age, trt))
dat.timedep = make.timedep.dataset (dat, "X", "d", "baseline.age", 6)
coxph(Surv(tstart,tstop,d) ~ trt*.timedep.agegrp, dat.timedep)
```

---

**math.functions**

**Math Functions**

**Description**

\( H \) calculates entropy.

**Usage**

- `as.binary(n, base = 2, r = FALSE)`
- `binom.coef(n, m)`
- `expit(x)`
- `logDiffExp(logx1, logx2)`
- `logit(x)`
- `logMeanExp(logx, B = NULL)`
- `logSumExp(logx)`
- `logSumExpFor2(logx, logy)`
- `permn(x, fun = NULL, ...)`
- `Stirling2(n, m)`
- `interpolate(pt1, pt2, x)`

**Arguments**

- `n` : tbdn
- `base` : tbdbase
**Examples**

\[ H(\text{rep}(1/5, 5)) \]
\[ H(\text{rep}(3, 5)) \]

---

**Description**

concatList returns a string that concatenates the elements of the input list or array.

**Usage**

\[ \text{AR1}(p, w) \]
\[ \text{concatList}(\text{lis}, \text{sep} = ")") \]
\[ \text{EXCH}(p, \rho) \]
\[ \text{fill.jagged.array}(a) \]
\[ \text{getMidPoints}(x) \]
\[ \text{getUpperRight}(\text{mat}, \text{func} = \text{NULL}) \]
\[ \text{last}(x, n = 1, \ldots) \]
\[ \text{mix}(a, b) \]
## S3 method for class 'data.frame'
rep(x, times = 1, ...)

## S3 method for class 'matrix'
rep(x, times = 1, each = 1, by.row = TRUE, ...)

## S3 method for class 'matrix.block'
rep(x, times = 2, ...)

shift.left(x, k = 1)

shift.right(x, k = 1)

thin.rows(dat, thin.factor = 10)

ThinRows(dat, thin.factor = 10)

tr(m)

### Arguments

- `p`
- `tbdp`
- `w`
- `tbdw`
- `lis`
- `tbdlis`  
  list or array
- `sep`
- `tbdsep`
- `rho`
- `tbdrho`
- `a`
- `tbdalpha`
- `x`
- `tbdx`
- `matri`
- `tbdmatri`  
  matrix
- `func`
- `tbdfunc`  
  function
- `n`
- `tbdn`
- `...`
- `tbd...`
- `b`
- `tbdbeta`
- `times`
- `tbdtimes`
- `each`
- `tbdeach`
- `by.row`
- `tbdby.row`
- `k`
- `tbdk`
- `dat`
- `tbdx`  
  data
- `thin.factor`
- `tbdthin.factor`  
  thin
- `m`
- `tbdm`

### Examples

```
concatList(1:3,”_”)
```
Matrix Functions that May Be Faster than

Description

DXD computes $D \times X \times D$, where $D$ is a diagonal matrix. tXDX computes $t(X) \times D \times X$. symprod computes $S \times X$ for symmetric $S$. txSy computes $t(x) \times S \times y$ for symmetric $S$.

Usage

DXD(d1, X, d2)
tXDX(X, D)
symprod(S, X)
txSy(x, S, y)
.as.double(x, stripAttributes = FALSE)

Arguments

d1 a diagonal matrix or an array
d2 a diagonal matrix or an array
x array
y array
S symmetric matrix
X matrix
D matrix
stripAttributes boolean

Details

.as.double does not copying whereas as.double(x) for older versions of R when using .C(DUP = FALSE) make duplicate copy of x. In addition, even if x is a 'double', since x has attributes (dim(x)) as.double(x) duplicates.

The functions do not check whether $S$ is symmetric. If it is not symmetric, then the result will be wrong. DXD offers a big gain, while symprod and txSy gains are more incremental.

Author(s)

Krisztian Sebestyen
Examples

d1=1:3
d2=4:6
X=matrix(1:9,3,3)
all(DDX(d1, X, d2) == diag(d1) %*% X %*% diag(d2))

S=matrix(c(1,2,3,2,4,5,3,5,8),3,3)
X=matrix(1:9,3,3)
all( symprod(S, X) == S %*% X )

x=1:3
y=4:6
S=matrix(c(1,2,3,2,4,5,3,5,8),3,3)
txSy(x, S, y) == drop(t(x) %*% S %*% y)

---

Misc Functions

Description

Misc functions. summ computes iterative sum, sort of like diff.

Usage

pava (x, wt = rep(1, length(x)))
summ(x)
empty2na(x)
## S3 method for class 'pcc'
predict(object, newdat, ...)
rank.inv.norm(x)
INT(x)
derc_to_binary (x,d)

Arguments

x tbdx
d number of digits in the returned binary representation, including leading 0’s
wt tbdvar.equal
object tbdvar.equal
newdat tbdvar.equal
... tbdvar.equal
Details

rank.inv.norm: rank-based inverse normal/gaussian transformation
dec_to_binary covert a decimal number to a binary representation with d digits

Value

summ returns

---

### p.adj.perm

**Permutation-based Multitesting P Values Adjustment**

**Description**

An implementation of Westfall and Young

**Usage**

