Package 'simstudy'

January 8, 2018

Type Package

Title Simulation of Study Data

Version 0.1.8

Date 2018-01-08

Author Keith Goldfeld [aut, cre]

Maintainer Keith Goldfeld <Keith.Goldfeld@nyumc.org>

Description Simulates data sets in order to explore modeling techniques or better understand data generating processes. The user specifies a set of relationships between covariates, and generates data based on these specifications. The final data sets can represent data from randomized control trials, repeated measure (longitudinal) designs, and cluster randomized trials. Missingness can be generated using various mechanisms (MCAR, MAR, NMAR).

Depends R (>= 3.2.2), data.table

License GPL-3

LazyData TRUE

Imports Rcpp, mvnfast

RoxygenNote 6.0.1

Suggests testthat, knitr, rmarkdown, ggplot2, scales, grid, gridExtra, survival, gee, splines, ordinal, formatR, mgcv

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-01-08 21:33:47 UTC

R topics documented:

`addColumns` *Add columns to existing data set*

Description

Add columns to existing data set

Usage

```
addColumns(dtDefs, dtOld)
```

Arguments

dtDefs	Name of definitions for added columns
dtOld	Name of data table that is to be updated

Value

An updated data.table that contains the added simulated data

Examples

```
# New data set

def <- defData(varname = "xNr", dist = "nonrandom", formula=7, id = "idnum")
def <- defData(def, varname="xUni", dist="uniform", formula="10;20")

dt <- genData(10, def)

# Add columns to dt

def2 <- defDataAdd(varname="y1", formula = 10, variance = 3)
def2 <- defDataAdd(def2, varname="y2", formula = .5, dist = "binary")
def2

dt <- addColumns(def2, dt)
dt
```

---

addCondition *Add a single column to existing data set based on a condition*

Description

Add a single column to existing data set based on a condition

Usage

```
addCondition(condDefs, dtOld, newvar)
```

Arguments

condDefs	Name of definitions for added column
dtOld	Name of data table that is to be updated
newvar	Name of new column to add

Value

An updated data.table that contains the added simulated data

Examples

```
# New data set

def <- defData(varname = "x", dist = "categorical", formula = ".33;.33")
def <- defData(def, varname="y", dist="uniform", formula="-5;5")

dt <- genData(1000, def)

# Define conditions

defC <- defCondition(condition = "x == 1", formula = "5 + 2*y-.5*y^2",
                     variance = 1,dist = "normal")
defC <- defCondition(defC, condition = "x == 2",
                     formula = "3 - 3*y + y^2", variance = 2, dist="normal")
defC <- defCondition(defC, condition = "x == 3",
                     formula = "abs(y)", dist="poisson")

# Add column

dt <- addCondition(defC, dt, "NewVar")

# Plot data

library(ggplot2)

ggplot(data = dt, aes(x=y, y=NewVar, group = x)) +
  geom_point(aes(color = factor(x)))
```

`addCorData` *Add correlated data to existing data.table*

Description

Add correlated data to existing data.table

Usage

```
addCorData(dtOld, idname, mu, sigma, corMatrix = NULL, rho, corstr = "ind",
  cnames = NULL)
```

Arguments

`dtOld`	Data table that is the new columns will be appended to.
`idname`	Character name of id field, defaults to "id".
`mu`	A vector of means. The length of mu must be nvars.

sigma	Standard deviation of variables. If standard deviation differs for each variable, enter as a vector with the same length as the mean vector mu. If the standard deviation is constant across variables, as single value can be entered.
corMatrix	Correlation matrix can be entered directly. It must be symmetrical and positive semi-definite. It is not a required field; if a matrix is not provided, then a structure and correlation coefficient rho must be specified.
rho	Correlation coefficient, -1 <= rho <= 1. Use if corMatrix is not provided.
corstr	Correlation structure of the variance-covariance matrix defined by sigma and rho. Options include "ind" for an independence structure, "cs" for a compound symmetry structure, and "ar1" for an autoregressive structure.
cnames	Explicit column names. A single string with names separated by commas. If no string is provided, the default names will be V#, where # represents the column.

Value

The original data table with the additional correlated columns

Examples

```
def <- defData(varname="xUni", dist="uniform", formula="10;20", id = "myID")
def <- defData(def, varname="xNorm", formula="xUni * 2", dist="normal", variance=8)

dt <- genData(250, def)

mu <- c(3, 8, 15)
sigma <- c(1, 2, 3)

dtAdd <- addCorData(dt, "myID", mu = mu, sigma = sigma, rho = .7, corstr = "cs")
dtAdd

round(var(dtAdd[,.(V1, V2, V3)]), 3)
round(cor(dtAdd[,.(V1, V2, V3)]), 2)

dtAdd <- addCorData(dt, "myID", mu = mu, sigma = sigma, rho = .7, corstr = "ar1")
round(cor(dtAdd[,.(V1, V2, V3)]), 2)

corMat <- matrix(c(1, .2, .8, .2, 1, .6, .8, .6, 1), nrow = 3)

dtAdd <- addCorData(dt, "myID", mu = mu, sigma = sigma, corMatrix = corMat)
round(cor(dtAdd[,.(V1, V2, V3)]), 2)
```

---

addCorFlex *Create multivariate (correlated) data - for general distributions*

Description

Create multivariate (correlated) data - for general distributions

Usage

```
addCorFlex(dt, defs, rho = 0, tau = NULL, corstr = "cs",
  corMatrix = NULL)
```

Arguments

dt	Data table that will be updated.
defs	Field definition table created by function ‘defDataAdd‘.
rho	Correlation coefficient, -1 <= rho <= 1. Use if corMatrix is not provided.
tau	Correlation based on Kendall’s tau. If tau is specified, then it is used as the correlation even if rho is specfied. If tau is NULL, then the specified value of rho is used, or rho defaults to 0.
corstr	Correlation structure of the variance-covariance matrix defined by sigma and rho. Options include "cs" for a compound symmetry structure and "ar1" for an autoregressive structure. Defaults to "cs".
corMatrix	Correlation matrix can be entered directly. It must be symmetrical and positive semi-definite. It is not a required field; if a matrix is not provided, then a structure and correlation coefficient rho must be specified.

