Package ‘gvcm.cat’

March 16, 2015

Type Package
Title Regularized Categorical Effects/Categorical Effect Modifiers/Continuous/Smooth Effects in GLMs
Version 1.9
Date 2015-03-16
Author Margret Oelker
Maintainer Margret Oelker <Margret.Oelker@stat.uni-muenchen.de>
Description Generalized structured regression models with regularized categorical effects, categorical effect modifiers, continuous effects and smooth effects.
Depends R (>= 2.10.0), Matrix, MASS, splines, mgcv
License GPL (>= 2)
LazyLoad yes
NeedsCompilation no
Repository CRAN
Date/Publication 2015-03-16 19:26:36

R topics documented:
cat_control ................................................................. 2
gvcm.cat ................................................................. 4
gvcm.cat-internal ....................................................... 9
gvcm.cat.flex ............................................................ 9
index ................................................................. 11
plot.gvcm.cat ........................................................... 13
predict.gvcm.cat ...................................................... 14
simulation ............................................................ 15
Index 17
**Description**

Auxiliary function for `gvcm.cat`. Modifies the algorithm's internal parameters.

**Usage**

```r
cat_control(center = FALSE, standardize = FALSE, accuracy = 2, digits = 4,
g = 0.5, epsilon = 10^(-5), maxi = 250, c = 10^(-5), gama = 20, steps = 25,
nu = 1, tuning.criterion = "GCV", K = 5, cv.refit = FALSE,
lambda.upper=50, lambda.lower=0, lambda.accuracy=.01, scaled.lik=FALSE,
adapted.weights=FALSE, adapted.weights.adj = FALSE, adapted.weights.ridge =
FALSE, assured.intercept=TRUE,
level.control = FALSE, case.control = FALSE, pairwise = TRUE,
grouped.cat.diffs = FALSE, bootstrap = 0, start.ml = FALSE, L0.log = TRUE,
subjspec.gr = FALSE, high = NULL, ...)
```

**Arguments**

- `center` logical; if TRUE, all metric covariates are centered by their empirical mean
- `standardize` logical; if TRUE, the design matrix is standardized by its (weighted) empirical variances
- `accuracy` integer; number of digits being compared when setting coefficients equal/to zero
- `digits` integer; number of digits for estimates
- `g` step length parameter for the PIRLS-algorithm; out of ]0,1[
- `epsilon` small, positive, real constant; the PIRLS-algorithm is terminated when the (scaled, absolute) difference of the coefficients of the current iteration and the coefficients of the previous iteration is smaller than epsilon
- `maxi` integer; maximal number of iterations in the fitting algorithm
- `c` small, positive, real constant; needed for the approximation of the absolute value function in the PIRLS-algorithm
- `gama` positive number; tuning parameter for the approximation of the L0 norm
- `steps` integer; tuning parameter for path-plotting; minimal number of estimates employed for path-plotting
- `nu` optional weighting parameter
- `tuning.criterion` loss criterion for cross-validation; one out of "GCV" (generalized cross validation criterion), "deviance" (K-fold cross-validation with the predictive deviance as criterion)
- `K` integer; number of folds for cross-validation
- `cv.refit` logical; if TRUE, cross-validation is based on a refit of the selected coefficients
lambda.upper integer; upper bound for cross-validation of lambda
lambda.lower integer; lower bound for cross-validation of lambda
lambda.accuracy numeric; how accurate shall lambda be cross-validated?; minimal absolute difference between two candidates for lambda
scaled.lik if TRUE, the likelihood in the objective function is scaled by 1/n
adapted.weights logical; if TRUE, penalty terms are weighted adaptively, that is by inverse ML-estimates; set to FALSE, if ML-estimates do not exist/are to close to zero; only for specials v, p, grouped, SCAD, elastic
adapted.weights.adj logical; if TRUE, adapted weights of several categorical covariates are scaled such that they are comparable
adapted.weights.ridge logical; if TRUE, adapted weights are based on an estimate that is slightly penalized by a Ridge penalty
assured.intercept logical; shall a constant intercept remain in the model in any case?
level.control logical; if TRUE, the penalty terms are adjusted for different number of penalty terms per covariate
case.control logical; if TRUE, the penalty terms are adjusted for the number of observations on each level of a categorical covariate
pairwise experimental option; disabled if TRUE
grouped.cat.diffs experimental option; disabled if FALSE
bootstrap experimental option; disabled if 0
start.ml logical; if TRUE, the initial value is the ML-estimate
L0.log experimental option; disabled if TRUE
subjspec.gr experimental option; disabled if FALSE
high experimental option; disabled if NULL
... further arguments passed to or from other methods

Value

Returns a list containing the (checked) input arguments.

