Package ‘rmutil’

June 9, 2020

Version 1.1.5

Title Utilities for Nonlinear Regression and Repeated Measurements Models

Depends R (>= 1.4)

Description A toolkit of functions for nonlinear regression and repeated measurements not to be used by itself but called by other Lindsey packages such as 'gnlm', 'stable', 'growth', 'repeated', and 'event' (available at <http://www.commanster.eu/rcode.html>).

License GPL-2

URL http://www.commanster.eu/rcode.html

BugReports https://github.com/swihart/rmutil/issues

Encoding UTF-8

LazyData true

LazyLoad true

NeedsCompilation yes

Author Bruce Swihart [cre, aut],
Jim Lindsey [aut] (Jim created this package, Bruce is maintaining the CRAN version)

Maintainer Bruce Swihart <bruce.swihart@gmail.com>

Repository CRAN

Date/Publication 2020-06-09 05:30:06 UTC

R topics documented:

`Beta Binomial` *Beta Binomial Distribution*

Description

These functions provide information about the beta binomial distribution with parameters `m` and `s`: density, cumulative distribution, quantiles, and random generation. Compared to the parameterization of ‘VGAM::pbetabinom.ab‘, `m = alpha/(alpha+beta)` and `s = (alpha+beta)`. See examples.

The beta binomial distribution with total $= n$ and `prob` $= m$ has density

$$p(y) = \frac{B(y + \sigma\mu, n - y + \sigma * (1 - \mu)) \binom{n}{y}}{B(sm, s(1 - m))}$$

for $y = 0, \ldots, n$ where $B()$ is the beta function.

Usage

```
dbetabinom(y, size, m, s, log=FALSE)
pbetabinom(q, size, m, s)
qbetabinom(p, size, m, s)
rbetabinom(n, size, m, s)
```

Arguments

`y`	vector of frequencies
`q`	vector of quantiles
`p`	vector of probabilities
`n`	number of values to generate
`size`	vector of totals
`m`	vector of probabilities of success; Compared to the parameterization of ‘VGAM::pbetabinom.ab‘, `m = alpha/(alpha+beta)` where `shape1=alpha` and `shape2=beta`. See examples.
`s`	vector of overdispersion parameters; Compared to the parameterization of ‘VGAM::pbetabinom.ab‘, `s = (alpha+beta)` where `shape1=alpha` and `shape2=beta`. See examples.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dbinom` for the binomial, `ddoublebinom` for the double binomial, and `dmultbinom` for the multiplicative binomial distribution.

Examples

```
# compute P(45 < y < 55) for y beta binomial(100,0.5,1.1)
sum(dbetabinom(46:54, 100, 0.5, 1.1))
pbetabinom(54,100,0.5,1.1)-pbetabinom(45,100,0.5,1.1)
pbetabinom(2,10,0.5,1.1)
qbetabinom(0.33,10,0.5,1.1)
rbetabinom(10,10,0.5,1.1)
## compare to VGAM
## Not run:
# The beta binomial distribution with total = n and prob = m has density
#
# p(y) = B(y+s m,n-y+s (1-m)) Choose(n,y) / B(s m,s (1-m))
#
# for y = 0, . . . , n where B() is the beta function.

## in `rmutil` from the .Rd file (excerpt above), the "alpha" is s*m
## in `rmutil` from the .Rd file (excerpt above), the "beta"  is s*(1-m)

## in `VGAM`, rho is 1/(1+alpha+beta)

qq = 2.2
zz = 100

alpha = 1.1
beta  = 2
VGAM::pbetabinom.ab(q=qq, size=zz, shape1=alpha, shape2=beta)

## for VGAM `rho`
rr = 1/(1+alpha+beta)
VGAM::pbetabinom   (q=qq, size=zz, prob=mm, rho = rr)

## for rmutil `m` and `s`:
mm = alpha / (alpha+beta)
ss = (alpha+beta)
rmutil::pbetabinom(q=qq, size=zz, m=mm, s=ss )

## End(Not run)
```

Box-Cox *Box-Cox Distribution*

Description

These functions provide information about the Box-Cox distribution with location parameter equal to `m`, dispersion equal to `s`, and power transformation equal to `f`: density, cumulative distribution, quantiles, log hazard, and random generation.

The Box-Cox distribution has density

$$f(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-((y^\nu/\nu - \mu)^2/(2\sigma^2)))/(1 - I(\nu < 0) - sign(\nu) * pnorm(0, \mu, sqrt(\sigma)))$$

where μ is the location parameter of the distribution, σ is the dispersion, ν is the family parameter, $I()$ is the indicator function, and $y > 0$.

$\nu = 1$ gives a truncated normal distribution.

Usage

```
dboxcox(y, m, s=1, f=1, log=FALSE)
pboxcox(q, m, s=1, f=1)
qboxcox(p, m, s=1, f=1)
rboxcox(n, m, s=1, f=1)
```

Arguments

`y`	vector of responses.
`q`	vector of quantiles.
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of location parameters.
`s`	vector of dispersion parameters.
`f`	vector of power parameters.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dnorm` for the normal or Gaussian distribution.

Examples

```
dboxcox(2, 5, 5, 2)
pboxcox(2, 5, 5, 2)
qboxcox(0.1, 5, 5, 2)
rboxcox(10, 5, 5, 2)
```

`Burr` *Burr Distribution*

Description

These functions provide information about the Burr distribution with location parameter equal to `m`, dispersion equal to `s`, and family parameter equal to `f`: density, cumulative distribution, quantiles, log hazard, and random generation.

The Burr distribution has density

$$f(y) = \frac{\nu\sigma(y/\mu)^{\sigma-1}}{\mu(1+(y/\mu)^{\sigma})^{\nu+1}}$$

where μ is the location parameter of the distribution, σ is the dispersion, and ν is the family parameter.

Usage

```
dburr(y, m, s, f, log=FALSE)
pburr(q, m, s, f)
qburr(p, m, s, f)
rburr(n, m, s, f)
```

Arguments

`y`	vector of responses.
`q`	vector of quantiles.
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of location parameters.
`s`	vector of dispersion parameters.
`f`	vector of family parameters.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

Examples

```
dburr(2, 5, 1, 2)
pburr(2, 5, 1, 2)
qburr(0.3, 5, 1, 2)
rburr(10, 5, 1, 2)
```

capply — *A Fast Simplified Version of* `tapply`

Description

a fast simplified version of `tapply`

Usage

```
capply(x, index, fcn=sum)
```

Arguments

`x`	x
`index`	index
`fcn`	default sum

Details

a fast simplified version of tapply

Value

Returns `ans` where `for(i in split(x,index))ans <-c(ans,fcn(i))`.

Consul — *Consul Distribution*

Description

These functions provide information about the Consul distribution with parameters `m` and `s`: density, cumulative distribution, quantiles, and random generation.

The Consul distribution with `mu` $= m$ has density

$$p(y) = \mu \exp(-(\mu + y(\lambda - 1))/\lambda)(\mu + y(\lambda - 1))^(y - 1)/(\lambda^y y!)$$

for $y = 0, \ldots$.

Usage

```
dconsul(y, m, s, log=FALSE)
pconsul(q, m, s)
qconsul(p, m, s)
rconsul(n, m, s)
```

Arguments

`y`	vector of counts
`q`	vector of quantiles
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of means
`s`	vector of overdispersion parameters
`log`	if TRUE, log probabilities are supplied.

See Also

`dpois` for the Poisson, `ddoublepois` for the double Poisson, `dmultpois` for the multiplicative Poisson, and `dpvfpois` for the power variance function Poisson.

Examples

```
dconsul(5,10,0.9)
pconsul(5,10,0.9)
qconsul(0.08,10,0.9)
rconsul(10,10,0.9)
```

`contrast` *Contrast Matrix for Constraints about the Mean*

Description

Return a matrix of contrasts for constraints about the mean.

Usage

```
contr.mean(n, contrasts = TRUE)
```

Arguments

`n`	A vector of levels for a factor or the number of levels.
`contrasts`	A logical value indicating whether or not contrasts should be computed.

Details

This function corrects `contr.sum` to display labels properly.

Value

A matrix of computed contrasts with `n` rows and `k` columns, with `k=n-1` if `contrasts` is `TRUE` and `k=n` if `contrasts` is `FALSE`. The columns of the resulting matrices contain contrasts which can be used for coding a factor with `n` levels.

See Also

contrasts, C, and contr.sum.

Examples

```
oldop <- options(contrasts=c("contr.sum","contra.poly"))
y <- rnorm(30)
x <- gl(3,10,labels=c("First","Second","Third"))
glm(y~x)
options(contrasts=c("contr.mean","contra.poly"))
x <- gl(3,10,labels=c("First","Second","Third"))
glm(y~x)
options(oldop)
```

DataMethods *Methods for response, tccov, tvcov, and repeated Data Objects*

Description

Objects of class, `response`, contain response values, and possibly the corresponding times, binomial totals, nesting categories, censor indicators, and/or units of precision/Jacobian. Objects of class, `tccov`, contain time-constant or inter-individual, baseline covariates. Objects of class, `tvcov`, contain time-varying or intra-individual covariates. Objects of class, `repeated`, contain a `response` object and possibly `tccov` and `tvcov` objects.

In formula and functions, the key words, `times` can be used to refer to the response times from the data object as a covariate, `individuals` to the index for individuals as a factor covariate, and `nesting` the index for nesting as a factor covariate. The latter two only work for W&R notation.

The following methods are available for accessing the contents of such data objects.

`as.data.frame`: places all of the variables in the data object in one dataframe, extending time-constant covariates to the length of the others unless the object has class, `tccov`. Binomial and censored response variables have two columns, respectively 'yes' and 'no' and response and censoring indicator, with the name given to the response.

`as.matrix`: places all of the variables in the data object in one matrix, extending time-constant covariates to the length of the others unless the object has class, `tccov`. If any covariates are factor variables (instead of the corresponding sets of indicator variables), the matrix will be character instead of numeric.

`covariates`: extracts covariate matrices from a data object (for formulae and functions, possibly for selected individuals. See `covariates.formulafn`).

`covind`: gives the indexing of the response by individual (that is, the nesting indicator for observations within individuals). It can be used to expand time-constant covariates to the size of the repeated measurements response.

`delta`: extracts the units of measurement vector and Jacobian of any transformation of the response, possibly for selected individuals. Note that, if the unit of measurement/Jacobian is available in the `response` object, this is automatically included in the calculation of the likelihood function in all library model functions.

units: prints the variable names and their description and returns the latter.

formula: gives the formula used to create the time-constant covariate matrix of a data object (for formulae and functions, see formula.formulafn).

names: extracts the names of the response and/or covariates.

nesting: gives the coding variable(s) for individuals (same as covind) and also for nesting within individuals if available, possibly for selected individuals.

nobs: gives the number of observations per individual.

plot: plots the variables in the data object in various ways. For repeated objects, name can be a response or a time-varying covariate.

print: prints summary information about the variables in a data object.

response: extracts the response vector, possibly for selected individuals. If there are censored observations, this is a two-column matrix, with the censor indicator in the second column. For binomial data, it is a two-column matrix with "positive" (y) and "negative" (totals-y) frequencies.

resptype: extracts the type of each response.

times: extracts the times vector, possibly for selected individuals.

transform: transforms variables. For example, transform(z,y=fcn1(y),times=fcn2(times)) where fcn1 and fcn2 are transformation functions. When the response is transformed, the Jacobian is automatically calculated. New response variables and covariates can be created in this way, if the left hand side is a new name (ynew=fcn3(y)), as well as replacing an old variable with the transformed one. If the transformation reverses the order of the responses, use its negative to keep the ordering and have a positive Jacobian; for example, ry=-1/y. For repeated objects, only the response and the times can be transformed.

units: prints the variable names and their units of measurement and returns the latter.

weights: extracts the weight vector, possibly for selected individuals.

Usage

```
as.data.frame(x, ...)
as.matrix(x, ...)
covariates(z, ...)
covind(z, ...)
delta(z, ...)
## S3 method for class 'tccov'
formula(x, ...)
## S3 method for class 'repeated'
formula(x, ...)
## S3 method for class 'tccov'
names(x, ...)
## S3 method for class 'repeated'
names(x, ...)
nesting(z, ...)
nobs(z, ...)
## S3 method for class 'response'
plot(x, name=NULL, nind=NULL, nest=1, ccov=NULL, add=FALSE, lty=NULL, pch=NULL,
main=NULL, ylim=NULL, xlim=NULL, xlab=NULL, ylab=NULL, ...)
```

```
## S3 method for class 'repeated'
plot(x, name=NULL, nind=NULL, nest=1, ccov=NULL, add=FALSE, lty=NULL, pch=NULL,
main=NULL, ylim=NULL, xlim=NULL, xlab=NULL, ylab=NULL, ...)
## S3 method for class 'tccov'
print(x, ...)
## S3 method for class 'repeated'
print(x, nindmax=50, ...)
response(z, ...)
resptype(z, ...)
times(z, ...)
## S3 method for class 'response'
transform(`_data`, times=NULL, units=NULL, ...)
## S3 method for class 'repeated'
transform(`_data`, times=NULL, ...)
units(x, ...)
## S3 method for class 'gnlm'
weights(object, ...)
## S3 method for class 'repeated'
weights(object, nind=NULL, ...)
## S3 method for class 'response'
weights(object, nind=NULL, ...)
```

Arguments

x,z	A response, tccov, tvcov, or repeated data object. For covind and nobs, this may also be a model.
times	The function, when the times are to be transformed.
names	The names of the response variable(s) or covariate(s).
nind	The numbers of individuals to be used. (For plotting, cannot be used simultaneously with ccov.)
ccov	For plotting: If a vector of values for the time-constant covariates is supplied, only individuals having that set of values will have profiles plotted. These values must be in the order in which the covariates appear when the data object is printed. For factor variables, the codes must be given. If the name of a covariate is supplied, a set of graphs is plotted, one for each covariate value, showing profiles of all individuals having that value. (The covariate can have a maximum of six values.) Cannot be used simultaneously with nind.
nest	For plotting: nesting category to plot.
add	For plotting: add to previous plot.
nindmax	For printing a response, tvcov, or repeated object, if the number of individuals is greater than nindmax, the range of numbers of observations per individual is printed instead of the vector of numbers.
name,lty,pch,main,ylim,xlim,xlab,ylab	See base plot.
_data,units,object	TBD.
...	Arguments to other methods

Value

These methods extract information stored in response, tccov, tvcov, and repeated data objects created respectively by restovec, tcctomat, tvctomat, and rmna.