Value

data.table with added column(s) of correlated data

Examples

```
defC <- defData(varname = "nInds", formula = 50, dist = "noZeroPoisson", id = "idClust")

dc <- genData(10, defC)
#### Normal only

dc <- addCorData(dc, mu = c(0,0,0,0), sigma = c(2, 2, 2, 2), rho = .2,
                 corstr = "cs", cnames = c("a","b","c","d"), idname = "idClust")

di <- genCluster(dc, "idClust", "nInds", "id")

defI <- defDataAdd(varname = "A", formula = "-1 + a", variance = 3,
                   dist = "normal")
defI <- defDataAdd(defI, varname = "B", formula = "4.5 + b", variance = .5,
                   dist = "normal")
defI <- defDataAdd(defI, varname = "C", formula = "5*c", variance = 3,
                   dist = "normal")
defI <- defDataAdd(defI, varname = "D", formula = "1.6 + d", variance = 1,
                   dist = "normal")

#### Generate new data

di <- addCorFlex(di, defI, rho = 0.4, corstr = "cs")

# Check correlations by cluster
```

```
for (i in 1:nrow(dc)) {
  print(cor(di[idClust == i, list(A, B, C, D)]))
}

# Check global correlations - should not be as correlated
cor(di[, list(A, B, C, D)])
```

---

addCorGen *Create multivariate (correlated) data - for general distributions*

Description

Create multivariate (correlated) data - for general distributions

Usage

```
addCorGen(dtOld, nvars, idvar, rho, corstr, corMatrix = NULL, dist, param1,
  param2 = NULL, cnames = NULL)
```

Arguments

dtOld	If an existing data.table is specified, then wide will be set to TRUE and n will be set to the nrow(dt) without any warning or error.
nvars	Number of new variables to create for each id.
idvar	String variable name of column represents individual level id for correlated data.
rho	Correlation coefficient, -1 <= rho <= 1. Use if corMatrix is not provided.
corstr	Correlation structure of the variance-covariance matrix defined by sigma and rho. Options include "cs" for a compound symmetry structure and "ar1" for an autoregressive structure.
corMatrix	Correlation matrix can be entered directly. It must be symmetrical and positive semi-definite. It is not a required field; if a matrix is not provided, then a structure and correlation coefficient rho must be specified.
dist	A string indicating "binary", "poisson" or "gamma".
param1	A string that represents the column in dtOld that contains the parameter for the mean of the distribution. In the case of the uniform distribution the column specifies the minimum.
param2	A string that represents the column in dtOld that contains a possible second parameter for the distribution. For the normal distribution, this will be the variance; for the gamma distribution, this will be the dispersion; and for the uniform distribution, this will be the maximum.
cnames	Explicit column names. A single string with names separated by commas. If no string is provided, the default names will be V#, where # represents the column.

Value

Original data.table with added column(s) of correlated data

Examples

```
# Wide example

def <- defData(varname = "xbase", formula = 5, variance = .4, dist = "gamma", id = "cid")
def <- defData(def, varname = "lambda", formula = ".5 + .1*xbase", dist="nonrandom", link = "log")
def <- defData(def, varname = "p", formula = "-2 + .3*xbase", dist="nonrandom", link = "logit")

dt <- genData(500, def)

dtX1 <- addCorGen(dtOld = dt, idvar = "cid", nvars = 3, rho = .7, corstr = "cs",
                  dist = "poisson", param1 = "lambda")

dtX2 <- addCorGen(dtOld = dtX1, idvar = "cid", nvars = 4, rho = .4, corstr = "ar1",
                  dist = "binary", param1 = "p")

# Long example

def <- defData(varname = "xbase", formula = 5, variance = .4, dist = "gamma", id = "cid")
def <- defData(def, "nperiods", formula = 3, dist = "noZeroPoisson")

def2 <- defDataAdd(varname = "lambda", formula = ".5+.5*period + .1*xbase",
                   dist="nonrandom", link = "log")
def2 <- defDataAdd(def2, varname = "p", formula = "-3+.2*period + .3*xbase",
                   dist="nonrandom", link = "logit")
def2 <- defDataAdd(def2, varname = "gammaMu", formula = ".2*period + .3*xbase",
                   dist="nonrandom", link = "log")
def2 <- defDataAdd(def2, varname = "gammaDis", formula = 1, dist="nonrandom")
def2 <- defDataAdd(def2, varname = "normMu", formula = "5+period + .5*xbase", dist="nonrandom")
def2 <- defDataAdd(def2, varname = "normVar", formula = 4, dist="nonrandom")
def2 <- defDataAdd(def2, varname = "unifMin", formula = "5 + 2*period + .2*xbase", dist="nonrandom")
def2 <- defDataAdd(def2, varname = "unifMax", formula = "unifMin + 20", dist="nonrandom")

dt <- genData(1000, def)

dtLong <- addPeriods(dt, idvars = "cid", nPeriods = 3)
dtLong <- addColumns(def2, dtLong)

# Poisson distribution

dtX3 <- addCorGen(dtOld = dtLong, idvar = "cid", nvars = 3, rho = .6, corstr = "cs",
                  dist = "poisson", param1 = "lambda", cnames = "NewPois")
dtX3

# Binomial distribution

dtX4 <- addCorGen(dtOld = dtLong, idvar = "cid", nvars = 3, rho = .6, corstr = "cs",
                  dist = "binary", param1 = "p", cnames = "NewBin")
```

```
dtX4

# Gamma distribution

dtX6 <- addCorGen(dtOld = dtLong, idvar = "cid", nvars = 3, rho = .6, corstr = "ar1",
                  dist = "gamma", param1 = "gammaMu", param2 = "gammaDis",
                  cnames = "NewGamma")

dtX6

# Normal distribution

dtX7 <- addCorGen(dtOld = dtLong, idvar = "cid", nvars = 3, rho = .6, corstr = "ar1",
                  dist = "normal", param1 = "normMu", param2 = "normVar",
                  cnames = "NewNorm")
```

---

addPeriods *Create longitudinal/panel data*

Description

Create longitudinal/panel data

Usage

```
addPeriods(dtName, nPeriods = NULL, idvars = "id", timevars = NULL,
  timevarName = "timevar", timeid = "timeID")
```

Arguments

dtName	Name of existing data table
nPeriods	Number of time periods for each record
idvars	Names of index variables (in a string vector) that will be repeated during each time period
timevars	Names of time dependent variables. Defaults to NULL.
timevarName	Name of new time dependent variable
timeid	Variable name for new index field. Defaults to "timevar"

Value

An updated data.table that that has multiple rows per observation in dtName

Examples

```
tdef <- defData(varname = "T", dist="binary", formula = 0.5)
tdef <- defData(tdef, varname = "Y0", dist = "normal", formula = 10, variance = 1)
tdef <- defData(tdef, varname = "Y1", dist = "normal", formula = "Y0 + 5 + 5 * T", variance = 1)
tdef <- defData(tdef, varname = "Y2", dist = "normal", formula = "Y0 + 10 + 5 * T", variance = 1)

dtTrial <- genData( 5, tdef)
dtTrial

dtTime <- addPeriods(dtTrial, nPeriods = 3, idvars = "id",
                     timevars = c("Y0", "Y1", "Y2"), timevarName = "Y")
dtTime

# Varying # of periods and intervals - need to have variables
# called nCount and mInterval

def <- defData(varname = "xbase", dist = "normal", formula = 20, variance = 3)
def <- defData(def,varname = "nCount", dist = "noZeroPoisson", formula = 6)
def <- defData(def, varname = "mInterval", dist = "gamma", formula = 30, variance = .01)
def <- defData(def, varname = "vInterval", dist = "nonrandom", formula = .07)

dt <- genData(200, def)
dt[id %in% c(8,121)]

dtPeriod <- addPeriods(dt)
dtPeriod[id %in% c(8,121)]  # View individuals 8 and 121 only
```