See Also

Function `gvcm.cat`
Description

The function fits generalized linear models with regularized categorical effects, categorical effect modifiers, continuous effects and smooth effects. The model is specified by giving a symbolic description of the linear predictor and a description of the error distribution. Estimation employs different regularization and model selection strategies. These strategies are either a penalty or a forward selection strategy employing AIC/BIC. For non-differentiable penalties, a local quadratic approximation is employed, see Oelker and Tutz (2013).

Usage

```r
gvcm.cat(formula, data, family = gaussian, method = c("lqa", "AIC", "BIC"), tuning = list(lambda=TRUE, specific=FALSE, phi=0.5, grouped.fused=0.5, elastic=0.5, vs=0.5, spl=0.5), weights, offset, start, control, model = FALSE, x = FALSE, y = FALSE, plot=FALSE, ...)

pest(x, y, indices, family = gaussian, tuning = list(lambda=TRUE, specific=FALSE, phi=0.5, grouped.fused=0.5, elastic=0.5, vs=0.5, spl=0.5), weights, offset, start = NULL, control = cat_control(), plot=FALSE, ...)

abc(x, y, indices, family = gaussian, tuning = c("AIC", "BIC"), weights, offset, start, control = cat_control(), plot=FALSE, ...)
```

Arguments

- **formula**: an object of class `formula`: a symbolic description of the model to be fitted. See details
- **data**: a data frame, containing the variables in the model
- **family**: a family object describing the error distribution and link function to be used in the model; this can be a character string naming a family function, a family function or the result of a call to a family function, see `family` for details; currently only gaussian, binomial, poisson, Gamma are working
- **method**: fitting method; one out of "lqa", "AIC" or "BIC"; method "lqa" induces penalized estimation; it employs a PIRLS-algorithm (see Fan and Li, 2001; Oelker and Tutz, 2013). Methods "AIC" and "BIC" employ a forward selection strategy
- **tuning**: a list; tuning parameters for penalized estimation; lambda is the scalar, overall penalty parameter; if lambda is a vector of values, these values are cross-validated; if lambda = TRUE, lambda is cross-validated on log scale between lambda.lower and lambda.upper; see `cat_control`. If lambda is a vector with the same length as elements in the formula and if specific equals a vector of proper length, the entries of specific are interpreted as specific tuning...
parameters for each entry of the formula. phi, grouped.fused, elastic, vs
and spl are parameters that weigh the terms of some penalties; must be out
of intervall (0,1); the default 0.5 corresponds to equal weights

weights
  an optional weight vector (for the observations)

offset
  an optional offset

start
  initial values for the PIRLS algorithm for method lqa

tuning
  a list of parameters for controlling the fitting process; if empty, set to cat_control();
  see cat_control

model
  for functions gvm.cat: a logical value indicating whether the employed model
  frame shall be returned or not

x, y
  for function gvm.cat: logical values indicating whether the response vector
  and the model matrix used in the fitting process shall be returned or not; for
  functions pest and abc: y must be a response vector, x a proper coded design
  matrix

plot
  logical; if TRUE, estimates needed to plot coefficient paths are computed

indices
  for pest and abc only: the to be used index argument; see function index

... further arguments passed to or from other methods

Details

A typical formula has the form response ~ 1 + terms; where response is the response vector
and terms is a series of terms which specifies a linear predictor. There are some special terms for
regularized terms:

- v(x, u, n="L1", bj=TRUE): varying coefficients enter the formula as v(x,u) where u
denotes the categorical effect modifier and x the modified covariate. A varying intercept is
denoted by v(1,u). Varying coefficients with categorical effect modifiers are penalized as
described in Oelker et al. 2012. The argument bj and the element phi in argument tuning
allow for the described weights.

