Package 'GJRM'

June 15, 2026

Type Package

Version 0.2-6.9

Title Generalised Joint Regression Modelling

Description Routines for fitting various joint (and univariate) regression models, with several types of covariate effects, in the presence of equations' errors association.

Author Giampiero Marra [aut, cre],
Rosalba Radice [aut]

Maintainer Giampiero Marra <giampiero.marra@ucl.ac.uk>

Depends R (>= 3.6.0)

Imports mgcv, magic, VGAM, survey, trust, VineCopula, graphics, stats, utils, grDevices, ggplot2, matrixStats, mnormt, gamlss.dist, Rmpfr, scam, psych, copula, numDeriv, evd, ismev, methods, distrEx, pracma

Enhances sp

LazyLoad yes

License GPL (>= 2)

URL https://www.ucl.ac.uk/mathematical-physical-sciences/statistics/people/academic-and-lecturing-staff/professor-giampiero-marra

Repository CRAN

Date/Publication 2026-06-15 12:30:12 UTC

NeedsCompilation no

Contents

GJRM-package *Generalised Joint Regression Modelling*

Description

This package provides a function for fitting various generalised joint regression models with several types of covariate effects and distributions. Many modelling options are supported and all parameters of the joint distribution can be specified as flexible functions of covariates.

The orginal name of this package was SemiParBIVProbit which was designed to fit flexible bivariate binary response models. However, since then the package has expanded so much that its orginal name no longer gave a clue about all modelling options available. The new name should more closely reflect past, current and future developments.

The main fitting functions are listed below.

gjrm() which fits bivariate regression models with binary responses (useful for fitting bivariate binary models in the presence of (i) non-random sample selection or (ii) associated responses/endogeneity or (iii) partial observability), bivariate models with binary/discrete/continuous/survival margins in the presence of associated responses/endogeneity, bivariate sample selection models with continuous/discrete response, trivariate binary models (with and without double sample selection). This function essentially merges all previously available fitting functions, namely SemiParBIV(), SemiParTRIV(), copulaReg() and copulaSampleSel().

gamlss() fits flexible univariate regression models where the response can be binary (only the extreme value distribution is allowed for), continuous, discrete and survival. The purpose of this function was only to provide, in some cases, starting values for the above functions, but it has now been made available in the form of a proper function should the user wish to fit univariate models using the general estimation approach of this package.

We are currently working on several multivariate extensions.

Details

GJRM provides functions for fitting general joint models in various situations. The estimation approach is based on a very generic penalized maximum likelihood based framework, where any (parametric) distribution can in principle be employed, and the smoothers (representing several types of covariate effects) are set up using penalised regression splines. Several marginal and copula distributions are available and the numerical routine carries out function minimization using a trust region algorithm in combination with an adaptation of an automatic multiple smoothing parameter estimation procedure for GAMs (see mgcv for more details on this last point). The smoothers supported by this package are those available in mgcv.

Confidence intervals for smooth components and nonlinear functions of the model parameters are derived using a Bayesian approach. P-values for testing individual smooth terms for equality to the zero function are also provided and based on the approach implemented in mgcv. The usual plotting and summary functions are also available. Model/variable selection is also possible via the use of shrinakge smoothers and/or information criteria.

Author(s)

Giampiero Marra (University College London, Department of Statistical Science) and Rosalba Radice (Bayes Business School, Faculty of Actuarial Science and Insurance, City, University of London)

with help and contributions from Panagiota Filippou (trivariate binary models), Francesco Donat (bivariate models with ordinal and continuous margins, and ordinal margins), Matteo Fasiolo (pdf and cdf, and related derivatives, of the Tweedie distribution), Alessia Eletti and Javier Rubio Alvarez (univariate survival models with mixed censoring and excess hazards), Kiron Das (Galambos copula), Eva Cantoni and William Aeberhard (robust gamlss).

Thanks to Bear Braumoeller for suggesting the implementation of bivariate models with partial observability, and Carmen Cadarso for suggesting the inclusion of various modelling extensions.

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Part funded by EPSRC: EP/J006742/1 and EP/T033061/1

References

Key methodological references (ordered by year of publication):

Marra G, Radice R (in press), Joint Modeling of In-Hospital Mortality and Length of Stay: A Copula Additive Distributional Regression Analysis of COVID-19 Patient Data. *Journal of the Royal Statistical Society Series A*.

Marra G, Radice R (2026), Modeling Physician Visit Frequency and Costs Using a Copula Additive Distributional Regression Approach. *Journal of the Royal Statistical Society Series C*, 75(2), 431-447.

Marra G., Radice R., Zimmer D. (2024), A Unifying Switching Regime Regression Framework with Applications in Health Economics. *Econometric Reviews*, 43(1), 52-70.

Eletti A., Marra G., Quaresma M., Radice R., Rubio F.J. (2022), A Unifying Framework for Flexible Excess Hazard Modeling with Applications in Cancer Epidemiology. *Journal of the Royal Statistical Society Series C*, 71(4), 1044-1062.

Petti D., Eletti A., Marra G., Radice R. (2022), Copula Link-Based Additive Models for Bivariate Time-to-Event Outcomes with General Censoring Scheme. *Computational Statistics and Data Analysis*, 107550.

Ranjbar S., Cantoni E., Chavez-Demoulin V., Marra G., Radice R., Jaton-Ogay K. (2022), Modelling the Extremes of Seasonal Viruses and Hospital Congestion: The Example of Flu in a Swiss Hospital. *Journal of the Royal Statistical Society Series C*, 71(4), 884-905.

Aeberhard W.H., Cantoni E., Marra G., Radice R. (2021), Robust Fitting for Generalized Additive Models for Location, Scale and Shape. *Statistics and Computing*, 31(11), 1-16.

Marra G., Farcomeni A., Radice R. (2021), Link-Based Survival Additive Models under Mixed Censoring to Assess Risks of Hospital-Acquired Infections. *Computational Statistics and Data Analysis*, 155, 107092.

Hohberg M., Donat F., Marra G., Kneib T. (2021), Beyond Unidimensional Poverty Analysis Using Distributional Copula Models for Mixed Ordered-Continuous Outcomes. *Journal of the Royal Statistical Society Series C*, 70(5), 1365-1390.

Dettoni R., Marra G., Radice R. (2020), Generalized Link-Based Additive Survival Models with Informative Censoring. *Journal of Computational and Graphical Statistics*, 29(3), 503-512.

Marra G., Radice R. (2020), Copula Link-Based Additive Models for Right-Censored Event Time Data. *Journal of the American Statistical Association*, 115(530), 886-895.

Filippou P., Kneib T., Marra G., Radice R. (2019), A Trivariate Additive Regression Model with Arbitrary Link Functions and Varying Correlation Matrix. *Journal of Statistical Planning and Inference*, 199, 236-248.

Klein N., Kneib T., Marra G., Radice R., Rokicki S., McGovern M.E. (2019), Mixed Binary-Continuous Copula Regression Models with Application to Adverse Birth Outcomes. *Statistics in Medicine*, 38(3), 413-436.

Filippou P., Marra G., Radice R. (2017), Penalized Likelihood Estimation of a Trivariate Additive Probit Model. *Biostatistics*, 18(3), 569-585.

Marra G., Radice R. (2017), Bivariate Copula Additive Models for Location, Scale and Shape. *Computational Statistics and Data Analysis*, 112, 99-113.

Marra G., Radice R., Barnighausen T., Wood S.N., McGovern M.E. (2017), A Simultaneous Equation Approach to Estimating HIV Prevalence with Non-Ignorable Missing Responses. *Journal of the American Statistical Association*, 112(518), 484-496.

Marra G., Radice R., Filippou P. (2017), Testing the Hypothesis of Exogeneity in Regression Spline Bivariate Probit Models. *Communications in Statistics - Simulation and Computation*, 46(3), 2283-2298.

Radice R., Marra G., Wojtys M. (2016), Copula Regression Spline Models for Binary Outcomes. *Statistics and Computing*, 26(5), 981-995.

Marra G., Radice R. (2013), A Penalized Likelihood Estimation Approach to Semiparametric Sample Selection Binary Response Modeling. *Electronic Journal of Statistics*, 7, 1432-1455.

Marra G., Radice R. (2013), Estimation of a Regression Spline Sample Selection Model. *Computational Statistics and Data Analysis*, 61, 158-173.

Marra G., Radice R. (2011), Estimation of a Semiparametric Recursive Bivariate Probit in the Presence of Endogeneity. *Canadian Journal of Statistics*, 39(2), 259-279.

For applied case studies see https://www.homepages.ucl.ac.uk/~ucakgm0/pubs.htm.

See Also

gjrm, gamlss

`adjCov` *Adjustment for the covariance matrix from a fitted gjrm model*

Description

`adjCov` can be used to adjust the covariance matrix of a fitted `gjrm` object.

Usage

```
adjCov(x, id)
```

Arguments

x	A fitted gjrm object as produced by the respective fitting function.
id	Cluster identifier.

Details

This adjustment can be made when dealing with clustered data and the cluster structure is neglected when fitting the model. The basic idea is that the model is fitted as though observations were independent, and subsequently adjust the covariance matrix of the parameter estimates. Using the terminology of Liang and Zeger (1986), this would correspond to using an independence structure within the context of generalized estimating equations. The parameter estimators are still consistent but are inefficient as compared to a model which accounts for the correct cluster dependence structure. The covariance matrix of the independence estimators can be adjusted as described in Liang and Zeger (1986, Section 2).

Value

This function returns a fitted object which is identical to that supplied in adjCov but with adjusted covariance matrix.

WARNINGS

This correction may not be appropriate for models fitted using penalties.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Liang K.-Y. and Zeger S. (1986), Longitudinal Data Analysis Using Generalized Linear Models. *Biometrika*, 73(1), 13-22.

See Also

GJRM-package, gjrm

---

adjCovSD — *Adjustment for the covariance matrix from a gjrm model fitted to complex survey data.*

Description

adjCovSD can be used to adjust the covariance matrix of a fitted gjrm object.

Usage

```
adjCovSD(x, design)
```

Arguments

x	A fitted gjrm object as produced by the respective fitting function.
design	A svydesign object as produced by svydesign() from the survey package.

Details

This function has been extracted from the survey package and adapted to the class of this package's models. It computes the sandwich variance estimator for a copula model fitted to data from a complex sample survey (Lumley, 2004).

Value

This function returns a fitted object which is identical to that supplied in adjCovSD but with adjusted covariance matrix.

WARNINGS

This correction may not be appropriate for models fitted using penalties.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Lumley T. (2004), Analysis of Complex Survey Samples. *Journal of Statistical Software*, 9(8), 1-19.

See Also

GJRM-package, gjrm

ATE *Average Treatment Effect of a binary or continuous treatment variable*

Description

ATE can be used to calculate the causal average treatment effect of a binary or continuous Gaussian treatment variable, with corresponding interval obtained using posterior simulation.

Usage

```
ATE(x, trt, trt.val = NULL, int.var = NULL, eq = NULL, joint = TRUE, n.sim = 100,
    prob.lev = 0.05, length.out = NULL, percentage = FALSE)
```

Arguments

x	A fitted gjrm object as produced by the respective fitting function.
trt	Name of the treatment variable.
trt.val	Numeric value for the treatment variable. This is only required when the endogenous variable is Gaussian.
int.var	A vector made up of the name of the variable interacted with trt, and a value for it. It can also be a list.
eq	Number of equation containing the treatment variable. This is only used for trivariate models.
joint	If FALSE then the effect is obtained from the univariate model which neglects the presence of unobserved confounders. When TRUE, the effect is obtained from the simultaneous model which accounts for observed and unobserved confounders.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. It may be increased if more precision is required.
prob.lev	Overall probability of the left and right tails of the AT distribution used for interval calculations.
length.out	Length of the sequence to be used when calculating the effect that a continuous treatment has on a binary outcome.
percentage	Only for the Roy model, when TRUE it provides results in terms of percentage.

Details

ATE measures the causal average difference in outcomes under treatment (the binary predictor or treatment assumes value 1) and under control (the binary treatment assumes value 0). Posterior simulation is used to obtain a confidence/credible interval. See the references below for details.

ATE can also calculate the effect that a continuous Gaussian endogenous variable has on a binary outcome. In this case the effect will depend on the unit increment chosen (as shown by the plot produced).

Value

res	It returns three values: lower confidence interval limit, estimated AT and upper interval limit.
prob.lev	Probability level used.
sim.ATE	It returns a vector containing simulated values of the average treatment effect. This is used to calculate intervals.
Effects	For the case of continuous/discrete endogenous variable and binary outcome, it returns a matrix made up of three columns containing the effects for each incremental value in the endogenous variable and respective intervals.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Marra G. and Radice R. (2011), Estimation of a Semiparametric Recursive Bivariate Probit in the Presence of Endogeneity. *Canadian Journal of Statistics*, 39(2), 259-279.

See Also

GJRM-package, gjrm

BCDF *Internal Function*

Description

It evaluates the cdf of several copulae.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

bcont *Internal Function*

Description

This and other similar internal functions provide the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when copula models with continuous margins are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

bdiscrcont *Internal Function*

Description

This and other similar internal functions provide the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when copula models with discrete and continuous margins are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bdiscrdiscr *Internal Function*

Description

This and other similar internal functions provide the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when copula models with discrete margins are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHs *Internal Function*

Description

It provides the log-likelihood, gradient and observed/Fisher information matrix for penalized/unpenalized maximum likelihood optimization when copula models with binary outcomes are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHsCont *Internal Function*

Description

It provides the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when copula models with binary and continuous margins are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHsContSS *Internal Function*

Description

It provides the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when copula sample selection models with continuous margins are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHsContUniv *Internal Function*

Description

It provides the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when fitting univariate models with discrete/continuous response.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHsDiscr1 *Internal Function*

Description

It provides the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when copula models with binary and discrete margins are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHsDiscr1SS *Internal Function*

Description

It provides the log-likelihood, gradient and observed information matrix for penalized/unpenalized maximum likelihood optimization when copula sample selection models with discrete margins are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHsPO *Internal Function*

Description

It provides the log-likelihood, gradient and observed or expected information matrix for penalized/unpenalized maximum likelihood optimization when bivariate probit models with partial observability are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

bprobgHsSS *Internal Function*

Description

It provides the log-likelihood, gradient and observed/Fisher information matrix for penalized/unpenalized maximum likelihood optimization when copula sample selection models with binary outcomes are employed.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

clarke.test *Clarke test*

Description

The Clarke test is a likelihood-ratio-based test that can be used for choosing between two non-nested models.

Usage

```
clarke.test(obj1, obj2, sig.lev = 0.05)
```

Arguments

obj1, obj2	Objects of the two fitted bivariate non-nested models.
sig.lev	Significance level used for testing.

Details

The Clarke (2007) test is a likelihood-ratio-based tests for model selection that use the Kullback-Leibler information criterion, and that can be employed for choosing between two bivariate models which are non-nested.

If the two models are statistically equivalent then the log-likelihood ratios of the observations should be evenly distributed around zero and around half of the ratios should be larger than zero. The test follows asymptotically a binomial distribution with parameters n and 0.5. Critical values can be obtained as shown in Clarke (2007). Intuitively, model obj1 is preferred over obj2 if the value of the test is significantly larger than its expected value under the null hypothesis ($n/2$), and vice versa. If the value is not significantly different from $n/2$ then obj1 can be thought of as equivalent to obj2.

Value

It returns a decision.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Clarke K. (2007), A Simple Distribution-Free Test for Non-Nested Model Selection. *Political Analysis*, 15, 347-363.

Examples

```
## see examples for gjrm
```

cond.mv *Conditional Mean/Variance from a Copula Model*

Description

Function cond.mv can be used to calculate conditional means/variances from a copula model, with corresponding interval obtained using posterior simulation.

Usage

```
cond.mv(x, eq, y1 = NULL, y2 = NULL, newdata, fun = "mean", n.sim = 100,
        prob.lev = 0.05, method = "adaptive")
```

Arguments

x	A fitted cond.mv object as produced by the respective fitting function.
eq	Equation of interest. From this, conditioning is also deduced.
y1, y2	Values for y1 and y2. Depending on the fitted model, one of them may be required.
newdata	A data frame with one row, which must be provided.
fun	Either mean or variance.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters.
prob.lev	Overall probability of the left and right tails of the simulated distribution used for interval calculations.
method	Character string specifying the numerical integration method to use. Possible values are: "adaptive" (default) and "quad". "adaptive" performs integration using an adaptive quadrature algorithm with automatic fallback to a Gauss-Legendre quadrature rule if numerical issues occur. When the integration bounds are infinite, they are approximated using quantile-based truncation of the distribution. "quad" (which is slower and tends to be less reliable) applies a transformation-based quadrature method suitable for improper integrals on infinite intervals. It is specifically designed to handle integrals with infinite or semi-infinite bounds and assumes that the integrand decreases sufficiently fast at infinity. No fallback or truncation strategy is used in this case. "quad" is often used as sensitivity analysis tool.

Details

cond.mv() calculates the conditional mean or variance of copula models. Posterior simulation is used to obtain a confidence/credible interval.

Value

res	It returns three values: lower confidence interval limit, estimated conditional mean or variance and upper interval limit.
prob.lev	Probability level used.
sim.mv	It returns a vector containing simulated values of the conditional mean or variance. This is used to calculate intervals.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm, cond.mv.pcc

---

cond.mv.pcc *Conditional Mean/Variance from a PCC-based Trivariate Copula Model*

Description

Function cond.mv.pcc calculates conditional means or variances from a trivariate copula model built via pair-copula constructions (PCC). Credible/confidence intervals are obtained using posterior simulation.

Usage

```
cond.mv.pcc(x, eq, y1 = NULL, y2 = NULL, y3 = NULL, newdata, fun = "mean",
            n.sim = 100, prob.lev = 0.05, method = "adaptive")
```

Arguments

x	A fitted cond.mv.pcc object, as produced by a trivariate PCC copula model fitting function.
eq	Equation of interest. The function infers the conditional structure based on this input.
y1	Value of the first response variable. Required if the equation or conditioning involves it.
y2	Value of the second response variable. Required if the equation or conditioning involves it.
y3	Value of the third response variable. Required if the equation or conditioning involves it.
newdata	A data frame with one row providing the covariate values for prediction. This must be supplied.
fun	Specifies the functional form to compute: either "mean" (default) or "variance".

n.sim	Number of posterior draws from the distribution of the model parameters used for simulating the conditional mean or variance.
prob.lev	The tail probability used for interval estimation. For example, 0.05 corresponds to a 95% credible interval.
method	The numerical integration method to be used. Options are: "adaptive" (default), which performs adaptive quadrature with fallback to Gauss-Legendre integration if necessary, or "quad" (which is slower and tends to be less reliable), a transformation-based quadrature approach designed for integrals with infinite or semi-infinite limits. This method does not implement truncation or fallback. "quad" is often used as sensitivity analysis tool.