---

defCondition *Add single row to definitions table of conditions that will be used to add data to an existing definitions table*

Description

Add single row to definitions table of conditions that will be used to add data to an existing definitions table

Usage

```
defCondition(dtDefs = NULL, condition, formula, variance = 0,
  dist = "normal", link = "identity")
```

Arguments

dtDefs	Name of definition table to be modified. Null if this is a new definition.
condition	Formula specifying condition to be checked
formula	An R expression for mean (string)
variance	Number
dist	Distribution. For possibilities, see details
link	The link function for the mean, see details

Value

A data.table named dtName that is an updated data definitions table

Examples

```
# New data set

def <- defData(varname = "x", dist = "noZeroPoisson", formula=5)
def <- defData(def, varname="y", dist="normal", formula=0, variance=9)

dt <- genData(10, def)

# Add columns to dt

defC <- defCondition(condition = "x == 1", formula = "5 + 2*y",
                     variance = 1,dist = "normal")

defC <- defCondition(defC, condition = "x <= 5 & x >= 2", formula = "3 - 2*y",
                     variance = 1, dist="normal")

defC <- defCondition(defC, condition = "x >= 6", formula = 1,
                     variance = 1, dist="normal")

defC

# Add conditional column with field name "z"

dt <- addCondition(defC, dt, "z")
dt
```

---

defData *Add single row to definitions table*

Description

Add single row to definitions table

Usage

```
defData(dtDefs = NULL, varname, formula, variance = 0, dist = "normal",
  link = "identity", id = "id")
```

Arguments

dtDefs	Definition data.table to be modified
varname	Name (string) of new variable
formula	An R expression for mean (string)
variance	Number

dist	Distribution. For possibilities, see details
link	The link function for the mean, see details
id	A string indicating the field name for the unique record identifier

Details

The possible data distributions include ""normal", "poisson", "noZeroPoisson", "binary", "uniform", "uniformInt", "categorical", "gamma", "negBinomial", and "nonrandom."

Value

A data.table named dtName that is an updated data definitions table

Examples

```
def <- defData(varname = "xNr", dist = "nonrandom", formula=7, id = "idnum")
def <- defData(def, varname="xUni", dist="uniform", formula="10;20")
def <- defData(def, varname="xNorm", formula="xNr + xUni * 2", dist="normal", variance=8)
def <- defData(def, varname="xPois", dist="poisson", formula="xNr - 0.2 * xUni", link="log")
def <- defData(def, varname="xCat", formula = "0.3;0.2;0.5", dist="categorical")
def <- defData(def, varname="xGamma", dist="gamma", formula = "5+xCat", variance = 1, link = "log")
def <- defData(def, varname = "xBin", dist = "binary" , formula="-3 + xCat", link="logit")
def
```

---

defDataAdd *Add single row to definitions table that will be used to add data to an existing data.table*

Description

Add single row to definitions table that will be used to add data to an existing data.table

Usage

```
defDataAdd(dtDefs = NULL, varname, formula, variance = 0, dist = "normal",
  link = "identity")
```

Arguments

dtDefs	Name of definition table to be modified. Null if this is a new definition.
varname	Name (string) of new variable
formula	An R expression for mean (string)
variance	Number
dist	Distribution. For possibilities, see details
link	The link function for the mean, see details

Value

A data.table named dtName that is an updated data definitions table

Examples

```
# New data set

def <- defData(varname = "xNr", dist = "nonrandom", formula=7, id = "idnum")
def <- defData(def, varname="xUni", dist="uniform", formula="10;20")

dt <- genData(10, def)

# Add columns to dt

def2 <- defDataAdd(varname="y1", formula = 10, variance = 3)
def2 <- defDataAdd(def2, varname="y2", formula = .5, dist = "binary")
def2

dt <- addColumns(def2, dt)
dt
```

---

defMiss *Definitions for missing data*

Description

Add single row to definitions table for missing data

Usage

```
defMiss(dtDefs = NULL, varname, formula, logit.link = FALSE,
  baseline = FALSE, monotonic = FALSE)
```

Arguments

dtDefs	Definition data.table to be modified
varname	Name of variable with missingness
formula	Formula to describe pattern of missingness
logit.link	Indicator set to TRUE when the probability of missingness is based on a logit model.
baseline	Indicator is set to TRUE if the variable is a baseline measure and should be missing throughout an entire observation period. This is applicable to repeated measures/longitudinal data.
monotonic	Indicator set to TRUE if missingness at time t is followed by missingness at all follow-up times > t.

Value

A data.table named dtName that is an updated data definitions table

See Also

genMiss, genObs

Examples

```
def1 <- defData(varname = "m", dist = "binary", formula = .5)
def1 <- defData(def1, "u", dist = "binary", formula = .5)
def1 <- defData(def1, "x1", dist="normal", formula = "20*m + 20*u", variance = 2)
def1 <- defData(def1, "x2", dist="normal", formula = "20*m + 20*u", variance = 2)
def1 <- defData(def1, "x3", dist="normal", formula = "20*m + 20*u", variance = 2)

dtAct <- genData(1000, def1)

defM <- defMiss(varname = "x1", formula = .15, logit.link = FALSE)
defM <- defMiss(defM, varname = "x2", formula = ".05 + m * 0.25", logit.link = FALSE)
defM <- defMiss(defM, varname = "x3", formula = ".05 + u * 0.25", logit.link = FALSE)
defM <- defMiss(defM, varname = "u", formula = 1, logit.link = FALSE) # not observed
defM

# Generate missing data matrix

missMat <- genMiss(dtName = dtAct, missDefs = defM, idvars = "id")
missMat

# Generate observed data from actual data and missing data matrix

dtObs <- genObs(dtAct, missMat, idvars = "id")
dtObs
```

---

defRead *Read external csv data set definitions*

Description

Read external csv data set definitions

Usage

```
defRead(filen, id = "id")
```

Arguments

filen	String file name, including full path. Must be a csv file.
id	string that includes name of id field. Defaults to "id"

Value

A data.table with data set definitions

Examples

```
# Create temporary external "csv" file

test1 <- c("varname,formula,variance,dist,link",
           "nr,7, 0,nonrandom,identity",
           "x1,.4, 0,binary,identity",
           "y1,nr + x1 * 2,8,normal,identity",
           "y2,nr - 0.2 * x1,0,poisson, log"
          )

tfcsv <- tempfile()
writeLines(test1, tfcsv)

# Read external csv file stored in file "tfcsv"

defs <- defRead(tfcsv, id = "myID")
defs

unlink(tfcsv)

# Generate data based on external definition

genData(5, defs)
```

---

defReadAdd *Read external csv data set definitions for adding columns*

Description

Read external csv data set definitions for adding columns

Usage

```
defReadAdd(filen)
```

Arguments

filen	String file name, including full path. Must be a csv file.