```r
p.adj.perm(p.unadj, p.perms, alpha = 0.05)
```

**Arguments**

- `p.unadj`: p.unadj
- `p.perms`: p.perms
- `alpha`: alpha

**Details**

This implementation is not as fast as the implementation from the package multtest. But usually the step to create p.perms is the rate-limiting step.

The smallest of the Westfall and Young FWER-controlling multitesting adjusted p values coincides with the p value for testing a global null without any assumptions. But for the multitesting adjustment to hold, it requires the subset pivotality condition.

**Author(s)**

Sue Li, sli@fredhutch.org

**References**

Plotting Functions

Description

mypostscript and mypdf sets the width and height based on mfrow input.

Usage

smoothed.scaled.hist (dat.ls, bin_width, scale.factors=NULL, cols=NULL, legend=NULL, cex.legend=1, ...)

myplot (object, ...)

## S3 method for class 'loess'
myplot(object, xlab="x", ylab="fitted", ...)

whiskers (x, s, ...)

abline.pt.slope(pt1, slope, x2=NULL, ...)

abline.pts(pt1, pt2 = NULL)

butterfly.plot(dat, dat2 = NULL, add = FALSE, xaxislabels = rep("", 4), x.ori = 0, xlab = ", ylab = ", cex.axis = 1, ...)

eempty.plot()

add.mtext.label (text, cex = 1.4, adj = -0.2)

mydev.off(file = "temp", ext = c("pdf"), res = 200, mydev = NULL, silent = TRUE)

getMfrow(len)

myhist (x, add.norm=TRUE, col.norm="blue", ...)

myforestplot(dat, xlim = NULL, xlab = ", main = ", col.1 = "red", col.2 = "blue", plot.labels = TRUE, order = FALSE, decreasing = FALSE, vline = TRUE, cols = NULL, log = ", null.val = NULL)

my.interaction.plot(dat, x.ori = 0, xaxislabels = rep("", 2), cex.axis = 1, add = FALSE, xlab = ", ylab = ", pcol = NULL, lcol = NULL, ...)

myboxplot(object, ...)

## S3 method for class 'formula'
myboxplot(formula, data, cex = 0.5, xlab = "", ylab = "", main = ", box = TRUE, at = NULL, na.action = NULL, p.val = NULL, pch = 1, col = 1, test = ", friedman.test.formula = NULL, reshape.formula = NULL, reshape.id = NULL, jitter = TRUE, add.interaction = FALSE, drop.unused.levels = TRUE, bg.pt = NULL, add = FALSE, seed = 1, write.p.at.top = FALSE, ...)## S3 method for class 'data.frame'
myboxplot(object, cex = 0.5, ylab = "", xlab = "", main = ", box = TRUE, at = NULL, pch = 1, col = 1, test = ", paired = FALSE, ...)## S3 method for class 'list'
myboxplot(object, paired = FALSE, ...)abline.shade.2(x, col=c(0,1,0))abline.shade(pt, type = 5, col = c(0, 1, 0), alpha = 0.3)mylegend(legend, x, y=NULL, lty = NULL, bty = "n", ...)mymatplot(x, y, type = "b", lty = c(1, 2, 1, 2, 1, 2), pch = NULL, col = rep(c("darkgray", "black"), each = 3), xlab = NULL, ylab = "", draw.x.axis = TRUE, bg = NA, lwd = 1, at = NULL, make.legend = TRUE, legend = NULL, impute.missing.for.line = TRUE, legend.x = 9, legend.title = NULL, legend.cex = 1, legend.lty = lty, legend.inset = 0, xaxt = "s", y.intersp = 1.5, x.intersp = 0.3, text.width = NULL, add = FALSE, ...)mypairs(dat, ladder = FALSE, show.data.cloud = TRUE, ladder.add.line = T, ladder.add.text = T, ...)wtd.hist (x, breaks = "Sturges", freq = NULL, probability = !freq, include.lowest = TRUE, right = TRUE, density = NULL, angle = 45, col = NULL, border = NULL, main = paste("Histogram of", xname), xlim = range(breaks), ylim = NULL, xlab = xname, ylab, axes = TRUE, plot = TRUE, labels = FALSE, nclass = NULL, weight = NULL, ...)mylines(x, y, type = "l", ...)myfigure(mfrow = c(1, 1), mfcol = NULL, width = NULL, height = NULL, oma = NULL, mar = NULL, main.outer = FALSE, bg=NULL)
mypdf(....)

mypng(....)
mytiff(....)

mypostscript(file = "temp", mfrow = c(1, 1), mfcol = NULL, width = NULL, height = NULL, ext = c("eps", "pdf", "png", "tiff"), oma = NULL, mar = NULL, main.outer = FALSE, save2file = TRUE, res = 200, silent = TRUE, ....)

panel.cor(x, y, digits = 2, prefix = ",", cex.cor, cor., leading0 = FALSE, cex.cor.dep = TRUE, ....)

panel.hist(x,....)

panel.nothing(x,....)

corplot(object,....)