Details

cond.mv.pcc extends cond.mv to trivariate copula models using pair-copula constructions (PCC). It computes conditional means or variances by integrating over relevant conditional distributions. Uncertainty intervals are estimated using simulated parameter values from the posterior distribution.

Value

res	A numeric vector containing: the lower bound of the interval, the estimated conditional mean or variance, and the upper bound.
prob.lev	The probability level used for constructing the interval.
sim.mv	A numeric vector of simulated conditional means or variances across posterior draws, used to derive the interval.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm, cond.mv

---

conv.check *Some convergence diagnostics*

Description

It takes a fitted model object and produces some diagnostic information about the fitting procedure.

Usage

```
conv.check(x, blather = FALSE)
```

Arguments

x	gjrm object.
blather	If TRUE then more diagnostic information is provided.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gamlss, gjrm

---

copgHs *Internal Function*

Description

This and other similar internal functions evaluate the first and second derivatives with respect to the margins and association parameter of several copulae.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

copula.prob *Copula probabilities (joint and conditional) from a fitted simultaneous model*

Description

copula.prob can be used to calculate the joint or conditional copula probabilities from a fitted simultaneous model with intervals obtained via posterior simulation.

Usage

```
copula.prob(x, y1, y2, y3 = NULL, newdata, joint = TRUE, cond = 0,
            intervals = FALSE, n.sim = 100, prob.lev = 0.05,
            theta = FALSE, tau = FALSE, min.pr = 1e-323, max.pr = 1,
            method = "adaptive", cumul = "no", tail = c("lower","lower"))
```

Arguments

x	A fitted gjrm object as produced by the respective fitting function.
y1	Value of response for first margin.
y2	Value of response for second margin.
y3	Value of response for third margin if a trivariate model is employed.
newdata	A data frame with one row, which must be provided.
joint	If TRUE then the calculation is done using the fitted joint model. If FALSE then the calculation is done from univariate fits.
cond	There are three possible values: 0 (joint probabilities are delivered), 1 (conditional probabilities are delivered and conditioning is with the respect to the first margin), 2 (as before but conditioning is with the respect to the second margin).
intervals	If TRUE then intervals for the probabilities are also produced.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used for interval calculations.
prob.lev	Overall probability of the left and right tails of the probabilities' distributions used for interval calculations.
theta	If TRUE the theta dependence parameter will be shown. This is especially useful for prediction purposes when theta is specified as a function of covariate effects.
tau	If TRUE the Kendall's tau will also be calculated and provided in output. Note that the calculation adopted here assumes continuous margins. In all other cases, this may provide a rough indication of dependence under certain assumptions. Note that, for the F, PL and J0 (and the related rotations), computing times may be longer than for the other cases. This is especially useful for prediction purposes when theta is specified as a function of covariate effects, with an interest in analysing a more interpretable measure of dependence for certain copulae.
min.pr, max.pr	Allowed minimum and maximum for estimated probabities.
method	For PCC model only, this indicates the numerical integration method to be used. Options are: "adaptive" (default), which performs adaptive quadrature with fallback to Gauss-Legendre integration if necessary, or "quad" (which is slower and tends to be less reliable), a transformation-based quadrature approach designed for integrals with infinite or semi-infinite limits. This method does not implement truncation or fallback. "quad" is often used as sensitivity analysis tool.
cumul	Currently only for the count-continuous margin case, if set to a character different from "no" then the joint probability will refer to the case in which both outcomes are smaller or equal to the specified values.
tail	A character vector of length two indicating which tail of each margin is used when computing the joint probability. Possible values are "lower" and "upper". The default is c("lower","lower"), which corresponds to the probability $P(Y_1 \leq y_1, Y_2 \leq y_2)$. The option c("upper","upper") returns the joint upper-tail probability $P(Y_1 > y_1, Y_2 > y_2)$. At present, this option is implemented only for copula models with continuous margins. Also, mixed tail combinations such as c("lower","upper") or c("upper","lower") are not currently supported. Users interested in these or other extensions are welcome to contact the package authors.

Details

This function calculates joint or conditional copula probabilities from a fitted simultaneous model or a model assuming independence, with intervals obtained via posterior simulation.

Value

`res`	It returns several values including: estimated probabilities (`p12`), with lower and upper interval limits (`CIpr`) if `intervals = TRUE`, and `p1`, `p2` and `p3` (the marginal probabilities).

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm

CopulaCLM *Internal fitting function*

Description

Internal fitting and set up function.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

copulaSampleSel *Internal fitting function*

Description

Internal fitting and set up function.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

cv.inform *Cross validation for informative censoring univariate survival models*

Description

cv.inform carries out cross validation to help choosing the set of informative covariates.

Usage

```
cv.inform(x, K = 5, data, informative = "yes")
```

Arguments

x	A fitted gamlss object as produced by the respective fitting function.
K	No. of folds.
data	Data.
informative	If no then cv is carried out for the case of no informative censoring. This is useful for comparison purposes.

Details

cv.inform carries out cross validation to help choosing the set of informative covariates.

Value

sl	Overall sum of predicted likelihood contributions.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gamlss

distrHs *Internal Function*

Description

This and other similar internal functions evaluate the margins' derivatives needed in the likelihood function for the binary, discrete and continuous cases.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Dpens *Differentiable penalties*

Description

work in progress, temp function

Usage

```
Dpens(params, type = "lasso", lambda = 1, w.alasso = NULL,
      gamma = 1, a = 3.7, eps = 1e-08)
```

Arguments

params	coefficients.
type	lasso, alasso or scad.
lambda	smoothing parameter.
w.alasso	for alasso.
gamma	default 1.
a	for scad.
eps	tolerance.

Details

work in progress.

Value

The function returns a penalty.

eta.tr *Internal Function*

Description

This and other similar internal functions map certain key quantities into a feasible parameter space. Some functions carry out some general consistency checks.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

g.tri *Internal Function*

Description

This and other similar internal functions calculate the score for trivariate binary models.

Author(s)

Author: Panagiota Filippou

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

gamlss *Generalised Additive Models for Location, Scale and Shape and Beyond*

Description

gamlss fits flexible univariate regression models for several continuous and discrete distributions as well as survival outcomes, and types of covariate effects. When first designed, the purpose of this function was only to provide, in some cases, starting values for the simultaneous models in the package. At a later stage, it was made available in the form of a proper function should the user wish to fit univariate models using the general estimation approach of this package. The continuous and discrete distributions used here are parametrised according to Rigby and Stasinopoulos (2005).

Usage

```
gamlss(formula, data = list(), weights = NULL, subset = NULL, offset = NULL,
       family = "N", cens = NULL, type.cens = "R", ub.t = NULL, left.trunc = 0,
       robust = FALSE, rc = 3, lB = NULL, uB = NULL, infl.fac = 1,
       rinit = 1, rmax = 100, iterlimsp = 50, tolsp = 1e-07,
       gc.l = FALSE, parscale, gev.par = -0.25,
       chunk.size = 10000, knots = NULL,
       informative = "no", inform.cov = NULL, family2 = "-cloglog",
       fp = FALSE, sp = NULL,
       drop.unused.levels = TRUE, siginit = NULL, shinit = NULL,
       sp.method = "perf", hrate = NULL, d.lchrate = NULL, d.rchrate = NULL,
       d.lchrate.td = NULL, d.rchrate.td = NULL, truncation.time = NULL,
       min.dn = 1e-40, min.pr = 1e-16, max.pr = 0.9999999, ygrid.tol = 1e-08)
```

Arguments

formula	List of equations. This should contain one or more equations.
data	A data frame.
weights	Optional vector of prior weights to be used in fitting.
subset	Optional vector specifying a subset of observations to be used in the fitting process.
offset	Optional vector specifying an offset for use in fitting. Option introduced for dealing with offset with discrete distributions.
family	Possible choices are Gaussian ("N"), truncated Gaussian ("tN"), Tweedie ("TW"), log-normal ("LN"), Gumbel ("GU"), reverse Gumbel ("rGU"), generelised Pareto ("GP"), generelised Pareto II ("GPII") where the shape parameter is forced to be > -0.5, generelised Pareto (with orthogonal parametrisation) ("GPo") where the shape parameter is forced to be > -0.5, discrete generelised Pareto ("DGP"), discrete generelised Pareto II ("DGPII") where the shape parameter is forced to be positive, discrete generelised Pareto derived under the scenario in which shape = 0 ("DGP0"), logistic ("LO"), Weibull ("WEI"), Inverse Gaussian ("IG"), gamma ("GA"), Dagum ("DAGUM"), Singh-Maddala ("SM"), beta ("BE"), Fisk ("FISK", also known as log-logistic), Poisson ("P"), truncated Poisson ("tP"), negative binomial - type I ("NBI"), negative binomial - type II ("NBII"), Poisson inverse Gaussian ("PIG"), truncated negative binomial - type I ("tNBI"), truncated negative binomial - type II ("tNBII"), truncated Poisson inverse Gaussian ("tPIG"), generalised extreme value link function ("GEVlink", this is used for binary responses and is more stable and faster than the R package bgeva). For survival models, family can be "-cloglog" (similar to generalised proportional hazards), "-logit" (similar to generalised proportional odds), "-probit" (generalised probit). Another avaialble distribution is the Beta Binomial ("BB"). Here, note that the response variable should be a matrix containing two columns, the first with the count of successes and the second with the count of failures.
cens	This is required for a survival model. When type.cens is different from mixed, this variable can be equal to 1 if the event occurred and 0 otherwise. If type.cens = "mixed" then cens is a mixed factor variable (made up of four possible categories: I for interval, L for left, R for right, and U for uncensored.
type.cens	Type of censoring mechanism. This can be "R", "L", "I" or "mixed".
ub.t	Variable name of right/upper bound when type.cens = "I" or type.cens = "mixed" and interval censoring is present.
left.trunc	Value of truncation at left. Currently done for count distributions only.
robust	If TRUE then the robust version of the model is fitted.
rc	Robust constant.
lB, uB	Bounds for integral in robust case.
infl.fac	Inflation factor for the model degrees of freedom in the approximate AIC. Smoother models can be obtained setting this parameter to a value greater than 1.
rinit	Starting trust region radius. The trust region radius is adjusted as the algorithm proceeds.

`rmax` Maximum allowed trust region radius. This may be set very large. If set small, the algorithm traces a steepest descent path.

`iterlimsp` A positive integer specifying the maximum number of loops to be performed before the smoothing parameter estimation step is terminated.

`tolsp` Tolerance to use in judging convergence of the algorithm when automatic smoothing parameter estimation is used.

`gc.l` This is relevant when working with big datasets. If `TRUE` then the garbage collector is called more often than it is usually done. This keeps the memory footprint down but it will slow down the routine.

`parscale` The algorithm will operate as if optimizing objfun(x / parscale, ...) where parscale is a scalar. If missing then no rescaling is done. See the documentation of `trust` for more details.

`gev.par` GEV link parameter.

`chunk.size` This is used for discrete robust models.

`knots` Optional list containing user specified knot values to be used for basis construction.

`informative` If "yes" then informative censoring is assumed when using a survival model.

`inform.cov` If above is "yes" then a set of informative covariates must be provided.

`family2` In the informative survival case, the family for the censored equation can be different from that of the survival equation. Choices are "-cloglog" (siilar to generalised proportional hazards), "-logit" (similar to generalised proportional odds), "-probit" (generalised probit).

`fp` If `TRUE` then a fully parametric model with unpenalised regression splines if fitted.

`sp` A vector of smoothing parameters can be provided here. Smoothing parameters must be supplied in the order that the smooth terms appear in the model equation(s).

`drop.unused.levels`
By default unused levels are dropped from factors before fitting. For some smooths involving factor variables this may have to be turned off (only use if you know what you are doing).

`siginit`, `shinit` For the GP and DGP distributions, initial values for sigma and shape may be provided.

`sp.method` Multiple smoothing automatic parameter selection is perf. efs is an alternative and only sensible option for robust models.

`hrate` Vector of population hazard rates computed at time of death of each uncensored patient. The length of `hrate` should be equal to the number of uncensored observations in the dataset. Needed in the context of excess hazard modelling when uncensored observations are present. Note that this includes left truncated uncensored observations as well.

`d.lchrate` Vector of differences of population cumulative excess hazards computed at the age of the patient when the left censoring occurred and at the initial age of the patient. The length of `d.lchrate` should be equal to the number of left

and/or interval censored observations in the dataset. Needed in the context of excess hazard modelling if left censored and/or interval censored observations are present. In the latter case, `d.rchrate` also need be provided.

`d.rchrate` Vector of differences of population cumulative excess hazards computed at the age of the patient when the at the right interval censoring time and at the initial age of the patient. The length of `d.rchrate` should be equal to the number of right censored and/or interval censored observations in the dataset. Needed in the context of excess hazard modelling if right censored and/or interval censored observations are present. In the latter case, `d.lchrate` also need be provided.

`d.lchrate.td` Vector of differences of population cumulative excess hazards computed at the age of the patient when the left censoring occurred and at the age of the patient when the truncation occurred. The length of `d.lchrate.td` should be equal to the number of left truncated left censored and/or left truncated interval censored observations in the dataset. Needed in the context of excess hazard modelling if left truncated left censored and/or left truncated interval censored observations are present. In the latter case, `d.rchrate.td` also need be provided.

`d.rchrate.td` Vector of differences of population cumulative excess hazards computed at the age of the patient when the right censoring occurred and at the age of the patient when the truncation occurred. The length of `d.rchrate.td` should be equal to the number of left truncated right censored and/or left truncated interval censored observations in the dataset. Needed in the context of excess hazard modelling if left truncated right censored and/or left truncated interval censored observations are present. In the latter case, `d.lchrate.td` also need be provided.

`truncation.time`
Variable name of truncation time.

`min.dn`, `min.pr`, `max.pr`
These values are used to set, depending on the model used for modelling, the minimum and maximum allowed for the densities and probabilities. These parameters are employed to avoid potential overflows/underflows in the calculations and the default values seem to offer a good compromise. Function `conv.check()` provides some relevant diagnostic information which can be used, for example, to check whether the lower bounds of `min.dn` and `min.pr` have been reached. So based on this or if the user wishes to do some sensitivity analysis then this can be easily carried out using these three arguments. However, the user has to be cautious. For instance, it would not make much sense to choose for `min.dn` and `min.pr` values bigger than the default ones. Bear in mind that the bounds can be reached for ill-defined models. For certain distributions/models, if convergence failure occurs and the bounds have been reached then the user can try a sensitivity analysis as mentioned above.

`ygrid.tol` Tolerance used to choose grid of response values for robust discrete models. Values smaller than 1e-160 are not allowed for.

Details

The underlying algorithm is described in ?gjrm.

There are many continuous/discrete distributions to choose from and we plan to include more options. Get in touch if you are interested in a particular distribution.

The "GEVlink" option is used for binary response additive models and is more stable and faster than the R package bgeva. This model has been incorporated into this package to take advantage of the richer set of smoother choices, and of the estimation approach. Details on the model can be found in Calabrese, Marra and Osmetti (2016).

Value

The function returns an object of class gamlss as described in gamlssObject.

WARNINGS

Convergence can be checked using conv.check which provides some information about the score and information matrix associated with the fitted model. The former should be close to 0 and the latter positive definite. gamlss() will produce some warnings if there is a convergence issue.

Convergence failure may sometimes occur. This is not necessarily a bad thing as it may indicate specific problems with a fitted model. In such a situation, the user may use rescaling (see parscale). However, the user should especially consider re-specifying/simplifying the model, and/or checking that the chosen distribution fits the response well. In our experience, we found that convergence failure typically occurs when the model has been misspecified and/or the sample size is low compared to the complexity of the model. It is also worth bearing in mind that the use of three parameter distributions requires the data to be more informative than a situation in which two parameter distributions are used instead.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Aeberhard W.H., Cantoni E., Marra G., Radice R. (2021), Robust Fitting for Generalized Additive Models for Location, Scale and Shape. *Statistics and Computing*, 31(11), 1-16.

Eletti A., Marra G., Quaresma M., Radice R., Rubio F.J. (2022), A Unifying Framework for Flexible Excess Hazard Modeling with Applications in Cancer Epidemiology. *Journal of the Royal Statistical Society Series C*, 71(4), 1044-1062.

Marra G., Farcomeni A., Radice R. (2021), Link-Based Survival Additive Models under Mixed Censoring to Assess Risks of Hospital-Acquired Infections. *Computational Statistics and Data Analysis*, 155, 107092.

Marra G., Radice R. (2017), Bivariate Copula Additive Models for Location, Scale and Shape. *Computational Statistics and Data Analysis*, 112, 99-113.

Ranjbar S., Cantoni E., Chavez-Demoulin V., Marra G., Radice R., Jaton-Ogay K. (2022), Modelling the Extremes of Seasonal Viruses and Hospital Congestion: The Example of Flu in a Swiss Hospital. *Journal of the Royal Statistical Society Series C*, 71(4), 884-905.

Calabrese R., Marra G., Osmetti SA (2016), Bankruptcy Prediction of Small and Medium Enterprises Using a Flexible Binary Generalized Extreme Value Model. *Journal of the Operational Research Society*, 67(4), 604-615.

Marincioni V., Marra G., Altamirano-Medina H. (2018), Development of Predictive Models for the Probabilistic Moisture Risk Assessment of Internal Wall Insulation. *Building and Environment*, 137, 5257-267.