Note that if a vector of binomial totals or a censoring indicator is present, this is extract as the second column of the matrix produced by the response method.

Author(s)

J.K. Lindsey

See Also

restovec, rmna, tcctomat, tvctomat.

Examples

```
# set up some data and create the objects
#
y <- matrix(rnorm(20),ncol=5)
tt <- c(1,3,6,10,15)
print(resp <- restovec(y, times=tt, units="m", type="duration"))
x <- c(0,0,1,1)
x2 <- as.factor(c("a","b","a","b"))
tcc <- tcctomat(data.frame(x=x,x2=x2))
z <- matrix(rpois(20,5),ncol=5)
tvc <- tvctomat(z)
print(reps <- rmna(resp, tvcov=tvc, ccov=tcc))
#
plot(resp)
plot(reps)
plot(reps, nind=1:2)
plot(reps, ccov=c(0,1))
plot(reps, ccov="x2")
plot(reps, name="z", nind=3:4, pch=1:2)
plot(reps, name="z", ccov="x2")
#
response(resp)
response(transform(resp, y=1/y))
response(reps)
response(reps, nind=2:3)
response(transform(reps,y=1/y))
#
times(resp)
times(transform(resp,times=times-6))
times(reps)
#
delta(resp)
delta(reps)
delta(transform(reps,y=1/y))
delta(transform(reps,y=1/y), nind=3)
#
```

```
nobs(resp)
nobs(tvc)
nobs(reps)
#
units(resp)
units(reps)
#
resptype(resp)
resptype(reps)
#
weights(resp)
weights(reps)
#
covariates(tcc)
covariates(tcc, nind=2:3)
covariates(tvc)
covariates(tvc, nind=3)
covariates(reps)
covariates(reps, nind=3)
covariates(reps,names="x")
covariates(reps,names="z")
#
names(tcc)
names(tvc)
names(reps)
#
nesting(resp)
nesting(reps)
#
# because individuals are the only nesting, this is the same as
covind(resp)
covind(reps)
#
as.data.frame(resp)
as.data.frame(tcc)
as.data.frame(tvc)
as.data.frame(reps)
#
# use in glm
rm(y,x,z)
glm(y~x+z, data=as.data.frame(reps))
```

`dftorep` *Transform a Dataframe to a repeated Object*

Description

`dftorep` forms an object of class, `repeated`, from a dataframe with the option of removing any observations where response and covariate values have NAs. For repeated measurements, observations on the same individual must be together in the table. A number of validity checks are performed on the data.

Such objects can be printed and plotted. Methods are available for extracting the response, the numbers of observations per individual, the times, the weights, the units of measurement/Jacobian, the nesting variable, the covariates, and their names: `response`, `nobs`, `times`, `weights`, `delta`, `nesting`, `covariates`, and `names`.

Usage

```
dftorep(dataframe, response, id=NULL, times=NULL, censor=NULL,
totals=NULL, weights=NULL, nest=NULL, delta=NULL,
coordinates=NULL, type=NULL, ccov=NULL, tvcov=NULL, na.rm=TRUE)
```

Arguments

`dataframe`	A dataframe.
`response`	A character vector giving the column name(s) of the dataframe for the response variable(s).
`id`	A character vector giving the column name of the dataframe for the identification numbers of the individuals. If the numbers are not consecutive integers, a warning is given. If NULL, one observation per individual is assumed if `times` is also NULL, other time series is assumed.
`times`	An optional character vector giving the column name of the dataframe for the times vector.
`censor`	An optional character vector giving the column name(s) of the dataframe for the censor indicator(s). This must be the same length as `response`. Responses without censor indicator can have a column either of all NAs or all 1s.
`totals`	An optional character vector giving the column name(s) of the dataframe for the totals for binomial data. This must be the same length as `response`. Responses without censor indicator can have a column all NAs.
`weights`	An optional character vector giving the column name of the dataframe for the weights vector.
`nest`	An optional character vector giving the column name of the dataframe for the nesting vector within individuals. This is the second level of nesting for repeated measurements, with the individual being the first level. Values for an individual must be consecutive increasing integers.
`delta`	An optional character vector giving the column name(s) of the dataframe for the units of measurement/Jacobian(s) of the response(s). This must be the same length as `response`. Responses without units of measurement/Jacobian can have a column all NAs. If all response variables have the same unit of measurement, this can be that one number. If each response variable has the same unit of measurement for all its values, this can be a numeric vector of length the number of response variables.

`coordinates` An optional character vector giving the two or three column name(s) of the dataframe for the spatial coordinates.

`type` An optional character vector giving the types of response variables: nominal, ordinal, discrete, duration, continuous, multivariate, or unknown.

`ccov` An optional character vector giving the column names of the dataframe for the time-constant or inter-individual covariates. For repeated measurements, if the value is not constant for all observations on an individual, an error is produced.

`tvcov` An optional character vector giving the column names of the dataframe for the time-varying or intra-individual covariates.

`na.rm` If TRUE, observations with NAs in any variables selected are removed in the object returned. Otherwise, the corresponding indicator variable is returned in a slot in the object.

Value

Returns an object of class, `repeated`, containing a list of the response object (`z$response`, so that, for example, the response vector is `z$response$y`; see `restovec`), and possibly the two classes of covariate objects (`z$ccov` and `z$tvcov`; see `tcctomat` and `tvctomat`).

Author(s)

J.K. Lindsey

See Also

`lvna`, `read.list`, `read.rep`, `restovec`, `rmna`, `tcctomat`, `tvctomat`

Examples

```
y <- data.frame(y1=rpois(20,5),y2=rpois(20,5))
y[2,2] <- NA
idd <- c(rep(1,5),rep(2,10),rep(3,5))
tt <- c(1:5,1:10,1:5)
totals <- data.frame(tot1=rep(12,20),tot2=rep(12,20))
x2 <- c(rep(1,5),rep(2,10),rep(3,5))
df <- data.frame(y,id=idd,tt=tt,totals,x1=rnorm(20),x2=x2)
df
dftorep(df,resp=c("y1","y2"),times="tt",id="id",totals=c("tot1","tot2"),
tvcov="x1",ccov="x2")
dftorep(df,resp=c("y1","y2"),times="tt",id="id",totals=c("tot1","tot2"),
tvcov="x1",ccov="x2",na.rm=FALSE)
# x1 is not a time-constant covariate
#dftorep(df,resp=c("y1","y2"),times="tt",id="id",ccov="x1",na.rm=FALSE)
```

`Double Binomial` *Double Binomial Distribution*

Description

These functions provide information about the double binomial distribution with parameters `m` and `s`: density, cumulative distribution, quantiles, and random generation.

The double binomial distribution with total $= n$ and `prob` $= m$ has density

$$p(y) = c(n, m, s)\binom{n}{y} n^{ns}(m/y)^(ys)((1-m)/(n-y))^((n-y)sy)y^y(n-y)^{(n-y)})$$

for $y = 0, \ldots, n$, where c(.) is a normalizing constant.

Usage

```
ddoublebinom(y, size, m, s, log=FALSE)
pdoublebinom(q, size, m, s)
qdoublebinom(p, size, m, s)
rdoublebinom(n, size, m, s)
```

Arguments

`y`	vector of frequencies
`q`	vector of quantiles
`p`	vector of probabilities
`n`	number of values to generate
`size`	vector of totals
`m`	vector of probabilities of success
`s`	vector of overdispersion parameters
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dbinom` for the binomial, `dmultbinom` for the multiplicative binomial, and `dbetabinom` for the beta binomial distribution.

Examples

```
# compute P(45 < y < 55) for y double binomial(100,0.5,1.1)
sum(ddoublebinom(46:54, 100, 0.5, 1.1))
pdoublebinom(54, 100, 0.5, 1.1)-pdoublebinom(45, 100, 0.5, 1.1)
pdoublebinom(2,10,0.5,1.1)
qdoublebinom(0.05,10,0.5,1.1)
rdoublebinom(10,10,0.5,1.1)
```

DoublePoisson *Double Poisson Distribution*

Description

These functions provide information about the double Poisson distribution with parameters `m` and `s`: density, cumulative distribution, quantiles, and random generation.

The double Poisson distribution with `mu` $= m$ has density

$$p(y) = c(\mu, \lambda)\lambda^(y/\mu)(\mu/y)^(y \log(\lambda))y^{(y-1)}/y!$$

for $y = 0, \ldots$, where c(.) is a normalizing constant.

Usage

```
ddoublepois(y, m, s, log=FALSE)
pdoublepois(q, m, s)
qdoublepois(p, m, s)
rdoublepois(n, m, s)
```

Arguments

`y`	vector of counts
`q`	vector of quantiles
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of means
`s`	vector of overdispersion parameters
`log`	if TRUE, log probabilities are supplied.

See Also

`dpois` for the Poisson, `dconsul` for the Consul generalized Poisson, `dgammacount` for the gamma count, `dmultpois` for the multiplicative Poisson, `dpvfpois` for the power variance function Poisson, and `dnbinom` for the negative binomial distribution.

Examples

```
ddoublepois(5,10,0.9)
pdoublepois(5,10,0.9)
qdoublepois(0.08,10,0.9)
rdoublepois(10,10,0.9)
```

`finterp` *Formula Interpreter*

Description

`finterp` translates a model formula into a function of the unknown parameters or of a vector of them. Such language formulae can either be in Wilkinson and Rogers notation or be expressions containing both known (existing) covariates and unknown (not existing) parameters. In the latter, factor variables cannot be used and parameters must be scalars.

The covariates in the formula are sought in the environment or in the data object provided. If the data object has class, `repeated` or `response`, then the key words, `times` will use the response times from the data object as a covariate, `individuals` will use the index for individuals as a factor covariate, and `nesting` the index for nesting as a factor covariate. The latter two only work for W&R notation.

Note that, in parameter displays, formulae in Wilkinson and Rogers notation use variable names whereas those with unknowns use the names of these parameters, as given in the formulae, and that the meaning of operators (*, /, :, etc.) is different in the two cases.

Usage

```
finterp(.z, ...)
## Default S3 method:
finterp(.z, .envir=parent.frame(), .formula=FALSE, .vector=TRUE,
.args=NULL, .start=1, .name=NULL, .expand=TRUE, .intercept=TRUE,
.old=NULL, .response=FALSE, ...)
```

Arguments

`.z`	A model formula beginning with ~, either in Wilkinson and Rogers notation or containing unknown parameters. If it contains unknown parameters, it can have several lines so that, for example, local variables can be assigned temporary values. In this case, enclose the formula in curly brackets.
`.envir`	The environment in which the formula is to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`.
`.formula`	If TRUE and the formula is in Wilkinson and Rogers notation, just returns the formula.
`.vector`	If FALSE and the formula contains unknown parameters, the function returned has them as separate arguments. If TRUE, it has one argument, the unknowns as a vector, unless certain parameter names are specified in `.args`. Always TRUE if `.envir` is a data object.

.args	If .vector is TRUE, names of parameters that are to be function arguments and not included in the vector.
.start	The starting index value of the parameter vector in the function returned when .vector is TRUE.
.name	Character string giving the name of the data object specified by .envir. Ignored unless the latter is such an object and only necessary when finterp is called within other functions.
.expand	If TRUE, expand functions with only time-constant covariates to return one value per observation instead of one value per individual. Ignored unless .envir is an object of class, repeated.
.intercept	If W&R notation is supplied and .intercept=F, a model function without intercept is returned.
.old	The name of an existing object of class formulafn which has common parameters with the one being created, or a list of such objects. Only used if .vector=TRUE. The value of .start should ensure that there is no conflict in indexing the vector.
.response	If TRUE, any response variable can be used in the function. If FALSE, checks are made that the response is not also used as a covariate.
...	Arguments passed to other functions.

Value

A function, of class formulafn, of the unknown parameters or of a vector of them is returned. Its attributes give the formula supplied, the model function produced, the covariate names, the parameter names, and the range of values of the index of the parameter vector. If formula is TRUE and a Wilkinson and Rogers formula was supplied, it is simply returned instead of creating a function.

Author(s)