## Default S3 method:
corplot(object, y,....)

## S3 method for class 'formula'
corplot(formula, data, main = ",", method = c("pearson", "spearman"), col = 1, cex = 0.5, add.diagonal.line = TRUE, add.lm.fit = FALSE, add.loess.fit = FALSE, col.lm = 2, add.deming.fit = FALSE, col.deming = 4, add = FALSE, log = ",", same.xylim = FALSE, xlim = NULL, ylim = NULL,....)

Arguments

dat.ls named list of vectors. A histogram is made for each vector.

bin_width width of bin for histograms

scale.factors named vector of scale factors to scale the histogram counts by

cex.legend cex for legend

silent tbdadd

legend.lty tbdadd
cex.cor.dep tbdadd

add.loess.fit tbdadd

leading0 tbdadd

null.val tbdadd

write.p.at.top tbdadd
text.width  tbdadd
text     tbdadd
cex      tbdcex
adj      tbdpt2
file     tbdfile
ext      tbdext
res      resolution.
add.norm  Boolean, whether to add normal approximation density line
col.norm  string, color of added normal density line
pt1      tbdpt1
s        tbdslope
ladder   tbdslope
slope    tbdslope
friedman.test.formula  tbdslope
reshape.id  tbdslope
impute.missing.for.line  tbdslope
cor.     tbdslope
mydev    tbdslope
jitter   Boolean
add.interaction  Boolean
...      tbd...
xaxt     tbdpt2
breaks   tbdpt2
freq     tbdpt2
bg.pt    tbdpt2
probability tbdpt2
include.lowest tbdpt2
right    tbdpt2
density  tbdpt2
angle    tbdpt2
border   tbdpt2
axes     tbdpt2
plot     tbdpt2
labels   tbdpt2
nclass   tbdpt2
weight  tbdpt2
pt2    tbdpt2
pt     tbdpt2
alpha  tbdpt2
dat    tbddat
lwd    line width.
x.intersp controls the look of legend.
y.intersp controls the look of legend.
legend.inset legend inset
dat2   tbddat2
add    tbdadd
log    log
add.lm.fit lm fit
add.deming.fit add
col.lm  col
col.deming col
reshape.formula a formula object.
xaxislabels tbdxaxislabels
x.ori   tbdx.ori
xlab    tbdxlab
ylab    tbdylab
cex.axis tbdcex.axis
len     tbdlen
same.xlim Boolean. Whether xlim and ylim should be the same
xlim    tbdxlim
ylim    tbdxlim
main    tbdmain
col.1   tbdcol.1
col.2   tbdcol.2
pcol    tbdpcol
lcol    tbdlcol
object  tbdobject
formula tbdformula
data    tbddata
box     tbdbox
at      tbdat
plotting

pch  tbdpch
col  tbdcol

**test** string. For example, "t","w","f","k", "tw"

legend  tbdlegend
x  tbdx
lty  tbdlty
bty  tbdbty
type  tbdtype
make.legend  tbdmake.legend
legend.x  tbdlegend.x
legend.title  tbdlegend.title
legend.cex  tbdlegend.cex
draw.x.axis  tbddraw.x.axis
bg  tbdbg

**method** tbdmethod
mfrow  tbdmfrow
mfcol  tbdmfcol
width  tbdwidth
height  tbdheight
oma  tbdoma
mar  tbdmar
main.outer  tbdmain.outer
save2file  tbdsave2file

**y** tbdy
digits  tbdigits
prefix  tbdprefix
cex.cor  cex.cor
plot.labels  Boolean
order  Boolean
decreasing  Boolean
add.diagonal.line  tbdadd.diagonal.line

x2  tbdadd.diagonal.line
vline  tbdadd.diagonal.line
cols  tbdadd.diagonal.line
na.action  tbdadd.diagonal.line
drop.unused.levels  tbdadd.diagonal.line
Details

myboxplot shows data points along with boxes. The data points are jittered and the pattern of jittering is made reproducible in repeated calls. The test can only take one type of test currently.

myforestplot is modified from code from Allan deCamp/SCHARP. dat should have three columns. first column should be point estimate, second and third ici and uci, fourth p value. col.1 is the color used for CIs that do not include null, col.2 is used for CIs that do include null. If order is TRUE, the rows are ordered by the first column of dat. decreasing can be used to change the behavior of order.

corplot.formula uses MethComp::Deming by Bendix Carstensen to fit Deming regression.

wtd.hist is copied from weights package, author: Josh Pasek.

mymatplot will use na.approx (zoo) to fill in NA before plotting in order to draw continuous lines. The filled-in values will not be shown as points.

smoothed.scaled.hist draws histograms and overlay densities on top.

Examples