See Also

GJRM-package, gamlssObject, conv.check, summary.gamlss

Examples

```
## Not run:

library(GJRM)

set.seed(0)

n <- 400

x1 <- round(runif(n))
x2 <- runif(n)
x3 <- runif(n)
f1 <- function(x) cos(pi*2*x) + sin(pi*x)
y1 <- -1.55 + 2*x1 + f1(x2) + rnorm(n)

dataSim <- data.frame(y1, x1, x2, x3)
resp.check(y1, "N")

eq.mu <- y1 ~ x1 + s(x2) + s(x3)
eq.s  <-    ~ s(x3)
fl    <- list(eq.mu, eq.s)

out <- gamlss(fl, data = dataSim)

conv.check(out)
res.check(out, intervals = TRUE)

plot(out, eq = 1, scale = 0, pages = 1, seWithMean = TRUE)
plot(out, eq = 2, seWithMean = TRUE)

summary(out)

AIC(out)
BIC(out)


################
# Robust example
################

eq.mu <- y1 ~ x1 + x2 + x3
fl    <- list(eq.mu)

out <- gamlss(fl, data = dataSim, family = "N", robust = TRUE,
                  rc = 3, lB = -Inf, uB = Inf)

conv.check(out)
summary(out)
```

```
rob.const(out, 100)

##

eq.s  <-    ~ x3
fl    <- list(eq.mu, eq.s)

out <- gamlss(fl, data = dataSim, family = "N", robust = TRUE)

conv.check(out)
summary(out)

##

eq.mu <- y1 ~ x1 + s(x2) + s(x3)
eq.s  <-    ~ s(x3)
fl    <- list(eq.mu, eq.s)

out1 <- gamlss(fl, data = dataSim, family = "N", robust = TRUE,
               sp.method = "efs")

conv.check(out1)
summary(out1)
AIC(out, out1)

plot(out1, eq = 1, all.terms = TRUE, pages = 1, seWithMean = TRUE)
plot(out1, eq = 2, seWithMean = TRUE)


##########################
## GEV link binary example
##########################
# this incorporates the bgeva
# model implemented in the bgeva package
# however this implementation is more general,
# stable and efficient

set.seed(0)

n <- 400

x1 <- round(runif(n)); x2 <- runif(n); x3 <- runif(n)

f1 <- function(x) cos(pi*2*x) + sin(pi*x)
f2 <- function(x) x+exp(-30*(x-0.5)^2)

y  <- ifelse(-3.55 + 2*x1 + f1(x2) + rnorm(n) > 0, 1, 0)

dataSim <- data.frame(y, x1, x2, x3)

out1 <- gamlss(list(y ~ x1 + x2 + x3), family = "GEVlink", data = dataSim)
out2 <- gamlss(list(y ~ x1 + s(x2) + s(x3)), family = "GEVlink", data = dataSim)
```

```
conv.check(out1)
conv.check(out2)
summary(out1)
summary(out2)
AIC(out1, out2)
BIC(out1, out2)

plot(out2, eq = 1, all.terms = TRUE, pages = 1, seWithMean = TRUE)

###################
# prediction of Pr
###################

# Calculate eta (that is, X*model.coef)
# For a new data set the argument newdata should be used

eta <- predict(out2, eq = 1, type = "link")

# extract gev tail parameter

gev.par <- out2$gev.par

# multiply gev tail parameter by eta

gevpeta <- gev.par*eta

# establish for which values the model is defined

gevpetaIND <- ifelse(gevpeta < -1, FALSE, TRUE)
gevpeta <- gevpeta[gevpetaIND]

# estimate probabilities

pr <- exp(-(1 + gevpeta)^(-1/gev.par))


###################################
## Flexible survival model examples
###################################

## Simulate proportional hazards data ##

set.seed(0)
n  <- 2000
c  <- runif(n, 3, 8)
u  <- runif(n, 0, 1)
z1 <- rbinom(n, 1, 0.5)
z2 <- runif(n, 0, 1)
t  <- rep(NA, n)

beta_0 <- -0.2357
beta_1 <- 1
```

```
f <- function(t, beta_0, beta_1, u, z1, z2){
  S_0 <- 0.7 * exp(-0.03*t^1.9) + 0.3*exp(-0.3*t^2.5)
  exp(-exp(log(-log(S_0))+beta_0*z1 + beta_1*z2))-u
}


for (i in 1:n){
   t[i] <- uniroot(f, c(0, 8), tol = .Machine$double.eps^0.5,
                   beta_0 = beta_0, beta_1 = beta_1, u = u[i],
                   z1 = z1[i], z2 = z2[i], extendInt = "yes" )$root
}

delta   <- ifelse(t < c, 1, 0)
u       <- apply(cbind(t, c), 1, min)
dataSim <- data.frame(u, delta, z1, z2)
1-mean(delta) # average censoring rate

# log(u) helps obtaining smoother hazards

out <- gamlss(list(u ~ s(log(u), bs = "mpi") + z1 + s(z2) ), data = dataSim,
              family = "-cloglog", cens = delta)
res.check(out, intervals = TRUE)
summary(out)
AIC(out)
BIC(out)
plot(out, eq = 1, scale = 0, pages = 1)
haz.surv(out, newdata = data.frame(z1 = 0, z2 = 0), shade = TRUE,
        n.sim = 1000, baseline = TRUE)
haz.surv(out, type = "haz", newdata = data.frame(z1 = 0, z2 = 0),
        shade = TRUE, n.sim = 1000, baseline = TRUE)

# library(mgcv)
# out1 <- mgcv::gam(u ~ z1 + s(z2), family = cox.ph(),
#                   data = dataSim, weights = delta)
# summary(out1)
# estimates of z1 and s(z2) are
# nearly identical between out and out1

####################################
## Simulate proportional odds data ##
####################################

set.seed(0)

n <- 2000
c <- runif(n, 4, 8)
u <- runif(n, 0, 1)
z <- rbinom(n, 1, 0.5)
beta_0 <- -1.05
t <- rep(NA, n)

f <- function(t, beta_0, u, z){
  S_0 <- 0.7 * exp(-0.03*t^1.9) + 0.3*exp(-0.3*t^2.5)
```

```
  1/(1 + exp(log((1-S_0)/S_0)+beta_0*z))-u
}


for (i in 1:n){
    t[i] <- uniroot(f, c(0, 8), tol = .Machine$double.eps^0.5,
                    beta_0 = beta_0, u = u[i], z = z[i],
                    extendInt="yes" )$root
}

delta   <- ifelse(t < c,1, 0)
u       <- apply(cbind(t, c), 1, min)
dataSim <- data.frame(u, delta, z)
1-mean(delta) # average censoring rate

out <- gamlss(list(u ~ s(log(u), bs = "mpi") + z ), data = dataSim,
              family = "-logit", cens = delta)
res.check(out, intervals = TRUE)
summary(out)
AIC(out)
BIC(out)
plot(out, eq = 1, scale = 0)
haz.surv(out, newdata = data.frame(z = 0), shade = TRUE, n.sim = 1000,
        baseline = TRUE)
haz.surv(out, type = "haz", newdata = data.frame(z = 0),
        shade = TRUE, n.sim = 1000)


############################
## Mixed censoring example ##
############################

f1 <- function(t, u, z1, z2, z3, z4, s1, s2){

    S_0 <- 0.7 * exp(-0.03*t^1.8) + 0.3*exp(-0.3*t^2.5)

    exp( -exp(log(-log(S_0)) + 1.3*z1 + 0.5*z2 + s1(z3) + s2(z4)  ) ) - u

  }

datagen <- function(n, z1, z2, z3, z4, s1, s2, f1){

  u <- runif(n, 0, 1)
  t <- rep(NA, n)

  for (i in 1:n) t[i] <- uniroot(f1, c(0, 100), tol = .Machine$double.eps^0.5,
                                 u = u[i], s1 = s1, s2 = s2, z1 = z1[i], z2 = z2[i],
                                 z3 = z3[i], z4 = z4[i], extendInt = "yes")$root

  c1 <-      runif(n, 0, 2)
  c2 <- c1 + runif(n, 0, 6)

  df <- data.frame(u1 = t, u2 = t, cens = character(n), stringsAsFactors = FALSE)
```

```
for (i in 1:n){

  if(t[i] <= c1[i]) {
        df[i, 1] <- c1[i]
        df[i, 2] <- NA
        df[i, 3] <- "L"

  }else if(c1[i] < t[i] && t[i] <= c2[i]){
        df[i, 1] <- c1[i]
        df[i, 2] <- c2[i]
        df[i, 3] <- "I"

  }else if(t[i] > c2[i]){
        df[i, 1] <- c2[i]
        df[i, 2] <- NA
        df[i, 3] <- "R"}

}

uncens <- (df[, 3] %in% c("L", "I")) + (rbinom(n, 1, 0.2) == 1) == 2

df[uncens, 1] <- t[uncens]
df[uncens, 2] <- NA
df[uncens, 3] <- "U"

dataSim <- data.frame(u1 = df$u1, u2 = df$u2, cens = as.factor(df$cens), z1, z2, z3, z4, t)
dataSim

}

set.seed(0)

n      <- 1000
SigmaC <- matrix(0.5, 4, 4); diag(SigmaC) <- 1
cov    <- rMVN(n, rep(0,4), SigmaC)
cov    <- pnorm(cov)
z1     <- round(cov[, 1])
z2     <- round(cov[, 2])
z3     <- cov[, 3]
z4     <- cov[, 4]
s1     <- function(x) -0.075*exp(3.2 * x)
s2     <- function(x) sin(2*pi*x)

eq1    <- u1 ~ s(log(u1), bs = "mpi") + z1 + z2 + s(z3) + s(z4)

dataSim <- datagen(n, z1, z2, z3, z4, s1, s2, f1)

out <- gamlss(list(eq1), data = dataSim, family = "-cloglog",
              cens = cens, type.cen = "mixed", ub.t = "u2")
res.check(out, intervals = TRUE)
conv.check(out)
summary(out)
```

```
plot(out, eq = 1, scale = 0, pages = 1)

ndf <- data.frame(z1 = 1, z2 = 0, z3 = 0.2, z4 = 0.5)

haz.surv(out, eq = 1, newdata = ndf, type = "surv")
haz.surv(out, eq = 1, newdata = ndf, type = "haz", n.sim = 1000)

## End(Not run)
```

`gamlssObject` *Fitted gamlssObject object*

Description

A fitted gamlss object returned by function `gamlss` and of class "gamlss" and "SemiParBIV".

Value

`fit`	List of values and diagnostics extracted from the output of the algorithm.
`gam1`, `gam2`, `gam3`	Univariate starting values' fits.
`coefficients`	The coefficients of the fitted model.
`weights`	Prior weights used during model fitting.
`sp`	Estimated smoothing parameters of the smooth components.
`iter.sp`	Number of iterations performed for the smoothing parameter estimation step.
`iter.if`	Number of iterations performed in the initial step of the algorithm.
`iter.inner`	Number of iterations performed within the smoothing parameter estimation step.
`n`	Sample size.
`X1`, `X2`, `X3`, ...	Design matrices associated with the linear predictors.
`X1.d2`, `X2.d2`, `X3.d2`, ...	Number of columns of `X1`, `X2`, `X3`, etc.
`l.sp1`, `l.sp2`, `l.sp3`, ...	Number of smooth components in the equations.
`He`	Penalized -hessian/Fisher. This is the same as `HeSh` for unpenalized models.
`HeSh`	Unpenalized -hessian/Fisher.
`Vb`	Inverse of `He`. This corresponds to the Bayesian variance-covariance matrix used for confidence/credible interval calculations.
`F`	This is obtained multiplying Vb by HeSh.
`t.edf`	Total degrees of freedom of the estimated bivariate model. It is calculated as `sum(diag(F))`.
`edf1`, `edf2`, `edf3`, ...	Degrees of freedom for the model's equations.

wor.c	Working model quantities.
eta1, eta2, eta3, ...	Estimated linear predictors.
y1	Response.
logLik	Value of the (unpenalized) log-likelihood evaluated at the (penalized or unpenalized) parameter estimates.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gamlss, summary.gamlss

---

ggmtrust *ggmtrust for penalised network*

Description

penalised network, work in progress.

Usage

```
ggmtrust(s, n, data = NULL, lambda = 1, pen = "lasso", params = NULL, method = "BHHH",
         w.alasso = NULL, gamma = 1, a = 3.7)
```

Arguments

s	Sample covariance matrix.
n	Sample size.
data	Data.
lambda	Regularisation parameter.
pen	Either "lasso" or "ridge".
params	If different from null then these are taken as the starting values.
method	Either "H" or "BHHH".
w.alasso	weight for alasso.
gamma	alasso param.
a	scad param.

Details

penalised network, work in progress.

Value

The function returns an object of class ggmtrust.

---

gjrm *Generalised Joint Regression Models with Binary/Continuous/Discrete/Survival Margins*

Description

gjrm fits flexible joint models with binary/continuous/discrete/survival margins, with several types of covariate effects, copula and marginal distributions.

Usage

```
gjrm(formula, data = list(), weights = NULL, subset = NULL,
     offset1 = NULL, offset2 = NULL,
     copula = "N", copula2 = "N", margins, model, dof = 3, dof2 = 3,
     cens1 = NULL, cens2 = NULL, cens3 = NULL, dep.cens = FALSE,
     ub.t1 = NULL, ub.t2 = NULL, left.trunc1 = 0, left.trunc2 = 0,
     uni.fit = FALSE, fp = FALSE, infl.fac = 1,
     rinit = 1, rmax = 100, iterlimsp = 50, tolsp = 1e-07,
     gc.l = FALSE, parscale, knots = NULL,
     penCor = "unpen", sp.penCor = 3,
     Chol = FALSE, gamma = 1, w.alasso = NULL,
     drop.unused.levels = TRUE,
     min.dn = 1e-40, min.pr = 1e-16, max.pr = 0.999999,
     fixed.margins = FALSE, fixed.margins.type = c("continuous", "continuous"),
     pf.margin1 = NULL, pf.margin2 = NULL)
```

Arguments

formula — In the basic setup this will be a list of two (or three) formulas, one for equation 1, the other for equation 2 and another one for equation 3 if a trivariate model is fitted to the data. Otherwise, more equations can be used depending on the number of distributional parameters. s terms are used to specify smooth functions of predictors; see the documentation of mgcv for further details on formula specifications. Note that if a selection model is employed (that is, model = "BSS" or model = "TSS") then the first formula (and the second as well for trivariate models) MUST refer to the selection equation(s). When one outcome is binary and the other continuous/discrete then the first equation should refer to the binary outcome whereas the second to the continuous/discrete one. When one outcome is discrete and the other continuous then the first equation has to be the discrete one.

`data` A data frame.

`weights` Optional vector of prior weights to be used in fitting.

`subset` Optional vector specifying a subset of observations to be used in the fitting process.

`offset1, offset2`
They can be used to supply model offsets for use in fitting. These have been introduced for dealing with offsets in the case of discrete marginal distributions.

`copula` Type of bivariate error distribution employed. Possible choices are "N", "C0", "C90", "C180", "C270", "GAL0", "GAL90", "GAL180", "GAL270", "J0", "J90", "J180", "J270", "G0", "G90", "G180", "G270", "F", "AMH", "FGM", "T", "PL", "HO" which stand for bivariate normal, Clayton, rotated Clayton (90 degrees), survival Clayton, rotated Clayton (270 degrees), Galambos, rotated Galambos (90 degrees), survival Galambos, rotated Galambos (270 degrees), Joe, rotated Joe (90 degrees), survival Joe, rotated Joe (270 degrees), Gumbel, rotated Gumbel (90 degrees), survival Gumbel, rotated Gumbel (270 degrees), Frank, Ali-Mikhail-Haq, Farlie-Gumbel-Morgenstern, Student-t with `dof`, Plackett, Hougaard. Each of the Clayton, Galambos, Joe and Gumbel copulae is allowed to be mixed with a rotated version of the same family. The options are: "C0C90", "C0C270", "C180C90", "C180C270", "GAL0GAL90", "GAL0GAL270", "GAL180GAL90", "GAL180GAL270", "G0G90", "G0G270", "G180G90", "G180G270", "J0J90", "J0J270", "J180J90" and "J180J270". This allows the user to model negative and positive tail dependencies.

`copula2` As above but used only for Roy models.

`margins` It indicates the distributions used for margins. Possible distributions are Gaussian ("N"), truncated Gaussian ("tN"), Tweedie ("TW"), log-normal ("LN"), Gumbel ("GU"), reverse Gumbel ("rGU"), logistic ("LO"), Weibull ("WEI"), Inverse Gaussian ("IG"), gamma ("GA"), Dagum ("DAGUM"), Singh-Maddala ("SM"), beta ("BE"), Fisk ("FISK", also known as log-logistic distribution), Poisson ("P"), truncated Poisson ("tP"), negative binomial - type I ("NBI"), negative binomial - type II ("NBII"), Poisson inverse Gaussian ("PIG"), truncated negative binomial - type I ("tNBI"), truncated negative binomial - type II ("tNBII"), truncated Poisson inverse Gaussian ("tPIG"). If the responses are binary then possible link functions are "probit", "logit", "cloglog". For survival models, the margins can be "-cloglog" (similar to generalised proportional hazards), "-logit" (similar to generalised proportional odds), "-probit" (generalised probit). For ordinal marginals, the choices are "ord.probit" and "ord.logit". For extreme value models, there are also options we are working on, which are already implemented in the univariate gamlss() function. These are the generelised Pareto ("GP"), generelised Pareto II ("GPII") where the shape parameter is forced to be > -0.5, generelised Pareto (with orthogonal parametrisation) ("GPo") where the shape parameter is forced to be > -0.5, discrete generelised Pareto ("DGP"), discrete generelised Pareto II ("DGPII") where the shape parameter is forced to be positive, discrete generelised Pareto derived under the scenario in which shape = 0 ("DGP0"). Regarding the Tweedie, this margin can currently only be used together with a binary margin; we are working on the discrete/continuous margin extension. Another avaialble distribution is the Beta Binomial ("BB"). Here, note that the response variable should

be a matrix containing two columns, the first with the count of successes and the second with the count of failures.

`model` Possible values are "B" (bivariate model), "T" (trivariate model), "BSS" (bivariate model with non-random sample selection), "TSS" (trivariate model with double non-random sample selection), "TESS" (trivariate model with endogeneity and non-random sample selection), "BPO" (bivariate model with partial observability) and "BPO0" (bivariate model with partial observability and zero correlation). Options "T", "TESS" and "TSS" are currently for trivariate binary models only. "BPO" and "BPO0" are for bivariate binary models only. "SWITCH" is for the switching regression model.

`dof` If `copula = "T"` then the degrees of freedom can be set to a value greater than 2 and smaller than 249. Only for continuous margins, this will be taken as a starting value and the dof estimated from the data.

`dof2` As above but used only for Roy models.