J.K. Lindsey

See Also

FormulaMethods, covariates, fnenvir, formula, model, parameters

Examples

```
x1 <- rpois(20,2)
x2 <- rnorm(20)
#
# Wilkinson and Rogers formula with three parameters
fn1 <- finterp(~x1+x2)
fn1
fn1(rep(2,3))
# the same formula with unknowns
fn2 <- finterp(~b0+b1*x1+b2*x2)
fn2
```

```
fn2(rep(2,3))
#
# nonlinear formulae with unknowns
# log link
fn2a <- finterp(~exp(b0+b1*x1+b2*x2))
fn2a
fn2a(rep(0.2,3))
# parameters common to two functions
fn2b <- finterp(~c0+c1*exp(b0+b1*x1+b2*x2), .old=fn2a, .start=4)
fn2b
# function returned also depends on values of another function
fn2c <- finterp(~fn2+c1*exp(b0+b1*x1+b2*x2), .old=fn2a,
.start=4, .args="fn2")
fn2c
args(fn2c)
fn2c(rep(0.2,4),fn2(rep(2,3)))
#
# compartment model
times <- 1:20
# exp() parameters to ensure that they are positive
fn3 <- finterp(~exp(absorption-volume)/(exp(absorption)-
exp(elimination))*(exp(-exp(elimination)*times)-
exp(-exp(absorption)*times)))
fn3
fn3(log(c(0.3,3,0.2)))
# a more efficient way
# (note that parameters do not appear in the same order)
form <- ~{
ka <- exp(absorption)
ke <- exp(elimination)
ka*exp(-volume)/(ka-ke)*(exp(-ke*times)-exp(-ka*times))}
fn3a <- finterp(form)
fn3a(log(c(0.3,0.2,3)))
#
# Poisson density
y <- rpois(20,5)
fn4 <- finterp(~mu^y*exp(-mu)/gamma(y+1))
fn4
fn4(5)
dpois(y,5)
#
# Poisson likelihood
# mean parameter
fn5 <- finterp(~-y*log(mu)+mu+lgamma(y+1),.vector=FALSE)
fn5
likefn1 <- function(p) sum(fn5(mu=p))
nlm(likefn1,p=1)
mean(y)
# canonical parameter
fn5a <- finterp(~-y*theta+exp(theta)+lgamma(y+1),.vector=FALSE)
fn5a
likefn1a <- function(p) sum(fn5a(theta=p))
nlm(likefn1a,p=1)
```

```
#
# likelihood for Poisson log linear regression
y <- rpois(20,fn2a(c(0.2,1,0.4)))
nlm(likefn1,p=1)
mean(y)
likefn2 <- function(p) sum(fn5(mu=fn2a(p)))
nlm(likefn2,p=c(1,0,0))
# or
likefn2a <- function(p) sum(fn5a(theta=fn2(p)))
nlm(likefn2a,p=c(1,0,0))
#
# likelihood for Poisson nonlinear regression
y <- rpois(20,fn3(log(c(3,0.3,0.2))))
nlm(likefn1,p=1)
mean(y)
likefn3 <- function(p) sum(fn5(mu=fn3(p)))
nlm(likefn3,p=log(c(1,0.4,0.1)))
#
# envir as data objects
y <- matrix(rnorm(20),ncol=5)
y[3,3] <- y[2,2] <- NA
x1 <- 1:4
x2 <- c("a","b","c","d")
resp <- restovec(y)
xx <- tcctomat(x1)
xx2 <- tcctomat(data.frame(x1,x2))
z1 <- matrix(rnorm(20),ncol=5)
z2 <- matrix(rnorm(20),ncol=5)
z3 <- matrix(rnorm(20),ncol=5)
zz <- tvctomat(z1)
zz <- tvctomat(z2,old=zz)
reps <- rmna(resp, ccov=xx, tvcov=zz)
reps2 <- rmna(resp, ccov=xx2, tvcov=zz)
rm(y, x1, x2 , z1, z2)
#
# repeated objects
#
# time-constant covariates
# Wilkinson and Rogers notation
form1 <- ~x1
print(fn1 <- finterp(form1, .envir=reps))
fn1(2:3)
print(fn1a <- finterp(form1, .envir=xx))
fn1a(2:3)
form1b <- ~x1+x2
print(fn1b <- finterp(form1b, .envir=reps2))
fn1b(2:6)
print(fn1c <- finterp(form1b, .envir=xx2))
fn1c(2:6)
# with unknown parameters
form2 <- ~a+b*x1
print(fn2 <- finterp(form2, .envir=reps))
fn2(2:3)
```

```
print(fn2a <- finterp(form2, .envir=xx))
fn2a(2:3)
#
# time-varying covariates
# Wilkinson and Rogers notation
form3 <- ~z1+z2
print(fn3 <- finterp(form3, .envir=reps))
fn3(2:4)
print(fn3a <- finterp(form3, .envir=zz))
fn3a(2:4)
# with unknown parameters
form4 <- ~a+b*z1+c*z2
print(fn4 <- finterp(form4, .envir=reps))
fn4(2:4)
print(fn4a <- finterp(form4, .envir=zz))
fn4a(2:4)
#
# note: lengths of x1 and z2 differ
# Wilkinson and Rogers notation
form5 <- ~x1+z2
print(fn5 <- finterp(form5, .envir=reps))
fn5(2:4)
# with unknown parameters
form6 <- ~a+b*x1+c*z2
print(fn6 <- finterp(form6, .envir=reps))
fn6(2:4)
#
# with times
# Wilkinson and Rogers notation
form7 <- ~x1+z2+times
print(fn7 <- finterp(form7, .envir=reps))
fn7(2:5)
form7a <- ~x1+x2+z2+times
print(fn7a <- finterp(form7a, .envir=reps2))
fn7a(2:8)
# with unknown parameters
form8 <- ~a+b*x1+c*z2+e*times
print(fn8 <- finterp(form8, .envir=reps))
fn8(2:5)
#
# with a variable not in the data object
form9 <- ~a+b*z1+c*z2+e*z3
print(fn9 <- finterp(form9, .envir=reps))
fn9(2:5)
# z3 assumed to be an unknown parameter:
fn9(2:6)
#
# multiline formula
form10 <- ~{
tmp <- exp(b)
a+tmp*z1+c*z2+d*times}
print(fn10 <- finterp(form10, .envir=reps))
fn10(2:5)
```

fmobj *Object Finder*

Description

fmobj inspects a formula and returns a list containing the objects referred to, with indicators as to which are unknown parameters, covariates, factor variables, and functions.

Usage

```
fmobj(z, envir=parent.frame())
```

Arguments

z	A model formula beginning with ~, either in Wilkinson and Rogers notation or containing unknown parameters.
envir	The environment in which the formula is to be interpreted.

Value

A list, of class fmobj, containing a character vector (objects) with the names of the objects used in a formula, and logical vectors indicating which are unknown parameters (parameters), covariates (covariates), factor variables (factors), and functions (functions).

Author(s)

J.K. Lindsey

See Also

finterp

Examples

```
x1 <- rpois(20,2)
x2 <- rnorm(20)
x3 <- gl(2,10)
#
# W&R formula
fmobj(~x1+x2+x3)
#
# formula with unknowns
fmobj(~b0+b1*x1+b2*x2)
#
# nonlinear formulae with unknowns
# log link
fmobj(~exp(b0+b1*x1+b2*x2))
```

fnenvir *Check Covariates and Parameters of a Function*

Description

fnenvir finds the covariates and parameters in a function and can modify it so that the covariates used in it are found in the data object specified by .envir.

If the data object has class, repeated, the key word times in a function will use the response times from the data object as a covariate.

Usage

```
fnenvir(.z, ...)
## Default S3 method:
fnenvir(.z, .envir=parent.frame(), .name=NULL, .expand=TRUE,
.response=FALSE, ...)
```

Arguments

.z	A function.
.envir	The environment or data object of class, repeated, tccov, or tvcov, in which the function is to be interpreted.
.name	Character string giving the name of the data object specified by .envir. Ignored unless the latter is such an object and only necessary when fnenvir is called within other functions.
.expand	If TRUE, expand functions with only time-constant covariates to return one value per observation instead of one value per individual. Ignored unless .envir is an object of class, repeated.
.response	If TRUE, any response variable can be used in the function. If FALSE, checks are made that the response is not also used as a covariate.
...	Arguments passed to other functions.

Value

The (modified) function, of class formulafn, is returned with its attributes giving the (new) model function, the covariate names, and the parameter names.

Author(s)

J.K. Lindsey

See Also

FormulaMethods,covariates, finterp, model, parameters

Examples

```
fn <- function(p) a+b*x
fnenvir(fn)
fn <- function(p) a+p*x
fnenvir(fn)
x <- 1:4
fnenvir(fn)
fn <- function(p) p[1]+exp(p[2]*x)
fnenvir(fn)
#
y <- matrix(rnorm(20),ncol=5)
y[3,3] <- y[2,2] <- NA
resp <- restovec(y)
xx <- tcctomat(x)
z1 <- matrix(rnorm(20),ncol=5)
z2 <- matrix(rnorm(20),ncol=5)
z3 <- matrix(rnorm(20),ncol=5)
zz <- tvctomat(z1)
zz <- tvctomat(z2,old=zz)
reps <- rmna(resp, ccov=xx, tvcov=zz)
rm(y, x, z1, z2)
#
# repeated objects
func1 <- function(p) p[1]+p[2]*x+p[3]*z2
print(fn1 <- fnenvir(func1, .envir=reps))
fn1(2:4)
#
# time-constant covariates
func2 <- function(p) p[1]+p[2]*x
print(fn2 <- fnenvir(func2, .envir=reps))
fn2(2:3)
print(fn2a <- fnenvir(func2, .envir=xx))
fn2a(2:3)
#
# time-varying covariates
func3 <- function(p) p[1]+p[2]*z1+p[3]*z2
print(fn3 <- fnenvir(func3, .envir=reps))
fn3(2:4)
print(fn3a <- fnenvir(func3, .envir=zz))
fn3a(2:4)
# including times
func3b <- function(p) p[1]+p[2]*z1+p[3]*z2+p[4]*times
print(fn3b <- fnenvir(func3b, .envir=reps))
fn3b(2:5)
#
# with typing error and a variable not in the data object
func4 <- function(p) p[1]+p2[2]*z1+p[3]*z2+p[4]*z3
print(fn4 <- fnenvir(func4, .envir=reps))
#
# first-order one-compartment model
# data objects for formulae
dose <- c(2,5)
```

```
dd <- tcctomat(dose)
times <- matrix(rep(1:20,2), nrow=2, byrow=TRUE)
tt <- tvctomat(times)
# vector covariates for functions
dose <- c(rep(2,20),rep(5,20))
times <- rep(1:20,2)
# functions
mu <- function(p) {
absorption <- exp(p[1])
elimination <- exp(p[2])
absorption*exp(-p[3])*dose/(absorption-elimination)*
(exp(-elimination*times)-exp(-absorption*times))}
shape <- function(p) exp(p[1]-p[2])*times*dose*exp(-exp(p[1])*times)
# response
conc <- matrix(rgamma(40,shape(log(c(0.1,0.4))),
scale=mu(log(c(1,0.3,0.2))))/shape(log(c(0.1,0.4))),ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
ncol=20,byrow=TRUE)[,1:19])
conc <- restovec(ifelse(conc>0,conc,0.01))
reps <- rmna(conc, ccov=dd, tvcov=tt)
#
print(fn5 <- fnenvir(mu,.envir=reps))
fn5(c(0,-1.2,-1.6))
```

FormulaMethods *Methods for formulafn Functions*

Description

Methods for accessing the contents of a function created from formula produced by `finterp` or a function modified by `fnenvir`.

`covariates`: extract the names of the covariates.

`formula`: extract the formula used to produce the function (`finterp` only).

`model`: extract the model function or model matrix if W&R notation was used.

`parameters`: extract the names of the parameters.

Usage

```
## S3 method for class 'formulafn'
covariates(z, ...)
## S3 method for class 'formulafn'
formula(x, ...)
model(z, ...)
parameters(z, ...)
## S3 method for class 'formulafn'
print(x, ...)
```

Arguments

`x,z` A function of class, `formulafn`.

`...` Arguments to other functions.

Value

These methods extract information about functions of class, `formulafn`, created by `finterp` or `fnenvir`.

Author(s)

J.K. Lindsey

See Also

`finterp`, `fnenvir`.

Examples

```
x1 <- rpois(20,2)
x2 <- rnorm(20)
#
# Wilkinson and Rogers formula with three parameters
fn1 <- finterp(~x1+x2)
fn1
covariates(fn1)
formula(fn1)
model(fn1)
parameters(fn1)
#
# nonlinear formula with unknowns
fn2 <- finterp(~exp(b0+b1*x1+b2*x2))
fn2
covariates(fn2)
formula(fn2)
model(fn2)
parameters(fn2)
#
# function transformed by fnenvir
fn3 <- fnenvir(function(p) p[1]+p[2]*x1)
covariates(fn3)
formula(fn3)
model(fn3)
parameters(fn3)
```

Gamma Count — *Gamma Count Distribution*

Description

These functions provide information about the gamma count distribution with parameters `m` and `s`: density, cumulative distribution, quantiles, and random generation.

The gamma count distribution with `prob` $= m$ has density

$$p(y) = pgamma(\mu\sigma, y\sigma, 1) - pgamma(\mu\sigma, (y+1)\sigma, 1)$$

for $y = 0, \ldots, n$ where $pgamma(\mu\sigma, 0, 1) = 1$.

Usage

```
dgammacount(y, m, s, log=FALSE)
pgammacount(q, m, s)
qgammacount(p, m, s)
rgammacount(n, m, s)
```

Arguments

`y`	vector of frequencies
`q`	vector of quantiles
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of probabilities
`s`	vector of overdispersion parameters
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dpois` for the Poisson, `dconsul` for the Consul generalized Poisson, `ddoublepois` for the double Poisson, `dmultpois` for the multiplicative Poisson distributions, and `dnbinom` for the negative binomial distribution.

Examples

```
dgammacount(5,10,0.9)
pgammacount(5,10,0.9)
qgammacount(0.08,10,0.9)
rgammacount(10,10,0.9)
```

`gauss.hermite` *Calculate Gauss-Hermite Quadrature Points*

Description

`gauss.hermite` calculates the Gauss-Hermite quadrature values for a specified number of points.

Usage

```
gauss.hermite(points, iterlim=10)
```

Arguments

`points`	The number of points.
`iterlim`	Maximum number of iterations in Newton-Raphson.

Value

`gauss.hermite` returns a two-column matrix containing the points and their corresponding weights.

Author(s)

J.K. Lindsey

Examples

```
gauss.hermite(10)
```

`Generalized Extreme Value` *Generalized Extreme Value Distribution*

Description

These functions provide information about the generalized extreme value distribution with location parameter equal to `m`, dispersion equal to `s`, and family parameter equal to `f`: density, cumulative distribution, quantiles, log hazard, and random generation.

The generalized extreme value distribution has density

$$f(y) = y^{\nu-1} \exp(y^\nu/\nu) \frac{\sigma}{\mu} \frac{\exp(y^\nu/\nu)}{\mu^{\sigma-1}/(1 - I(\nu > 0) + sign(\nu) exp(-\mu^- \sigma))} \exp(-(\exp(y^\nu \nu)/\mu)^\sigma)$$

where μ is the location parameter of the distribution, σ is the dispersion, ν is the family parameter, $I()$ is the indicator function, and $y > 0$.

$\nu = 1$ a truncated extreme value distribution.

Usage

```
dgextval(y, s, m, f, log=FALSE)
pgextval(q, s, m, f)
qgextval(p, s, m, f)
rgextval(n, s, m, f)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of location parameters.
s	vector of dispersion parameters.
f	vector of family parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dweibull for the Weibull distribution.

Examples

```
dgextval(1, 2, 1, 2)
pgextval(1, 2, 1, 2)
qgextval(0.82, 2, 1, 2)
rgextval(10, 2, 1, 2)
```

Generalized Gamma *Generalized Gamma Distribution*

Description

These functions provide information about the generalized gamma distribution with scale parameter equal to m, shape equal to s, and family parameter equal to f: density, cumulative distribution, quantiles, log hazard, and random generation.

The generalized gamma distribution has density

$$f(y) = \frac{\nu y^{\nu-1}}{(\mu/\sigma)^{\nu\sigma} Gamma(\sigma)} y^{\nu(\sigma-1)} \exp(-(y\sigma/\mu)^{\nu})$$

where μ is the scale parameter of the distribution, σ is the shape, and ν is the family parameter.

$\nu = 1$ yields a gamma distribution, $\sigma = 1$ a Weibull distribution, and $\sigma = \infty$ a log normal distribution.

Usage

```
dggamma(y, s, m, f, log=FALSE)
pggamma(q, s, m, f)
qggamma(p, s, m, f)
rggamma(n, s, m, f)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of location parameters.
s	vector of dispersion parameters.
f	vector of family parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dgamma` for the gamma distribution, `dweibull` for the Weibull distribution, `dlnorm` for the log normal distribution.