```r
set.seed(1)
x=1:50+rnorm(50,0,4)
y=1:50+rnorm(50,0,4)

dat=data.frame(x, y)
corplot(y~x, dat, add.lm.fit=TRUE, add.deming.fit=TRUE, col.lm="red", col.deming="blue")

dat=data.frame(y=c(1:10,2:11), x=rep(c("a","b"),each=10), ptid=c(1:10,1:10))
par(mfrow=c(1,2))
myboxplot(y~x, dat, test="w", jitter=FALSE)
myboxplot(y~x, dat, test="f", add.interaction=TRUE, reshape.formula=y~x, reshape.id="ptid")

myboxplot(list(jitter(1:10), jitter(3:12)), test="w")
myboxplot(list(jitter(1:10), jitter(3:12)), test="w", paired=TRUE)

smoothed.scaled.hist(list(A=rnorm(100,0,1)), bin_width=0.1, xlab="x")
smoothed.scaled.hist(list(A=rnorm(100,0,1), B=rnorm(500,10,2)),
                      bin_width=0.1, xlab="x")
```
## Not run:
myfigure(mfrow=c(1,2))
plot(1:10)
plot(1:10)
mydev.off(ext="png,pdf", file="tmp")

## End(Not run)

#myboxplot x axis may look weird if log="xy"

---

**Print Functions**

### Description

roundup prints a specified number of digits after decimal point even if 0s are needed at the end. formatInt prints a specified number of digits before decimal point even if 0s are needed at the beginning.

### Usage

myprint(object, ...)

## Default S3 method:
myprint(..., newline = TRUE, digits = 3, print.name=TRUE)

## S3 method for class 'matrix'
myprint(object, ...)

formatInt(x, digits, fill = "0", ...)

prettyprint (value, digit=2)

make.latex.coef.table(models, model.names = NULL, row.major = FALSE, round.digits = NULL)

mysanitize.text(str)

mysanitize.numbers(x)

mytex(dat = NULL, file.name = "temp", digits = NULL, display = NULL, align = "r", include.rownames = TRUE, include.colnames = TRUE, col.headers = NULL, comment = FALSE, floating = FALSE, lines = TRUE, hline.after = NULL, add.to.row = NULL, sanitize.text.function = NULL, append = FALSE, preamble = "$", input.foldername = NULL, save2input.only = NULL, caption = NULL, label = NULL)
= paste("tab", last(strsplit(file.name, "/")[[1]]),
    sep = " " ), table.placement = "h!",
    add.clear.page.between.tables = FALSE, longtable =
    FALSE, verbose = FALSE, silent = TRUE, ...)

mytex.begin(file.name, preamble = "")

mytex.end(file.name)

mywrite(x, ...)

mywrite.csv(x, file = "tmp", row.names = FALSE, digits = NULL,
    silent = TRUE, ...)

roundup (value, digits, na.to.empty=TRUE, remove.leading0=TRUE)

formatDouble(value, digits, na.to.empty=TRUE, remove.leading0=TRUE)

Arguments

digit tbdigit
silent tbd newline
input.foldername tbd newline
object tbd newline
newline tbd newline
print.name tbd digits
save2input.only Boolean
include.colnames Boolean
col.headers string. Column headers
comment Boolean, whether to include the version and timestamp comment
hline.after vector
add.to.row a list
sanitize.text.function a function
str tbd value
remove.leading0 tbd value
caption tbd value
longtable tbd value
label default to be the same as file.name stem
print.functions  

*table.placement*  
tbdvalue  
*na.to.empty*  
tbdvalue  
*value*  
tbdvalue  
*digits*  
tbddigits  
*fill*  
tbdfill  
*models*  
tbdmodels  
*model.names*  
tbdmodel.names  
*row.major*  
tbdrow.major  
*round.digits*  
tbdround.digits  
*dat*  
tbddat  
*file.name*  
tbdfilename  
&display*  
tbddisplay  
*align*  
tbdalign  
*append*  
tbdappend  
*preamble*  
tbdpreamble  
*include.rownames*  
tbdinclude.rownames  
*floating*  
tbdfloating  
*lines*  
tbdlines  
*...*  
tbd...  
*verbose*  
tbd...  
*x*  
tbdx  
*file*  
tbdfile  
*row.names*  
tbdrow.names  
*add.clear.page.between.tables*  
tbdrow.names

**Examples**

```r
roundup (3.1, 2) # 3.10
```

```r
formatInt(3, 2) # 03
```

```r
## Not run:
# demo of dimnames
tab=diag(1:4); rownames(tab)<-colnames(tab)<-1:4; names(dimnames(tab))=c("age","height")
# for greek letter in the labels, we need sanitize.text.function=identity
trownames(tab)[1]="\$\alpha\$
# note that to use caption, floating needs to be TRUE
mytex (tab, file="tmp1", sanitize.text.function=identity,
```
random.functions

Random Functions

Description
Generate samples from random variables.

Usage

dbern(x, prob, log = FALSE)
dcorbern(x, p, a, log = FALSE)
dmixnorm(x, mix.p, sd1, sd2, log = FALSE)
dnorm.norm.gamma(x, p, same.distr = FALSE, log = FALSE)

rbern(n, prob, generalized = FALSE)

rbigamma(n, shape.1, shape.2, rate.1, rate.2, rho)

rbilogistic(n, loc.1, loc.2, scale.1, scale.2, rho)

rejective.sampling(N, n, pik)

rnorm.ar(n, sd, rho)

rnorm.norm.gamma(n, mu.0, lambda, alpha, beta)

rmixnorm(n, mix.p, mu1, mu2, sd1, sd2)

rdoublexp(n, location=0, scale=1)

ddoublexp(x, location=0, scale=1)

qdoublexp(p, location=0, scale=1)

pdoublexp(q, location=0, scale=1)

rbidoublexp(n, loc.1, loc.2, scale.1, scale.2, rho)

Arguments

- q: tbdx
- location: tbdx
- scale: tbdx
- x: tbdx
- prob: tbdprob
- log: tbdlog
- p: tbdp
- a: tbd
- mix.p: tbdmix.p
- sd1: tbdsd1
- sd2: tbdsd2
- same.distr: tbdsame.distr
- n: tbdn
- generalized: tbdgeneralized
- N: tbdN
- pik: tbdpik
- mu: tbdmu
Details

rbern generates Bernoulli random variables.
rbilogistic generates a bivariate logistic distribution for correlation coefficient 0.5, or [-0.271, 0.478]. In the former case it is generated by calling rbilogis, part of the VGAM package; in the latter case it is generated via the AMH copular.
rnorm.ar simulate autoregressive normal random variables, correlation is $\rho^d$ between $x_1$ and $x_{(1+d)}$

Examples