- p(u, n="L1"): ordinal/nominal covariates u given as p(u) are penalized as described in
Gertheiss and Tutz (2010). For numeric covariates, p(u) indicates a pure Lasso penalty.

- grouped(u, ...): penalizes a group of covariates with the grouped Lasso penalty of Yuan
and Lin (2006); so far, working for categorical covariates only

- sp(x, knots=20, n="L2"): implements a continuous x covariate non-parametrically as f(x);
f(x) is represented by centered evaluations of basis functions (cubic B-splines with number
of knots = knots); for n="L2", the curvature of f(x) is penalized by a Ridge penalty; see
Eilers and Marx (1996)

- SCAD(u): penalizes a covariate u with the SCAD penalty by Fan and Li (2001); for categorical
covariates u, differences of coefficients are penalized by a SCAD penalty, see Gertheiss and
Tutz (2010)

- elastic(u): penalizes a covariate u with the elastic net penalty by Zou and Hastie (2005); for
categorical covariates u, differences of coefficients are penalized by the elastic net penalty,
see Gertheiss and Tutz (2010)
If the formula contains no (varying) intercept, gvcm.cat assumes a constant intercept. There is no way to avoid an intercept.

For specials p and v, there is the special argument n: if n="L1", the absolute values in the penalty are replaced by squares of the same terms; if n="L2", the absolute values in the penalty are replaced by quadratic, Ridge-type terms; if n="L0", the absolute values in the penalty are replaced by an indicator for non-zero entries of the same terms.

For methods "AIC" and "BIC", the coefficients are not penalized but selected by a forward selection strategy whenever it makes sense; for special v(x,u), the selection strategy is described in Oelker et. al. 2012; the approach for the other specials corresponds to this idea.

For binomial families the response can also be a success/failure rate or a two-column matrix with the columns giving the numbers of successes and failures.

Function pest computes penalized estimates, that is, it implements method "lqa" (PIRLS-algorithm). Function abc implements the forward selection strategy employing AIC/BIC.

Categorical effect modifiers and penalized categorical covariates are dummy coded as required by the penalty. If x in v(x,u) is binary, it is effect coded (first category refers to -1). Other covariates are coded like given by getOption.

There is a summary function: summary.gvcm.cat

Value

gvcm.cat returns an object of class “gvcm.cat” which inherits from class “glm” which inherits from class “lm”. An object of class “gvcm.cat” contains:

- coefficients: named vector of coefficients
- coefficients.reduced: reduced vector of coefficients; selected coefficients/differences of coefficients are set to zero
- coefficients.refitted: refitted vector of coefficients; i.e. maximum likelihood estimate of that model containing selected covariates only; same length as coefficients.reduced
- coefficients.oml: maximum likelihood estimate of the full model
- residuals: deviance residuals
- fitted.values: fitted mean values
- rank: degrees of freedom model; for method="lqa" estimated by the trace of the generalized head matrix; for methods "AIC", "BIC" estimated like default in glm.fit
- family: the family object used
- linear.predictors: linear fit on link scale
- deviance: scaled deviance
- aic: a version of Akaike’s Information Criterion; minus twice the maximized log-likelihood plus twice the rank. For binomial and Poison families the dispersion is fixed at one. For a gaussian family the dispersion is estimated from the residual deviance, and the number of parameters is the rank plus one.
- null.deviance: the deviance for the null model, comparable with deviance; the null model includes a non-varying intercept only
iter
weights
df.residual
df.null
converged
boundary
offset
control
contrasts
na.action
plot
tuning
indices
number.selectable.parameters
number.removed.parameters
x.reduction
beta.reduction
call
formula
terms
data
x, y
model
xlevels
bootstrap.errors
method

Note

Please note that the functions gvcm.cat, pest and the fitting procedure for penalized estimation gvcmcatfit are organized like the functions glm/glm.fit whenever possible. This was done to avoid mistakes and to provide a well-known structure.
Author(s)
Margret-Ruth Oelker (<margret.oelker@stat.uni-muenchen.de>)

References

See Also
Functions index, cat_control, plot.gvcm.cat, predict.gvcm.cat, simulation