`cens1` Censoring indicator for the first equation. For the case of right censored data only, this variable can be equal to 1 if the event occurred and 0 otherwise. However, if there are several censoring mechanisms then `cens` will have to be specified as a factor variable made up of four possible categories: `I` for interval, `L` for left, `R` for right, and `U` for uncensored.

`cens2` Same as above but for the second equation.

`cens3` Binary censoring indicator employed only when `dep.cens = TRUE` and administrative censoring is present.

`dep.cens` If TRUE then the dependence censored model is employed.

`ub.t1`, `ub.t2` Variable names of right/upper bounds when interval censoring is present.

`left.trunc1`, `left.trunc2` Values of truncation at left. Currently done for count distributions only.

`uni.fit` If `uni.fit = TRUE` then gamlss or gam univariate models are also fitted. This is useful for obtaining starting values, for instance.

`fp` If `TRUE` then a fully parametric model with unpenalised regression splines if fitted. See the Example 2 below.

`infl.fac` Inflation factor for the model degrees of freedom in the approximate AIC. Smoother models can be obtained setting this parameter to a value greater than 1.

`rinit` Starting trust region radius. The trust region radius is adjusted as the algorithm proceeds. See the documentation of `trust` for further details.

`rmax` Maximum allowed trust region radius. This may be set very large. If set small, the algorithm traces a steepest descent path.

`iterlimsp` A positive integer specifying the maximum number of loops to be performed before the smoothing parameter estimation step is terminated.

`tolsp` Tolerance to use in judging convergence of the algorithm when automatic smoothing parameter estimation is used.

`gc.l` This is relevant when working with big datasets. If `TRUE` then the garbage collector is called more often than it is usually done. This keeps the memory footprint down but it will slow down the routine.

parscale The algorithm will operate as if optimizing objfun(x / parscale, ...) where parscale is a scalar. If missing then no rescaling is done. See the documentation of trust for more details.

knots Optional list containing user specified knot values to be used for basis construction.

penCor This and the arguments below are only for trivariate binary models. Type of penalty for correlation coefficients. Possible values are "unpen", "lasso", "ridge", "alasso".

sp.penCor Starting value for smoothing parameter of penCor.

Chol If TRUE then the Cholesky method instead of the eigenvalue method is employed for the correlation matrix.

gamma Inflation factor used only for the alasso penalty.

w.alasso When using the alasso penalty a weight vector made up of three values must be provided.

drop.unused.levels
By default unused levels are dropped from factors before fitting. For some smooths involving factor variables this may have to be turned off (only use if you know what you are doing).

min.dn, min.pr, max.pr
These values are used to set, depending on the model used for modelling, the minimum and maximum allowed for the densities and probabilities; recall that the margins of copula models have to be in the range (0,1). These parameters are employed to avoid potential overflows/underflows in the calculations and the default values seem to offer a good compromise. Function conv.check() provides some relevant diagnostic information which can be used, for example, to check whether the lower bounds of min.dn and min.pr have been reached. So based on this or if the user wishes to do some sensitivity analysis then this can be easily carried out using these three arguments. However, the user has to be cautious. For instance, it would not make much sense to choose for min.dn and min.pr values bigger than the default ones. Bear in mind that the bounds can be reached for ill-defined models. For certain distributions/models, if convergence failure occurs and the bounds have been reached then the user can try a sensitivity analysis as mentioned above.

fixed.margins If TRUE, the marginal distributions are treated as fixed and are not estimated when fitting the dependence structure. This affects the form of the likelihood used for estimating the copula parameter. Not currently implemented for all model types.

fixed.margins.type
Specifies the type of marginal distributions assumed when fixed.margins = TRUE. The default is "continuous", which uses the continuous-margin copula likelihood based on marginal CDFs. When set to "count", the likelihood is computed using rectangle probabilities appropriate for count-valued margins. Ignored if fixed.margins = FALSE.

pf.margin1, pf.margin2
Probability mass functions. Required as additional inputs for models with count margins.

Details

The joint models considered by this function consist of two or three model equations which depend on flexible linear predictors and whose dependence between the responses is modelled through one or more parameters of a chosen multivariate distribution. The additive predictors of the equations are flexibly specified using parametric components and smooth functions of covariates. The same can be done for the dependence parameter(s) if it makes sense. Estimation is achieved within a penalized likelihood framework with integrated automatic multiple smoothing parameter selection. The use of penalty matrices allows for the suppression of that part of smooth term complexity which has no support from the data. The trade-off between smoothness and fitness is controlled by smoothing parameters associated with the penalty matrices. Smoothing parameters are chosen to minimise an approximate AIC.

For sample selection models, if there are factors in the model then before fitting the user has to ensure that the numbers of factor variables' levels in the selected sample are the same as those in the complete dataset. Even if a model could be fitted in such a situation, the model may produce fits which are not coherent with the nature of the correction sought. As an example consider the situation in which the complete dataset contains a factor variable with five levels and that only three of them appear in the selected sample. For the outcome equation (which is the one of interest) only three levels of such variable exist in the population, but their effects will be corrected for non-random selection using a selection equation in which five levels exist instead. Having differing numbers of factors' levels between complete and selected samples will also make prediction not feasible (an aspect which may be particularly important for selection models); clearly it is not possible to predict the response of interest for the missing entries using a dataset that contains all levels of a factor variable but using an outcome model estimated using a subset of these levels.

There are many continuous/discrete/survival distributions and copula functions to choose from and we plan to include more options. Get in touch if you are interested in a particular distribution.

Value

The function returns an object of class `gjrm` as described in `gjrmObject`.

WARNINGS

Convergence can be checked using `conv.check` which provides some information about the score and information matrix associated with the fitted model. The former should be close to 0 and the latter positive definite. `gjrm()` will produce some warnings if there is a convergence issue.

Convergence failure may sometimes occur. This is not necessarily a bad thing as it may indicate specific problems with a fitted model. In such a situation, the user may use rescaling (see `parscale`). Using `uni.fit = TRUE` is typically more effective than the first two options as this will provide better calibrated starting values as compared to those obtained from the default starting value procedure. The default option is, however, `uni.fit = FALSE` only because it tends to be computationally cheaper and because the default procedure has typically been found to do a satisfactory job in most cases. (The results obtained when using `uni.fit = FALSE` and `uni.fit = TRUE` could also be compared to check if starting values make any difference.)

The above suggestions may help, especially the latter option. However, the user should also consider re-specifying/simplifying the model, and/or using a diferrent dependence structure and/or checking that the chosen marginal distributions fit the responses well. In our experience, we found that convergence failure typically occurs when the model has been misspecified and/or the sample size

is low compared to the complexity of the model. Examples of misspecification include using a Clayton copula rotated by 90 degrees when a positive association between the margins is present instead, using marginal distributions that do not fit the responses, and employing a copula which does not accommodate the type and/or strength of the dependence between the margins (e.g., using AMH when the association between the margins is strong). When using smooth functions, if the covariate's values are too sparse then convergence may be affected by this. It is also worth bearing in mind that the use of three parameter marginal distributions requires the data to be more informative than a situation in which two parameter distributions are used instead.

In the contexts of endogeneity and non-random sample selection, extra attention is required when specifying the dependence parameter as a function of covariates. This is because in these situations the dependence parameter mainly models the association between the unobserved confounders in the two equations. Therefore, this option would make sense when it is believed that the strength of the association between the unobservables in the two equations varies based on some grouping factor or across geographical areas, for instance. In any case, a clear rationale is typically needed in such cases.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

See help("GJRM-package").

See Also

adjCov, vuong.test, clarke.test, GJRM-package, gjrmObject, conv.check, summary.gjrm