```r
set.seed(1)
rbern(n=10, p=1/2)
rbern(n=2, p=c(.999,.001))
```
Regression Model Functions

Description

getFormattedSummary prints a table of regression coefficient estimates and standard errors.

Usage

getFormattedSummary(fits, type = 12, est.digits = 2, se.digits = 2, robust, random = FALSE, VE = FALSE, to.trim = FALSE, rows = NULL, coef.direct = FALSE, trunc.large.est = TRUE, scale.factor = 1, p.digits = 3, remove.leading0 = FALSE, p.adj.method = "fdr", ...)  
getVarComponent(object, ...)  
getFixedEf(object, ...)  
risk.cal(risk, binary.outcome, weights = NULL, ngroups = NULL, cuts = NULL, main = "", add = FALSE, show.emp.risk = TRUE, lcol = 2, ylim = NULL, scale = c("logit", "risk"))  
interaction.table(fit, v1, v2, v1.type = "continuous", v2.type = "continuous", logistic.regression = TRUE)  

## S3 method for class 'coxph'  
getFixedEf(object, exp=FALSE,robust=FALSE, ...)  

## S3 method for class 'gam'  
getFixedEf(object, ...)  

## S3 method for class 'gee'  
getFixedEf(object, exp = FALSE, ...)  

## S3 method for class 'geese'  
getFixedEf(object, robust = TRUE, ...)  

## S3 method for class 'tps'  
getFixedEf(object, exp=FALSE, robust=TRUE, ...)  

## S3 method for class 'glm'  
getFixedEf(object, exp = FALSE, robust = TRUE, ret.robcov = FALSE, ...)  

## S3 method for class 'svyglm'  
getFixedEf(object, exp = FALSE, robust = TRUE, ...)

regression.model.functions
## S3 method for class 'svy_vglm'
getFixedEf(object, exp = FALSE, robust = TRUE, ...)

## S3 method for class 'svycoxph'
getFixedEf(object, exp = FALSE, robust = TRUE, ...)

## S3 method for class 'inla'
getFixedEf(object, ...)

## S3 method for class 'lm'
getFixedEf(object, ...)

## S3 method for class 'lme'
getFixedEf(object, ...)

## S3 method for class 'logistf'
getFixedEf(object, exp = FALSE, ...)

## S3 method for class 'matrix'
getFixedEf(object, ...)

## S3 method for class 'M1result'
getFixedEf(object, ...)

## S3 method for class 'hyperpar.inla'
getVarComponent(object, transformation = NULL, ...)

## S3 method for class 'matrix'
getVarComponent(object, ...)

## S3 method for class 'geese'
coef(object, ...)

## S3 method for class 'tps'
coef(object, ...)

## S3 method for class 'geese'
predict(object, x, ...)

## S3 method for class 'tps'
predict(object, newdata = NULL, type = c("link", "response"), ...)

## S3 method for class 'geese'
residuals(object, y, x,...)

## S3 method for class 'geese'
vcov(object, ...)

## S3 method for class 'tps'
vcov(object, robust, ...)
## S3 method for class 'logistf'
vcov(object, ...)

### Arguments

- ...`: tbd...
- `object`: `tbdobject`
- `fit`: `tbdfit`
- `coef.direct`: `tbdfit`
- `robust`: Boolean, whether to return robust variance estimate
- `exp`: `tbdexp`
- `remove.leading0`: `tbdexp`
- `p.adj.method`: `tbdexp`
- `cuts`: `tbdfits`
- `ret.robcov`: `tbdfits`
- `fits`: `tbdfits`
- `type`: `tbdtype`
- `est.digits`: `tbdest.digits`
- `se.digits`: `tbdse.digits`
- `p.digits`: `tbdse.digits`
- `random`: `tbdrandom`
- `VE`: `tbdrandom`
- `transformation`: `tbdtransformation`
- `weights`: `tbdv1`
- `v1`: `tbdv1`
- `v2`: `tbdv2`
- `v1.type`: `tbdv1.type`
- `v2.type`: `tbdv2.type`
- `logistic.regression`: `tbdlogistic.regression`
- `newdata`: `tbdx`
- `x`: `tbdx`
- `y`: `tbdy`
- `to.trim`: `tbdy`
- `rows`: `tbdy`
- `risk`: `tbdfit`
- `binary.outcome`: `tbdfit`
- `ngroups`: `tbdfit`
Detailed usage:

- getFormattedSummary: from a list of fits, say lmer, inla fits, return formatted summary controlled by "type". For a matrix, return Monte Carlo variance random=TRUE returns variance components
type=1: est type=2: est (se) type=3: est (2.5 percent, 97.5 percent) type=4: est se
- getFixedEf returns a matrix, first column coef, second column se,
- getFixedEf.matrix used to get mean and sd from a jags or winbugs sample, getVarComponent.matrix and getFixedEf.matrix do the same thing. Each column of samples is a variable
- interaction.table expects coef and vcov to work with fit.

Examples

```r
## Annette Dobson (1990) "An Introduction to Generalized Linear Models"
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
glm.D9 <- glm(weight ~ group)
getFormattedSummary (list(lm.D9, glm.D9), robust=FALSE)
```

---

**Description**

ROC/AUC methods. fastauc calculates the AUC using a sort operation, instead of summing over pairwise differences in R.
- computeRoc computes an ROC curve.
- plotRoc plots an ROC curve.
- addRoc adds an ROC curve to a plot.
- classification.error computes classification error.
Usage

fastauc (score, outcome, t0 = 0, t1 = 1, reverse.sign.if.nece = TRUE, quiet = FALSE)
computeRoc (score, outcome, reverse.sign.if.nece = TRUE, cutpoints = NULL)
plotRoc(x, add = FALSE, type = "l", diag.line=TRUE,...)
addRoc (x,...)
classification.error(score, outcome, threshold=NULL, verbose=FALSE)

Arguments

score           a vector. Linear combination or score.
outcome         a vector of 0 and 1. Outcome.
t0              a number between 0 and 1 that is the lower boundary of pAUC
t1              a number between 0 and 1 that is the upper boundary of pAUC
reverse.sign.if.nece
                a boolean. If TRUE, score is multiplied by -1 if AUC is less than 0.5.
x               a list of two elements: sensitivity and specificity.
diag.line
                boolean. If TRUE, a diagonal line is plotted
add              boolean. If TRUE, add to existing plot. If FALSE, create a new plot.
quiet
threshold       threshold
verbose         boolean
type
                line type for lines
...              arguments passed to plot or lines

Details

These functions originally come from Thomas Lumley and Tianxi Cai et al.

Value

computeRoc returns a list of sensitivity and specificity.
plotRoc and addRoc plots ROC curves.

Author(s)

Shuxin Yin <>
Youyi Fong <youyifong@gmail.com>
Krisztian Sebestyen <>
Examples

n=1e2
score=c(rnorm(n/2,1), rnorm(n/2,0))
outcome=rep(1:0, each=n/2)
# cannot print due to r cmd check
#plotRoc(computeRoc(score, outcome))

# commented out b/c slower on pc and cause note when r cmd check
## test, fastauc2 is a version without all the checking
#score=rnorm(1e5)
#outcome=rbinom(1e5,1,.5)
#system.time(for (i in 1:1e2) fastauc(score,outcome)) # 4.9 sec
#system.time(for (i in 1:1e2) fastauc2(score,outcome)) # 3.8 sec

---

**sim.dat.tvarying.two**  
_Simulation Functions for Time-dependent Proportional Hazard Model_

**Description**

`sim.dat.tvarying.three` simulates from a model with time varying age group variable of three levels, `sim.dat.tvarying.two` two.

**Usage**