Value

A data.table with data set definitions

Examples

```
# Create temporary external "csv" files

test1 <- c("varname,formula,variance,dist,link",
           "nr,7, 0,nonrandom,identity"
          )

tfcsv1 <- tempfile()
writeLines(test1, tfcsv1)

test2 <- c("varname,formula,variance,dist,link",
           "x1,.4, 0,binary,identity",
           "y1,nr + x1 * 2,8,normal,identity",
           "y2,nr - 0.2 * x1,0,poisson, log"
          )

tfcsv2 <- tempfile()
writeLines(test2, tfcsv2)

# Generate data based on external definitions

defs <- defRead(tfcsv1)
dt <- genData(5, defs)
dt

# Add additional data based on external definitions

defs2 <- defReadAdd(tfcsv2)
dt <- addColumns(defs2, dt)
dt

unlink(tfcsv1)
unlink(tfcsv2)
```

`defReadCond` *Read external csv data set definitions for adding columns*

Description

Read external csv data set definitions for adding columns

Usage

```
defReadCond(filen)
```

Arguments

`filen`	String file name, including full path. Must be a csv file.

Value

A data.table with data set definitions

Examples

```
# Create temporary external "csv" files

test1 <- c("varname,formula,variance,dist,link",
           "x,0.3;0.4;0.3,0,categorical,identity"
          )

tfcsv1 <- tempfile()
writeLines(test1, tfcsv1)

test2 <- c("condition,formula,variance,dist,link",
           "x == 1, 0.4,0,binary,identity",
           "x == 2, 0.6,0,binary,identity",
           "x >= 3, 0.8,0,binary,identity"
          )

tfcsv2 <- tempfile()
writeLines(test2, tfcsv2)

# Generate data based on external definitions

defs <- defRead(tfcsv1)
dt <- genData(2000, defs)
dt

# Add column based on

defsCond <- defReadCond(tfcsv2)
dt <- addCondition(defsCond, dt, "y")
dt

dt[, mean(y), keyby = x]

unlink(tfcsv1)
unlink(tfcsv2)
```

---

defSurv *Add single row to survival definitions*

Description

Add single row to survival definitions

Usage

```
defSurv(dtDefs = NULL, varname, formula = 0, scale, shape = 1)
```

Arguments

dtDefs	Definition data.table to be modified
varname	Variable name
formula	Covariates predicting survival
scale	Scale parameter for the Weibull distribution.
shape	The shape of the Weibull distribution. Shape = 1 for an exponential distribution

Value

A data.table named dtName that is an updated data definitions table

Examples

```
# Baseline data definitions

def <- defData(varname = "x1", formula = .5, dist = "binary")
def <- defData(def,varname = "x2", formula = .5, dist = "binary")
def <- defData(def,varname = "grp", formula = .5, dist = "binary")

# Survival data definitions

sdef <- defSurv(varname = "survTime", formula = "1.5*x1",
                scale = "grp*50 + (1-grp)*25", shape = "grp*1 + (1-grp)*1.5")

sdef <- defSurv(sdef, varname = "censorTime", scale = 80, shape = 1)

sdef

# Baseline data definitions

dtSurv <- genData(300, def)

# Add survival times

dtSurv <- genSurv(dtSurv, sdef)

head(dtSurv)
```

---

delColumns *Delete columns from existing data set*

Description

Delete columns from existing data set

Usage

```
delColumns(dtOld, vars)
```

Arguments

dtOld	Name of data table that is to be updated
vars	Vector of column names (as strings)

Value

An updated data.table that contains the added simulated data

Examples

```
# New data set

def <- defData(varname = "x", dist = "noZeroPoisson", formula=7, id = "idnum")
def <- defData(def, varname="xUni", dist="uniformInt", formula="x-3;x+3")

dt <- genData(10, def)
dt

# Delete column

dt <- delColumns(dt, "x")
dt
```

---

gammaGetShapeRate *Convert gamma mean and dispersion parameters to shape and rate parameters*

Description

Convert gamma mean and dispersion parameters to shape and rate parameters

Usage

```
gammaGetShapeRate(mean, dispersion)
```

Arguments

mean	The mean of a gamma distribution
dispersion	The dispersion parameter of a gamma distribution

Details

In simstudy, users specify the gamma distribution as a function of two parameters - a mean and dispersion. In this case, the variance of the specified distribution is (mean^2)*dispersion. The base R function rgamma uses the shape and rate parameters to specify the gamma distribution. This function converts the mean and dispersion into the shape and rate.

Value

A list that includes the shape and rate parameters of the gamma distribution

Examples

```
set.seed(12345)
mean = 5; dispersion = 1.5
rs <- gammaGetShapeRate(mean, dispersion)
c(rs$shape, rs$rate)
vec <- rgamma(1000, shape = rs$shape, rate = rs$rate)
(estMoments <- c(mean(vec), var(vec)))
(theoryMoments <- c(mean, mean^2*dispersion))
(theoryMoments <- c(rs$shape/rs$rate, rs$shape/rs$rate^2))
```

---

genCluster *Simulate clustered data*

Description

Simulate data set that is one level down in a multilevel data context. The level "2" data set must contain a field that specifies the number of individual records in a particular cluster.

Usage

```
genCluster(dtClust, cLevelVar, numIndsVar, level1ID, allLevel2 = TRUE)
```

Arguments

dtClust	Name of existing data set that contains the level "2" data
cLevelVar	Variable name (string) of cluster id in dtClust
numIndsVar	Variable name (string) of number of observations per cluster in dtClust
level1ID	Name of id field in new level "1" data set
allLevel2	Indicator: if set to TRUE (default), the returned data set includes all of the Level 2 data columns. If FALSE, the returned data set only includes the Levels 1 and 2 ids.

Value

A simulated data table with level "1" data

Examples

```
gen.school <- defData(varname="s0", dist = "normal",
 formula = 0, variance = 3, id = "idSchool"
)
gen.school <- defData(gen.school, varname = "nClasses",
                      dist = "noZeroPoisson", formula = 3
)

dtSchool <- genData(3, gen.school)#'
dtSchool

dtClass <- genCluster(dtSchool, cLevelVar = "idSchool",
                       numIndsVar = "nClasses", level1ID = "idClass")
dtClass
```

---

genCorData *Create correlated data*

Description

Create correlated data

Usage

```
genCorData(n, mu, sigma, corMatrix = NULL, rho, corstr = "ind",
  cnames = NULL, idname = "id")
```

Arguments

n	Number of observations
mu	A vector of means. The length of mu must be nvars.
sigma	Standard deviation of variables. If standard deviation differs for each variable, enter as a vector with the same length as the mean vector mu. If the standard deviation is constant across variables, as single value can be entered.
corMatrix	Correlation matrix can be entered directly. It must be symmetrical and positive semi-definite. It is not a required field; if a matrix is not provided, then a structure and correlation coefficient rho must be specified.
rho	Correlation coefficient, -1 <= rho <= 1. Use if corMatrix is not provided.
corstr	Correlation structure of the variance-covariance matrix defined by sigma and rho. Options include "ind" for an independence structure, "cs" for a compound symmetry structure, and "ar1" for an autoregressive structure.
cnames	Explicit column names. A single string with names separated by commas. If no string is provided, the default names will be V#, where # represents the column.
idname	The name of the index id name. Defaults to "id."