Examples

```
library(GJRM)

####################################
# JOINT MODELS WITH BINARY MARGINS #
####################################

###############
## EXAMPLE 1 ##


set.seed(0)

n <- 400

Sigma <- matrix(0.5, 2, 2); diag(Sigma) <- 1
u     <- rMVN(n, rep(0,2), Sigma)

x1 <- round(runif(n)); x2 <- runif(n); x3 <- runif(n)

f1   <- function(x) cos(pi*2*x) + sin(pi*x)
f2   <- function(x) x+exp(-30*(x-0.5)^2)
```

```
y1 <- ifelse(-1.55 + 2*x1    + f1(x2) + u[,1] > 0, 1, 0)
y2 <- ifelse(-0.25 - 1.25*x1 + f2(x2) + u[,2] > 0, 1, 0)

dataSim <- data.frame(y1, y2, x1, x2, x3)

## CLASSIC BIVARIATE PROBIT

out  <- gjrm(list(y1 ~ x1 + x2 + x3,
                  y2 ~ x1 + x2 + x3),
                  data = dataSim,
                  margins = c("probit", "probit"),
                  model = "B")
conv.check(out)
summary(out)
AIC(out)
BIC(out)

## Not run:


## BIVARIATE PROBIT with Splines

out  <- gjrm(list(y1 ~ x1 + s(x2) + s(x3),
                  y2 ~ x1 + s(x2) + s(x3)),
                  data = dataSim,
                  margins = c("probit", "probit"),
                  model = "B")
conv.check(out)
summary(out)
AIC(out)


## estimated smooth function plots

plot(out, eq = 1, pages = 1, seWithMean = TRUE, scale = 0)
plot(out, eq = 2, pages = 1, seWithMean = TRUE, scale = 0)


## BIVARIATE PROBIT with Splines and
## varying dependence parameter

eq.mu.1  <- y1 ~ x1 + s(x2)
eq.mu.2  <- y2 ~ x1 + s(x2)
eq.theta <-    ~ x1 + s(x2)

fl <- list(eq.mu.1, eq.mu.2, eq.theta)

outD <- gjrm(fl, data = dataSim,
             margins = c("probit", "probit"),
             model = "B")

conv.check(outD)
```

```
summary(outD)
summary(outD$theta)

plot(outD, eq = 3, seWithMean = TRUE)

###############
## EXAMPLE 2 ##

## Generate data with one endogenous variable
## and exclusion restriction (or instrument)

set.seed(0)

n <- 400

Sigma <- matrix(0.5, 2, 2); diag(Sigma) <- 1
u     <- rMVN(n, rep(0,2), Sigma)

cov   <- rMVN(n, rep(0,2), Sigma)
cov   <- pnorm(cov)
x1 <- round(cov[,1]); x2 <- cov[,2]

f1   <- function(x) cos(pi*2*x) + sin(pi*x)
f2   <- function(x) x+exp(-30*(x-0.5)^2)

y1 <- ifelse(-1.55 + 2*x1    + f1(x2) + u[,1] > 0, 1, 0)
y2 <- ifelse(-0.25 - 1.25*y1 + f2(x2) + u[,2] > 0, 1, 0)

dataSim <- data.frame(y1, y2, x1, x2)

#
## Testing the hypothesis of absence of endogeneity...
#

LM.bpm(list(y1 ~ x1 + s(x2), y2 ~ y1 + s(x2)), dataSim, model = "B")


## CLASSIC RECURSIVE BIVARIATE PROBIT

out <- gjrm(list(y1 ~ x1 + x2,
                 y2 ~ y1 + x2),
                 data = dataSim,
                 margins = c("probit", "probit"), model = "B")
conv.check(out)
summary(out)

## FLEXIBLE RECURSIVE BIVARIATE PROBIT

out <- gjrm(list(y1 ~ x1 + s(x2),
                 y2 ~ y1 + s(x2)),
                 data = dataSim,
                 margins = c("probit", "probit"),
                 model = "B")
```

```
conv.check(out)
summary(out)

#
## Testing the hypothesis of absence of endogeneity post estimation...

gt.bpm(out)

#
## Causal effects
## average treatment effect, risk ratio and odds ratio with CIs

mb(y1, y2, model = "B")
ATE(out, trt = "y1")
RR(out, trt = "y1")
OR(out, trt = "y1")
ATE(out, trt = "y1", joint = FALSE)

## try a Clayton copula model...

outC <- gjrm(list(y1 ~ x1 + s(x2),
                  y2 ~ y1 + s(x2)),
                  data = dataSim, copula = "C0",
                  margins = c("probit", "probit"),
                  model = "B")
conv.check(outC)
summary(outC)
ATE(outC, trt = "y1")

## try a Joe copula model...

outJ <- gjrm(list(y1 ~ x1 + s(x2),
                  y2 ~ y1 + s(x2)),
                  data = dataSim, copula = "J0",
                  margins = c("probit", "probit"),
                  model = "B")
conv.check(outJ)
summary(outJ)
ATE(outJ, "y1")

vuong.test(out, outJ)
clarke.test(out, outJ)


#
## recursive bivariate probit modelling with unpenalized splines
## can be achieved as follows

outFP <- gjrm(list(y1 ~ x1 + s(x2, bs = "cr", k = 5),
                   y2 ~ y1 + s(x2, bs = "cr", k = 6)),
                   fp = TRUE, data = dataSim,
                   margins = c("probit", "probit"),
                   model = "B")
```

```
conv.check(outFP)
summary(outFP)

# in the above examples a third equation could be introduced
# as illustrated in Example 1

###############
## EXAMPLE 3 ##

## Generate data with a non-random sample selection mechanism
## and exclusion restriction

set.seed(0)

n <- 2000

Sigma <- matrix(0.5, 2, 2); diag(Sigma) <- 1
u     <- rMVN(n, rep(0,2), Sigma)

SigmaC <- matrix(0.5, 3, 3); diag(SigmaC) <- 1
cov    <- rMVN(n, rep(0,3), SigmaC)
cov    <- pnorm(cov)
bi <- round(cov[,1]); x1 <- cov[,2]; x2 <- cov[,3]

f11 <- function(x) -0.7*(4*x + 2.5*x^2 + 0.7*sin(5*x) + cos(7.5*x))
f12 <- function(x) -0.4*( -0.3 - 1.6*x + sin(5*x))
f21 <- function(x) 0.6*(exp(x) + sin(2.9*x))

ys <-  0.58 + 2.5*bi + f11(x1) + f12(x2) + u[, 1] > 0
y  <- -0.68 - 1.5*bi + f21(x1) +         + u[, 2] > 0
yo <- y*(ys > 0)

dataSim <- data.frame(y, ys, yo, bi, x1, x2)

#
## Testing the hypothesis of absence of non-random sample selection...

LM.bpm(list(ys ~ bi + s(x1) + s(x2), yo ~ bi + s(x1)), dataSim, model = "BSS")

# p-value suggests presence of sample selection

#
## SEMIPARAMETRIC SAMPLE SELECTION BIVARIATE PROBIT
## the first equation MUST be the selection equation

out <- gjrm(list(ys ~ bi + s(x1) + s(x2),
                 yo ~ bi + s(x1)),
                 data = dataSim, model = "BSS",
                 margins = c("probit", "probit"))
conv.check(out)
gt.bpm(out)

## compare the two summary outputs below
```

```
## the second output produces a summary of the results obtained when
## selection bias is not accounted for

summary(out)
summary(out$gam2)

## corrected predicted probability that 'yo' is equal to 1

mb(ys, yo, model = "BSS")
prev(out)
prev(out, joint = FALSE)

## estimated smooth function plots
## the red line is the true curve
## the blue line is the univariate model curve not accounting for selection bias

x1.s <- sort(x1[dataSim$ys>0])
f21.x1 <- f21(x1.s)[order(x1.s)]-mean(f21(x1.s))

plot(out, eq = 2, ylim = c(-1.65,0.95)); lines(x1.s, f21.x1, col="red")
par(new = TRUE)
plot(out$gam2, se = FALSE, col = "blue", ylim = c(-1.65,0.95),
     ylab = "", rug = FALSE)

#
#
## try a Clayton copula model...

outC <- gjrm(list(ys ~ bi + s(x1) + s(x2),
                  yo ~ bi + s(x1)),
                  data = dataSim, model = "BSS", copula = "C0",
                  margins = c("probit", "probit"))
conv.check(outC)
summary(outC)
prev(outC)

###############
## EXAMPLE 4 ##

## Generate data with partial observability

set.seed(0)

n <- 1000

Sigma <- matrix(0.5, 2, 2); diag(Sigma) <- 1
u     <- rMVN(n, rep(0,2), Sigma)

x1 <- round(runif(n)); x2 <- runif(n); x3 <- runif(n)

y1 <- ifelse(-1.55 + 2*x1 + x2 + u[,1] > 0, 1, 0)
y2 <- ifelse( 0.45 - x3        + u[,2] > 0, 1, 0)
y  <- y1*y2
```

```
dataSim <- data.frame(y, x1, x2, x3)


## BIVARIATE PROBIT with Partial Observability

out  <- gjrm(list(y ~ x1 + x2,
                  y ~ x3),
                  data = dataSim, model = "BPO",
                  margins = c("probit", "probit"))
conv.check(out)
summary(out)

# first ten estimated probabilities for the four events from object out

cbind(out$p11, out$p10, out$p00, out$p01)[1:10,]


# case with smooth function

f1 <- function(x) cos(pi*2*x) + sin(pi*x)

y1 <- ifelse(-1.55 + 2*x1 + f1(x2) + u[,1] > 0, 1, 0)
y2 <- ifelse( 0.45 - x3            + u[,2] > 0, 1, 0)
y  <- y1*y2

dataSim <- data.frame(y, x1, x2, x3)

out  <- gjrm(list(y ~ x1 + s(x2),
                  y ~ x3),
                  data = dataSim, model = "BPO",
                  margins = c("probit", "probit"))

conv.check(out)
summary(out)

plot(out, eq = 1, scale = 0)

#########################################################
# JOINT MODELS WITH BINARY AND/OR CONTINUOUS MARGINS #
#########################################################

###############
## EXAMPLE 5 ##

## Generate data
## Correlation between the two equations 0.5 - Sample size 400

set.seed(0)

n <- 400

Sigma <- matrix(0.5, 2, 2); diag(Sigma) <- 1
```

```
u     <- rMVN(n, rep(0,2), Sigma)

x1 <- round(runif(n)); x2 <- runif(n); x3 <- runif(n)

f1   <- function(x) cos(pi*2*x) + sin(pi*x)
f2   <- function(x) x+exp(-30*(x-0.5)^2)

y1 <- -1.55 + 2*x1    + f1(x2) + u[,1]
y2 <- -0.25 - 1.25*x1 + f2(x2) + u[,2]

dataSim <- data.frame(y1, y2, x1, x2, x3)

resp.check(y1, "N")
resp.check(y2, "N")

eq.mu.1     <- y1 ~ x1 + s(x2) + s(x3)
eq.mu.2     <- y2 ~ x1 + s(x2) + s(x3)
eq.sigma1   <-    ~ 1
eq.sigma2   <-    ~ 1
eq.theta    <-    ~ x1

fl <- list(eq.mu.1, eq.mu.2, eq.sigma1, eq.sigma2, eq.theta)

# the order above is the one to follow when
# using more than two equations

out  <- gjrm(fl, data = dataSim, margins = c("N", "N"),
             model = "B")

conv.check(out)
res.check(out, intervals = TRUE)
res.check(out, joint = TRUE, intervals = TRUE)
summary(out)
AIC(out)
BIC(out)

nd <- data.frame(x1 = 1, x2 = 0.4, x3 = 0.6)
copula.prob(out, y1 = 1.4, y2 = 2.3, newdata = nd, intervals = TRUE)

###############
## EXAMPLE 6 ##

## Generate data with one endogenous binary variable
## and continuous outcome

set.seed(0)

n <- 1000

Sigma <- matrix(0.5, 2, 2); diag(Sigma) <- 1
u     <- rMVN(n, rep(0,2), Sigma)

cov   <- rMVN(n, rep(0,2), Sigma)
```

```
cov   <- pnorm(cov)
x1 <- round(cov[,1]); x2 <- cov[,2]

f1   <- function(x) cos(pi*2*x) + sin(pi*x)
f2   <- function(x) x+exp(-30*(x-0.5)^2)

y1 <- ifelse(-1.55 + 2*x1    + f1(x2) + u[,1] > 0, 1, 0)
y2 <-        -0.25 - 1.25*y1 + f2(x2) + u[,2]

dataSim <- data.frame(y1, y2, x1, x2)


## RECURSIVE Model

out <- gjrm(list(y1 ~ x1 + x2,
                 y2 ~ y1 + x2),
                 data = dataSim, margins = c("probit","N"),
                 model = "B")
conv.check(out)
summary(out)
res.check(out, intervals = TRUE)

## SEMIPARAMETRIC RECURSIVE Model

eq.mu.1   <- y1 ~ x1 + s(x2)
eq.mu.2   <- y2 ~ y1 + s(x2)
eq.sigma  <-    ~ 1
eq.theta  <-    ~ 1

fl <- list(eq.mu.1, eq.mu.2, eq.sigma, eq.theta)

out <- gjrm(fl, data = dataSim,
            margins = c("probit","N"), uni.fit = TRUE,
            model = "B")
conv.check(out)
summary(out)
res.check(out, intervals = TRUE)
ATE(out, trt = "y1")
ATE(out, trt = "y1", joint = FALSE)

#
#

################
## EXAMPLE 7 ##

## Generate data with one endogenous continuous exposure
## and binary outcome

set.seed(0)

n <- 1000
```

```
Sigma <- matrix(0.5, 2, 2); diag(Sigma) <- 1
u     <- rMVN(n, rep(0,2), Sigma)

cov   <- rMVN(n, rep(0,2), Sigma)
cov   <- pnorm(cov)
x1 <- round(cov[,1]); x2 <- cov[,2]

f1   <- function(x) cos(pi*2*x) + sin(pi*x)
f2   <- function(x) x+exp(-30*(x-0.5)^2)

y1 <-        -0.25 - 2*x1    + f2(x2) + u[,2]
y2 <- ifelse(-0.25 - 0.25*y1 + f1(x2) + u[,1] > 0, 1, 0)

dataSim <- data.frame(y1, y2, x1, x2)

eq.mu.1   <- y2 ~ y1 + s(x2)
eq.mu.2   <- y1 ~ x1 + s(x2)
eq.sigma  <-    ~ 1
eq.theta  <-    ~ 1

fl <- list(eq.mu.1, eq.mu.2, eq.sigma, eq.theta)

out <- gjrm(fl, data = dataSim,
            margins = c("probit","N"),
            model = "B")
conv.check(out)
summary(out)
ATE(out, trt = "y1", trt.val = 1)
ATE(out, trt = "y1", trt.val = 1, joint = FALSE)
 RR(out, trt = "y1", trt.val = 1)
 RR(out, trt = "y1", trt.val = 1, joint = FALSE)
 OR(out, trt = "y1", trt.val = 1)
 OR(out, trt = "y1", trt.val = 1, joint = FALSE)

#
#

######################
## EXAMPLE 8       ##
## SURVIVAL MODELS ##

set.seed(0)

n  <- 2000
c  <- runif(n, 3, 8)
u  <- runif(n, 0, 1)
z1 <- rbinom(n, 1, 0.5)
z2 <- runif(n, 0, 1)
t  <- rep(NA, n)

beta_0 <- -0.2357
beta_1 <- 1
```

```
f <- function(t, beta_0, beta_1, u, z1, z2){
  S_0 <- 0.7 * exp(-0.03*t^1.9) + 0.3*exp(-0.3*t^2.5)
  exp(-exp(log(-log(S_0))+beta_0*z1 + beta_1*z2))-u
}


for (i in 1:n){
   t[i] <- uniroot(f, c(0, 8), tol = .Machine$double.eps^0.5,
                   beta_0 = beta_0, beta_1 = beta_1, u = u[i],
                   z1 = z1[i], z2 = z2[i], extendInt = "yes" )$root
}

delta1  <- ifelse(t < c, 1, 0)
u1      <- apply(cbind(t, c), 1, min)
dataSim <- data.frame(u1, delta1, z1, z2)


c <- runif(n, 4, 8)
u <- runif(n, 0, 1)
z <- rbinom(n, 1, 0.5)
beta_0 <- -1.05
t      <- rep(NA, n)

f <- function(t, beta_0, u, z){
  S_0 <- 0.7 * exp(-0.03*t^1.9) + 0.3*exp(-0.3*t^2.5)
  1/(1 + exp(log((1-S_0)/S_0)+beta_0*z))-u
}



for (i in 1:n){
    t[i] <- uniroot(f, c(0, 8), tol = .Machine$double.eps^0.5,
                    beta_0 = beta_0, u = u[i], z = z[i],
                    extendInt="yes" )$root
}

delta2 <- ifelse(t < c,1, 0)
u2     <- apply(cbind(t, c), 1, min)
dataSim$delta2 <- delta2
dataSim$u2     <- u2
dataSim$z      <- z



eq1 <- u1 ~ s(log(u1), bs = "mpi") + z1 + s(z2)
eq2 <- u2 ~ s(log(u2), bs = "mpi") + z
eq3 <-    ~ s(z2)

out <- gjrm(list(eq1, eq2), data = dataSim,
            margins = c("-cloglog", "-logit"),
            cens1 = delta1, cens2 = delta2, model = "B")

conv.check(out)
```

```
res.check(out, intervals = TRUE)

summary(out)
AIC(out); BIC(out)
plot(out, eq = 1, scale = 0, pages = 1)
plot(out, eq = 2, scale = 0, pages = 1)

haz.surv(out, eq = 1, newdata = data.frame(z1 = 0, z2 = 0),
        shade = TRUE, n.sim = 100, baseline = TRUE)
haz.surv(out, eq = 1, newdata = data.frame(z1 = 0, z2 = 0),
        shade = TRUE, n.sim = 100, type = "haz", baseline = TRUE,
             intervals = FALSE)
haz.surv(out, eq = 2, newdata = data.frame(z = 0),
        shade = FALSE, n.sim = 100, baseline = TRUE)
haz.surv(out, eq = 2, newdata = data.frame(z = 0),
        shade = TRUE, n.sim = 100, type = "haz", baseline = TRUE)

newd0 <- newd1 <- data.frame(z = 0, z1 = mean(dataSim$z1),
                             z2 = mean(dataSim$z2),
                             u1 =  mean(dataSim$u1) + 1,
                             u2 =  mean(dataSim$u2) + 1)
newd1$z <- 1

copula.prob(out, newdata = newd0, intervals = TRUE)
copula.prob(out, newdata = newd1, intervals = TRUE)

out1 <- gjrm(list(eq1, eq2, eq3), data = dataSim,
                  margins = c("-cloglog", "-logit"),
                  cens1 = delta1, cens2 = delta2, uni.fit = TRUE,
                  model = "B")


####################################################
## Joint continuous and survival outcomes
####################################################
# this is complete, just testing, get in touch if interested
#
# eq1 <- z2 ~ z1
# eq2 <- u2 ~ s(u2, bs = "mpi") + z
# eq3 <-    ~ s(z2)
# eq4 <-    ~ s(z2)
#
# f.l <- list(eq1, eq2, eq3, eq4)
#
# out3 <- gjrm(f.l, data = dataSim,
#                   margins = c("N", "-logit"),
#                   cens1 = NULL, cens2 = delta2,
#                   uni.fit = TRUE, model = "B")
#
# conv.check(out3)
# res.check(out3)
# summary(out3)
# AIC(out3); BIC(out3)
```

```
# plot(out3, eq = 2, scale = 0, pages = 1)
# plot(out3, eq = 3, scale = 0, pages = 1)
# plot(out3, eq = 4, scale = 0, pages = 1)
#
# newd <- newd1 <- data.frame(z = 0, z1 = mean(dataSim$z1),
#                             z2 = mean(dataSim$z2),
#                             u2 =  mean(dataSim$u2) + 1)
#
# copula.prob(out3, y1 = 0.6, newdata = newd, intervals = TRUE)

########################################
# JOINT MODELS WITH THREE BINARY MARGINS #
########################################

###############
## EXAMPLE 9 ##

## Generate data
## Correlation between the two equations 0.5 - Sample size 400

set.seed(0)

n <- 400

Sigma <- matrix(0.5, 3, 3); diag(Sigma) <- 1
u     <- rMVN(n, rep(0,3), Sigma)

x1 <- round(runif(n)); x2 <- runif(n); x3 <- runif(n)

f1   <- function(x) cos(pi*2*x) + sin(pi*x)
f2   <- function(x) x+exp(-30*(x-0.5)^2)

y1 <- ifelse(-1.55 +    2*x1 - f1(x2) + u[,1] > 0, 1, 0)
y2 <- ifelse(-0.25 - 1.25*x1 + f2(x2) + u[,2] > 0, 1, 0)
y3 <- ifelse(-0.75 + 0.25*x1          + u[,3] > 0, 1, 0)

dataSim <- data.frame(y1, y2, y3, x1, x2)

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ x1 + s(x2),
            y3 ~ x1)

margs <- c("probit", "probit", "probit")

out  <- gjrm(f.l, data = dataSim, model = "T", margins = margs)
out1 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)

conv.check(out)
summary(out)
plot(out, eq = 1)
plot(out, eq = 2)
AIC(out)
BIC(out)
```

```
margs <- c("probit","logit","cloglog")


out  <- gjrm(f.l, data = dataSim, model = "T", margins = margs)
out1 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)
conv.check(out)
summary(out)
plot(out, eq = 1)
plot(out, eq = 2)

margs <- c("probit", "probit", "probit")

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ x1 + s(x2),
            y3 ~ x1,
               ~ 1, ~ 1, ~ 1)

out1 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ x1 + s(x2),
            y3 ~ x1,
               ~ 1, ~ s(x2), ~ 1)

out2 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ x1 + s(x2),
            y3 ~ x1,
               ~ x1, ~ s(x2), ~ x1 + s(x2))

out2 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ x1 + s(x2),
            y3 ~ x1,
               ~ x1, ~ x1, ~ s(x2))

out2 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ x1 + s(x2),
            y3 ~ x1,
               ~ x1, ~ x1 + x2, ~ s(x2))

out2 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ x1 + s(x2),
            y3 ~ x1,
               ~ x1 + x2, ~ x1 + x2, ~ x1 + x2)
```

```
out2 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T", margins = margs)

nw <- data.frame(x1 = 1, x2 = 0.7)

copula.prob(out2, 1, 1, 1, newdata = nw, cond = 0, intervals = TRUE, n.sim = 100)


# with endogenous variable

f.l <- list(y1 ~ x1 + s(x2),
            y2 ~ y1 + x1 + s(x2),
            y3 ~ x1)

margs <- c("probit", "probit", "probit")

out <- gjrm(f.l, data = dataSim, model = "T", margins = margs)

conv.check(out)
summary(out)

ATE(out, trt = "y1", eq = 2, joint = TRUE)
ATE(out, trt = "y1", eq = 2, joint = FALSE)

#################
## EXAMPLE 10 ##

## Generate data
## with double sample selection

set.seed(0)

n <- 5000

Sigma <- matrix(c(1,   0.5, 0.4,
                  0.5,   1, 0.6,
                  0.4, 0.6,   1 ), 3, 3)

u <- rMVN(n, rep(0,3), Sigma)

f1   <- function(x) cos(pi*2*x) + sin(pi*x)
f2   <- function(x) x+exp(-30*(x-0.5)^2)

x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
x4 <- runif(n)

y1 <-  1    + 1.5*x1 -     x2 + 0.8*x3 - f1(x4) + u[, 1] > 0
y2 <-  1    - 2.5*x1 + 1.2*x2 +     x3          + u[, 2] > 0
y3 <-  1.58 + 1.5*x1 - f2(x2)                   + u[, 3] > 0

dataSim <- data.frame(y1, y2, y3, x1, x2, x3, x4)
```

```
f.l <- list(y1 ~ x1 + x2 + x3 + s(x4),
            y2 ~ x1 + x2 + x3,
            y3 ~ x1 + s(x2))

out <- gjrm(f.l, data = dataSim, model = "TSS",
            margins = c("probit", "probit", "probit"))
conv.check(out)
summary(out)
plot(out, eq = 1)
plot(out, eq = 3)
prev(out)
prev(out, joint = FALSE)

#############
## EXAMPLE 11

set.seed(0)

n <- 2000

rh <- 0.5

sigmau <- matrix(c(1, rh, rh, 1), 2, 2)
u      <- rMVN(n, rep(0,2), sigmau)

sigmac <- matrix(rh, 3, 3); diag(sigmac) <- 1
cov    <- rMVN(n, rep(0,3), sigmac)
cov    <- pnorm(cov)

bi <- round(cov[,1]); x1 <- cov[,2]; x2 <- cov[,3]

f11 <- function(x) -0.7*(4*x + 2.5*x^2 + 0.7*sin(5*x) + cos(7.5*x))
f12 <- function(x) -0.4*( -0.3 - 1.6*x + sin(5*x))
f21 <- function(x) 0.6*(exp(x) + sin(2.9*x))

ys <-  0.58 + 2.5*bi + f11(x1) + f12(x2) + u[, 1] > 0
y  <- -0.68 - 1.5*bi + f21(x1) +           u[, 2]
yo <- y*(ys > 0)

dataSim <- data.frame(ys, yo, bi, x1, x2)

## CLASSIC SAMPLE SELECTION MODEL
## the first equation MUST be the selection equation

resp.check(yo[ys > 0], "N")

out <- gjrm(list(ys ~ bi + x1 + x2,
                 yo ~ bi + x1),
                 data = dataSim, model = "BSS",
                 margins = c("probit", "N"))
conv.check(out)
res.check(out, intervals = TRUE)
summary(out)
```

```
AIC(out)
BIC(out)


## SEMIPARAMETRIC SAMPLE SELECTION MODEL

out <- gjrm(list(ys ~ bi + s(x1) + s(x2),
                 yo ~ bi + s(x1)),
                 data = dataSim, model = "BSS",
                 margins = c("probit", "N"))
conv.check(out)
res.check(out, intervals = TRUE)
AIC(out)

## compare the two summary outputs
## the second output produces a summary of the results obtained when only
## the outcome equation is fitted, i.e. selection bias is not accounted for

summary(out)
summary(out$gam2)

## estimated smooth function plots
## the red line is the true curve
## the blue line is the naive curve not accounting for selection bias

x1.s <- sort(x1[dataSim$ys>0])
f21.x1 <- f21(x1.s)[order(x1.s)] - mean(f21(x1.s))

plot(out, eq = 2, ylim = c(-1, 0.8)); lines(x1.s, f21.x1, col = "red")
par(new = TRUE)
plot(out$gam2, se = FALSE, lty = 3, lwd = 2, ylim = c(-1, 0.8),
     ylab = "", rug = FALSE)


##

## SEMIPARAMETRIC SAMPLE SELECTION MODEL with association
## and dispersion parameters
## depending on covariates as well

eq.mu.1   <- ys ~ bi + s(x1) + s(x2)
eq.mu.2   <- yo ~ bi + s(x1)
eq.sigma  <-    ~ bi
eq.theta  <-    ~ bi + x1

fl <- list(eq.mu.1, eq.mu.2, eq.sigma, eq.theta)

out <- gjrm(fl, data = dataSim, model = "BSS",
                 margins = c("probit", "N"))
conv.check(out)
res.check(out)
summary(out)
```

```
summary(out$sigma)
summary(out$theta)

nd <- data.frame(bi = 0, x1 = 0.2, x2 = 0.8)
copula.prob(out, 0, 0.3, newdata = nd, intervals = TRUE)

outC0 <- gjrm(fl, data = dataSim, copula = "C0", model = "BSS",
              margins = c("probit", "N"))
conv.check(outC0)
res.check(outC0)


## End(Not run)
```

---

gjrm.pcc *Three-Dimensional PCC Models with Continuous Margins*

Description

Fits a 3-dimensional copula model using the pair-copula construction approach by selecting and estimating three bivariate copulas, optionally incorporating covariates or smooth effects.

Usage

```
gjrm.pcc(obj, data, cond.var, eq.theta  = NULL, simplified = FALSE,
         criterion = "aic", sig.lev = 0.05, dof.grid = 3:12,
         infl.fac = c(1, 1, 1), copulas = NULL, dofs = NULL)
```

Arguments

obj	A list containing three gamlss objects. Each element has to be fitted using GJRM::gamlss().
data	Compulsory data.frame.
cond.var	Integer (1, 2, or 3) indicating the variable to condition on in the third pair-copula.
eq.theta	Optional formula for covariate effects in the copula parameter.
simplified	Logical. If TRUE, then a simplified PCC is fitted.
criterion	Model selection criterion. One of: "aic", "bic".
sig.lev	Significance level for copula selection when using the Vuong and Clarke tests.
dof.grid	Integer vector of candidate degrees of freedom for the Student-t copula.
infl.fac	Inflation factor for the effective degrees of freedom in the approximate AIC. Values greater than 1 increase the penalty on model complexity, leading to smoother and more parsimonious fits. Can be a scalar or a vector matching the number of pair-copula components.

copulas	Optional character vector specifying the copula family used for each pair-copula in the PCC. Each element must correspond to a pair-copula in the decomposition (in the order defined by the model structure). Allowed values depend on the implementation (e.g. Gaussian, Clayton, Gumbel, Student-t, etc.). If NULL, the copula families are selected automatically using the specified criterion. If a Student-t copula is specified (e.g., "T"), the corresponding degrees of freedom must be provided in dofs.
dofs	Optional numeric vector of degrees of freedom associated with Student-t copulas in copulas. This argument must have the same length as copulas. Entries are only used when the corresponding element of copulas is a Student-t copula (e.g., "T"); otherwise they are ignored. If a Student-t copula is included in copulas, the corresponding element in dofs must be provided; otherwise the function will stop with an error.