```r
sim.dat.tvarying.three(n, followup.length, incidence.density,
    age.sim = c("tvaryinggroup", "baselinegroup", "continuous","bt"),
    random.censoring.rate = 0.05, seed)
```

```r
sim.dat.tvarying.two(n, followup.length, incidence.density,
    age.sim = c("tvaryinggroup", "baselinegroup", "continuous","bt"),
    random.censoring.rate = 0.05, seed)
```

**Arguments**

- `n` integer. Sample size.
- `followup.length` numeric. Length of followup, in years.
- `incidence.density` numeric. Incidence rate per year.
- `age.sim` string. Choose between one of three possibilities. `tvaryinggroup`: age group is time-varying covariate; `baselinegroup`: age group is a baseline covariate; `continuous`: age is a continuous covariate; `bt`: age group by treatment interaction uses baseline age group, while age group main effect uses time-dependent age group.
- `random.censoring.rate` numeric. Amount of random censoring.
- `seed` integer. Random number generator seed.
Details

In `sim.dat.tvarying.three`, baseline age is uniformly distributed between 2.0 and 16.0, and divided into three groups at 6 and 12. In `sim.dat.tvarying.two`, baseline age is uniformly distributed between 2.0 and 12.0, and divided into two groups at 6.

Value

Return a data frame with the following columns:

- **ptid**: subject identifier
- **trt**: treatment indicator 0/1
- **for.non.tvarying.ana**: Boolean, used to subset dataset for non-time dependent analysis
- **C**: censoring time
- **baseline.age**: age years at baseline
- **agegrp**: a factor with levels [0,6) [6,12) [12,100)
- **baseline.agegrp**: a factor with levels [0,6) [6,12) [12,100)
- **tstart**: left bound of time interval
- **tstop**: right bound of time interval
- **d**: event indicator
- **X**: followup time, in years

Author(s)

Youyi Fong

See Also

`make.timedep.dataset`

Examples

```r
library(survival)

dat=sim.dat.tvarying.three(n=6000,followup.length=3, incidence.density=0.05,
    age.sim="tvaryinggroup", seed=1)
f.tvarying = Surv(tstart,tstop,d) ~ trt*agegrp
f = Surv(X,d) ~ trt*baseline.agegrp
fits=list()
fits["tvarying"]=coxph(f.tvarying, dat)
fits["baseline"]=coxph(f, subset(dat, for.non.tvarying.ana))
fits
```
**Description**

H calculates entropy.

**Usage**

```r
H(p, logbase = c("e", "2"))
```

```r
mutual.info(two.way.table, logbase = c("e", "2"))
```

```r
cor.mixed(x, ...)
```

```r
## Default S3 method:
cor.mixed(x, na.fun, method="pearson", "spearman", ...)
## S3 method for class 'vector'
cor.mixed(x, y, na.fun, method="pearson", "spearman", ...)
## S3 method for class 'formula'
cor.mixed(formula, data, na.fun, method=c("pearson", "spearman"), ...)
```

```r
skew(x, na.rm = FALSE)
```

```r
info.cor(two.way.table)
```

```r
yule.y(two.by.two.matrix)
```

```r
kappacor(two.by.two.matrix, weight = c(1, 1), maximum = FALSE)
```

```r
l.measure(two.by.two.matrix)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>either a count vector or a probability vector, but can not be a vector of membership indicator</td>
</tr>
<tr>
<td>logbase</td>
<td>tbdlogbase</td>
</tr>
<tr>
<td>na.rm</td>
<td>tbdlogbase</td>
</tr>
<tr>
<td>two.way.table</td>
<td>tbdtwo.way.table</td>
</tr>
<tr>
<td>x</td>
<td>tbdx</td>
</tr>
<tr>
<td>...</td>
<td>tbd...</td>
</tr>
<tr>
<td>na.fun</td>
<td>tbdna.fun</td>
</tr>
</tbody>
</table>
string.functions

method          tbdmethod
y              tbdy
formula        tbdformula
data           tbddata
two.by.two.matrix  tbdtwo.by.two.matrix
weight         tbdweight
maximum        tbdmaximum

Examples

\( H(\text{rep}(1/5, 5)) \)
\( H(\text{rep}(3, 5)) \)

---

**Description**

```
%+% concatenates its arguments and returns a string.
``` 

**Usage**

```
a %.% b

contain(s1, s2)
trim (x, trim.trailing=TRUE, trim.leading=TRUE)
escapeUnderline(name)
fileStem(file.name)
firstIndex(s1, s2)
getExt(file.name)
getFileStem(file.name)
getStem(file.name)
lastIndex(s1, s2)
remove.prefix(s, sep = "_")
```
Arguments

a  a
b  b
s1 s1
s2 s2
name name
file.name file.name s  s
sep  sep x  sep trim.leading  sep trim.trailing  sep

Examples

x = 1
x %.% "b" %.% "c"

testing.functions  Testing Functions

Description

Testing functions.