Examples
```r
## example for function simulation()
covariates <- list(x1=list("unif", c(0,2)),
                  x2=list("unif", c(0,2)),
                  x3=list("unif", c(0,2)),
                  u=list("multinom",c(0.3,0.4,0.3), "nominal")
true.f <- y ~ 1 + v(x1,u) + x2
true.coefs <- c(0.2, 0.3,.7,.7, -.5)
data <- simulation(400, covariates, NULL, true.f, true.coefs, binomial(), seed=456)
## example for function gvcm.cat()
f <- y ~ v(1,u) + v(x1,u) + v(x2,u)
m1 <- gvcm.cat(f, data, binomial(), plot=TRUE, control=cat_control(lambda.upper=19))
summary(m1)
## example for function predict.gvcm.cat
newdata <- simulation(200, covariates, NULL, true.f, true.coefs, binomial(), seed=789)
prediction <- predict.gvcm.cat(m1, newdata)
## example for function plot.gvcm.cat
plot(m1)
plot(m1, type="score")
plot(m1, type="coefs")
```
Internal Function of `gvcm.cat()`

**Description**

For internal use only.

**See Also**

Function `gvcm.cat`

Regularized Effects with Flexible Smoothing Parameters

**Description**

The function fits the same models with the same approximation as in `gvcm.cat` but the choice of the tuning parameter lambda for the penalty differs: instead of weighting the penalty terms and choosing on global tuning parameter based on (generalized) cross-validation methods that again rely on the converged model, `gvcm.cat.flex` estimates several penalty parameteres lambda_i by linking the local quadratic approximation of `gvcm.cat` with the fantastic methods implemented in the package mgcv. This is why the arguments of `gvcm.cat` and `gvcm.cat.flex` differ. `gvcm.cat.flex` is not as well-developed as `gvcm.cat`.

**Usage**

```r
gvcm.cat.flex(whichCoefs, intercept = TRUE, data, family = gaussian(), method = "REML",
tuning = NULL, indexNrCoefs, indexPenNorm, indexPenA, indexPenWeight,
control = list(c=1e-05, epsilon=1e-07, gama=35, maxi=1500, nu=.5))
```

**Arguments**

- `whichCoefs` : vector with covariates (as characters)
- `intercept` : logical
- `data` : a data frame, with named and coded covariates
- `family` : a family object describing the error distribution and link function to be used in the model; see family for details; everyl family that is compatible with `gam` is working
- `method` : see `gam`
- `tuning` : for function `gam`: argument sp
- `indexNrCoefs` : vector with number of coefficients per covariate
- `indexPenNorm` : vector with norm of the employed penalty (as.character)
- `indexPenA` : list with the penalty matrices A_j for each covariate j
- `indexPenWeight` : list, possible weights for the penalty terms (each entry is a vector)
- `control` : a list of parameters for controlling the fitting process; must be NULL or contain all named elements
Details

The local quadratic approximation are linked to the methods of mgcv by alternating the update of the penalty and the update of the PIRLS algorithm/estimating the tuning parameters lambda_i via mgcv. Therefore, gvcm.cat.flex can be slow (but will be faster than gvcm.cat for the most part).

Value

A gamObject.

See Also

Function gvcm.cat.

Examples

```r
## Not run:
# compare gvcm.cat.flex and gvcm.cat for Lasso-type penalties:
n <- 100
ncov <- 7
set.seed(123)
X <- matrix(rnorm(n*nncov, sd=5), ncol=ncov)
coefs <- rpois(ncov + 1, 2)
y <- cbind(1, X)
data <- as.data.frame(cbind(y, X))
names(data) <- c("y", paste("x", 1:ncov, sep=""))

m1 <- gvcm.cat.flex(
  whichCoefs = paste("x", 1:ncov, sep=""),
  data=data,
  indexNrCoefs=rep(1, ncov),
  indexPenNorm=rep("L1", ncov),
  indexPen=list(1,1,1,1,1,1,1),
  indexPenWeight=list(1,1,1,1,1,1,1)
)

m2 <- gvcm.cat(y ~ 1 + p(x1) + p(x2) + p(x3) + p(x4) + p(x5) + p(x6) + p(x7),
  data=data, tuning=list(lambda=m1$sp, specific=TRUE), start=rep(1, 8))
rbind(m1$coefficients, m2$coefficients)

# Lasso-type fusion penalty with gvcm.cat.flex
n <- 100
ncat <- 8
set.seed(567)
X <- t(rmultinom(n, 1, rep(1/ncat, ncat))[, -1]
coefs <- c(rpois(1, 2), sort(rpois(ncat-1, 1)))
y <- cbind(1, X)
data <- as.data.frame(y)
data$x1 <- X
names(data) <- c("y", "x1")
```