Examples

```
dggamma(2, 5, 4, 2)
pggamma(2, 5, 4, 2)
qggamma(0.75, 5, 4, 2)
rggamma(10, 5, 4, 2)
```

`Generalized Inverse Gaussian`
Generalized Inverse Gaussian Distribution

Description

These functions provide information about the generalized inverse Gaussian distribution with mean equal to `m`, dispersion equal to `s`, and family parameter equal to `f`: density, cumulative distribution, quantiles, log hazard, and random generation.

The generalized inverse Gaussian distribution has density

$$f(y) = \frac{y^{\nu-1}}{2\mu^{\nu} K(1/(\sigma\mu), abs(\nu))} \exp(-(1/y + y/\mu^2)/(2 * \sigma))$$

where μ is the mean of the distribution, σ the dispersion, ν is the family parameter, and $K()$ is the fractional Bessel function of the third kind.

$\nu = -1/2$ yields an inverse Gaussian distribution, $\sigma = \infty$, $\nu > 0$ a gamma distribution, and $\nu = 0$ a hyperbola distribution.

Usage

```
dginvgauss(y, m, s, f, log=FALSE)
pginvgauss(q, m, s, f)
qginvgauss(p, m, s, f)
rginvgauss(n, m, s, f)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of means.
s	vector of dispersion parameters.
f	vector of family parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dinvgauss for the inverse Gaussian distribution.

Examples

```
dginvgauss(10, 3, 1, 1)
pginvgauss(10, 3, 1, 1)
qginvgauss(0.4, 3, 1, 1)
rginvgauss(10, 3, 1, 1)
```

`Generalized Logistic` *Generalized Logistic Distribution*

Description

These functions provide information about the generalized logistic distribution with location parameter equal to `m`, dispersion equal to `s`, and family parameter equal to `f`: density, cumulative distribution, quantiles, log hazard, and random generation.

The generalized logistic distribution has density

$$f(y) = \frac{\nu\sqrt{3}\exp(-\sqrt{3}(y-\mu)/(\sigma\pi))}{\sigma\pi(1+\exp(-\sqrt{3}(y-\mu)/(\sigma\pi)))^{\nu+1}}$$

where μ is the location parameter of the distribution, σ is the dispersion, and ν is the family parameter.

$\nu = 1$ gives a logistic distribution.

Usage

```
dglogis(y, m=0, s=1, f=1, log=FALSE)
pglogis(q, m=0, s=1, f=1)
qglogis(p, m=0, s=1, f=1)
rglogis(n, m=0, s=1, f=1)
```

Arguments

`y`	vector of responses.
`q`	vector of quantiles.
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of location parameters.
`s`	vector of dispersion parameters.
`f`	vector of family parameters.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dlogis` for the logistic distribution.

Examples

```
dglogis(5, 5, 1, 2)
pglogis(5, 5, 1, 2)
qglogis(0.25, 5, 1, 2)
rglogis(10, 5, 1, 2)
```

Generalized Weibull *Generalized Weibull Distribution*

Description

These functions provide information about the generalized Weibull distribution, also called the exponentiated Weibull, with scale parameter equal to m, shape equal to s, and family parameter equal to f: density, cumulative distribution, quantiles, log hazard, and random generation.

The generalized Weibull distribution has density

$$f(y) = \frac{\sigma \nu y^{\sigma-1}(1-\exp(-(y/\mu)^{\sigma}))^{\nu-1}\exp(-(y/\mu)^{\sigma})}{\mu^{\sigma}}$$

where μ is the scale parameter of the distribution, σ is the shape, and ν is the family parameter.

$\nu = 1$ gives a Weibull distribution, for $\sigma = 1$, $\nu < 0$ a generalized F distribution, and for $\sigma > 0$, $\nu \leq 0$ a Burr type XII distribution.

Usage

```
dgweibull(y, s, m, f, log=FALSE)
pgweibull(q, s, m, f)
qgweibull(p, s, m, f)
rgweibull(n, s, m, f)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of location parameters.
s	vector of dispersion parameters.
f	vector of family parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dweibull for the Weibull distribution, df for the F distribution, dburr for the Burr distribution.

Examples

```
dgweibull(5, 1, 3, 2)
pgweibull(5, 1, 3, 2)
qgweibull(0.65, 1, 3, 2)
rgweibull(10, 1, 3, 2)
```

---

gettvc *Find the Most Recent Value of a Time-varying Covariate Before Each Observed Response*

Description

gettvc finds the most recent value of a time-varying covariate before each observed response and possibly adds them to a list of other time-varying covariates. It compares the times of response observations with those of time-varying covariates to find the most recent observed time-varying covariate for each response. These are either placed in a new object of class, tvcov, added to an already existing list of matrices containing other time-varying covariates and a new object of class, tvcov, created, or added to an existing object of class, tvcov.

If there are response observation times before the first covariate time, the covariate for these times is set to zero.

Usage

```
gettvc(response, times=NULL, tvcov=NULL, tvctimes=NULL,
oldtvcov=NULL, ties=TRUE)
```

Arguments

response	A list of two column matrices with response values and times for each individual, one matrix or dataframe of response values, or an object of class, response (created by restovec).
times	When response is a matrix, a vector of possibly unequally spaced times for the response, when they are the same for all individuals or a matrix of times. Not necessary if equally spaced.
tvcov	A list of two column matrices with time-varying covariate values and corresponding times for each individual or one matrix or dataframe of such covariate values. Times need not be the same as for responses.
tvctimes	When the time-varying covariate is a matrix, a vector of possibly unequally spaced times for the covariate, when they are the same for all individuals or a matrix of times. Not necessary if equally spaced.

`oldtvcov` A list of matrices with time-varying covariate values, observed at the event times in `response`, for each individual, or an object of class, `tvcov`. If not provided, a new object is created.

`ties` If TRUE, when the response and covariate times are identical, the response depends on that new value (as in observational studies); if FALSE, only the next response depends on that value (for example, if the covariate is a new treatment just applied at that time).

Value

An object of class, `tvcov`, is returned containing the new time-varying covariate and, possibly, those in `oldtvcov`.

Author(s)

J.K. Lindsey and D.F. Heitjan

See Also

`read.list`, `restovec`, `tvctomat`.

Examples

```
## Not run:
y <- matrix(rnorm(20), ncol=5)
resp <- restovec(y, times=c(1,3,6,10,15))
z <- matrix(rpois(20,5),ncol=5)
z
# create a new time-varying covariate object for the response
newtvc <- gettvc(resp, tvcov=z, tvctimes=c(1,2,5,12,14))
covariates(newtvc)
# add another time-varying covariate to the object
z2 <- matrix(rpois(20,5),ncol=5)
z2
newtvc2 <- gettvc(resp, tvcov=z2, tvctimes=c(0,4,5,12,16), oldtvc=newtvc)
covariates(newtvc2)

## End(Not run)
```

`Hjorth` *Hjorth Distribution*

Description

These functions provide information about the Hjorth distribution with location parameter equal to `m`, dispersion equal to `s`, and family parameter equal to `f`: density, cumulative distribution, quantiles, log hazard, and random generation.

The Hjorth distribution has density

$$f(y) = (1 + \sigma y)^{-\nu/\sigma} \exp(-(y/\mu)^2/2)(\frac{y}{\mu^2} + \frac{\nu}{1 + \sigma y})$$

where μ is the location parameter of the distribution, σ is the dispersion, and ν is the family parameter.

Usage

```
dhjorth(y, m, s, f, log=FALSE)
phjorth(q, m, s, f)
qhjorth(p, m, s, f)
rhjorth(n, m, s, f)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of location parameters.
s	vector of dispersion parameters.
f	vector of family parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

Examples

```
dhjorth(5, 5, 5, 2)
phjorth(5, 5, 5, 2)
qhjorth(0.8, 5, 5, 2)
rhjorth(10, 5, 5, 2)
```

---

int *Vectorized Numerical Integration*

Description

`int` performs numerical integration of a given function using either Romberg integration or algorithm 614 of the collected algorithms from ACM. Only the former is vectorized. The latter uses formulae optimal in certain Hardy spaces h(p,d).

Functions may have singularities at one or both end-points of the interval (a,b).

Usage

```
int(f, a=-Inf, b=Inf, type="Romberg", eps=0.0001, max=NULL, d=NULL, p=0)
```

Arguments

f	The function (of one variable) to integrate, returning either a scalar or a vector.
a	A scalar or vector (only Romberg) giving the lower bound(s). A vector cannot contain both -Inf and finite values.
b	A scalar or vector (only Romberg) giving the upper bound(s). A vector cannot contain both Inf and finite values.
type	The algorithm to be used, by default Romberg integration. Otherwise, it uses the TOMS614 algorithm.
eps	Precision.
max	For Romberg, the maximum number of steps, by default set to 16. For TOMS614, the maximum number of function evaluations, by default set to 100.
d	For Romberg, the number of extrapolation points so that 2d is the order of integration, by default set to 5; d=2 is Simpson's rule. For TOMS614, heuristic termination = any real number; deterministic termination = a number in the range 0 < d < pi/2 by default, set to 1.
p	For TOMS614, p = 0: heuristic termination, p = 1: deterministic termination with the infinity norm, p > 1: deterministic termination with the p-th norm.

Value

The vector of values of the integrals of the function supplied.

Author(s)

J.K. Lindsey

References

ACM algorithm 614 appeared in

ACM-Trans. Math. Software, Vol.10, No. 2, Jun., 1984, p. 152-160.

See also

Sikorski,K., Optimal quadrature algorithms in HP spaces, Num. Math., 39, 405-410 (1982).

Examples

```
f <- function(x) sin(x)+cos(x)-x^2
int(f, a=0, b=2)
int(f, a=0, b=2, type="TOMS614")
#
f <- function(x) exp(-(x-2)^2/2)/sqrt(2*pi)
int(f, a=0:3)
int(f, a=0:3, d=2)
1-pnorm(0:3, 2)
```

int2 *Vectorized Two-dimensional Numerical Integration*

Description

int performs vectorized numerical integration of a given two-dimensional function.

Usage

```
int2(f, a=c(-Inf,-Inf), b=c(Inf,Inf), eps=1.0e-6, max=16, d=5)
```

Arguments

f	The function (of two variables) to integrate, returning either a scalar or a vector.
a	A two-element vector or a two-column matrix giving the lower bounds. It cannot contain both -Inf and finite values.
b	A two-element vector or a two-column matrix giving the upper bounds. It cannot contain both Inf and finite values.
eps	Precision.
max	The maximum number of steps, by default set to 16.
d	The number of extrapolation points so that 2k is the order of integration, by default set to 5; d=2 is Simpson's rule.

Value

The vector of values of the integrals of the function supplied.

Author(s)

J.K. Lindsey

Examples

```
f <- function(x,y) sin(x)+cos(y)-x^2
int2(f, a=c(0,1), b=c(2,4))
#
fn1 <- function(x, y) x^2+y^2
fn2 <- function(x, y) (1:4)*x^2+(2:5)*y^2
int2(fn1, c(1,2), c(2,4))
int2(fn2, c(1,2), c(2,4))
int2(fn1, matrix(c(1:4,1:4),ncol=2), matrix(c(2:5,2:5),ncol=2))
int2(fn2, matrix(c(1:4,1:4),ncol=2), matrix(c(2:5,2:5),ncol=2))
```

Inverse Gaussian *Inverse Gaussian Distribution*

Description

These functions provide information about the inverse Gaussian distribution with mean equal to m and dispersion equal to s: density, cumulative distribution, quantiles, log hazard, and random generation.

The inverse Gaussian distribution has density

$$f(y) = \frac{1}{\sqrt{2\pi\sigma y^3}} e^{-(y-\mu)^2/(2y\sigma m^2)}$$

where μ is the mean of the distribution and σ is the dispersion.

Usage

```
dinvgauss(y, m, s, log=FALSE)
pinvgauss(q, m, s)
qinvgauss(p, m, s)
rinvgauss(n, m, s)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of means.
s	vector of dispersion parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dnorm for the normal distribution and dlnorm for the *Log*normal distribution.

Examples

```
dinvgauss(5, 5, 1)
pinvgauss(5, 5, 1)
qinvgauss(0.8, 5, 1)
rinvgauss(10, 5, 1)
```

iprofile *Produce Individual Time Profiles for Plotting*

Description

iprofile is used for plotting individual profiles over time for objects obtained from dynamic models. It produces output for plotting recursive fitted values for individual time profiles from such models.

See mprofile for plotting marginal profiles.

Usage

```
## S3 method for class 'iprofile'
plot(x, nind=1, observed=TRUE, intensity=FALSE,
add=FALSE, lty=NULL, pch=NULL, ylab=NULL, xlab=NULL,
main=NULL, ylim=NULL, xlim=NULL, ...)
```

Arguments

x	An object of class iprofile, e.g. x = iprofile(z,plotsd=FALSE), where z is an object of class recursive, from carma, elliptic, gar, kalcount, kalseries, kalsurv, or nbkal. If plotsd is If TRUE, plots standard deviations around profile (carma and elliptic only).
nind	Observation number(s) of individual(s) to be plotted.
observed	If TRUE, plots observed responses.
intensity	If z has class, kalsurv, and this is TRUE, the intensity is plotted instead of the time between events.
add	If TRUE, the graph is added to an existing plot.
lty,pch,main,ylim,xlim,xlab,ylab	See base plot.
...	Arguments passed to other functions.

Value

iprofile returns information ready for plotting by plot.iprofile.

Author(s)

J.K. Lindsey

See Also

mprofile plot.residuals.