Usage

hosmerlem(y, yhat, g = 10)

quick.t.test(x, y, var.equal = FALSE)

sign.test(x)

tukey.mtest(mu, ms, n)

vector.t.test(mean.x, mean.y, var.x, var.y, n)

myfisher.test(x, y, ...)

mycor.test(x, method = c("pearson", "kendall", "spearman"), idx = NULL)
Arguments

...  tbd
y   tbdy
yhat tbdyhat
g   tbdg
x   tbdx
var.equal  tbdvar.equal
method  tbdmethod
mu   tbdmu
ms   tbdms
n   tbdn
mean.x  tbdmean.x
mean.y  tbdmean.y
var.x  tbdvar.x
var.y  tbdvar.y
idx  tbdvar.y

Examples

signtest(runif(10))

VEplot

Vaccine Efficacy Plots

Description

Vaccine efficacy plots.

Usage

VEplot (object, ...)

# S3 method for class 'cox.zph'
VEplot(object, resid = TRUE, se = TRUE, df = 4, nsmo = 40,
       var, ylab="VE", xlab="Time", xaxt="s", cex.axis=1, ...)

# S3 method for class 'glm'
VEplot(object, X1, X2, x, ...)

# S3 method for class 'cox.zph'
myplot(object, resid = TRUE, se = TRUE, df = 4, nsmo = 40, var,
       coef.transform=NULL,
       ylab=NULL,
       xlab="Time", xaxt="s", cex.axis=1,
       ...)


Arguments

- **object**: An object
- **resid**: Boolean, whether to plot residuals
- **se**: Boolean, whether to plot confidence band
- **df**: degrees of freedom
- **nsmo**: number of points used to plot the fitted spline
- **var**: estimated variance matrix from the Cox model fit
- **xlab**: x label
- **xaxt**: x axis
- **cex.axis**: cex for axis
- **ylab**: y label
- **coef.transform**: a function to transform Cox hazard ratio estimate
- **X1**: a matrix of dimension k by p, where k is the length of x (see below) and p is the length of coef(object)
- **X2**: a matrix of dimension k by p, where k is the length of x (see below) and p is the length of coef(object)
- **x**: a vector of length k that represents the x coordinate of the VE plot
- ... additional parameters

Details

VEplot and myplot.cox.zph are extensions of survival::plot.cox.zph to plot VE curve and other transformations.

myplot.cox.zph adds the following parameters to the original list of parameters in plot.cox.zph: coef.transform: a function to transform the coefficients ylab: y axis label xlab: x axis label

VEplot.glm computes a series of k VEs: for i in 1..k, VE[i] = P(Y=1|X1[i,])/P(Y=1|X2[i,]). It returns a 3 by k matrix, whose first row contains VE estimates and the second and third rows contain lower and upper bounds, respectively.

Author(s)

Youyi Fong, Dennis Chao

References

Examples

library(survival)
vfit <- coxph(Surv(time,status) ~ trt + factor(celltype) +
             karno + age, data=veteran, x=TRUE)
temp <- cox.zph(vfit)

par(mfrow=c(2,2))
for (v in c("trt","age")) {
    VEplot(temp, var=v, resid=FALSE, main=v, ylab="VE", cex.axis=1.5)
    plot(temp, var=v, resid=FALSE, main=v)
}

library(survival)
fit <- glm(status ~ trt + trt*age, data=veteran)
summary(fit)
age=seq(min(veteran$age),max(veteran$age),length=10)
out = VEplot(fit, X1=cbind(1,1,age,1*age), X2=cbind(1,0,age,0*age), x=age)
out
Index

* time varying
  * make.timedep.dataset, 21
    * .as.double (matrix2), 25
    * %.% (string.functions), 49
  * abline.pt.slope (plotting), 28
  * abline.pts (plotting), 28
  * abline.shade (plotting), 28
  * add.mtext.label (plotting), 28
  * addRoc (roc), 44
  * age_calc, 2
  * AR1 (matrix.array.functions), 23
  * array.functions  (matrix.array.functions), 23
  * as.binary (math.functions), 22
  * auc, 3
  * base.functions, 4
  * binary (base.functions), 4
  * binary2 (base.functions), 4
  * binaryloess, 7
  * binom.coef (math.functions), 22
  * butterfly.plot (plotting), 28
  * cbind.uneven (base.functions), 4
  * classification.error (roc), 44
  * coef.auc (auc), 3
  * coef.Deming (Deming), 10
  * coef.geese  (regression.model.functions), 41
  * coef.tps (regression.model.functions), 41
  * computeRoc (roc), 44
  * concatList (matrix.array.functions), 23
  * contain (string.functions), 49
  * cor.mixed (stat.functions), 48
  * corplot (plotting), 28
  * cox.zph, 2
  * crossvalidation, 9
  * dbinom (random.functions), 38
  * dcorbern (random.functions), 38
  * ddoublexp (random.functions), 38
  * dec_to_binary (misc), 26
  * Deming, 10
  * DMHeatMap, 11
  * dmixnorm (random.functions), 38
  * dnorm.norm.gamma (random.functions), 38
  * DXD (matrix2), 25
  * empty.plot (plotting), 28
  * empty2na (misc), 26
  * escapeUnderline (string.functions), 49
  * EXCH (matrix.array.functions), 23
  * expit (math.functions), 22
  * f2c (base.functions), 4
  * fastauc (roc), 44
  * fileStem (string.functions), 49
  * fill.jagged.array  (matrix.array.functions), 23
  * firstIndex (string.functions), 49
  * formatDouble (print.functions), 35
  * formatInt (print.functions), 35
  * ftoi (base.functions), 4
  * get.kfold.splits (crossvalidation), 9
  * get.sim.res, 13
  * get.splits (crossvalidation), 9
  * get_count_from_xy_coor, 16
  * getExt (string.functions), 49
  * getFileStem (string.functions), 49
  * getFixedEf  (regression.model.functions), 41
  * getFixedEf.Deming (Deming), 10
  * getFixedEf2  (regression.model.functions), 41
  * getFormattedMCSummary (get.sim.res), 13
INDEX

getFormattedSummary  
  (regression.model.functions), 41
getK, 15
getMfrow (plotting), 28
getMidPoints (matrix.array.functions), 23
getStem (string.functions), 49
getUpperRight (matrix.array.functions), 23
getVarComponent  
  (regression.model.functions), 41