index

\[
A \leftarrow a(1:(\text{ncat}-1), \text{ncat}-2)
\]

\[
m3 \leftarrow \text{gvcm.cat.flex}
\]

\[
\text{whichCoefs} = c("x1"),
\]

\[
\text{data} = \text{data},
\]

\[
\text{indexNrCoefs} = c(\text{ncat}-1),
\]

\[
\text{indexPenNorm} = c("L1"),
\]

\[
\text{indexPenA} = \text{list}(A),
\]

\[
\text{indexPenWeight} = \text{list}(\text{rep}(1, \text{ncol}(A))),
\]

\[
\text{tuning} = 100 \# \text{fixed and large – in order to demonstrate the fusion of the coefficients}
\]

\[
m3$\text{coefficients}
\]

## End(Not run)

---

### Functions to Build Design Matrices and Indices for Function `gvcm.cat()`

**Description**

`design()` builds design matrices for function `gvcm.cat`; `index()` computes indices with information about the terms of the formula.

**Usage**

```r
design(formula, data)
index(dsgn, data = data, formula = formula)
```

**Arguments**

- `formula` an object of class "formula"; see `gvcm.cat`
- `data` a data frame; see `gvcm.cat`
- `dsgn` value of function `design()`

**Details**

Function `index` returns a matrix with one indicator vector per row. The columns refer to the elements of the formula (same order). The indicator/indices are:

- `index1`: gives the number of coefficients belonging to each term in the formula. An entry is 1 if the according term is metric, it equals the number of the coded variable’s categories, if the variable is a factor. If a continuous variable is modified by a factor \(u\) the entry equals the number of \(u\)’s categories
- `index2`: indicates varying coefficients. An entry is 0 if the according coefficient is not varying, it is -1 if the according coefficient is nominal, 1 if it is ordinal
• index2b: conforms to indicator $b_j$ in Oelker et. al. 2012
• index3: indicates penalized covariates $p(u)$. An entry is 0 if the according covariate is not penalized, it is -1 if the according covariate is nominal, 1 if it is ordinal or metric
• index4: indicates penalized covariates grouped($u$). An entry is 0 if the according covariate is not penalized, it is -1 if the according covariate is nominal, 1 if it is ordinal or metric
• index5: experimental
• index6: indicates penalized covariates sp
• index7: indicates penalized covariates SCAD. An entry is 0 if the according covariate is not penalized, it is -1 if the according covariate is nominal, 1 if it is ordinal or metric
• index8: indicates penalized covariates elastic. An entry is 0 if the according covariate is not penalized, it is -1 if the according covariate is nominal, 1 if it is ordinal or metric
• index9: experimental

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>the model matrix</td>
</tr>
<tr>
<td>Terms</td>
<td>the according terms.object</td>
</tr>
<tr>
<td>m</td>
<td>the model frame</td>
</tr>
<tr>
<td>int</td>
<td>either 0, indicating that the intercept is varying, or 1 indicating that the intercept is constant</td>
</tr>
<tr>
<td>formula</td>
<td>sorted version of the given formula, index vectors will refer to this formula</td>
</tr>
<tr>
<td>a matrix</td>
<td>value of function index</td>
</tr>
</tbody>
</table>

References


See Also

Functions pest, abc

Examples

```r
## example for function simulation()
covariates <- list(x1=list("unif", c(0,2)),
                    x2=list("unif", c(0,2)),
                    x3=list("unif", c(0,2)),
                    u=list("multinom",c(0.3,0.4,0.3), "nominal")
)
true.f <- y ~ 1 + v(x1,u) + x2
true.coefs <- c(0.2, 0.3,.7,.7, -.5)
data <- simulation(400, covariates, NULL, true.f, true.coefs, binomial(), seed=456)
## example for function index()
f <- y ~ v(1,u) + v(x1,u) + v(x2,u)
```
plot.gvcm.cat

Plot Method for gvcm.cat Objects

Description
Function to visualize a gvcm.cat object.