Value

A data.table with n rows and the k + 1 columns, where k is the number of means in the vector mu.

Examples

```
mu <- c(3, 8, 15)
sigma <- c(1, 2, 3)

corMat <- matrix(c(1, .2, .8, .2, 1, .6, .8, .6, 1), nrow = 3)

dtcor1 <- genCorData(1000, mu = mu, sigma = sigma, rho = .7, corstr = "cs")
dtcor2 <- genCorData(1000, mu = mu, sigma = sigma, corMatrix = corMat)

dtcor1
dtcor2

round(var(dtcor1[,.(V1, V2, V3)]), 3)
round(cor(dtcor1[,.(V1, V2, V3)]), 2)

round(var(dtcor2[,.(V1, V2, V3)]), 3)
round(cor(dtcor2[,.(V1, V2, V3)]), 2)
```

---

genCorFlex *Create multivariate (correlated) data - for general distributions*

Description

Create multivariate (correlated) data - for general distributions

Usage

```
genCorFlex(n, defs, rho = 0, tau = NULL, corstr = "cs",
  corMatrix = NULL)
```

Arguments

n	Number of observations
defs	Field definition table created by function ‘defData‘. All definitions must be scalar. Definition specifies distribution, mean, and variance, with all caveats for each of the distributions. (See defData).
rho	Correlation coefficient, -1 <= rho <= 1. Use if corMatrix is not provided.
tau	Correlation based on Kendall’s tau. If tau is specified, then it is used as the correlation even if rho is specfied. If tau is NULL, then the specified value of rho is used, or rho defaults to 0.
corstr	Correlation structure of the variance-covariance matrix defined by sigma and rho. Options include "cs" for a compound symmetry structure and "ar1" for an autoregressive structure. Defaults to "cs".

corMatrix — Correlation matrix can be entered directly. It must be symmetrical and positive semi-definite. It is not a required field; if a matrix is not provided, then a structure and correlation coefficient rho must be specified. This is only used if tau is not specified.

Value

data.table with added column(s) of correlated data

Examples

```
def <- defData(varname = "xNorm", formula = 0, variance = 4, dist = "normal")
def <- defData(def, varname = "xGamma1", formula = 15, variance = 2, dist = "gamma")
def <- defData(def, varname = "xBin", formula = 0.5, dist = "binary")
def <- defData(def, varname = "xUnif1", formula = "0;10", dist = "uniform")
def <- defData(def, varname = "xPois", formula = 15, dist = "poisson")
def <- defData(def, varname = "xUnif2", formula = "23;28", dist = "uniform")
def <- defData(def, varname = "xUnif3", formula = "100;150", dist = "uniform")
def <- defData(def, varname = "xGamma2", formula = 150, variance = 0.003, dist = "gamma")

dt <- genCorFlex(1000, def, tau = 0.3 , corstr = "cs")

cor(dt[,-"id"])
cor(dt[,-"id"], method = "kendall")
var(dt[,-"id"])
apply(dt[,-"id"], 2, mean)
```

---

genCorGen — *Create multivariate (correlated) data - for general distributions*

Description

Create multivariate (correlated) data - for general distributions

Usage

```
genCorGen(n, nvars, params1, params2 = NULL, dist, rho, corstr,
  corMatrix = NULL, wide = FALSE, cnames = NULL)
```

Arguments

n — Number of observations

nvars — Number of variables

params1 — A single vector specifying the mean of the distribution. The vector is of length 1 if the mean is the same across all observations, otherwise the vector is of length nvars. In the case of the uniform distribution the vector specifies the minimum.

params2	A single vector specifying a possible second parameter for the distribution. For the normal distribution, this will be the variance; for the gamma distribution, this will be the dispersion; and for the uniform distribution, this will be the maximum. The vector is of length 1 if the mean is the same across all observations, otherwise the vector is of length nvars.
dist	A string indicating "binary", "poisson" or "gamma", "normal", or "uniform".
rho	Correlation coefficient, -1 <= rho <= 1. Use if corMatrix is not provided.
corstr	Correlation structure of the variance-covariance matrix defined by sigma and rho. Options include "cs" for a compound symmetry structure and "ar1" for an autoregressive structure.
corMatrix	Correlation matrix can be entered directly. It must be symmetrical and positive semi-definite. It is not a required field; if a matrix is not provided, then a structure and correlation coefficient rho must be specified.
wide	The layout of the returned file - if wide = TRUE, all new correlated variables will be returned in a single record, if wide = FALSE, each new variable will be its own record (i.e. the data will be in long form). Defaults to FALSE.
cnames	Explicit column names. A single string with names separated by commas. If no string is provided, the default names will be V#, where # represents the column.

Value

data.table with added column(s) of correlated data

Examples

```
l <- c(8, 10, 12)

genCorGen(1000, nvars = 3, params1 = l, dist = "poisson", rho = .7, corstr = "cs")
genCorGen(1000, nvars = 3, params1 = 5, dist = "poisson", rho = .7, corstr = "cs")
genCorGen(1000, nvars = 3, params1 = l, dist = "poisson", rho = .7, corstr = "cs", wide = TRUE)
genCorGen(1000, nvars = 3, params1 = 5, dist = "poisson", rho = .7, corstr = "cs", wide = TRUE)

genCorGen(1000, nvars = 3, params1 = l, dist = "poisson", rho = .7, corstr = "cs",
          cnames = "new_var")
genCorGen(1000, nvars = 3, params1 = l, dist = "poisson", rho = .7, corstr = "cs",
          wide = TRUE, cnames = "a, b, c")

genCorGen(1000, nvars = 3, params1 = c(.3, .5, .7), dist = "binary", rho = .3, corstr = "cs")
genCorGen(1000, nvars = 3, params1 = l, params2 = c(1,1,1), dist = "gamma", rho = .3,
          corstr = "cs", wide = TRUE)
```

---

genData *Calling function to simulate data*

Description

Calling function to simulate data

Usage

```
genData(n, dtDefs = NULL, id = "id")
```

Arguments

n	the number of observations required in the data set.
dtDefs	name of definitions data.table/data.frame. If no definitions are provided a data set with ids only is generated.
id	The string defining the id of the record

Value

A data.table that contains the simulated data.