H (stat.functions), 48
hosmerlem (testing.functions), 50
info.cor (stat.functions), 48
INT (misc), 26
interaction.table  
  (regression.model.functions), 41
interpolate (math.functions), 22
iorw, 17
kappacor (stat.functions), 48
keepWarnings (base.functions), 4
kfold.split (crossvalidation), 9
kid, 19
kyotil, 20

l.measure (stat.functions), 48
last (matrix.array.functions), 23
lastIndex (string.functions), 49
logDiffExp (math.functions), 22
logit (math.functions), 22
logMeanExp (math.functions), 22
logSumExp (math.functions), 22
logSumExpFor2 (math.functions), 22
lpo.split (crossvalidation), 9

make.latex.coef.table  
  (print.functions), 35
make.timedep.dataset, 21, 47
math.functions, 22
matrix.array.functions, 23
matrix.functions  
  (matrix.array.functions), 23
matrix2, 25
MCsummary (get.sim.res), 13
meanmed (base.functions), 4
misc, 26
mix (matrix.array.functions), 23
multi.outer (base.functions), 4
mutual.info (stat.functions), 48
my.interaction.plot (plotting), 28
myaggregate (base.functions), 4
myboxplot (plotting), 28
mycor (base.functions), 4
mycor.test (testing.functions), 50
mydev.off (plotting), 28
myfigure (plotting), 28
myfisher.test (testing.functions), 50
myforestplot (plotting), 28
myhist (plotting), 28
mylegend (plotting), 28
mylines (plotting), 28
mymatplot (plotting), 28
mypairs (plotting), 28
mypdf (plotting), 28
myplot (plotting), 28
myplot.cox.zph (VEplot), 51
mypng (plotting), 28
mypostscript (plotting), 28
myprint (print.functions), 35
myreshapelong (base.functions), 4
myreshapewide (base.functions), 4
mysanitize.numbers (print.functions), 35
mysanitize.text (print.functions), 35
mysapply (base.functions), 4
myscale (base.functions), 4
mysystem (base.functions), 4
mytable (base.functions), 4
mytapply (base.functions), 4
mytex (print.functions), 35
mytext (plotting), 28
mywrite (print.functions), 35

p.adj.perm, 27
panel.cor (plotting), 28
panel.hist (plotting), 28
panel.nothing (plotting), 28
pava (misc), 26
pdoublexp (random.functions), 38
permn (math.functions), 22
plotRoc (roc), 44
plotting, 28
predict.auc (auc), 3
predict.Deming (Deming), 10
predict.geese
  (regression.model.functions), 41
predict.pcc (misc), 26
predict.tps
  (regression.model.functions), 41
prettyprint (print.functions), 35
print.auc (auc), 3
print.functions, 35
print.iorw (iorw), 17
qdoublexp (random.functions), 38
quick.t.test (testing.functions), 50
ran.kfold.split (crossvalidation), 9
random.functions, 38
rank.inv.norm (misc), 26
ratio (auc), 3
rbern (random.functions), 38
rbidoublexp (random.functions), 38
rbigamma (random.functions), 38
rbilogistic (random.functions), 38
rdoublexp (random.functions), 38
read.sv (base.functions), 4
read.tsv (base.functions), 4
regression.model.functions, 41
rejective.sampling (random.functions), 38
remove.prefix (string.functions), 49
rep.data.frame
  (matrix.array.functions), 23
rep.matrix (matrix.array.functions), 23
residuals.geese
  (regression.model.functions), 41
risk.cal (regression.model.functions), 41
rmixnorm (random.functions), 38
rnorm.ar (random.functions), 38
rnorm.norm.gamma (random.functions), 38
roc, 44
roundup (print.functions), 35
sample.for.cv (crossvalidation), 9
shift.left (matrix.array.functions), 23
shift.right (matrix.array.functions), 23
signtest (testing.functions), 50
sim.dat.tvarying.three
  (sim.dat.tvarying.two), 46
sim.dat.tvarying.two, 46
skew (stat.functions), 48
smoothed.scaled.hist (plotting), 28
stat.functions, 48
Stirling2 (math.functions), 22
string.functions, 49
summ (misc), 26
summary.auc (auc), 3
summary.Deming (Deming), 10
symprod (matrix2), 25
table.cases (base.functions), 4
table.prop (base.functions), 4
testing.functions, 50
thin.rows (matrix.array.functions), 23
ThinRows (matrix.array.functions), 23
t (matrix.array.functions), 23
trainauc (auc), 3
trim (string.functions), 49
tukey.mtest (testing.functions), 50
tXDX (matrix2), 25
txSy (matrix2), 25
unix (base.functions), 4
v cov. Deming (Deming), 10
v cov. geese
  (regression.model.functions), 41
v cov. logistf
  (regression.model.functions), 41
v cov. tps (regression.model.functions), 41
vector.t.test (testing.functions), 50
VEplot, 51
whiskers (plotting), 28
wtd.hist (plotting), 28
yule.y (stat.functions), 48