Usage
## S3 method for class 'gvcm.cat'
plot(x, accuracy = 2, type = "path", individual = FALSE,
     xlim, ylim, main = NULL, indent = 0, color = TRUE, xscale = "lambda",
     label = TRUE, intercept = TRUE, ...) 

Arguments
x
a gvcm.cat object; for type="path", a gvcm.cat object with value plot unequal NA is required
accuracy
integer; number of digits being compared when setting coefficients equal/to zero for plotting
type
one out of "path", "score", "coeffs"; defines the type of the plot
individual
logical; for type="path" and type="coeffs" only; for type="path", it indicates whether the paths of all coefficients shall be plotted into one common figure (default) or in an individual figure per covariate; paths of single covariates can be selected by giving a vector containing the covariates (as characters and as given in the formula, e.g.: individual.paths=c("v(1,u)", "v(x1,u1)")) for type="coeffs", the default is one plot per covariate. individual allows to select single covariates.
xlim
the x limits (x1, x2) of the plot
ylim
the y limits (y1, y2) of the plot
main
title of the plot
indent
numeric; if larger zero, coefficient names printed on top of each other are adjusted
color
logical; if FALSE, lines are gray and dotted/dashed
xscale
for type="path" only; if xscale="lambda", the x-axis is scaled as $1 - \lambda/\lambda_{max}$; if xscale="beta", the scale of the x-axis is the scaled L1 norm of the penalized coefficients.
label
omits additional information printed in the plot, if FALSE
intercept
for type="coeffs" and type="path" only; if FALSE, for type="path", the path of the intercept is not plotted; if FALSE, for type="coeffs", intercept is not added to smooth functions
...

further arguments passed to or from other methods
Details

Default option type="path" delivers a graphic with the coefficient paths between 0 (= maximal penalization) and 1 (= no penalization). Maximal penalization is defined by the minimal penalty parameter lambda that sets all penalized coefficients to zero (to constant relating to the intercept and assured intercept = TRUE). Minimal penalization means no penalization at all, i.e. lambda = 0. Of course the minimal penalty parameter causing maximal penalization depends on how selection and clustering of coefficients is defined (see function gvcm.cat and cat_control). Coefficients belonging to one covariate are plotted in the same color, coefficients that are not modified are plotted as dashed lines. Paths are drawn by connecting steps estimates related to different values of lambda, see cat_control.

Option type="score" plots the cross-validation score (depending on criterion in cat_control) as a function of penalty parameter lambda and marks the chosen penalty parameter as a dotted line. Option type="coeffs" plots the penalized coefficients whenever possible.

So far, there is no plot for methods "AIC" and "BIC".

Value

A plot.

See Also

Function gvcm.cat

Examples

## see example for function gvcm.cat

### predict.gvcm.cat

**Predict Method for gvcm.cat Fits**

Description

Obtains predictions from a fitted gvcm.cat object.

Usage

```r
## S3 method for class 'gvcm.cat'
predict(object, newdata, type = "link", ...)
```

Arguments

- `object`: a fitted object of class `gvcm.cat`
- `newdata`: a data frame in which to look for variables with which to predict
- `type`: the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities
- `...`: further arguments passed to or from other methods
Details

Observations containing NAs are always omitted.

Value

- `fit`: predictions
- `fit.refitted`: predictions assuming refitted coefficients
- `fit.OM`: predictions assuming maximum likelihood estimates
- `na.action`: information returned by `model.frame` on the special handling of NAs; currently always `na.omit`

See Also

Function `gvcm.cat`

Examples

```r
## see example for function gvcm.cat
```

---

**simulation**

*Simulates data with categorical covariates*

Description

Simulates data with categorical covariates/categorical effect modifiers

Usage

```r
simulation(n, covariates, correlation = NULL, formula, coefficients, family, sd = 1, seed = rpois(1, 2348) * rnorm(1))
```