Value

A list containing the fitted pair-copula models, selected copulas, estimated parameters, total AIC/BIC and additional model details. The usual summary and plotting functions can be employed for each single fitted model.

---

gjrmObject *Fitted gjrm object*

Description

A fitted joint model returned by function gjrm and of class "gjrm", "SemiParBIV", "SemiParTRIV", etc.

Value

fit	List of values and diagnostics extracted from the output of the algorithm.
gam1	Univariate fit for equation 1. See the documentation of mgcv for full details.
gam2, gam3, ...	Univariate fit for equation 2, equation 3, etc.
coefficients	The coefficients of the fitted model.
weights	Prior weights used during model fitting.
sp	Estimated smoothing parameters of the smooth components.
iter.sp	Number of iterations performed for the smoothing parameter estimation step.
iter.if	Number of iterations performed in the initial step of the algorithm.
iter.inner	Number of iterations performed within the smoothing parameter estimation step.
theta	Estimated dependence parameter linking the two equations.
n	Sample size.
X1, X2, X3, ...	Design matrices associated with the linear predictors.
X1.d2, X2.d2, X3.d2, ...	Number of columns of X1, X2, X3, etc.

`l.sp1`, `l.sp2`, `l.sp3`, ...
: Number of smooth components in the equations.

`He`
: Penalized -hessian/Fisher. This is the same as `HeSh` for unpenalized models.

`HeSh`
: Unpenalized -hessian/Fisher.

`Vb`
: Inverse of `He`. This corresponds to the Bayesian variance-covariance matrix used for confidence/credible interval calculations.

`F`
: This is obtained multiplying Vb by HeSh.

`t.edf`
: Total degrees of freedom of the estimated bivariate model. It is calculated as `sum(diag(F))`.

`edf1`, `edf2`, `edf3`, ...
: Degrees of freedom for the two equations of the fitted bivariate model (and for the third and fourth equations if present. They are calculated when splines are used.

`bs.mgfit`
: List of values and diagnostics extracted from `magic` in `mgcv`.

`conv.sp`
: If `TRUE` then the smoothing parameter selection algorithm stopped before reaching the maximum number of iterations allowed.

`wor.c`
: Working model quantities.

`eta1`, `eta2`, `eta3`, ...
: Estimated linear predictors for the two equations (as well as the third and fourth equations if present).

`y1`, `y2`
: Responses of the two equations.

`logLik`
: Value of the (unpenalized) log-likelihood evaluated at the (penalized or unpenalized) parameter estimates.

`respvec`
: List containing response vectors.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

`gjrm`, `summary.gjrm`

---

gt.bpm *Gradient test*

Description

`gt.bpm` can be used to test the hypothesis of absence of endogeneity, correlated model equations/errors or non-random sample selection in binary bivariate probit models.

Usage

```
gt.bpm(x)
```

Arguments

`x`	A fitted `gjrm` object.

Details

The gradient test was first proposed by Terrell (2002) and it is based on classic likelihood theory. See Marra et al. (2017) for full details.

Value

It returns a numeric p-value corresponding to the null hypothesis that the correlation, θ, is equal to 0.

WARNINGS

This test's implementation is only valid for bivariate binary probit models with normal errors.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Marra G., Radice R. and Filippou P. (2017), Regression Spline Bivariate Probit Models: A Practical Approach to Testing for Exogeneity. *Communications in Statistics - Simulation and Computation*, 46(3), 2283-2298.

Terrell G. (2002), The Gradient Statistic. *Computing Science and Statistics*, 34, 206-215.

Examples

```
## see examples for gjrm
```

H.tri *Internal Function*

Description

This and other similar internal functions calculate the Hessian for trivariate binary models.

Author(s)

Author: Panagiota Filippou

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

`haz.surv` *Post-estimation calculation of hazard, cumulative hazard and survival functions*

Description

This function produces estimated values, intervals and plots for the hazard, cumulative hazard and survival functions.

Usage

```
haz.surv(x, eq, newdata, type = "surv", t.range = NULL, t.vec = NULL,
        intervals = TRUE, n.sim = 100, prob.lev = 0.05, shade = FALSE,
        bars = FALSE, ylim, ylab, xlab, pch, ls = 100, baseline = FALSE,
        min.dn = 1e-200, min.pr = 1e-200, max.pr = 1, plot = TRUE,
        print.progress = TRUE, ...)
```

Arguments

`x`	A fitted `gamlss`/`gjrm` object.
`eq`	Equation number. This can be ignored for univariate models.
`newdata`	A data frame or list containing the values of the model covariates at which predictions are required. This must always be provided. For the individual survival/hazard/cumulative hazard function, the data frame must have one row containing the values of the model covariates corresponding to the individual of interest. For the (sub-)population survival/hazard/cumulative hazard function, the data frame must have as many rows as there are individuals in the (sub-)population of interest. Each row must contain the values of the model covariates of the corresponding individual.
`type`	Either `"surv"`, `"haz"` or `"cum.haz"`. In the excess hazard setting these are, respectively, the net survival, the excess hazard and the cumulative excess hazard.
`t.range`	Time variable range. This must be a vector with only two elements: the minimum and maximum of the time range. If `NULL` then it is determined automatically based on the observed data.
`t.vec`	Vector of time values. This can also be a single time. Note you cannot provide both `t.range` and `t.vec` as they are two mutually exclusive ways of defining the time variable. If `NULL` then it is determined automatically based on the observed data.
`intervals`	If `TRUE` then intervals are also produced.
`n.sim`	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used for interval calculations.

prob.lev	Overall probability of the left and right tails of the probabilities' distributions used for interval calculations.
shade	If TRUE then it produces shaded regions as confidence bands.
bars	If TRUE then the confidence intervals are plotted as bars rather than continuous curves. If t.vec is used and only one time value is provided, this is the only possible plotting option for the confidence intervals. Note shade and bars are mutually exclusive.
ylim, ylab, xlab, pch	Usual plot arguments.
ls	Length of sequence to use for time variable.
baseline	If baseline is desired; this will set all covariate/smooth effects to zero.
min.dn, min.pr, max.pr	Allowed minimum and maximum for estimated probabities and densities for survival, hazard and cumulative hazard calculations.
plot	If FALSE then the function does not produce a plot. The default is TRUE.
print.progress	If FALSE then the function does not print progress made. The default is TRUE.
...	Other arguments to pass to plot.

Value

It produces estimated values, intervals and plots for the hazard, cumulative hazard and survival functions.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

hfunc *Compute h-functions (conditional distributions) for a bivariate copula*

Description

hfunc computes both h-functions for a given bivariate copula, using a pair of uniform pseudo-observations and the copula parameters. These h-functions are essential for vine copula constructions and likelihood evaluation in pair-copula decompositions.

Usage

```
hfunc(p1, p2, theta, copula = "N", dof = NULL)
```

Arguments

p1	A numeric vector of pseudo-observations (marginal uniform variables) from the first variable.
p2	A numeric vector of pseudo-observations from the second variable.
theta	Dependence copula parameter.
copula	A character string indicating the bivariate copula family.
dof	Degrees of freedom parameter, only required for the Student t-copula.

Details

This function computes the so called *h-functions* which are required when performing pair-copula constructions, simulating from vine copulas or evaluating vine log-likelihoods.

The function returns a named list with two components:

- h1: Partial derivative of $C(u_1, u_2)$ with respect to u_1.
- h2: Partial derivative of $C(u_1, u_2)$ with respect to u_2.

Author(s)

Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

k.tau *Kendall's tau a fitted joint model*

Description

k.tau can be used to calculate the Kendall's tau from a fitted joint model with intervals obtained via posterior simulation.

Usage

```
k.tau(x, prob.lev = 0.05)
```

Arguments

x	A fitted gjrm object as produced by the respective fitting function.
prob.lev	Overall probability of the left and right tails of the probabilities' distributions used for interval calculations.

Details

This function calculates the Kendall's tau a fitted simultaneous model, with intervals obtained via posterior simulation. Note that this is derived under the assumption of continuous margins.

Value

`res`	It returns the estimated tau with lower and upper interval limits.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm

---

llpsi *Internal Function*

Description

Log-logistic robust function.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

LM.bpm *Lagrange Multiplier Test (Score Test)*

Description

Before fitting a bivariate probit model, `LM.bpm` can be used to test the hypothesis of absence of endogeneity, correlated model equations/errors or non-random sample selection.

Usage

```
LM.bpm(formula, data = list(), weights = NULL, subset = NULL, model, hess = TRUE)
```

Arguments

`formula`	A list of two formulas, one for equation 1 and the other for equation 2. `s` terms are used to specify smooth smooth functions of predictors. Note that if `model = "BSS"` then the first formula MUST refer to the selection equation.
`data`	An optional data frame, list or environment containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`weights`	Optional vector of prior weights to be used in fitting.
`subset`	Optional vector specifying a subset of observations to be used in the fitting process.

model	It indicates the type of model to be used in the analysis. Possible values are "B" (bivariate model) and "BSS" (bivariate model with sample selection). The two marginal equations have probit links.
hess	If FALSE then the expected (rather than observed) information matrix is employed.

Details

This Lagrange multiplier test (also known as score test) is used here for testing the null hypothesis that θ is equal to 0 (i.e. no endogeneity, non-random sample selection or correlated model equations/errors, depending on the model being fitted). Its main advantage is that it does not require an estimate of the model parameter vector under the alternative hypothesis. Asymptotically, it takes a Chi-squared distribution with one degree of freedom. Full details can be found in Marra et al. (2014) and Marra et al. (2017).

Value

It returns a numeric p-value corresponding to the null hypothesis that the correlation, θ, is equal to 0.

WARNINGS

This test's implementation is ONLY valid for bivariate binary probit models with normal errors.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Marra G., Radice R. and Filippou P. (2017), Regression Spline Bivariate Probit Models: A Practical Approach to Testing for Exogeneity. *Communications in Statistics - Simulation and Computation*, 46(3), 2283-2298.

Marra G., Radice R. and Missiroli S. (2014), Testing the Hypothesis of Absence of Unobserved Confounding in Semiparametric Bivariate Probit Models. *Computational Statistics*, 29(3-4), 715-741.

See Also

gjrm

Examples

```
## see examples for gjrm
```

`lmc` *Linear Model Fitting with Constraints*

Description

Linear model fitting with positivity and sum-to-one constraints on the model's coefficients.

Usage

```
lmc(y, X, start.v = NULL, lambda = 1, pen = "none", gamma = 1, a = 3.7)
```

Arguments

`y`	Response vector.
`X`	Design matrix.
`start.v`	Starting values.
`lambda`	Tuning parameter.
`pen`	Type of penalty. Choices are: none, ridge, lasso, alasso, scad.
`gamma`	Power parameter of adaptive lasso.
`a`	Scad parameter.

Details

Linear model fitting with positivity and sum-to-one constraints on the model's coefficients.

Value

The function returns an object of class `lmc`.

Examples

```
## Not run:

library(GJRM)

set.seed(1)

n    <- 1000
beta <- c(0.07, 0.08, 0.21, 0.12, 0.15, 0.17, 0.2)
l    <- length(beta)
X    <- matrix(runif(n*l), n, l)

y    <- X%*%beta + rnorm(n)

out <- lmc(y, X)
```

```
conv.check(out)

out1 <- lmc(y, X, start.v = beta)
conv.check(out1)


coef(out)                    # estimated   coefficients
round(out$c.coefficients, 3) # constrained coefficients
sum(out$c.coefficients)

round(out1$c.coefficients, 3)
sum(out1$c.coefficients)


# penalised estimation

out1 <- lmc(y, X, pen = "alasso", lambda = 0.02)
conv.check(out1)

coef(out1)
round(out1$c.coefficients, 3)
sum(out1$c.coefficients)


AIC(out, out1)
BIC(out, out1)

round(cbind(out$c.coefficients, out1$c.coefficients), 3)

# scad

n    <- 10000
beta <- c(0.2, 0, 0, 0.02, 0.01, 0.01, 0.01, 0.08, 0.21, 0.12, 0.15, 0.17, 0.02)
l    <- length(beta)
X    <- matrix(runif(n*l), n, l)

y    <- X%*%beta + rnorm(n)

out1 <- lmc(y, X, pen = "scad", lambda = 0.01)
conv.check(out1)

coef(out1)
sum(out1$c.coefficients)

round(cbind(beta, out1$c.coefficients), 2)


## End(Not run)
```

`logLik.SemiParBIV` *Extract the log likelihood for a fitted copula model*

Description

It extracts the log-likelihood for a fitted gjrm model.

Usage

```
## S3 method for class 'SemiParBIV'
logLik(object, ...)
```

Arguments

object	A fitted gjrm object.
...	Un-used for this function.

Details

Modification of the classic logLik which accounts for the estimated degrees of freedom used in gjrm. This function is provided so that information criteria work correctly by using the correct number of degrees of freedom.

Value

Standard logLik object.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

AIC, BIC

---

marg.mv *Marginal Mean/Variance*

Description

Function marg.mv can be used to calculate marginal means/variances, with corresponding interval obtained using posterior simulation.

Usage

```
marg.mv(x, eq, newdata, fun = "mean", n.sim = 100, prob.lev = 0.05,
        bin.model = NULL, bd = NULL)
```

Arguments

x	A fitted marg.mv object as produced by the respective fitting function.
eq	Margin of interest.
newdata	A data frame with one row, which must be provided.
fun	Either mean or variance.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters.
prob.lev	Overall probability of the left and right tails of the simulated distribution used for interval calculations.
bin.model	If a two part or hurdle model is used then this is the object of a binary regression model fitted using gam() from mgcv.
bd	Binomial denominator. To be supplied when using a distribution that requires it (e.g., BB).

Details

marg.mv() calculates the marginal mean or variance. Posterior simulation is used to obtain a confidence/credible interval.

Value

res	It returns three values: lower confidence interval limit, estimated marginal mean or variance and upper interval limit.
prob.lev	Probability level used.
sim.mv	It returns a vector containing simulated values of the marginal mean or variance. This is used to calculate intervals.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm

---

mb *Nonparametric (worst-case and IV) Manski's bounds*

Description

mb can be used to calculate the (worst-case and IV) Manski's bounds and confidence interval covering the true effect of interest with a fixed probability.

Usage

```
mb(treat, outc, IV = NULL, model, B = 100, sig.lev = 0.05)
```

Arguments

treat	Binary treatment/selection variable.
outc	Binary outcome variable.
IV	An instrumental binary variable can be used if available.
model	Possible values are "B" (model with endogenous variable) and "BSS" (model with non-random sample selection).
B	Number of bootstrap replicates. This is used to obtain some components needed for confidence interval calculations.
sig.lev	Significance level.

Details

Based on Manski (1990), this function returns the nonparametric lower and upper (worst-case) Manski's bounds for the average treatment effect (ATE) when `model = "B"` or prevalence when `model = "BSS"`. When an IV is employed the function returns IV Manski bounds.

For comparison, it also returns the estimated effect assuming random assignment (i.e., the treatment received or selection relies on the assumption of ignorable observed and unobserved selection). Note that this is equivalent to what provided by `ATE` or `prev` when `type = "naive"`, and is different from what obtained by `ATE` or `prev` when `type = "univariate"` as observed confounders are accounted for and the assumption here is of ignorable unobserved selection.

A confidence interval covering the true ATE/prevalence with a fixed probability is also provided. This is based on the approach described in Imbens and Manski (2004). NOTE that this interval is typically very close (if not identical) to the lower and upper bounds.

The ATE can be at most 1 (or 100 in percentage) and the worst-case Manski's bounds have width 1. This means that 0 is always included within the possibilites of these bounds. Nevertheless, this may be useful to check whether the effect from a bivariate recursive model is included within the possibilites of the bounds.

When estimating a prevalance the worst-case Manski's bounds have width equal to the non-response probability, which provides a measure of the uncertainty about the prevalence caused by non-response. Again, this may be useful to check whether the prevalence from a bivariate non-random sample selection model is included within the possibilites of the bounds.

See `gjrm` for some examples.

Value

LB, UP	Lower and upper bounds for the true effect of interest.
CI	Confidence interval covering the true effect of interest with a fixed probability.
ate.ra	Estimated effect of interest assuming random assignment.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Manski C.F. (1990), Nonparametric Bounds on Treatment Effects. *American Economic Review, Papers and Proceedings*, 80(2), 319-323.

Imbens G.W. and Manski C.F (2004), Confidence Intervals for Partially Identified Parameters. *Econometrica*, 72(6), 1845-1857.

See Also

gjrm

Examples

```
## see examples for gjrm
```

---

mc.copula.prob *Monte Carlo Copula Probabilities from a Fitted PCC Model*

Description

mc.copula.prob computes probabilities of user-defined events from a fitted PCC model using Monte Carlo simulation. Intervals are obtained via repeated posterior simulation of the model parameters.

Usage

```
mc.copula.prob(obj, y.vec, newdata, event.fun, joint = TRUE,
               n.draw = 50000, n.sim = 100, prob.lev = 0.05)
```

Arguments

obj	A fitted gjrm.pcc object as produced by the respective fitting function.
y.vec	A vector containing the threshold values used inside event.fun.
newdata	A data frame with one row, containing the covariate values at which the probability is to be evaluated.
event.fun	A user-supplied function of the form function(simY, y.vec) returning a logical vector indicating whether the event of interest occurred for each simulated draw.
joint	If TRUE then the calculation is done using the fitted joint model. If FALSE then the calculation is done from univariate fits.
n.draw	Number of Monte Carlo draws from the fitted model used to compute the point estimate.

n.sim	Number of posterior simulations used to compute interval estimates.
prob.lev	Overall probability mass in the two tails of the simulated distribution used to form the interval.

Details

The function first simulates n.draw draws from the fitted joint distribution with fixed parameters to obtain the point estimate of the probability.