Examples

```
## Not run:
## try this after you have repeated package installed
library(repeated)
times <- rep(1:20,2)
dose <- c(rep(2,20),rep(5,20))
mu <- function(p) exp(p[1]-p[3])*(dose/(exp(p[1])-exp(p[2]))*
(exp(-exp(p[2])*times)-exp(-exp(p[1])*times)))
shape <- function(p) exp(p[1]-p[2])*times*dose*exp(-exp(p[1])*times)
conc <- matrix(rgamma(40,1,scale=mu(log(c(1,0.3,0.2)))),ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
ncol=20,byrow=TRUE)[,1:19])
conc <- ifelse(conc>0,conc,0.01)
z <- gar(conc, dist="gamma", times=1:20, mu=mu, shape=shape,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=log(c(1,0.2)))
# plot individual profiles and the average profile
plot(iprofile(z), nind=1:2, pch=c(1,20), lty=3:4)
plot(mprofile(z), nind=1:2, lty=1:2, add=TRUE)

## End(Not run)
```

Laplace *Laplace Distribution*

Description

These functions provide information about the Laplace distribution with location parameter equal to m and dispersion equal to s: density, cumulative distribution, quantiles, log hazard, and random generation.

The Laplace distribution has density

$$f(y) = \frac{\exp(-abs(y-\mu)/\sigma)}{(2\sigma)}$$

where μ is the location parameter of the distribution and σ is the dispersion.

Usage

```
dlaplace(y, m=0, s=1, log=FALSE)
plaplace(q, m=0, s=1)
qlaplace(p, m=0, s=1)
rlaplace(n=1, m=0, s=1)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities

n	number of values to generate
m	vector of location parameters.
s	vector of dispersion parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dexp for the exponential distribution and dcauchy for the Cauchy distribution.

Examples

```
dlaplace(5, 2, 1)
plaplace(5, 2, 1)
qlaplace(0.95, 2, 1)
rlaplace(10, 2, 1)
```

Levy — *Levy Distribution*

Description

These functions provide information about the Levy distribution with location parameter equal to m and dispersion equal to s: density, cumulative distribution, quantiles, and random generation.

The Levy distribution has density

$$f(y) = \sqrt{\frac{\sigma}{2\pi(y-\mu)^3}}\exp(-\sigma/(2(y-\mu)))$$

where μ is the location parameter of the distribution and σ is the dispersion, and $y > \mu$.

Usage

```
dlevy(y, m=0, s=1, log=FALSE)
plevy(q, m=0, s=1)
qlevy(p, m=0, s=1)
rlevy(n, m=0, s=1)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of location parameters.
s	vector of dispersion parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dnorm for the normal distribution and dcauchy for the Cauchy distribution, two other stable distributions.

Examples

```
dlevy(5, 2, 1)
plevy(5, 2, 1)
qlevy(0.6, 2, 1)
rlevy(10, 2, 1)
```

---

lin.diff.eqn *Solution of Autonomous Linear Differential Equations*

Description

lin.diff.eqn numerically solves a system of autonomous linear differential equations with given initial conditions by matrix exponentiation.

Usage

```
lin.diff.eqn(A, initial, t=1)
```

Arguments

A	A square matrix giving the coefficients of the equations.
initial	The vector of initial values of the system.
t	A scalar or vector of values of the independent variable for which solutions are sought.

Value

A matrix of solutions with one row for each value of t.

Author(s)

J.K. Lindsey

Examples

```
a <- matrix(c(1,0,1,0,0,0,0,0,-1),ncol=3,byrow=TRUE)
x <- c(5,7,6)
lin.diff.eqn(a,x,1)
# function giving the exact solution
exact <- function(t) c(8*exp(t)-3*exp(-t),7,6*exp(-t))
exact(1)
```

---

lvna — *Create a repeated Object, Leaving NAs*

Description

lvna forms an object of class, repeated, from a response object and possibly time-varying or intra-individual covariate (tvcov), and time-constant or inter-individual covariate (tccov) objects. If there are NAs in any variables, it also creates a logical vector indicating which observations have NAs either in the response or the covariate values. Subjects must be in the same order in all (three) objects to be combined.

Such objects can be printed and plotted. Methods are available for extracting the response, the numbers of observations per individual, the times, the weights, the units of measurement/Jacobian, the nesting variable, the covariates, and their names: response, nobs, times, weights, delta, nesting, covariates, and names.

Usage

```
lvna(response, ccov=NULL, tvcov=NULL)
```

Arguments

response	An object of class, response (created by restovec), containing the response variable information.
ccov	An object of class, tccov (created by tcctomat), containing the time-constant or inter-individual covariate information.
tvcov	An object of class, tvcov (created by tvctomat), containing the time-varying or intra-individual covariate information.

Value

Returns an object of class, `repeated`, containing a list of the response object (`z$response`, so that, for example, the response vector is `z$response$y`; see `restovec`), possibly the two classes of covariate objects (`z$ccov` and `z$tvcov`; see `tcctomat` and `tvctomat`), and a logical vector (`z$NAs`) indicating which observations have an NA in the response or some covariate.

Author(s)

J.K. Lindsey

See Also

`DataMethods`, `covariates`, `covind`, `delta`, `dftorep`, `names`, `nesting`, `nobs`, `read.list`, `read.surv`, `response`, `resptype`, `restovec`, `rmna`, `tcctomat`, `times`, `transform`, `tvctomat`, `units`, `weights`

Examples

```
y <- matrix(rnorm(20),ncol=5)
y[2,3] <- NA
tt <- c(1,3,6,10,15)
print(resp <- restovec(y,times=tt))
x <- c(0,0,1,1)
tcc <- tcctomat(x)
z <- matrix(rpois(20,5),ncol=5)
tvc <- tvctomat(z)
print(reps <- lvna(resp, tvcov=tvc, ccov=tcc))
response(reps)
response(reps, nind=2:3)
times(reps)
nobs(reps)
weights(reps)
covariates(reps)
covariates(reps,names="x")
covariates(reps,names="z")
names(reps)
nesting(reps)
# because individuals are the only nesting, this is the same as
covind(reps)
# binomial
y <- matrix(rpois(20,5),ncol=5)
y[2,3] <- NA
print(respb <- restovec(y,totals=y+matrix(rpois(20,5),ncol=5),times=tt))
print(repsb <- lvna(respb, tvcov=tvc, ccov=tcc))
response(repsb)
# censored data
y <- matrix(rweibull(20,2,5),ncol=5)
print(respc <- restovec(y,censor=matrix(rbinom(20,1,0.9),ncol=5),times=tt))
print(repsc <- lvna(respc, tvcov=tvc, ccov=tcc))
# if there is no censoring, censor indicator is not printed
response(repsc)
# nesting clustered within individuals
```

```
nest <- c(1,1,2,2,2)
print(respn <- restovec(y,censor=matrix(rbinom(20,1,0.9),ncol=5),
times=tt,nest=nest))
print(repsn <- lvna(respn, tvcov=tvc, ccov=tcc))
response(respn)
times(respn)
nesting(respn)
```

`mexp` *Matrix Exponentiation*

Description

`mexp` calculates `exp(t*x)` for the square matrix, `x`, by spectral decomposition or series expansion.

Usage

```
mexp(x, t=1, type="spectral decomposition", n=20, k=3)
```

Arguments

`x`	A square matrix.
`t`	Constant multiplying the matrix.
`type`	Algorithm used: spectral decomposition or series approximation.
`n`	Number of terms in the series expansion.
`k`	Constant divisor to avoid over- or underflow (series approximation only).

Value

`mexp` returns the exponential of a matrix.

Author(s)

J.K. Lindsey

Examples

```
x <- matrix(c(1,2,3,4),nrow=2)
mexp(x)
```

mpower *Power of a Matrix*

Description

`%^%` calculates `x^p` for the square matrix, `x`, by spectral decomposition.

Usage

```
x%^%p
```

Arguments

`x`	A square matrix.
`p`	The power to which the matrix is to be raised.

Value

`%^%` returns the power of a matrix.

Author(s)

J.K. Lindsey

Examples

```
## Not run:
x <- matrix(c(0.4,0.6,0.6,0.4),nrow=2)
x%^%2
x%^%10
x%^%20

## End(Not run)
```

mprofile *Produce Marginal Time Profiles for Plotting*

Description

`mprofile` is used for plotting marginal profiles over time for models obtained from dynamic models, for given fixed values of covariates. These are either obtained from those supplied by the model, if available, or from a function supplied by the user.

See `iprofile` for plotting individual profiles from recursive fitted values.

Usage

```
## S3 method for class 'mprofile'
plot(x, nind=1, intensity=FALSE, add=FALSE, ylim=range(z$pred, na.rm = TRUE),
lty=NULL, ylab=NULL, xlab=NULL, ...)
```

Arguments

`x`	An object of class `mprofile`, e.g. `x = mprofile(z,times=NULL,mu=NULL,ccov,plotse=TRUE)`, where `z`An object of class `recursive`, from `carma`, `elliptic`, `gar`, `kalcount`, `kalseries`, `kalsurv`, or `nbkal`; `times` is a vector of time points at which profiles are to be plotted; `mu` is the location regression as a function of the parameters and the times for the desired covariate values; `ccov` is covariate values for the profiles (`carma` only); and `plotse` when TRUE plots standard errors (`carma` only).
`nind`	Observation number(s) of individual(s) to be plotted. (Not used if `mu` is supplied.)
`intensity`	If TRUE, the intensity is plotted instead of the time between events. Only for models produced by `kalsurv`.
`add`	If TRUE, add contour to previous plot instead of creating a new one.
`lty,ylim,xlab,ylab`	See base plot.
`...`	Arguments passed to other functions.

Value

`mprofile` returns information ready for plotting by `plot.mprofile`.

Author(s)

J.K. Lindsey

See Also

`iprofile`, `plot.residuals`.

Examples

```
## Not run:
## try after you get the repeated package
library(repeated)
times <- rep(1:20,2)
dose <- c(rep(2,20),rep(5,20))
mu <- function(p) exp(p[1]-p[3])*(dose/(exp(p[1])-exp(p[2]))*
(exp(-exp(p[2])*times)-exp(-exp(p[1])*times)))
shape <- function(p) exp(p[1]-p[2])*times*dose*exp(-exp(p[1])*times)
conc <- matrix(rgamma(40,1,scale=mu(log(c(1,0.3,0.2)))),ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
ncol=20,byrow=TRUE)[,1:19])
conc <- ifelse(conc>0,conc,0.01)
```

```
z <- gar(conc, dist="gamma", times=1:20, mu=mu, shape=shape,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=log(c(1,0.2)))
# plot individual profiles and the average profile
plot(iprofile(z), nind=1:2, pch=c(1,20), lty=3:4)
plot(mprofile(z), nind=1:2, lty=1:2, add=TRUE)

## End(Not run)
```

`Multiplicative Binomial` *Multiplicative Binomial Distribution*

Description

These functions provide information about the multiplicative binomial distribution with parameters `m` and `s`: density, cumulative distribution, quantiles, and random generation.

The multiplicative binomial distribution with total $= n$ and `prob` $= m$ has density

$$p(y) = c(n, m, s) \binom{n}{y} m^y (1-m)^{n-y} s^{(y(n-y))}$$

for $y = 0, \ldots, n$, where c(.) is a normalizing constant.

Usage

```
dmultbinom(y, size, m, s, log=FALSE)
pmultbinom(q, size, m, s)
qmultbinom(p, size, m, s)
rmultbinom(n, size, m, s)
```

Arguments

`y`	vector of frequencies
`q`	vector of quantiles
`p`	vector of probabilities
`n`	number of values to generate
`size`	vector of totals
`m`	vector of probabilities of success
`s`	vector of overdispersion parameters
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dbinom for the binomial, ddoublebinom for the double binomial, and dbetabinom for the beta binomial distribution.

Examples

```
# compute P(45 < y < 55) for y multiplicative binomial(100,0.5,1.1)
sum(dmultbinom(46:54, 100, 0.5, 1.1))
pmultbinom(54, 100, 0.5, 1.1)-pmultbinom(45, 100, 0.5, 1.1)
pmultbinom(2,10,0.5,1.1)
qmultbinom(0.025,10,0.5,1.1)
rmultbinom(10,10,0.5,1.1)
```

MultPoisson *Multiplicative Poisson Distribution*

Description

These functions provide information about the multiplicative Poisson distribution with parameters m and s: density, cumulative distribution, quantiles, and random generation.

The multiplicative Poisson distribution with mu $= m$ has density

$$p(y) = c(\mu, \lambda) \exp(-\mu) \mu^y \lambda^(y^2)/y!$$

with $s <= 1$ for $y = 0, \ldots$, where c(.) is a normalizing constant.

Note that it only allows for underdispersion, not being defined for $s > 1$.

Usage

```
dmultpois(y, m, s, log=FALSE)
pmultpois(q, m, s)
qmultpois(p, m, s)
rmultpois(n, m, s)
```

Arguments

y	vector of counts
q	vector of quantiles
p	vector of probabilities
n	number of values to generate
m	scalar or vector of means
s	scalar or vector of overdispersion parameters, all of which must lie in (0,1).
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dpois` for the Poisson, `ddoublepois` for the double Poisson, `dpvfpois` for the power variance function Poisson, `dconsul` for the Consul generalized Poisson, `dgammacount` for the gamma count, and `dnbinom` for the negative binomial distribution.

Examples

```
dmultpois(5,10,0.9)
pmultpois(5,10,0.9)
qmultpois(0.85,10,0.9)
rmultpois(10,10,0.9)
```

`Pareto` *Pareto Distribution*

Description

These functions provide information about the Pareto distribution with location parameter equal to `m` and dispersion equal to `s`: density, cumulative distribution, quantiles, log hazard, and random generation.

The Pareto distribution has density

$$f(y) = \frac{\sigma}{\mu(\sigma - 1)(1 + y/(\mu(\sigma - 1)))^{\sigma+1}}$$

where μ is the mean parameter of the distribution and σ is the dispersion.

This distribution can be obtained as a mixture distribution from the exponential distribution using a gamma mixing distribution.

Usage

```
dpareto(y, m, s, log=FALSE)
ppareto(q, m, s)
qpareto(p, m, s)
rpareto(n, m, s)
```

Arguments

`y`	vector of responses.
`q`	vector of quantiles.
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of location parameters.
`s`	vector of dispersion parameters.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dexp for the exponential distribution.

Examples

```
dpareto(5, 2, 2)
ppareto(5, 2, 2)
qpareto(0.9, 2, 2)
rpareto(10, 2, 2)
```

---

pkpd *Pharmacokinetic Compartment Models*

Description

Mean functions for use in fitting pharmacokineticcompartment models models.