Arguments

- `n`: number of observations; must be large enough, so that all categories of all factor variables exist and therefore vector coefficients fits
- `covariates`: description of the covariates and effect modifiers included in the model; format: `list(name of variable = list("distribution", c(parameters), "level of measurement")`
- `correlation`: optional matrix, specifies the correlation of Gaussian covariates
- `formula`: formula like in `gvcm.cat` (all variables contained in `formula` must be defined in `covariates`)
- `coefficients`: true parameter vector
- `family`: a `family` object; currently only gaussian, binomial, poisson, Gamma
- `sd`: if `family` = gaussian, standard deviation of response; if `family` = Gamma the rate parameter like in `rgamma`
- `seed`: specifies the to be used seed
Details

Remarks on covariates:

- all parameterizations like default in `Distributions`.
- possible distributions of covariates (required as characters), their parameters (required as vectors) and constraints (in parentheses):
  - `beta`: `shape1 (>0), shape2 (>0)`
  - `exp`: rate (>0)
  - `gamma`: `shape (>0)`
  - `lnorm`: `mean, sd (>0)`
  - `multinom`: vector of the categories’ probabilities (all elements must be >0, sum over all elements must be 1)
  - `norm`: `mean, sd (>0)`
  - `pois`: `lambda (>0)`
  - `unif`: `min, max`

- level of measurement is only needed for distribution = “multinom”, must be “nominal” or “ordinal”.
- If any, the covariates’ correlation is specified by argument `correlation`. Correlations are defined for Gaussian covariates only. Matrix `correlation` refers to these covariates according to the order they are listed in `covariates`. So that the dimensions of `correlation` must fit to the number of normal distributed variables in `covariates`.

Value

A data frame containing all specified covariates (even if they are not included in `formula`) and the response (named `y`)

See Also

Function `gvcmNcat`

Examples

```r
## example function simulation
covariates <- list(
  list(x1=list("unif", c(0,2)), x2=list("unif", c(0,2)),
       x3=list("unif", c(0,2)), u=list("multinom", c(0.3,0.4,0.3), "nominal")
  )
)
true.f <- y ~ 1 + v(x1,u) + x2
ture.coefs <- c(0.2, 0.3, 0.3, 0.3)
data <- simulation(400, covariates, NULL, true.f, true.coefs, binomial(), seed=456)
```
Index

∗Topic gvcm.cat
  cat_control, 2
  index, 11
  plot.gvcm.cat, 13
  predict.gvcm.cat, 14
  simulation, 15

a(gvcm.cat-internal), 9
abc, 12
abc(gvcm.cat), 4
abc.a.coefs(gvcm.cat-internal), 9
abc.fit(gvcm.cat-internal), 9

bootstrap(gvcm.cat-internal), 9

cat_control, 2, 4, 5, 8, 14
check.simulation(gvcm.cat-internal), 9
contr.effect(gvcm.cat-internal), 9
cv.lambda(gvcm.cat-internal), 9
cv.vectors(gvcm.cat-internal), 9

design(index), 11
Distributions, 16

elastic(gvcm.cat-internal), 9

family, 4, 6, 9, 15
formula, 4–7, 15, 16

gam, 9
gamObject, 10
getOption, 6
glm, 6, 7
glm.fit, 6, 7
grouped(gvcm.cat-internal), 9
gvcm.cat, 2, 3, 4, 9–11, 13–16
gvcm.cat-internal, 9
gvcm.cat.flex, 9, 9, 10
gvcmcatfit(gvcm.cat-internal), 9
gvcmcatfitridge(gvcm.cat-internal), 9

index, 5, 7, 8, 11
lm, 6

mgcv, 9, 10
model.frame, 7, 15

NA, 7, 15
na.omit, 7, 15

p(gvcm.cat-internal), 9
path.matrix(gvcm.cat-internal), 9
pest, 12
pest(gvcm.cat), 4
plot.gvcm.cat, 8, 13
predict.gvcm.cat, 8, 14
print.gvcm.cat(gvcm.cat), 4
pspline(gvcm.cat-internal), 9

reduce(gvcm.cat-internal), 9
rgamma, 15

SCAD(gvcm.cat-internal), 9
simulation, 8, 15
sp(gvcm.cat-internal), 9
summary.gvcm.cat(gvcm.cat), 4
terms, 7
terms.object, 12

v(gvcm.cat-internal), 9
vspline(gvcm.cat-internal), 9

weight.function(gvcm.cat-internal), 9