Examples

```
genData(5)
genData(5, id = "grpID")

def <- defData(varname = "xNr", dist = "nonrandom", formula=7, id = "idnum")
def <- defData(def, varname="xUni", dist="uniform", formula="10;20")
def <- defData(def, varname="xNorm", formula="xNr + xUni * 2", dist="normal", variance=8)
def <- defData(def, varname="xPois", dist="poisson", formula="xNr - 0.2 * xUni", link="log")
def <- defData(def, varname="xCat", formula = "0.3;0.2;0.5", dist="categorical")
def <- defData(def, varname="xGamma", dist="gamma", formula = "5+xCat", variance = 1, link = "log")
def <- defData(def, varname = "xBin", dist = "binary" , formula="-3 + xCat", link="logit")
def

genData(5, def)
```

---

genDummy *Create dummy variables from a factor or integer variable*

Description

Create dummy variables from a factor or integer variable

Usage

```
genDummy(dtName, varname, sep = ".", replace = FALSE)
```

Arguments

dtName	Data table with column
varname	Name of factor
sep	Character to be used in creating new name for dummy fields. Valid characters include all letters and "_". Will default to ".". If an invalid character is provided, it will be replaced by default.
replace	If replace is set to TRUE (defaults to FALSE) the field referenced varname will be removed.

Examples

```
# First example:

def <- defData(varname = "cat", formula = ".2;.3;.5", dist="categorical")
def <- defData(def, varname = "x", formula = 5, variance = 2)

dx <- genData(200,def)
dx

dx <- genFactor(dx, "cat", labels = c("one", "two", "three"), replace = TRUE)
dx <- genDummy(dx, varname = "fcat", sep = "_")

dx

# Second example:

dx <- genData(15)
dx <- trtAssign(dtName = dx, 3, grpName = "arm")
dx <- genDummy(dx, varname = "arm")
dx
```

---

genFactor *Create factor variable from an existing (non-double) variable*

Description

Create factor variable from an existing (non-double) variable

Usage

```
genFactor(dtName, varname, labels = NULL, prefix = "f", replace = FALSE)
```

Arguments

dtName	Data table with column
varname	Name of field that is to be converted
labels	Factor level labels. If not provided, the generated factor levels will be used as the labels.
prefix	By default, the new field name will be a concatenation of "f" and the old field name. A prefix string can be provided.
replace	If replace is set to TRUE (defaults to FALSE) the field referenced varname will be removed.

Examples

```
# First example:

def <- defData(varname = "cat", formula = ".2;.3;.5", dist="categorical")
def <- defData(def, varname = "x", formula = 5, variance = 2)

dx <- genData(200,def)
dx

dx <- genFactor(dx, "cat", labels = c("one", "two", "three"))
dx

# Second example:

dx <- genData(10)
dx <- trtAssign(dtName = dx, 2, grpName = "studyArm")
dx <- genFactor(dx, varname = "studyArm", labels = c("control", "treatment"), prefix = "t_")
dx
```

---

genFormula *Generate a linear formula*

Description

Formulas for additive linear models can be generated with specified coefficient values and variable names.

Usage

```
genFormula(coefs, vars)
```

Arguments

coefs	A numerical vector that contains the values of the coefficients. If length(coefs) == length(vars), then no intercept is assumed. Otherwise, an intercept is assumed.
vars	A vector of strings that specify the names of the explanatory variables in the equation.

Value

A string that represents the desired formula

Examples

```
genFormula(c(.5, 2, 4), c("A", "B", "C"))
genFormula(c(.5, 2, 4), c("A", "B"))

changeX <- c(7, 10)
genFormula(c(.5, 2, changeX[1]), c("A", "B"))
genFormula(c(.5, 2, changeX[2]), c("A", "B"))
genFormula(c(.5, 2, changeX[2]), c("A", "B", "C"))

newForm <- genFormula(c(-2, 1), c("A"))

def1 <- defData(varname = "A", formula = 0, variance = 3, dist = "normal")
def1 <- defData(def1, varname = "B", formula = newForm, dist = "binary", link = "logit")

set.seed(2001)
dt <- genData(500, def1)
summary(glm(B ~ A, data = dt, family = binomial))
```

---

genMiss *Generate missing data*

Description

Generate missing data

Usage

```
genMiss(dtName, missDefs, idvars, repeated = FALSE, periodvar = "period")
```

Arguments

dtName	Name of complete data set
missDefs	Definitions of missingness
idvars	Index variables
repeated	Indicator for longitudinal data
periodvar	Name of variable that contains period

Value

Missing data matrix indexed by idvars (and period if relevant)

See Also

defMiss, genObs

Examples

```
def1 <- defData(varname = "m", dist = "binary", formula = .5)
def1 <- defData(def1, "u", dist = "binary", formula = .5)
def1 <- defData(def1, "x1", dist="normal", formula = "20*m + 20*u", variance = 2)
def1 <- defData(def1, "x2", dist="normal", formula = "20*m + 20*u", variance = 2)
def1 <- defData(def1, "x3", dist="normal", formula = "20*m + 20*u", variance = 2)

dtAct <- genData(1000, def1)

defM <- defMiss(varname = "x1", formula = .15, logit.link = FALSE)
defM <- defMiss(defM, varname = "x2", formula = ".05 + m * 0.25", logit.link = FALSE)
defM <- defMiss(defM, varname = "x3", formula = ".05 + u * 0.25", logit.link = FALSE)
defM <- defMiss(defM, varname = "u", formula = 1, logit.link = FALSE) # not observed
defM

# Generate missing data matrix

missMat <- genMiss(dtAct, defM, idvars = "id")
missMat

# Generate observed data from actual data and missing data matrix

dtObs <- genObs(dtAct, missMat, idvars = "id")
dtObs
```

---

genObs *Create an observed data set that includes missing data*

Description

Create an observed data set that includes missing data

Usage

```
genObs(dtName, dtMiss, idvars)
```

Arguments

dtName	Name of complete data set
dtMiss	Name of missing data matrix
idvars	Index variables that cannot be missing

Value

A data table that represents observed data, including missing data

See Also

defMiss, genMiss

Examples

```
def1 <- defData(varname = "m", dist = "binary", formula = .5)
def1 <- defData(def1, "u", dist = "binary", formula = .5)
def1 <- defData(def1, "x1", dist="normal", formula = "20*m + 20*u", variance = 2)
def1 <- defData(def1, "x2", dist="normal", formula = "20*m + 20*u", variance = 2)
def1 <- defData(def1, "x3", dist="normal", formula = "20*m + 20*u", variance = 2)

dtAct <- genData(1000, def1)

defM <- defMiss(varname = "x1", formula = .15, logit.link = FALSE)
defM <- defMiss(defM, varname = "x2", formula = ".05 + m * 0.25", logit.link = FALSE)
defM <- defMiss(defM, varname = "x3", formula = ".05 + u * 0.25", logit.link = FALSE)
defM <- defMiss(defM, varname = "u", formula = 1, logit.link = FALSE) # not observed
defM

# Generate missing data matrix

missMat <- genMiss(dtAct, defM, idvars = "id")
missMat

# Generate observed data from actual data and missing data matrix

dtObs <- genObs(dtAct, missMat, idvars = "id")
dtObs
```

---

genOrdCat *Generate ordinal categorical data*

Description

Ordinal categorical data is added to an existing data set.