`mu1.0o1c`: open zero-order one-compartment model

`mu1.1o1c`: open first-order one-compartment model

`mu1.1o2c`: open first-order two-compartment model (ordered)

`mu1.1o2cl`: open first-order two-compartment model (ordered, absorption and transfer equal)

`mu1.1o2cc`: open first-order two-compartment model (circular)

Simultaneous models for parent drug and metabolite:

`mu2.0o1c`: zero-order one-compartment model

`mu2.0o2c1`: zero-order two-compartment for parent, one-compartment for metabolite, model

`mu2.0o2c2`: zero-order two-compartment model for both parent and metabolite

`mu2.1o1c`: first-order one-compartment model

`mu2.0o1cfp`: zero-order one-compartment first-pass model

`mu2.0o2c1fp`: zero-order two-compartment for parent, one-compartment for metabolite, model with first-pass

`mu2.0o2c2fp`: zero-order two-compartment model for both parent and metabolite with first-pass

`mu2.1o1cfp`: first-order one-compartment first-pass model

Usage

```
mu1.0o1c(p, times, dose=1, end=0.5)
mu1.1o1c(p, times, dose=1)
mu1.1o2c(p, times, dose=1)
mu1.1o2cl(p, times, dose=1)
mu1.1o2cc(p, times, dose=1)
mu2.0o1c(p, times, dose=1, ind, end=0.5)
mu2.0o2c1(p, times, dose=1, ind, end=0.5)
mu2.0o2c2(p, times, dose=1, ind, end=0.5)
mu2.1o1c(p, times, dose=1, ind)
mu2.0o1cfp(p, times, dose=1, ind, end=0.5)
mu2.0o2c1fp(p, times, dose=1, ind, end=0.5)
mu2.0o2c2fp(p, times, dose=1, ind, end=0.5)
mu2.1o1cfp(p, times, dose=1, ind)
```

Arguments

`p`	Vector of parameters. See the source file for details.
`times`	Vector of times.
`dose`	Vector of dose levels.
`ind`	Indicator whether parent drug or metabolite.
`end`	Time infusion ends.

Value

The profile of mean concentrations for the given times and doses is returned.

Author(s)

J.K. Lindsey

Examples

```
## Not run:
library(repeated)
times <- rep(1:20,2)
dose <- c(rep(2,20),rep(5,20))
# set up a mean function for gar based on mu1.1o1c:
mu <- function(p) {
ka <- exp(p[2])
ke <- exp(p[3])
exp(p[2]-p[1])/(ka-ke)*(exp(-ke*times)-exp(-ka*times))}
conc <- matrix(rgamma(40,2,scale=mu(log(c(1,0.3,0.2)))/2),ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
ncol=20,byrow=TRUE)[,1:19])
conc <- ifelse(conc>0,conc,0.01)
gar(conc, dist="gamma", times=1:20, mu=mu, preg=log(c(1,0.4,0.1)),
pdepend=0.1, pshape=1)
# changing variance
```

```
shape <- mu
gar(conc, dist="gamma", times=1:20, mu=mu, preg=log(c(0.5,0.4,0.1)),
pdep=0.1, shape=shape, pshape=log(c(0.5,0.4,0.1)))

## End(Not run)
```

`plot.residuals` *Plot Residuals*

Description

`plot.residuals` is used for plotting residuals from models obtained from dynamic models for given subsets of the data.

Usage

```
## S3 method for class 'residuals'
plot(x, X=NULL, subset=NULL, ccov=NULL, nind=NULL,
recursive=TRUE, pch=20, ylab="Residual", xlab=NULL,
main=NULL, ...)
```

Arguments

`x`	An object of class recursive, from `carma`, `gar`, `kalcount`, `kalseries`, `kalsurv`, or `nbkal`.
`X`	Vector of of values for the x-axis. If missing, time is used. It can also be specified by the strings "response" or "fitted".
`subset`	A logical vector defining which observations are to be used.
`ccov`	If the name of a time-constant covariate is supplied, separate plots are made for each distinct value of that covariate.
`nind`	Observation number(s) of individual(s) to be plotted.
`recursive`	If TRUE, plot recursive residuals, otherwise ordinary residuals.
`pch,ylab,xlab,main,...`	Plotting control options.

Author(s)

J.K. Lindsey

See Also

`carma`, `gar`, `kalcount`, `kalseries`, `kalsurv`, `nbkal` `plot.iprofile`, `plot.mprofile`.

Examples

```
## Not run:
library(repeated)
times <- rep(1:20,2)
dose <- c(rep(2,20),rep(5,20))
mu <- function(p) exp(p[1]-p[3])*(dose/(exp(p[1])-exp(p[2]))*
(exp(-exp(p[2])*times)-exp(-exp(p[1])*times)))
shape <- function(p) exp(p[1]-p[2])*times*dose*exp(-exp(p[1])*times)
conc <- matrix(rgamma(40,2,scale=mu(log(c(1,0.3,0.2)))/2),ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
ncol=20,byrow=TRUE)[,1:19])
conc <- ifelse(conc>0,conc,0.01)
z <- gar(conc, dist="gamma", times=1:20, mu=mu, shape=shape,
preg=log(c(1,0.4,0.1)), pdepend=0.1, pshape=log(c(1,0.2)))
plot.residuals(z, subset=1:20, main="Dose 1")
plot.residuals(z, x="fitted", subset=1:20, main="Dose 1")
plot.residuals(z, x="response", subset=1:20, main="Dose 1")

## End(Not run)
```

PowerExponential *Power Exponential Distribution*

Description

These functions provide information about the power exponential distribution with mean parameter equal to m, dispersion equal to s, and family parameter equal to f: density, cumulative distribution, quantiles, log hazard, and random generation.

The power exponential distribution has density

$$f(y) = \frac{\exp(-(absy - \mu/\sqrt{\sigma})^{2\nu}/2)}{\sqrt{\sigma}Gamma(1 + 1/(2\nu))2^{1+1/(2\nu)}}$$

where μ is the mean of the distribution, σ is the dispersion, and ν is the family parameter. $\nu = 1$ yields a normal distribution, $\nu = 0.5$ a Laplace distribution, and $\nu = \infty$ a uniform distribution.

Usage

```
dpowexp(y, m=0, s=1, f=1, log=FALSE)
ppowexp(q, m=0, s=1, f=1)
qpowexp(p, m=0, s=1, f=1)
rpowexp(n, m=0, s=1, f=1)
```

Arguments

y	vector of responses.
q	vector of quantiles.

`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of means.
`s`	vector of dispersion parameters.
`f`	vector of family parameters.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

Examples

```
dpowexp(5, 5, 1, 2)
ppowexp(5, 5, 1, 2)
qpowexp(0.5, 5, 1, 2)
rpowexp(10, 5, 1, 2)
```

`PvfPoisson` *Power Variance Function Poisson Distribution*

Description

These functions provide information about the overdispersed power variance function Poisson distribution with parameters `m`, `s`, and `f`: density, cumulative distribution, quantiles, and random generation. This function is obtained from a Poisson distribution as a mixture with a power variance distribution. In the limit, for `f=0`, the mixing distribution is gamma so that it is a negative binomial distribution. For `f=0.5`, the mixing distribution is inverse Gaussian. For `f<0`, the mixing distribution is a compound distribution of the sum of a Poisson number of gamma distributions. For `f=1`, it is undefined.

The power variance function Poisson distribution with `m` $= \mu$, the mean, `s` $= \theta$, and `f` $= \alpha$ has density

$$p(y) = \frac{\exp(-\mu((\theta+1)^{\alpha}/\theta^{\alpha} - \theta)/\alpha)}{y!} \sum_{i=1}^{y} c_{yi}(\alpha)\mu^{i}(\theta+1)^{i\alpha-y}/\theta^{i(\alpha-1)}$$

for $y = 0, \ldots$, where `c_{yi}(f)` are coefficients obtained by recursion.

Usage

```
dpvfpois(y, m, s, f, log=FALSE)
ppvfpois(q, m, s, f)
qpvfpois(p, m, s, f)
rpvfpois(n, m, s, f)
```

Arguments

`y`	vector of counts
`q`	vector of quantiles
`p`	vector of probabilities
`n`	number of values to generate
`m`	scalar or vector of means
`s`	scalar or vector of overdispersion parameters
`f`	scalar or vector of family parameters, all < 1
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dpois` for the Poisson, `ddoublepois` for the double Poisson, `dmultpois` for the multiplicative Poisson, `dconsul` for the Consul generalized Poisson, `dgammacount` for the gamma count, and `dnbinom` for the negative binomial distribution.

Examples

```
dpvfpois(5,10,0.9,0.5)
ppvfpois(5,10,0.9,0.5)
qpvfpois(0.85,10,0.9,0.5)
rpvfpois(10,10,0.9,0.5)
```

`read.list` *Read a List of Matrices from a File for Unbalanced Repeated Measurements*

Description

`read.list` reads sets of lines of data from a file and creates a list of matrices. Different sets of lines may be have different lengths.

Usage

```
read.list(file="", skip=0, nlines=2, order=NULL)
```

Arguments

`file`	Name of the file to read
`skip`	Number of lines to skip at the beginning of the file
`nlines`	Number of lines per matrix
`order`	Order in which the lines are to be used as columns of the matrix. If NULL, they are placed in the order read.

Value

The list of matrices, each with `nlines` columns, is returned.

Author(s)

J.K. Lindsey

See Also

lvna, read.rep, read.surv, restovec, rmna, tvctomat

Examples

```
## Not run: y <- read.list("test.dat")
```

---

read.rep — *Read a Rectangular Data Set from a File to Create a repeated Object*

Description

`dftorep` forms an object of class, `repeated`, from data read from a file with the option of removing any observations where response and covariate values have NAs. For repeated measurements, observations on the same individual must be together in the file. A number of validity checks are performed on the data.

Such objects can be printed and plotted. Methods are available for extracting the response, the numbers of observations per individual, the times, the weights, the units of measurement/Jacobian, the nesting variable, the covariates, and their names: response, nobs, times, weights, delta, nesting, covariates, and names.

Usage

```
read.rep(file, header=TRUE, skip=0, sep = "", na.strings="NA",
response, id=NULL, times=NULL, censor=NULL, totals=NULL,
weights=NULL, nest=NULL, delta=NULL, coordinates=NULL,
type=NULL, ccov=NULL, tvcov=NULL, na.rm=TRUE)
```

Arguments

`file`	A file name from which to read the data with variables as columns and observations as rows.
`header`	A logical value indicating whether the file contains the names of the variables as the line before the first row of data.
`skip`	The number of lines of the file to skip before beginning to read data.
`sep`	The field separator character. Values on each line of the file are separated by this character.
`na.strings`	A vector of strings defining what values are to be assigned NA.

response
: A character vector giving the column name(s) of the dataframe for the response variable(s).

id
: A character vector giving the column name of the dataframe for the identification numbers of the individuals. If the numbers are not consecutive integers, a warning is given.

  If NULL, one observation per individual is assumed if `times` is also NULL, other time series is assumed.

times
: An optional character vector giving the column name of the dataframe for the times vector.

censor
: An optional character vector giving the column name(s) of the dataframe for the censor indicator(s). This must be the same length as `response`. Responses without censor indicator can have a column either of all NAs or all 1s.

totals
: An optional character vector giving the column name(s) of the dataframe for the totals for binomial data. This must be the same length as `response`. Responses without censor indicator can have a column all NAs.

weights
: An optional character vector giving the column name of the dataframe for the weights vector.

nest
: An optional character vector giving the column name of the dataframe for the nesting vector within individuals.

  This is the second level of nesting for repeated measurements, with the individual being the first level. Values for an individual must be consecutive increasing integers.

delta
: An optional character vector giving the column name(s) of the dataframe for the units of measurement/Jacobian(s) of the response(s). This must be the same length as `response`. Responses without units of measurement/Jacobian can have a column all NAs.

  If all response variables have the same unit of measurement, this can be that one number. If each response variable has the same unit of measurement for all its values, this can be a numeric vector of length the number of response variables.

coordinates
: An optional character vector giving the two or three column name(s) of the dataframe for the spatial coordinates.

type
: An optional character vector giving the types of response variables: nominal, ordinal, discrete, duration, continuous, multivariate, or unknown.

ccov
: An optional character vector giving the column names of the dataframe for the time-constant or inter-individual covariates. For repeated measurements, if the value is not constant for all observations on an individual, an error is produced.

tvcov
: An optional character vector giving the column names of the dataframe for the time-varying or intra-individual covariates.

na.rm
: If TRUE, observations with NAs in any variables selected are removed in the object returned. Otherwise, the corresponding indicator variable is returned in a slot in the object.

Value

Returns an object of class, `repeated`, containing a list of the response object (`z$response`, so that, for example, the response vector is `z$response$y`; see `restovec`), and possibly the two classes of covariate objects (`z$ccov` and `z$tvcov`; see `tcctomat` and `tvctomat`).

Author(s)

J.K. Lindsey

See Also

`dftorep`, `lvna`, `read.list`, `restovec`, `rmna`, `tcctomat`, `tvctomat`

Examples

```
## Not run: read.rep("test.dat", resp=c("y1","y2"), times="tt", id="id",
## Not run:  totals=c("tot1","tot2"), tvcov="x",ccov="x2")
```

`read.surv` *Read a List of Matrices from a File for Repeated Times to Events*

Description

`read.surv` reads sets of lines of data from a file. Each set may contain a series of duration times followed by a censor indicator for the last value (`all=FALSE`) or a series of pairs of times followed by their censor indicators (`all=TRUE`).

Usage

```
read.surv(file="", skip=0, nlines=1, cumulative=TRUE, all=TRUE)
```

Arguments

`file`	Name of the file to read
`skip`	Number of lines to skip at the beginning of the file
`nlines`	Number of lines in each series of duration times
`cumulative`	If TRUE, the times are cumulative and differences are taken to obtain times between events. Otherwise, the times are used unchanged.
`all`	If TRUE, all times have accompanying censor indicators; otherwise, only the last one does.

Value

A list containing a list of vectors with the series of times and a vector of censor indicators for the last time of each series is returned.

Author(s)

J.K. Lindsey

See Also

lvna, read.list, read.rep, restovec, rmna

Examples

```
## Not run: y <- read.surv("test.dat")
```

---

restovec *Create a response Object*

Description

restovec can produce an object of class, response, from a vector of (1) independent univariate responses or (2) a single time series.

It can produce such an object from repeated measurements in the form of (1) a list of vectors of event histories, (2) a list of two or more column matrices with times, response values, and and other information or (3) a matrix or dataframe of response values. The first two are for unbalanced data and the third for balanced data.