It then performs n.sim posterior simulations of the model parameters, and for each draw computes the corresponding probability. The empirical distribution of these n.sim values is used to form interval estimates.

The event of interest is defined entirely by the user through event.fun, which receives the simulated responses and must return a logical vector. This allows arbitrary probabilities of events to be evaluated.

Value

A named numeric vector containing:

pr	The Monte Carlo estimate of the probability of the event.
2.5%	Lower interval bound based on the empirical bootstrap distribution.
97.5%	Upper interval bound based on the empirical bootstrap distribution.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm, copula.prob

---

numgh *Internal Function*

Description

This and other similar internal functions calculate numerical derivatives.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

OR *Causal odds ratio of a binary/continuous treatment variable*

Description

OR can be used to calculate the causal odds ratio of a binary/continuous treatment variable, with corresponding interval obtained using posterior simulation.

Usage

```
OR(x, trt, trt.val = NULL, int.var = NULL, joint = TRUE, n.sim = 100, prob.lev = 0.05,
   length.out = NULL)
```

Arguments

x	A fitted gjrm object.
trt	Name of the treatment variable.
trt.val	Numeric value for the treatment variable. This is only required when the endogenous variable is Gaussian.
int.var	A vector made up of the name of the variable interacted with trt, and a value for it. It can also be a list.
joint	If FALSE then the effect is obtained from the univariate model which neglects the presence of unobserved confounders. When TRUE, the effect is obtained from the simultaneous model which accounts for observed and unobserved confounders.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used when delta = FALSE. It may be increased if more precision is required.
prob.lev	Overall probability of the left and right tails of the OR distribution used for interval calculations.
length.out	Ddesired length of the sequence to be used when calculating the effect that a continuous treatment has on a binary outcome.

Details

OR calculates the causal odds ratio for a binary/continuous Gaussian treatment. Posterior simulation is used to obtain a confidence/credible interval.

Value

prob.lev	Probability level used.
sim.OR	It returns a vector containing simulated values of the average OR. This is used to calculate intervals.
Ratios	For the case of continuous endogenous treatment and binary outcome, it returns a matrix made up of three columns containing the odds ratios for each incremental value in the endogenous variable and respective intervals.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm

PE *Partial effect from a binary bivariate model*

Description

PE can be used to calculate the sample treatment effect from a a binary bivariate model, with corresponding interval obtained using posterior simulation.

Usage

```
PE(x1, idx, n.sim = 100, prob.lev = 0.05,
   plot = FALSE,
   main = "Histogram of Simulated Average Effects",
   xlab = "Simulated Average Effects", ...)
```

Arguments

x1	A fitted gjrm object.
idx	This is useful to pick a particular individual and must be provided.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used when delta = FALSE. It may be increased if more precision is required.
prob.lev	Overall probability of the left and right tails of the AT distribution used for interval calculations.
plot	If TRUE then a plot of the histogram and kernel density estimate of the simulated average effects is produced.
main	Title for the plot.
xlab	Title for the x axis.
...	Other graphics parameters to pass on to plotting commands. These are used only when hd.plot = TRUE.

Details

PE measures the sample average effect from a binary bivariate model when a binary response (associated with a continuous outcome) takes values 0 and 1. Posterior simulation is used to obtain a confidence/credible interval.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm

---

pen *Internal Function*

Description

It provides an overall penalty matrix in a format suitable for estimation conditional on smoothing parameters.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

plot.SemiParBIV *Plotting function*

Description

It takes a fitted gjrm object produced by gjrm() and plots the estimated smooth functions on the scale of the linear predictors. This function is a wrapper of plot.gam() in mgcv. Please see the documentation of plot.gam() for full details.

Usage

```
## S3 method for class 'SemiParBIV'
plot(x, eq, ...)
```

Arguments

x	A fitted gjrm object.
eq	The equation from which smooth terms should be considered for printing.
...	Other graphics parameters to pass on to plotting commands, as described for plot.gam() in mgcv.

Details

This function produces plots showing the smooth terms of a fitted semiparametric bivariate probit model. In the case of 1-D smooths, the x axis of each plot is labelled using the name of the regressor, while the y axis is labelled as `s(regr, edf)` where `regr` is the regressor's name, and `edf` the effective degrees of freedom of the smooth. For 2-D smooths, perspective plots are produced with the x axes labelled with the first and second variable names and the y axis is labelled as `s(var1, var2, edf)`, which indicates the variables of which the term is a function and the `edf` for the term.

If `seWithMean = TRUE` then the intervals include the uncertainty about the overall mean. Note that the smooths are still shown centred. The theoretical arguments and simulation study of Marra and Wood (2012) suggest that `seWithMean = TRUE` results in intervals with close to nominal frequentist coverage probabilities.

Value

The function generates plots.

WARNING

The function can not deal with smooths of more than 2 variables.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Marra G. and Wood S.N. (2012), Coverage Properties of Confidence Intervals for Generalized Additive Model Components. *Scandinavian Journal of Statistics*, 39(1), 53-74.

See Also

`gjrm`

`polys.map` *Geographic map with regions defined as polygons*

Description

This function produces a map with geographic regions defined by polygons. It is essentially the same function as `polys.plot()` in `mgcv` but with added arguments `zlim` and `rev.col` and a wider set of choices for `scheme`.

Usage

```
polys.map(lm, z, scheme = "gray", lab = "", zlim, rev.col = FALSE, ...)
```

Arguments

lm	Named list of matrices where each matrix has two columns. The matrix rows each define the vertex of a boundary polygon.
z	A vector of values associated with each area (item) of lm.
scheme	Possible values are "heat", "terrain", "topo", "cm" and "gray", indicating how to fill the polygons in accordance with the value of z.
lab	label for plot.
zlim	If missing then the range of z will be chosen using pretty(z) otherwise the range provided will be used.
rev.col	If TRUE then coloring scheme is reversed.
...	other arguments to pass to plot.

Details

See help file of polys.plot in mgcv.

Value

It produces a plot.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

polys.setup *Set up geographic polygons*

Description

This function creates geographic polygons in a format suitable for smoothing.

Usage

```
polys.setup(object)
```

Arguments

object	An RDS file object as extracted from http://www.gadm.org.

Value

It produces a list with polygons (polys), and various names (names0, names1 - first level of aggregation, names2 - second level of aggregation).

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Thanks to Guy Harling for suggesting the implementation of this function.

Examples

```
?hiv
```

---

pred.gp *Function to predict quantiles from GP and DGP distributions*

Description

It takes a fitted gamlss object produced by gamlss() and produces the desired quntities and respective intervals.

Usage

```
pred.gp(x, p = 0.5, newdata, n.sim = 100, prob.lev = 0.05)
```

Arguments

x	A fitted gamlss object.
p	Value of p.
newdata	A data frame or list containing the values of the model covariates at which predictions are required. If not provided then predictions corresponding to the original data are returned. When newdata is provided, it should contain all the variables needed for prediction.
n.sim	The number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used to calculate intervals. It may be increased if more precision is required.
prob.lev	Probability of the left and right tails of the posterior distribution used for interval calculations.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gamlss

predict.CopulaCLM *Prediction function*

Description

It takes a fitted gjrm object for the ordinal-continuous case and, for each equation, produces predictions for a new set of values of the model covariates or the original values used for the model fit. Standard errors of predictions can be produced and are based on the posterior distribution of the model coefficients.

Usage

```
## S3 method for class 'CopulaCLM'
predict(object, eq, type = "link", ...)
```

Arguments

object	A fitted gjrm object.
eq	The equation to be considered for prediction.
type	Type of prediction.
...	Other arguments as in predict.gam() in mgcv.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gjrm

predict.SemiParBIV *Prediction function*

Description

It takes a fitted gjrm object and, for each equation, produces predictions for a new set of values of the model covariates or the original values used for the model fit. Standard errors of predictions can be produced and are based on the posterior distribution of the model coefficients. This function is a wrapper for predict.gam() in mgcv. Please see the documentation of predict.gam() for full details.

Usage

```
## S3 method for class 'SemiParBIV'
predict(object, eq, ...)
```

Arguments

object	A fitted gjrm object.
eq	The equation to be considered for prediction.
...	Other arguments as in predict.gam() in mgcv.

WARNINGS

When type = "response" this function will provide prediction assuming that the identity link function is adopted. This means that type = "link" and type = "response" will produce the same results, which for some distributions is fine. This is because, for internal reasons, the model object used always assumes an identity link. There are other functions in the package which will produce predictions for the response correctly and we are currently working on extending them to all models in the package. For all the other type values the function will produce the correct results.

When predicting based on a new data set, this function can not return correct predictions for models based on a copula value of "C0C90", "C0C270", "C180C90", "C180C270", "G0G90", "G0G270", "G180G90", "G180G270", "J0J90", "J0J270", "J180J90" or "J180J270".

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gjrm

---

prev *Estimated overall prevalence from a sample selection model*

Description

prev can be used to calculate the overall estimated prevalence from a sample selection model with binay outcome, with corresponding interval obtained using posterior simulation.

Usage

```
prev(x, sw = NULL, joint = TRUE, n.sim = 100, prob.lev = 0.05)
```

Arguments

x	A fitted gjrm object.
sw	Survey weights.
joint	If FALSE then the prevalence is obtained from the univariate model which neglects the presence of unobserved confounders. When TRUE, the prevalence is obtained from the simultaneous model which accounts for observed and unobserved confounders.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. It may be increased if more precision is required.
prob.lev	Overall probability of the left and right tails of the prevalence distribution used for interval calculations.

Details

prev estimates the overall prevalence of a disease (e.g., HIV) when there are missing values that are not at random. An interval for the estimated prevalence can be obtained using posterior simulation.

Value

res	It returns three values: lower confidence interval limit, estimated prevalence and upper confidence interval limit.
prob.lev	Probability level used.
sim.prev	Vector containing simulated values of the prevalence. This is used to calculate an interval.

Author(s)

Authors: Giampiero Marra, Rosalba Radice, Guy Harling, Mark E McGovern

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Marra G., Radice R., Barnighausen T., Wood S.N. and McGovern M.E. (2017), A Simultaneous Equation Approach to Estimating HIV Prevalence with Non-Ignorable Missing Responses. *Journal of the American Statistical Association*, 112(518), 484-496.

See Also

GJRM-package, gjrm

print.ATE *Print an ATE object*

Description

The print method for an ATE object.

Usage

```
## S3 method for class 'ATE'
print(x, ...)
```

Arguments

x	ATE object produced by ATE().
...	Other arguments.

Details

print.ATE prints the lower confidence interval limit, estimated ATE and upper confidence interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

ATE

print.cond.mv *Print a cond.mv object*

Description

The print method for a cond.mv object.

Usage

```
## S3 method for class 'cond.mv'
print(x, ...)
```

Arguments

x	cond.mv object produced by cond.mv().
...	Other arguments.

Details

print.cond.mv prints the lower confidence interval limit, estimated conditonal mean or variance and upper confidence interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

cond.mv

---

print.copulaSampleSel *Print a copulaSampleSel object*

Description

The print method for a copulaSampleSel object.

Usage

```
## S3 method for class 'copulaSampleSel'
print(x, ...)
```

Arguments

x	copulaSampleSel object.
...	Other arguments.

Details

It prints out the family, model equations, total number of observations, estimated association coefficient, etc for the penalized or unpenalized model.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

print.gamlss *Print a gamlss object*

Description

The print method for a gamlss object.

Usage

```
## S3 method for class 'gamlss'
print(x, ...)
```

Arguments

x	gamlss object produced by gamlss().
...	Other arguments.

Details

print.gamlss prints out the family, model equations, total number of observations, etc for the penalized or unpenalized model.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gamlss

print.gjrm *Print a gjrm/gjrm.pcc object*

Description

The print method for a gjrm or gjrm.pcc object.

Usage

```
## S3 method for class 'gjrm'
print(x, ...)
## S3 method for class 'gjrm.pcc'
print(x, ...)
```

Arguments

x	gjrm/gjrm.pcc object produced by gjrm()/gjrm.pcc().
...	Other arguments.

Details

print.gjrm/print.gjrm.pcc prints out the family, model equations, total number of observations, estimated association coefficient, etc for the penalized or unpenalized model.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gjrm, gjrm.pcc

---

print.marg.mv *Print a marg.mv object*

Description

The print method for a marg.mv object.

Usage

```
## S3 method for class 'marg.mv'
print(x, ...)
```

Arguments

x	marg.mv object produced by marg.mv().
...	Other arguments.

Details

print.marg.mv prints the lower confidence interval limit, estimated conditonal mean or variance and upper confidence interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

marg.mv

---

print.mb *Print an mb object*

Description

The print method for an mb object.

Usage

```
## S3 method for class 'mb'
print(x, ...)
```

Arguments

x	mb object produced by mb().
...	Other arguments.

Details

print.mb prints the lower and upper bounds, confidence interval, and effect assuming random assignment.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

mb

print.OR *Print an OR object*

Description

The print method for an OR object.

Usage

```
## S3 method for class 'OR'
print(x, ...)
```

Arguments

x	OR object produced by OR().
...	Other arguments.

Details

print.OR prints the lower confidence interval limit, estimated OR and upper confidence interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

OR

print.PE *Print an PE object*

Description

The print method for an PE object.

Usage

```
## S3 method for class 'PE'
print(x, ...)
```

Arguments

x	PE object produced by PE().
...	Other arguments.

Details

print.PE prints the lower confidence interval limit, estimated PE and upper confidence interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

PE

---

print.prev *Print an prev object*

Description

The print method for an prev object.

Usage

```
## S3 method for class 'prev'
print(x, ...)
```

Arguments

x	prev object produced by prev().
...	Other arguments.

Details

print.prev prints the lower interval limit, estimated prevalence and upper interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

prev

print.RR *Print an RR object*

Description

The print method for an RR object.

Usage

```
## S3 method for class 'RR'
print(x, ...)
```

Arguments

x	RR object produced by RR().
...	Other arguments.

Details

print.RR prints the lower confidence interval limit, estimated RR and upper confidence interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

RR

print.SATE *Print an SATE object*

Description

The print method for a SATE object.

Usage

```
## S3 method for class 'SATE'
print(x, ...)
```

Arguments

x	SATE object produced by SATE().
...	Other arguments.

Details

print.SATE prints the lower confidence interval limit, estimated SATE and upper confidence interval limit.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

SATE

---

print.SemiParBIV *Print a SemiParBIV object*

Description

The print method for a SemiParBIV object.

Usage

```
## S3 method for class 'SemiParBIV'
print(x, ...)
```

Arguments

x	SemiParBIV object.
...	Other arguments.

Details

It prints out the family, model equations, total number of observations, estimated association coefficient and total effective degrees of freedom for the penalized or unpenalized model.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

print.SemiParROY *Print a SemiParROY object*

Description

The print method for a SemiParROY object.

Usage

```
## S3 method for class 'SemiParROY'
print(x, ...)
```

Arguments

x	SemiParROY object.
...	Other arguments.

Details

It prints out the family, model equations, total number of observations, estimated association coefficient, etc for the penalized or unpenalized model.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

print.SemiParTRIV *Print a SemiParTRIV object*

Description

The print method for a SemiParTRIV object.

Usage

```
## S3 method for class 'SemiParTRIV'
print(x, ...)
```

Arguments

x	SemiParTRIV object.
...	Other arguments.

Details

It prints out the family, model equations, total number of observations, estimated association coefficient and total effective degrees of freedom for the penalized or unpenalized model.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

probm *Internal Function*

Description

Internal fitting function.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

regH *Internal Function*

Description

It applies one of two regularisations on the information matrix if desired. These are based on the Cholesky and eigen decompositions.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

res.check *Diagnostic plots for discrete/continuous response margins*

Description

It produces diagnostic plots based on (randomised) quantile residuals.

Usage

```
res.check(x, joint = FALSE, intervals = FALSE, n.sim = 1000, prob.lev = 0.05)
```

Arguments

x	A fitted gjrm object.
joint	If TRUE then a qqplot for overall model adequacy is produced, otherwise qqplots for the marginal models are displayed.
intervals	If TRUE then intervals for the qqplots are produced.
n.sim	Number of simulated variates from the assumed distribution.
prob.lev	Overall probability of the left and right tails of the probabilities' distributions used for interval calculations.

Details

If the model fits the response well then the plots should look normally distributed. When fitting models with discrete and/or continuous margins, four plots will be produced. In this case, the arguments main2 and xlab2 come into play and allow for different labelling across the plots.

Value

qr	It returns the (randomised) quantile residuals for the continuous or discrete margin when fitting a model that involves a binary response.
qr1	As above but for first equation (this applies when fitting models with continuous/discrete margins).
qr2	As above but for second equation.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gjrm

resp.check *Diagnostic plot for a variable*

Description

It produces a normal Q-Q plot for the (randomised) normalised quantile response. It also provides the log-likelihood for AIC calculation, for instance. It is also used for internal purposes.

Usage

```
resp.check(y, margin = "N", print.par = FALSE, plots = TRUE,
           loglik = FALSE, os = FALSE, i.f = FALSE,
           min.dn = 1e-40, min.pr = 1e-16, max.pr = 0.999999,
           left.trunc = 0)
```

Arguments

y	Response.
margin	The distributions allowed are: normal ("N"), log-normal ("LN"), generelised Pareto ("GP"), discrete generelised Pareto ("DGP"), Gumbel ("GU"), reverse Gumbel ("rGU"), logistic ("LO"), Weibull ("WEI"), inverse Gaussian ("iG"), gamma ("GA"), Dagum ("DAGUM"), Singh-Maddala ("SM"), beta ("BE"), Fisk ("FISK"), Poisson ("P"), zero truncated Poisson ("ZTP"), negative binomial - type I ("NBI"), negative binomial - type II ("NBII"), Poisson inverse Gaussian ("PIG").
print.par	If TRUE then the estimated parameters used to construct the plot are returned.
plots	If FALSE then no plot is produced and only parameter estimates returned.
loglik	If TRUE then it returns the logLik.
os	If TRUE then the estimated parameters are returned on the original scale.
i.f	Internal fitting option. This is not for user purposes.
min.dn, min.pr, max.pr	Allowed minimum and maximum for estimated probabities and densities for parameter estimation.
left.trunc	Value of truncation at left. Currently done for count distributions only.