Usage

```
genOrdCat(dtName, adjVar, baseprobs, catVar = "cat", asFactor = TRUE)
```

Arguments

dtName	Name of complete data set
adjVar	Adjustment variable name in dtName - determines logistic shift. This is specified assuming a cumulative logit link.
baseprobs	Baseline probability expressed as a vector of probabilities. The values must sum to <= 1. If sum(baseprobs) < 1, an additional category is added with probability 1 - sum(baseprobs).
catVar	Name of the new categorical field. Defaults to "cat"
asFactor	If asFactor == TRUE (default), new field is returned as a factor. If asFactor == FALSE, new field is returned as an integer.

Value

Original matrix with added categorical field

Examples

```
#### Set definitions

def1 <- defData(varname = "male", formula = 0.45, dist = "binary", id = "idG")
def1 <- defData(def1, varname = "z", formula = "1.2*male", dist = "nonrandom")

#### Generate data

set.seed(20)

dx <- genData(1000, def1)

probs<-c(0.40, 0.25, 0.15)
dx <- genOrdCat(dx, adjVar = "z", probs, catVar = "grp")

# test ordinal model
m1 <- ordinal::clm(grp ~ male, data = dx, link = "logit")
summary(m1)

# check true cumulative log odds
log(cumsum(probs)/(1-cumsum(probs)))
```

---

genSpline *Generate spline curves*

Description

Generate spline curves

Usage

```
genSpline(dt, newvar, predictor, theta, knots = c(0.25, 0.5, 0.75),
  degree = 3, newrange = NULL, noise.var = 0)
```

Arguments

dt	data.table that will be modified
newvar	Name of new variable to be created
predictor	Name of field in old data.table that is predicting new value
theta	A vector or matrix of values between 0 and 1. Each column of the matrix represents the weights/coefficients that will be applied to the basis functions determined by the knots and degree. Each column of theta represents a separate spline curve.
knots	A vector of values between 0 and 1, specifying quartile cut-points for splines. Defaults to c(0.25, 0.50, 0.75).
degree	Integer specifying polynomial degree of curvature.
newrange	Range of the spline function , specified as a string with two values separated by a semi-colon. The first value represents the minimum, and the second value represents the maximum. Defaults to NULL, which sets the range to be between 0 and 1.
noise.var	Add to normally distributed noise to observation - where mean is value of spline curve.

Value

A modified data.table with an added column named newvar.

Examples

```
ddef <- defData(varname = "age", formula = "0;1", dist = "uniform")

theta1 = c(0.1, 0.8, 0.6, 0.4, 0.6, 0.9, 0.9)
knots <- c(0.25, 0.5, 0.75)

viewSplines(knots = knots, theta = theta1, degree = 3)

set.seed(234)
dt <- genData(1000, ddef)

dt <- genSpline(dt = dt, newvar = "weight",
                predictor = "age", theta = theta1,
                knots = knots, degree = 3,
                noise.var = .025)

dt
```

genSurv *Generate survival data*

Description

Survival data is added to an existing data set.

Usage

```
genSurv(dtName, survDefs, digits = 3)
```

Arguments

dtName	Name of complete data set
survDefs	Definitions of survival
digits	Number of digits for rounding

Value

Original matrix with survival time

Examples

```
# Baseline data definitions

def <- defData(varname = "x1", formula = .5, dist = "binary")
def <- defData(def,varname = "x2", formula = .5, dist = "binary")
def <- defData(def,varname = "grp", formula = .5, dist = "binary")

# Survival data definitions

sdef <- defSurv(varname = "survTime", formula = "1.5*x1",
                scale = "grp*50 + (1-grp)*25", shape = "grp*1 + (1-grp)*1.5")

sdef <- defSurv(sdef, varname = "censorTime", scale = 80, shape = 1)

sdef

# Baseline data definitions

dtSurv <- genData(300, def)

# Add survival times

dtSurv <- genSurv(dtSurv, sdef)

head(dtSurv)
```

mergeData *Merge two data tables*

Description

Merge two data tables

Usage

```
mergeData(dt1, dt2, idvars)
```

Arguments

dt1	Name of first data.table
dt2	Name of second data.table
idvars	Vector of string names to merge on

Value

A new data table that merges dt2 with dt1

Examples

```
def1 <- defData(varname="x", formula = 0, variance = 1)
def1 <- defData(varname="xcat", formula = ".3;.2", dist = "categorical")

def2 <- defData(varname="yBin", formula = 0.5, dist = "binary", id="xcat")
def2 <- defData(def2, varname="yNorm", formula = 5, variance = 2)

dt1 <- genData(20, def1)
dt2 <- genData(3, def2)

dtMerge <- mergeData(dt1, dt2, "xcat")
dtMerge
```

negbinomGetSizeProb *Convert negative binomial mean and dispersion parameters to size and prob parameters*

Description

Convert negative binomial mean and dispersion parameters to size and prob parameters

Usage

```
negbinomGetSizeProb(mean, dispersion)
```

Arguments

mean	The mean of a gamma distribution
dispersion	The dispersion parameter of a gamma distribution

Details

In simstudy, users specify the negative binomial distribution as a function of two parameters - a mean and dispersion. In this case, the variance of the specified distribution is mean + (mean^2)*dispersion. The base R function rnbinom uses the size and prob parameters to specify the negative binomial distribution. This function converts the mean and dispersion into the size and probability parameters.

Value

A list that includes the size and prob parameters of the neg binom distribution

Examples

```
set.seed(12345)
mean = 5; dispersion = 0.5
sp <- negbinomGetSizeProb(mean, dispersion)
c(sp$size, sp$prob)
vec <- rnbinom(1000, size = sp$size, prob = sp$prob)
(estMoments <- c(mean(vec), var(vec)))
(theoryMoments <- c(mean, mean + mean^2*dispersion))
(theoryMoments <- c(sp$size*(1-sp$prob)/sp$prob, sp$size*(1-sp$prob)/sp$prob^2))
```

---

trtAssign *Assign treatment*

Description

Assign treatment

Usage

```
trtAssign(dtName, nTrt = 2, balanced = TRUE, strata = NULL,
  grpName = "trtGrp")
```

Arguments

dtName	data table
nTrt	number of treatment groups
balanced	indicator for treatment assignment process
strata	vector of strings representing stratifying variables
grpName	string representing variable name for treatment or exposure group

Value

An integer (group) ranging from 1 to length of the probability vector

See Also

trtObserve

Examples

```
dt <- genData(15)

dt1 <- trtAssign(dt, nTrt = 3, balanced = TRUE)
dt1[, .N, keyby = trtGrp]

dt2 <- trtAssign(dt, nTrt = 3, balanced = FALSE)
dt2[, .N, keyby = trtGrp]

def <- defData(varname = "male", formula = .4, dist = "binary")
dt <- genData(1000, def)
dt

dt3 <- trtAssign(dt, nTrt = 5, strata = "male", balanced = TRUE, grpName = "Group")
dt3
dt3[, .N, keyby = .(male, Group)]
dt3[, .N, keyby = .(Group)]

dt4 <- trtAssign(dt, nTrt = 5, strata = "male", balanced = FALSE, grpName = "Group")
dt4[, .N, keyby = .(male, Group)]
dt4[, .N, keyby = .(Group)]

dt5 <- trtAssign(dt, nTrt = 5, balanced = TRUE, grpName = "Group")
dt5[, .N, keyby = .(male, Group)]
dt5[, .N, keyby = .(Group)]
```