Multivariate responses can be supplied as (1) a three-dimensional array of balanced repeated measurements, (2) lists of matrices for unbalanced repeated measurements, or (3) a matrix with either (a) several time series or (b) single observations per individual on several variables.

In formula and functions, the key words, times can be used to refer to the response times from the data object as a covariate, individuals to the index for individuals as a factor covariate, and nesting the index for nesting as a factor covariate. The latter two only work for W&R notation.

NAs can be detected with lvna or removed with rmna (where necessary, in coordination with the appropriate covariates) to create a repeated object.

response objects can be printed and plotted. Methods are available for extracting the response, the numbers of observations per individual, the times, the weights, the units of measurement/Jacobian, and the nesting variable: response, nobs, times, weights, delta, and nesting.

The response and or the times may be transformed using transform(z,newy=fcn1(y),times=fcn2(times)) where fcn1 and fcn2 are transformations and y is the name of a response variable. When the response is transformed, the Jacobian is automatically calculated. Note that, if the unit of precision/Jacobian (delta) is available in the response object, this is automatically included in the calculation of the likelihood function in all library model functions.

Usage

```
restovec(response=NULL, times=NULL, nest=NULL, coordinates=NULL,
censor=NULL, totals=NULL, weights=NULL, delta=NULL,
type=NULL, names=NULL, units=NULL, oldresponse=NULL,
description=NULL)
```

Arguments

`response` For (1) independent univariate responses with one observation per individual or (2) a single time series, one vector may be supplied (in the latter case, the times must be given even if equally spaced).

Univariate repeated measurements responses can be given (1) if balanced, as a matrix or dataframe of response values with dimensions: number of individuals by number of responses/individual, (2) a list of vectors of event histories, or (3) a list of one or more column matrices, for each individual, with response values in the first column and times in the second (if there are no times, set `times` to FALSE), possibly followed by columns with nesting categories, binomial totals, censoring indicators, and/or units of measurement.

Multivariate responses can be supplied as (1) a three-dimensional array of balanced repeated measurements with dimensions: number of individuals by number of responses/individual by number of variables, (2) a list of matrices for unbalanced repeated measurements each with dimensions: number of responses on that individual by number of variables, plus a column for times if available (otherwise set `times` to FALSE), or (3) a matrix with either (a) several time series, having dimensions: length of time series by by number of times series, or (b) single observations per individual on several variables with dimensions: number of individuals by number of variables. In all but case (1), `type` must be a character vector with length equal to the number of responses. In case (2), where applicable, `censor`, `totals`, and `delta` must be supplied as lists of matrices of the same size as for `response`, and `nest` and `weights` as lists of vectors of lengths equal to the number of observations on each individual.

`times` When `response` is a matrix or multivariate array, these can be (1) a vector when the times are the same for all individuals, possibly unequally-spaced, or (2) a matrix with dimensions: number of individuals by number of responses/individual. Not necessary if times are equally spaced, except if a vector containing a single time series is supplied (if not given in this case, it takes the responses to be independent, not a time series). For clustered data with no time ordering, set to FALSE.

`nest` This is the second level of nesting, with the individual being the first level. Values for an individual must be consecutive increasing integers with all responses in the same cluster grouped together. For example, with three clusters of four observations each, the code would be 1,1,1,1,2,2,2,2,3,3,3,3.

When `response` is a matrix or multivariate array, this can be a vector of length equal to the number of responses/individual indicating which responses belong to which nesting category.

If `response` is a multivariate list, this must also be a list.

When `response` is a univariate list of unbalanced repeated measurements, the nesting indicator may instead be included in that list but must respect the same ordering as described above.

`coordinates` When `response` is a vector, a two-column matrix giving the coordinates for spatial data.

`censor` When `response` is a matrix, this can be (1) a vector of the same length as the number of individuals, containing a binary indicator, with a one indicating that the last time period in the series terminated with an event and zero that it was censored, or (2) a matrix of the same size as `response`.

When `response` is a multivariate array, this can be (1) a matrix with dimensions: number of individuals by number of responses, or (2) an array of the same size as `response`. In the first case, for each column corresponding to a duration response, it should contain a binary indicator, with a one indicating that the last time period in the series terminated with an event and zero that it was censored, and NAs in columns not containing durations. In the second case, layers not corresponding to duration responses should contain NAs.

If `response` is a multivariate list, this must also be a list.

For event history data, even with no censoring, an appropriate vector of ones must be supplied.

When `response` is a univariate list of unbalanced repeated measurements, the censoring indicator may instead be included in that list.

`totals` If the response is a matrix of binomial counts, this can be (1) a corresponding vector (one total per individual) or (2) a matrix of totals.

When `response` is a multivariate array, this can be (1) a matrix with dimensions: number of individuals by number of responses if all binomial responses for an individual have the same total, or (2) an array of the same size as `response`. In the first case, for each column corresponding to a binomial response, it should contain the corresponding totals, with NAs in columns not containing binomial. In the second case, layers not corresponding to binomial responses should contain NAs.

If `response` is a multivariate list, this must also be a list.

When `response` is a univariate list of unbalanced repeated measurements, the totals may instead be included in that list.

`weights` A vector, matrix, array, or list of vectors of frequencies or weights, with one value per `response`. In other words, a multivariate response has only one corresponding weight value.

`delta` For continuous measurements, the unit of precision (if not equal to unity) for each response: a scalar, vector, matrix, array, or list of the same dimensions as `response`. For example, if responses have two decimal places (`12.34`), `delta=0.01`. If the response has been transformed, this should be multiplied by the numerical values of the Jacobian. When the `transform` method is applied to the `response` object, this is automatically updated.

`type` The type(s) of observations: nominal, ordinal, discrete, duration, continuous, or unknown. If not specified otherwise, those responses with `delta` and no `censor` are assumed to be `continuous`, those with `censor` indicator are assumed to be `duration`, those with `totals` are assumed to be `nominal`, and all others `unknown`.

`names` Optional name(s) of the response variable(s).

`units` Optional character vector giving units of measurement of response(s).

`oldresponse` An existing `response` object to which the new data are to be added.

`description` An optional named list of character vectors with names of some or all response variables containing their descriptions.

Value

Returns an object of class, `response`, containing a vector with the responses (`z$y`), a corresponding vector of times (`z$times`) if applicable, a vector giving the number of observations per individual (`z$nobs`, set to a scalar 1 if observations are independent), type (`z$delta`), and possibly binomial totals (`z$n`), nesting (clustering, `z$nest`), censoring (`z$censor`), weights (`z$wt`), unit of precision/Jacobian (`z$delta`), units of measurement (`z$units`), and description (`z$description`) information.

Author(s)

J.K. Lindsey

See Also

`DataMethods`, `covind`, `delta`, `description`, `lvna`, `names`, `nesting`, `nobs`, `read.list`, `read.surv`, `response`, `resptype`, `rmna`, `tcctomat`, `times`, `transform`, `tvctomat`, `units`, `weights`

Examples

```
#
#continuous response
y <- matrix(rnorm(20),ncol=5)
# times assumed to be 1:5
restovec(y, units="m")
#unequally-spaced times
tt <- c(1,3,6,10,15)
print(resp <- restovec(y, times=tt, units="m",
description=list(y="Response measured in metres")))
response(resp)
response(resp, nind=2:3)
response(transform(resp, y=1/y))
transform(resp, y=1/y, units="1/m")
units(resp)
description(resp)
times(resp)
times(transform(resp, times=times-6))
nobs(resp)
weights(resp)
nesting(resp)
# because individuals are the only nesting, this is the same as
covind(resp)
#
# binomial response
y <- matrix(rpois(20,5),ncol=5)
# responses summarized as relative frequencies
print(respb <- restovec(y, totals=y+matrix(rpois(20,5),ncol=5), times=tt))
response(respb)
#
# censored data
y <- matrix(rweibull(20,2,5),ncol=5)
print(respc <- restovec(y, censor=matrix(rbinom(20,1,0.9),ncol=5), times=tt))
```

```
# if there is no censoring, censor indicator is not printed
response(respc)
# nesting clustered within individuals
nest <- c(1,1,2,2,2)
print(respn <- restovec(y, censor=matrix(rbinom(20,1,0.9),ncol=5),
times=tt,nest=nest))
response(respn)
times(respn)
nesting(respn)
#
# multivariate response
restovec(y, censor=matrix(rbinom(20,1,0.9),ncol=5),
units=c("m","days","l","cm","mon"),
type=c("continuous","duration","continuous","continuous","duration"),
description=list(y1="First continuous variable",
y2="First duration variable",y3="Second continuous variable",
y4="Third continuous variable",y5="Second duration variable"))
restovec(y, censor=matrix(rbinom(20,1,0.9),ncol=5),
names=c("a","b","c","d","e"), units=c("m","days","l","cm","mon"),
type=c("continuous","duration","continuous","continuous","duration"),
description=list(a="First continuous variable",
b="First duration variable",c="Second continuous variable",
d="Third continuous variable",e="Second duration variable"))
```

---

rmna *Create a repeated Object, Removing NAs*

Description

rmna forms an object of class, repeated, from a response object and possibly time-varying or intra-individual covariate (tvcov), and time-constant or inter-individual covariate (tccov) objects, removing any observations where response and covariate values have NAs. Subjects must be in the same order in all (three) objects to be combined.

Such objects can be printed and plotted. Methods are available for extracting the response, the numbers of observations per individual, the times, the weights, the units of measurement/Jacobian, the nesting variable, the covariates, and their names: response, nobs, times, weights, delta, nesting, covariates, and names.

Usage

```
rmna(response, ccov=NULL, tvcov=NULL)
```

Arguments

response	An object of class, response (created by restovec), containing the response variable information.
ccov	An object of class, tccov (created by tcctomat), containing the time-constant or inter-individual covariate information.
tvcov	An object of class, tvcov (created by tvctomat), containing the time-varying or intra-individual covariate information.

Value

Returns an object of class, repeated, containing a list of the response object (z$response, so that, for example, the response vector is z$response$y; see restovec), and possibly the two classes of covariate objects (z$ccov and z$tvcov; see tcctomat and tvctomat).

Author(s)

J.K. Lindsey

See Also

DataMethods, covariates, covind, delta, dftorep, lvna, names, nesting, nobs, read.list, read.surv, response, resptype, restovec, tcctomat, times, transform, tvctomat, units, weights

Examples

```
y <- matrix(rnorm(20),ncol=5)
tt <- c(1,3,6,10,15)
print(resp <- restovec(y,times=tt))
x <- c(0,0,1,1)
tcc <- tcctomat(x)
z <- matrix(rpois(20,5),ncol=5)
tvc <- tvctomat(z)
print(reps <- rmna(resp, tvcov=tvc, ccov=tcc))
response(reps)
response(reps, nind=2:3)
times(reps)
nobs(reps)
weights(reps)
covariates(reps)
covariates(reps,names="x")
covariates(reps,names="z")
names(reps)
nesting(reps)
# because individuals are the only nesting, this is the same as
covind(reps)
#
# use in glm
rm(y,x,z)
glm(y~x+z,data=as.data.frame(reps))
#
# binomial
y <- matrix(rpois(20,5),ncol=5)
print(respb <- restovec(y,totals=y+matrix(rpois(20,5),ncol=5),times=tt))
print(repsb <- rmna(respb, tvcov=tvc, ccov=tcc))
response(repsb)
#
# censored data
y <- matrix(rweibull(20,2,5),ncol=5)
print(respc <- restovec(y,censor=matrix(rbinom(20,1,0.9),ncol=5),times=tt))
```

```
print(repsc <- rmna(respc, tvcov=tvc, ccov=tcc))
# if there is no censoring, censor indicator is not printed
response(repsc)
#
# nesting clustered within individuals
nest <- c(1,1,2,2,2)
print(respn <- restovec(y,censor=matrix(rbinom(20,1,0.9),ncol=5),
times=tt,nest=nest))
print(repsn <- rmna(respn, tvcov=tvc, ccov=tcc))
response(respn)
times(respn)
nesting(respn)
```

---

rmutil *Utilities for Repeated Measurements Library*

Description

%^% Power of a Matrix

covariates Extract Covariate Matrices from a Data Object

covind Nesting Indicator for Observations within Individuals in a Data Object

dbetabinom Density of Beta Binomial Distribution

dboxcox Density of Box-Cox Distribution

dburr Density of Burr Distribution

ddoublebinom Density of Double Binomial Distribution

ddoublepois Density of Double Poisson Distribution

delta Extract Units of Measurement Vector from a Data Object

dftorep Transform a Dataframe to a repeated Object

dgammacount Density of Gamma Count Distribution

dgextval Density of Generalized Extreme Value Distribution

dggamma Density of Generalized Gamma Distribution

dginvgauss Density of Generalized Inverse Gaussian Distribution

dglogis Density of Generalized Logistic Distribution

dgweibull Density of Generalized Weibull Distribution

dhjorth Density of Hjorth Distribution

dinvgauss Density of Inverse Gaussian Distribution

dlaplace Density of Laplace Distribution

dlevy Density of Levy Distribution

dmultbinom Density of Multiplicative Binomial Distribution

dmultpois Density of Multiplicative Poisson Distribution

dpareto Density of Pareto Distribution

dpowexp Density of Power Exponential Distribution

dpvfpois Density of Power Variance Function Poisson Distribution

dsimplex Density of Simplex Distribution

dskewlaplace Density of Skew Laplace Distribution

finterp Formula Interpreter

fmobj Object Finder in Formulae

fnenvir Check Covariates and Parameters of a Function

formula Extract Formula Used to Create Time-constant Covariate Matrix in a Data Object

gauss.hermite Calculate Gauss-Hermite Quadrature Points

gettvc Create Time-varying Covariates

int Vectorized One-dimensional Numerical Integration

int2 Vectorized Two-dimensional Numerical Integration

iprofile Produce Individual Time Profiles for Plotting

lin.diff.eqn Solution of Autonomous Linear Differential Equations

lvna Create a Repeated Object Leaving NAs

mexp Matrix Exponentiation

mprofile Produce Marginal Time Profiles for Plotting

names Extract Names of Covariates from a Data Object

nesting Extract Nesting Indicators from a Data Object

nobs Extract Number of Observations per Individual from a Data Object

pbetabinom Distribution Function of Beta Binomial Distribution

pboxcox Distribution Function of Box-Cox Distribution

pburr Distribution Function of Burr Distribution

pdoublebinom Distribution Function of Double Binomial Distribution

pdoublepois Distribution Function of Double Poisson Distribution

pgammacount Distribution Function of Gamma Count Distribution

pgextval Distribution Function of Generalized Extreme Value Distribution

pggamma Distribution Function of Generalized Gamma Distribution

pginvgauss Distribution Function of Generalized Inverse Gaussian Distribution

pglogis Distribution Function of Generalized Logistic Distribution

pgweibull Distribution Function of Generalized Weibull Distribution

phjorth Distribution Function of Hjorth Distribution

pinvgauss Distribution Function of Inverse Gaussian Distribution

pkpd Pharmacokinetic Model Functions

plaplace Distribution Function of Laplace Distribution

plevy Distribution Function of Levy Distribution

plot.residuals Plot Residuals for Carma

`response` Extract Response Vector from a Data Object

`restovec` Create a Response Object

`rgammacount` Random Number Generation for Gamma Count Distribution

`rgextval` Random Number Generation for Generalized Extreme Value Distribution

`rggamma` Random Number Generation for Generalized Gamma Distribution

`rginvgauss` Random Number Generation for Generalized Inverse Gaussian Distribution

`rglogis` Random Number Generation for Generalized Logistic Distribution

`rgweibull` Random Number Generation for Generalized Weibull Distribution

`rhjorth` Random Number Generation for Hjorth Distribution

`rinvgauss` Random Number Generation for Inverse Gaussian Distribution

`rlaplace` Random Number Generation for Laplace Distribution

`rlevy` Random Number Generation for Levy Distribution

`rmna` Create a Repeated Object

`rmultbinom` Random Number Generation for Multiplicative Binomial Distribution

`rmultpois` Random Number Generation for Multiplicative Poisson Distribution

`rpareto` Random Number Generation for Pareto Distribution

`rpowexp` Random Number Generation for Power Exponential Distribution

`rpvfpois` Random Number Generation for Power Variance Function Poisson Distribution

`rsimplex` Random Number Generation for Simplex Distribution

`rskewlaplace` Random Number Generation for Skew Laplace Distribution

`runge.kutta` Runge-Kutta Method for Solving Differential Equations

`tcctomat` Create a Time-constant Covariate (tccov) Object

`times` Extract Times Vector from a Data Object

`transform` Transform Variables in a Data Object

`tvctomat` Create a Time-varying Covariate (tvcov) Object

`wr` Find the Response Vector and Design Matrix for a Model Formula

---

`runge.kutta` *Runge-Kutta Method for Solving Differential Equations*

Description

`runge.kutta` numerically solves a differential equation by the fourth-order Runge-Kutta method.

Usage

```
runge.kutta(f, initial, x)
```

Arguments

f	A function dy/dx=func(y,x).
initial	The initial value of y.
x	A vector of values of x for which the values or y are required.

Value

A vector of values of y as solution of the function f corresponding to the values in x.

Author(s)

J.K. Lindsey

Examples

```
fn <- function(y,x) (x*y-y^2)/x^2
soln <- runge.kutta(fn,2,seq(1,3,by=1/128))
## exact solution
exact <- seq(1,3,by=1/128)/(0.5+log(seq(1,3,by=1/128)))
rbind(soln, exact)
```

Simplex *Simplex Distribution*

Description

These functions provide information about the simplex distribution with location parameter equal to m and shape equal to s: density, cumulative distribution, quantiles, and random generation.

The simplex distribution has density

$$f(y) = \frac{1}{\sqrt{(2\pi\sigma(y(1-y))^3)}} \exp(-((y-\mu)/(\mu(1-\mu)))^2/(2y(1-y)\sigma))$$

where μ is the location parameter of the distribution and σ is the shape, and $0 < y < 1$.

Usage

```
dsimplex(y, m, s, log=FALSE)
psimplex(q, m, s)
qsimplex(p, m, s)
rsimplex(n, m, s)
```

Arguments

`y`	vector of responses.
`q`	vector of quantiles.
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of location parameters.
`s`	vector of shape parameters.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

`dbeta` for the beta distribution and `dtwosidedpower` for the two-sided power distribution, other distributions for proportions between zero and one.

Examples

```
dsimplex(0.3, 0.5, 1)
psimplex(0.3, 0.5, 1)
qsimplex(0.1, 0.5, 1)
rsimplex(10, 0.5, 1)
```

`SkewLaplace` *Skew Laplace Distribution*

Description

These functions provide information about the skew Laplace distribution with location parameter equal to `m`, dispersion equal to `s`, and skew equal to `f`: density, cumulative distribution, quantiles, log hazard, and random generation. For `f=1`, this is an ordinary (symmetric) Laplace distribution.

The skew Laplace distribution has density

$$f(y) = \frac{\nu \exp(-\nu(y-\mu)/\sigma)}{(1+\nu^2)\sigma}$$

if $y \geq \mu$ and else

$$f(y) = \frac{\nu \exp((y-\mu)/(\nu\sigma))}{(1+\nu^2)\sigma}$$

where μ is the location parameter of the distribution, σ is the dispersion, and ν is the skew.

The mean is given by $\mu + \frac{\sigma(1-\nu^2)}{\sqrt{2}\nu}$ and the variance by $\frac{\sigma^2(1+\nu^4)}{2\nu^2}$.

Note that this parametrization of the skew (family) parameter is different than that used for the multivariate skew Laplace distribution in `elliptic`.

Usage

```
dskewlaplace(y, m=0, s=1, f=1, log=FALSE)
pskewlaplace(q, m=0, s=1, f=1)
qskewlaplace(p, m=0, s=1, f=1)
rskewlaplace(n, m=0, s=1, f=1)
```

Arguments

y	vector of responses.
q	vector of quantiles.
p	vector of probabilities
n	number of values to generate
m	vector of location parameters.
s	vector of dispersion parameters.
f	vector of skew parameters.
log	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

See Also

dexp for the exponential distribution, dcauchy for the Cauchy distribution, and dlaplace for the Laplace distribution.

Examples

```
dskewlaplace(5, 2, 1, 0.5)
pskewlaplace(5, 2, 1, 0.5)
qskewlaplace(0.95, 2, 1, 0.5)
rskewlaplace(10, 2, 1, 0.5)
```

---

tcctomat *Create a Time-constant, Inter-individual Covariate (tccov) Object*

Description

tcctomat creates an object of class, tccov, from a vector or matrix containing time-constant or inter-individual baseline covariates or a model formula. It can also combine two such objects.

Such objects can be printed. Methods are available for extracting the covariates, their names, and the formula: covariates, names, and formula. The method, transform, can transform variables in place or by adding new variables to the object.

To obtain the indexing to expand time-constant or inter-individual covariates to the size of a repeated measurements response, use covind.

Usage

```
tcctomat(ccov, names=NULL, units=NULL, oldccov=NULL, dataframe=TRUE,
description=NULL)
```

Arguments

`ccov`	A vector, matrix, or dataframe containing time-constant or inter-individual baseline covariates with one row per individual, a model formula using vectors of the same size, or an object of class, `tccov`. In the first two cases, the variables may be factors; if `dataframe=FALSE`, these are transformed to indicator variables.
`units`	Optional character vector specifying units of measurements of covariates.
`names`	The names of the covariates (if the matrix does not have column names).
`oldccov`	An object of class, `tccov`, to which `ccov` is to be added.
`dataframe`	If TRUE and factor variables are present, the covariates are stored as a dataframe; if FALSE, they are expanded to indicator variables. If no factor variables are present, covariates are always stored as a matrix.
`description`	An optional named list of character vectors with names of some or all covariates containing their descriptions.

Value

Returns an object of class, `tccov`, containing one matrix or dataframe for the covariates (`z$ccov`) with one row per individual and possibly the model formula (`z$linear`).

Author(s)

J.K. Lindsey

See Also

`DataMethods`, `covariates`, `description`, `formula`, `lvna`, `names`, `restovec`, `rmna`, `transform`, `tvctomat`, `units`

Examples

```
x1 <- gl(4,1)
print(tcc1 <- tcctomat(~x1))
covariates(tcc1)
covariates(tcc1, name="x12")
tcctomat(x1)
tcctomat(x1, dataframe=FALSE)
x2 <- c(0,0,1,1)
print(tcc2 <- tcctomat(~x2, units="days"))
covariates(tcc2)
print(tcc3 <- tcctomat(~x1+x2))
covariates(tcc3)
covariates(tcc3, names=c("x12","x2"))
formula(tcc3)
names(tcc3)
```

```
print(tcc4 <- tcctomat(data.frame(x1,x2), units=c(NA,"days")))
covariates(tcc4)
print(tcc5 <- tcctomat(data.frame(x1,x2), dataframe=FALSE, units=c(NA,"days")))
covariates(tcc5)
```

---

tvctomat *Create a Time-varying, Intra-individual Covariate (tvcov) Object*

Description

tvctovmat creates an object of class, tvcov, from a list of matrices with time-varying or intra-individual covariates for each individual or one matrix or dataframe of such covariate values. It can also combine two such objects or add interactions among covariates.

Such objects can be printed. Methods are available for extracting the covariates and their names: covariates and names. The method, transform, can transform variables in place or by adding new variables to the object.

Usage

```
tvctomat(tvcov, names=NULL, units=NULL, interaction=NULL, ccov=NULL,
oldtvcov=NULL, dataframe=TRUE, description=NULL)
```

Arguments

tvcov	Either (1) if unbalanced, a list of matrices or dataframes with time-varying or intra-individual covariate values for each individual (one column per variable), (2) if balanced, one matrix or dataframe of such covariate values (when there is only one such covariate) with dimensions: number of individuals by number of observations/individual, or (3) an object of class, tvcov. In the first two cases, the variables may be factors; if dataframe=FALSE, these are transformed to indicator variables.
names	The names of the time-varying or intra-individual covariates in tvcov (if the matrices do not have column names) or the names of the time-constant covariates for which interactions are to be created.
units	Optional character vector specifying units of measurements of covariates.
interaction	A pair of index numbers or names of variables in tvcov, with that class, for which an interaction is to be added or, if ccov is provided, a set of such names of time-varying or intra-individual covariates for creating interactions with the time-constant covariates.
ccov	Time-constant or inter-individual covariates for which an interaction is to be introduced with time-varying or intra-individual covariates in tvcov.
oldtvcov	An object of class, tvcov, to which tvcov is to be added.
dataframe	If TRUE and factor variables are present, the covariates are stored as a dataframe; if FALSE, they are expanded to indicator variables. If no factor variables are present, covariates are always stored as a matrix.
description	An optional named list of character vectors with names of some or all covariates containing their descriptions.

Value

Returns an object of class, `tvcov`, containing a matrix or dataframe for the covariates (`z$tvcov`) with one row per response per individual and a vector giving the number of observations per individual (`z$nobs`).

Author(s)

J.K. Lindsey

See Also

DataMethods, covariates, description, formula, gettvc, lvna, names, restovec, rmna, tcctomat, transform, units

Examples

```
z <- matrix(rpois(20,5),ncol=5)
print(tvc <- tvctomat(z, units="days"))
covariates(tvc)
names(tvc)
v <- data.frame(matrix(rep(c("a","b","c","d","e"),4),ncol=5),stringsAsFactors=TRUE)
print(tvc2 <- tvctomat(v, oldtvc=tvc, units=NA))
covariates(tvc2)
print(tvc3 <- tvctomat(v, oldtvc=tvc, dataframe=FALSE, units=NA))
covariates(tvc3)
print(tvc4 <- tvctomat(tvc2, interaction=c("z","v")))
covariates(tvc4)
x1 <- 1:4
x2 <- gl(4,1)
xx <- tcctomat(data.frame(x1,x2), dataframe=FALSE)
tvctomat(tvc3, interaction="z", ccov=xx)
tvctomat(tvc3, interaction="z", ccov=xx, names="x1")
tvctomat(tvc3, interaction="z", ccov=xx, names=c("x22","x23","x24"))
xx <- tcctomat(data.frame(x1,x2), dataframe=TRUE)
tvctomat(tvc2, interaction="z", ccov=xx)
tvctomat(tvc2, interaction="z", ccov=xx, names="x1")
tvctomat(tvc2, interaction="z", ccov=xx, names="x2")
```

`Two-Sided Power` *Two-Sided Power Distribution*

Description

These functions provide information about the two-sided power distribution with location parameter equal to `m` and shape equal to `s`: density, cumulative distribution, quantiles, and random generation.

The two-sided power distribution has density

$$f(y) = s(\frac{y}{m})^{s-1}, y <= m$$

$$f(y) = s(\frac{1-y}{1-m})^{s-1}, y >= m$$

where μ is the location parameter of the distribution and σ is the shape, and $0 < y < 1$.

For $\sigma = 1$, this is the uniform distribution and for $\sigma = 2$, it is the triangular distribution.

Usage

```
dtwosidedpower(y, m, s=2, log=FALSE)
ptwosidedpower(q, m, s=2)
qtwosidedpower(p, m, s=2)
rtwosidedpower(n, m, s=2)
```

Arguments

`y`	vector of responses.
`q`	vector of quantiles.
`p`	vector of probabilities
`n`	number of values to generate
`m`	vector of location parameters.
`s`	vector of shape parameters.
`log`	if TRUE, log probabilities are supplied.

Author(s)

J.K. Lindsey

References

van Dorp, J.R. and Kotz, S. (2002) A novel extension of the triangular distribution and its parameter estimation. The Statistician 51, 63-79.

See Also

`dbeta` for the beta distribution and `dsimplex` for the simplex distribution, other distributions for proportions between zero and one.

Examples

```
dtwosidedpower(0.3, 0.5, 3)
ptwosidedpower(0.3, 0.5, 3)
qtwosidedpower(0.1, 0.5, 3)
rtwosidedpower(10, 0.5, 3)
```

wr — *Find the Response Vector and Design Matrix for a W\&R Model Formula*

Description

wr gives the response vector and design matrix for a formula in Wilkinson and Rogers notation.

Usage

```
wr(formula, data=NULL, expand=TRUE)
```

Arguments

formula	A model formula.
data	A data object or environment.
expand	If FALSE, the covariates are read from the tccov object without expanding to the length of the response variable.

Value

wr returns a list containing the response vector (z$response), if included in the formula, and the design matrix (z$design) from the data object or environment supplied or from the global environment for the formula supplied.

Author(s)

J.K. Lindsey

Examples

```
y <- rnorm(20)
x <- gl(4,5)
z <- rpois(20,2)
wr(y~x+z)
```

Index