---

trtObserve *Observed exposure or treatment*

Description

Observed exposure or treatment

Usage

```
trtObserve(dt, formulas, logit.link = FALSE, grpName = "trtGrp")
```

Arguments

dt	data table
formulas	collection of formulas that determine probabilities
logit.link	indicator that specifies link. If TRUE, then logit link is used. If FALSE, the identity link is used.
grpName	character string representing name of treatment/exposure group variable

Value

An integer (group) ranging from 1 to length of the probability vector

See Also

trtAssign

Examples

```
def <- defData(varname = "male", dist = "binary", formula = .5 , id="cid")
def <- defData(def, varname = "over65", dist = "binary", formula = "-1.7 + .8*male", link="logit")
def <- defData(def, varname = "baseDBP", dist = "normal", formula = 70, variance = 40)

dtstudy <- genData(1000, def)
dtstudy

formula1 <- c("-2 + 2*male - .5*over65", "-1 + 2*male + .5*over65")
dtObs <- trtObserve(dtstudy, formulas = formula1, logit.link = TRUE, grpName = "exposure")
dtObs

# Check actual distributions

dtObs[, .(pctMale = round(mean(male),2)), keyby = exposure]
dtObs[, .(pctMale = round(mean(over65),2)), keyby = exposure]

dtSum <- dtObs[, .N, keyby = .(male, over65, exposure)]
dtSum[, grpPct :=round(N/sum(N), 2), keyby = .(male, over65)]
dtSum
```

---

updateDef *Update definition table*

Description

Updates row definition table created by function defData or defRead. (For tables created using defDataAdd and defReadAdd use updateDefAdd.)

Usage

```
updateDef(dtDefs, changevar, newformula = NULL, newvariance = NULL,
  newdist = NULL, newlink = NULL, remove = FALSE)
```

Arguments

dtDefs	Definition table that will be modified
changevar	Name of field definition that will be changed
newformula	New formula definition (defaults to NULL)
newvariance	New variance specification (defaults to NULL)
newdist	New distribution definition (defaults to NULL)
newlink	New link specification (defaults to NULL)
remove	If set to TRUE, remove definition (defaults to FALSE)

Value

A string that represents the desired formula

Examples

```
# Example 1

defs <- defData(varname = "x", formula = 0, variance = 3, dist = "normal")
defs <- defData(defs, varname = "y", formula = "2 + 3*x", variance = 1, dist = "normal")
defs <- defData(defs, varname = "z", formula = "4 + 3*x - 2*y", variance = 1, dist = "normal")

defs

updateDef(dtDefs = defs, changevar = "y", newformula = "x + 5", newvariance = 2)
updateDef(dtDefs = defs, changevar = "z", newdist = "poisson", newlink = "log")

# Example 2

defs <- defData(varname = "w", formula = 0, variance = 3, dist = "normal")
defs <- defData(defs, varname = "x", formula = "1 + w", variance = 1, dist = "normal")
defs <- defData(defs, varname = "z", formula = 4, variance = 1, dist = "normal")

defs

updateDef(dtDefs = defs, changevar = "x", remove = TRUE)
updateDef(dtDefs = defs, changevar = "z", remove = TRUE)
```

---

updateDefAdd *Update definition table*

Description

Updates row definition table created by functions defDataAdd and defReadAdd. (For tables created using defData or defRead use updateDef.)

Usage

```
updateDefAdd(dtDefs, changevar, newformula = NULL, newvariance = NULL,
  newdist = NULL, newlink = NULL, remove = FALSE)
```

Arguments

dtDefs	Definition table that will be modified
changevar	Name of field definition that will be changed
newformula	New formula definition (defaults to NULL)
newvariance	New variance specification (defaults to NULL)
newdist	New distribution definition (defaults to NULL)
newlink	New link specification (defaults to NULL)
remove	If set to TRUE, remove definition (defaults to FALSE)

Value

A string that represents the desired formula

Examples

```
# Define original data

defs <- defData(varname = "w", formula = 0, variance = 3, dist = "normal")
defs <- defData(defs, varname = "x", formula = "1 + w", variance = 1, dist = "normal")
defs <- defData(defs, varname = "z", formula = 4, variance = 1, dist = "normal")

# Define additional columns

defsA <- defDataAdd(varname = "a", formula = "w + x + z", variance = 2, dist = "normal")

set.seed(2001)
dt <- genData(10, defs)
dt <- addColumns(defsA, dt)
dt

# Modify definition of additional column

defsA <- updateDefAdd(dtDefs = defsA, changevar = "a", newformula = "w+z", newvariance = 1)

set.seed(2001)
dt <- genData(10, defs)
dt <- addColumns(defsA, dt)
dt
```

viewBasis *Plot basis spline functions*

Description

Plot basis spline functions

Usage

```
viewBasis(knots, degree)
```

Arguments

knots	A vector of values between 0 and 1, specifying cut-points for splines
degree	Integer specifying degree of curvature.

Value

A ggplot object that contains a plot of the basis functions. In total, there will be length(knots) + degree + 1 functions plotted.

Examples

```
knots <- c(0.25, 0.50, 0.75 )
viewBasis(knots, degree = 1)

knots <- c(0.25, 0.50, 0.75 )
viewBasis(knots, degree = 2)

knots <- c(0.25, 0.50, 0.75 )
viewBasis(knots, degree = 3)
```

viewSplines *Plot spline curves*

Description

Plot spline curves

Usage

```
viewSplines(knots, degree, theta)
```

Arguments

knots	A vector of values between 0 and 1, specifying cut-points for splines
degree	Integer specifying degree of curvature.
theta	A vector or matrix of values between 0 and 1. Each column of the matrix represents the weights/coefficients that will be applied to the basis functions determined by the knots and degree. Each column of theta represents a separate spline curve.

Value

A ggplot object that contains a plot of the spline curves. The number of spline curves in the plot will equal the number of columns in the matrix (or it will equal 1 if theta is a vector).

Examples

```
knots <- c(0.25, 0.5, 0.75)
theta1 = c(0.1, 0.8, 0.4, 0.9, 0.2, 1.0)

viewSplines(knots, degree = 2, theta1)

theta2 = matrix(c(0.1, 0.2, 0.4, 0.9, 0.2, 0.3,
                  0.1, 0.3, 0.3, 0.8, 1.0, 0.9,
                  0.1, 0.4, 0.3, 0.8, 0.7, 0.5,
                  0.1, 0.9, 0.8, 0.2, 0.1, 0.6),
                ncol = 4)

viewSplines(knots, degree = 2, theta2)
```

Index