Details

Prior to fitting a model with discrete and/or continuous margins, the distributions for the outcome variables may be chosen by checking the normalised quantile responses. These will provide a rough guide to the adequacy of the chosen distribution. The latter are defined as the quantile standard normal function of the cumulative distribution function of the response with scale and location estimated by MLE. These should behave approximately as normally distributed variables (even though the original observations are not). Therefore, a normal Q-Q plot is appropriate here.

If loglik = TRUE then this function also provides the log-likelihood for AIC calculation, for instance.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gjrm

---

rMVN *Multivariate Normal Variates*

Description

This function simply generates random multivariate normal variates.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

---

rob.const *Bootstrap procedure to help select the robust constant in a GAMLSS*

Description

It helps finding the robust constant for a GAMLSS.

Usage

```
rob.const(x, B = 100, left.trunc = 0)
```

Arguments

x	A fitted gjrm object.
B	Number of bootstrap replicates.
left.trunc	If a truncated count distribution is employed then this is the left truncation point.

Details

It helps finding the robust constant for a GAMLSS based on the mean or median.

Value

rc	Robust constant used in fitting.
sw	Sum of weights for each bootstrap replicate.
m1	Mean.
m2	Median.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gamlss

---

rob.int *Tool for tuning bounds of integral in robust models*

Description

Tool for tuning bounds of integral in robust GAMLSS with continuous distribution.

Usage

```
rob.int(x, rc, l.grid = 1000, tol = 1e-4, var.range = NULL)
```

Arguments

x	A fitted gjrm object, typically from a non-robust fit.
rc	Robust tuning constant.
l.grid	Length of grid.
tol	Tolerance
var.range	Range of values, min and max, to use in calculations.

Details

Tool for tuning bounds of integral in robust GAMLSS.

Value

lB, uB	Lower and upper bounds.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gamlss

rpcc *Simulation from a Fitted Trivariate PCC Copula Model*

Description

rpcc generates Monte Carlo draws from a fitted trivariate pair-copula construction (PCC) model estimated via gjrm.pcc. The function simulates the three response variables jointly, using the fitted marginal distributions and the estimated PCC copula structure. Posterior simulation of model parameters is available for uncertainty propagation.

Usage

```
rpcc(obj, newdata, n.draw = 1000, sim.par = FALSE, independent = FALSE)
```

Arguments

obj	A fitted gjrm.pcc object containing a trivariate PCC model.
newdata	A data frame with exactly one row, containing the covariate values at which the simulation is to be performed.
n.draw	Number of Monte Carlo draws to generate. Must be greater than 2.
sim.par	If TRUE, model parameters are drawn from their posterior distribution before simulation, allowing uncertainty in the fitted model to be propagated. If FALSE, parameters are fixed at their estimated values.
independent	If TRUE, the three margins are simulated independently using uniform random variables. If FALSE, dependence is generated using the fitted PCC copula structure.

Details

The function returns a data frame with simulated values for the three response variables. It is primarily intended for use inside higher-level Monte Carlo procedures such as mc.copula.prob.

Value

A data frame with three columns:

Y1	Simulated values for the first response.
Y2	Simulated values for the second response.
Y3	Simulated values for the third response.

Each column contains n.sim simulated draws.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

mc.copula.prob, copula.prob, gjrm

RR *Causal risk ratio of a binary/continuous treatment variable*

Description

RR can be used to calculate the causal risk ratio of a binary/continuous treatment variable, with corresponding interval obtained using posterior simulation.

Usage

```
RR(x, trt, trt.val = NULL, int.var = NULL, joint = TRUE, n.sim = 100, prob.lev = 0.05,
   length.out = NULL)
```

Arguments

x	A fitted gjrm object.
trt	Name of the treatment variable.
trt.val	Numeric value for the treatment variable. This is only required when the endogenous variable is Gaussian.
int.var	A vector made up of the name of the variable interacted with trt, and a value for it. It can also be a list.
joint	If FALSE then the effect is obtained from the univariate model which neglects the presence of unobserved confounders. When TRUE, the effect is obtained from the simultaneous model which accounts for observed and unobserved confounders.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used when delta = FALSE. It may be increased if more precision is required.
prob.lev	Overall probability of the left and right tails of the RR distribution used for interval calculations.
length.out	Ddesired length of the sequence to be used when calculating the effect that a continuous treatment has on a binary outcome.

Details

RR calculates the causal risk ratio of the probabilities of positive outcome under treatment (the binary predictor or treatment assumes value 1) and under control (the binary treatment assumes value 0). Posterior simulation is used to obtain a confidence/credible interval.

RR works also for the case of continuous Gaussian endogenous treatment variable.

Value

prob.lev	Probability level used.
sim.RR	It returns a vector containing simulated values of the average RR. This is used to calculate intervals.
Ratios	For the case of continuous endogenous variable and binary outcome, it returns a matrix made up of three columns containing the risk ratios for each incremental value in the endogenous variable and respective intervals.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm

S.m *Internal Function*

Description

It provides penalty matrices in a format suitable for automatic multiple smoothing parameter estimation.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

SATE *Survival Average Treatment Effects of a binary treatment variable*

Description

SATE can be used to calculate the survival treatment effects of a binary treatment variable, with corresponding interval obtained using posterior simulation.

Usage

```
SATE(x, trt, surv.t = NULL, int.var = NULL, joint = TRUE,
     n.sim = 100, prob.lev = 0.05, ls = 10, plot.type = "none", ...)
```

Arguments

x	A fitted gjrm object as produced by the respective fitting function.
trt	Name of the treatment variable.
surv.t	Numeric value or vector for time. If not provided, the function will be calculate the SATE for each time point of a grid of lenght ls, calculated from the observed outcome.
int.var	A vector made up of the name of the variable interacted with trt, and a value for it. It can also be a list.
joint	If FALSE then the effects are obtained from the univariate model which neglects the presence of unobserved confounders. When TRUE, the effects are obtained from the simultaneous model which accounts for observed and unobserved confounders.
n.sim	Number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. It may be increased if more precision is required.
prob.lev	Overall probability of the left and right tails of the SAT distribution used for interval calculations.
ls	Length of sequence to use for time variable. Only used when surv.t = NULL.
plot.type	Values allowed are: "none", "survival" (for survival plots under treatment = 0 (grey lines) and treatment = 1 (black lines)) and "sate" (SATE evaluated at several time points).
...	Other graphics parameters to pass on to plotting commands.

Details

SATE measures the average survival difference in outcomes under treatment (the binary predictor or treatment assumes value 1) and under control (the binary treatment assumes value 0). Posterior simulation is used to obtain a confidence/credible interval.

Value

res	It returns three values: lower interval limit(s), estimated SATE(s) and upper interval limit(s).
prob.lev	Probability level used.
sim.SATE	It returns a vector containing simulated values of the survival average treatment effect for the case in which a specific time is chosen. This is used to calculate intervals.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

GJRM-package, gjrm

SemiParBIV *Internal fitting function*

Description

Internal fitting set up function.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

SemiParBIV.fit *Internal Function*

Description

Wrapper of core algorithm.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

SemiParBIV.fit.post *Internal Function*

Description

This and other similar internal functions calculate useful post estimation quantities.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

SemiParROY *Internal fitting function*

Description

Internal fitting set up function.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

SemiParTRIV *Internal fitting function*

Description

Internal fitting set up function.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

summary.copulaSampleSel *copulaSampleSel summary*

Description

It takes a fitted copulaSampleSel object and produces some summaries from it.

Usage

```
## S3 method for class 'copulaSampleSel'
summary(object, n.sim = 100, prob.lev = 0.05, ...)

## S3 method for class 'summary.copulaSampleSel'
print(x, digits = max(3, getOption("digits") - 3),
           signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	A fitted copulaSampleSel object.
x	summary.copulaSampleSel object produced by summary.copulaSampleSel().
n.sim	The number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used to calculate intervals for the association parameter, dispersion coefficient, for instance It may be increased if more precision is required.
prob.lev	Probability of the left and right tails of the posterior distribution used for interval calculations.
digits	Number of digits printed in output.
signif.stars	By default significance stars are printed alongside output.
...	Other arguments.

Details

print.summary.copulaSampleSel prints model term summaries.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Examples

```
## see examples for gjrm
```

---

summary.gamlss *gamlss summary*

Description

It takes a fitted gamlss object and produces some summaries from it.

Usage

```
## S3 method for class 'gamlss'
summary(object, n.sim = 100, prob.lev = 0.05, ...)



## S3 method for class 'summary.gamlss'
print(x, digits = max(3, getOption("digits") - 3),
           signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	A fitted gamlss object.
x	summary.gamlss object produced by summary.gamlss().
n.sim	The number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used to calculate intervals for various parameters. It may be increased if more precision is required.
prob.lev	Probability of the left and right tails of the posterior distribution used for interval calculations.
digits	Number of digits printed in output.
signif.stars	By default significance stars are printed alongside output.
...	Other arguments.

Details

print.summary.gamlss prints model term summaries.

Value

tableP1	Table containing parametric estimates, their standard errors, z-values and p-values for equation 1.
tableP2, tableP3	As above but for equations 2 and 3 if present.
tableNP1	Table of nonparametric summaries for each smooth component including effective degrees of freedom, estimated rank, approximate Wald statistic for testing the null hypothesis that the smooth term is zero and corresponding p-value, for equation 1.
tableNP2, tableNP3	As above but for equations 2 and 3.
n	Sample size.
sigma, nu	Estimated distribution specific parameters.
formula1, formula2, formula3	Formulas used for the model equations.
l.sp1, l.sp2, l.sp3	Number of smooth components in model equation.
t.edf	Total degrees of freedom of the estimated bivariate model.
CIsig, CInu	Intervals for distribution specific parameters.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Examples

```
## see examples for gamlss
```

---

summary.gjrm *gjrm summary*

Description

It takes a fitted gjrm object and produces some summaries from it.

Usage

```
## S3 method for class 'gjrm'
summary(object, n.sim = 100, prob.lev = 0.05, ...)



## S3 method for class 'summary.gjrm'
print(x, digits = max(3, getOption("digits") - 3),
           signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	A fitted gjrm object.
x	summary.gjrm object produced by summary.gjrm().
n.sim	The number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used to calculate intervals for the association parameter, dispersion coefficient etc. It may be increased if more precision is required.
prob.lev	Probability of the left and right tails of the posterior distribution used for interval calculations.
digits	Number of digits printed in output.
signif.stars	By default significance stars are printed alongside output.
...	Other arguments.

Details

print.summary.gjrm prints model term summaries.

Value

tableP1	Table containing parametric estimates, their standard errors, z-values and p-values for equation 1.
tableP2, tableP3, ...	As above but for equation 2 and equations 3 and 4 if present.
tableNP1	Table of nonparametric summaries for each smooth component including effective degrees of freedom, estimated rank, approximate Wald statistic for testing the null hypothesis that the smooth term is zero and corresponding p-value, for equation 1.
tableNP2, tableNP3, ...	As above but for equation 2 and equations 3 and 4 if present.
n	Sample size.
theta	Estimated dependence parameter linking the two equations.
sigma1, sigma2	Estimated distribution specific parameters for equations 1 and 2.
nu1, nu2	Estimated distribution specific parameters for equations 1 and 2.
formula1, formula2, formula3, ...	Formulas used for the model equations.
l.sp1, l.sp2, l.sp3, ...	Number of smooth components in model equations.
t.edf	Total degrees of freedom of the estimated bivariate model.
CItheta	Interval(s) for θ.
CIsig1, CIsig2, CInu1, CInu2	Intervals for distribution specific parameters

WARNINGS

Note that the Kendall's tau (and related interval), as implemented here, is a valid measure of dependence for continuous margins and it will only provide a crude indication of dependence in other cases.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

`summary.gjrm.pcc` *Summary Method for* `gjrm.pcc` *Objects*

Description

Produces summaries for pair-copula components of a fitted `gjrm.pcc` object. The method extracts and displays information for each estimated pair-copula in the PCC decomposition, including conditional copulas.

Usage

```
## S3 method for class 'gjrm.pcc'
summary(object, ...)

## S3 method for class 'summary.gjrm.pcc'
print(x, ...)
```

Arguments

`object`	A fitted `gjrm.pcc` object.
`x`	An object of class `summary.gjrm.pcc` produced by `summary.gjrm.pcc()`.
`...`	Additional arguments passed to other methods.

Details

The function `summary.gjrm.pcc` returns the object unchanged but assigns it the class `"summary.gjrm.pcc"` so that the corresponding print method `print.summary.gjrm.pcc` is dispatched.

The print method displays each estimated pair-copula component in a structured format, using labelled blocks such as `C12`, `C23` and `C13|2`.

This method is intended to provide a clear and interpretable summary of the dependence structure in the fitted PCC model.

Value

An object of class `summary.gjrm.pcc`, which is identical to the input `gjrm.pcc` object but with an additional class attribute to enable custom printing.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

See Also

gjrm.pcc, print.gjrm.pcc

---

summary.SemiParBIV *SemiParBIV summary*

Description

It takes a fitted SemiParBIV object and produces some summaries from it.

Usage

```
## S3 method for class 'SemiParBIV'
summary(object, n.sim = 100, prob.lev = 0.05, gm = FALSE, ...)

## S3 method for class 'summary.SemiParBIV'
print(x, digits = max(3, getOption("digits") - 3),
           signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	A fitted SemiParBIV object.
x	summary.SemiParBIV object produced by summary.SemiParBIV().
n.sim	The number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used to calculate intervals for the association parameter, dispersion coefficient and other measures (e.g., gamma measure). It may be increased if more precision is required.
prob.lev	Probability of the left and right tails of the posterior distribution used for interval calculations.
gm	If TRUE then intervals for the gamma measure and odds ratio are calculated.
digits	Number of digits printed in output.
signif.stars	By default significance stars are printed alongside output.
...	Other arguments.

Details

Using some low level functions in mgcv, based on the results of Marra and Wood (2012), 'Bayesian p-values' are returned for the smooth terms. These have better frequentist performance than their frequentist counterpart. See the help file of summary.gam in mgcv for further details. Covariate selection can also be achieved using a single penalty shrinkage approach as shown in Marra and Wood (2011).

Posterior simulation is used to obtain intervals of nonlinear functions of parameters, such as the association and dispersion parameters as well as the odds ratio and gamma measure discussed by Tajar et al. (2001) if gm = TRUE.

print.summary.SemiParBIV prints model term summaries.

Value

tableP1 — Table containing parametric estimates, their standard errors, z-values and p-values for equation 1.

tableP2, tableP3, ...
: As above but for equation 2 and equations 3 and 4 if present.

tableNP1 — Table of nonparametric summaries for each smooth component including effective degrees of freedom, estimated rank, approximate Wald statistic for testing the null hypothesis that the smooth term is zero and corresponding p-value, for equation 1.

tableNP2, tableNP3, ...
: As above but for equation 2 and equations 3 and 4 if present.

n — Sample size.

theta — Estimated dependence parameter linking the two equations.

formula1, formula2, formula3, ...
: Formulas used for the model equations.

l.sp1, l.sp2, l.sp3, ...
: Number of smooth components in model equations.

t.edf — Total degrees of freedom of the estimated bivariate model.

CItheta — Interval(s) for θ.

n.sel — Number of selected observations in the sample selection case.

OR, CIor — Odds ratio and related CI. The odds ratio is a measure of association between binary random variables and is defined as p00p11/p10p01. In the case of independence this ratio is equal to 1. It can take values in the range (-Inf, Inf) and it does not depend on the marginal probabilities (Tajar et al., 2001). Interval is calculated using posterior simulation.

GM, CIgm — Gamma measure and related CI. This measure of association was proposed by Goodman and Kruskal (1954). It is defined as (OR - 1)/(OR + 1), can take values in the range (-1, 1) and does not depend on the marginal probabilities. Interval is calculated using posterior simulation.

WARNINGS

Note that the Kendall's tau (and related interval), as implemented here, is a valid measure of dependence for continuous margins and it will only provide a crude indication of dependence in other cases.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Marra G. and Wood S.N. (2011), Practical Variable Selection for Generalized Additive Models. *Computational Statistics and Data Analysis*, 55(7), 2372-2387.

Marra G. and Wood S.N. (2012), Coverage Properties of Confidence Intervals for Generalized Additive Model Components. *Scandinavian Journal of Statistics*, 39(1), 53-74.

Tajar M., Denuit M. and Lambert P. (2001), Copula-Type Representation for Random Couples with Bernoulli Margins. Discussion Papaer 0118, Universite Catholique De Louvain.

See Also

ATE, prev

`summary.SemiParROY` *SemiParROY summary*

Description

It takes a fitted `SemiParROY` object and produces some summaries from it.

Usage

```
## S3 method for class 'SemiParROY'
summary(object, n.sim = 100, prob.lev = 0.05, ...)

## S3 method for class 'summary.SemiParROY'
print(x, digits = max(3, getOption("digits") - 3),
           signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

`object`	A fitted `SemiParROY` object.
`x`	`summary.SemiParROY` object produced by `summary.SemiParROY()`.
`n.sim`	The number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used to calculate intervals for the association parameter, dispersion coefficient, for instance It may be increased if more precision is required.

prob.lev	Probability of the left and right tails of the posterior distribution used for interval calculations.
digits	Number of digits printed in output.
signif.stars	By default significance stars are printed alongside output.
...	Other arguments.

Details

print.summary.SemiParROY prints model term summaries.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Examples

```
## see examples for gjrm
```

---

summary.SemiParTRIV *SemiParTRIV summary*

Description

It takes a fitted SemiParTRIV object and produces some summaries from it.

Usage

```
## S3 method for class 'SemiParTRIV'
summary(object, n.sim = 100, prob.lev = 0.05, ...)

## S3 method for class 'summary.SemiParTRIV'
print(x, digits = max(3, getOption("digits") - 3),
           signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	A fitted SemiParTRIV object.
x	summary.SemiParTRIV object produced by summary.SemiParTRIV().
n.sim	The number of simulated coefficient vectors from the posterior distribution of the estimated model parameters. This is used to calculate intervals for the association parameter and other measures. It may be increased if more precision is required.
prob.lev	Probability of the left and right tails of the posterior distribution used for interval calculations.
digits	Number of digits printed in output.
signif.stars	By default significance stars are printed alongside output.
...	Other arguments.

Details

`print.summary.SemiParTRIV` prints model term summaries.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Examples

```
## see examples for gjrm
```

`TRIapprox` *Internal Function*

Description

It approximates the trivariate normal integral.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

`triprobgHs` *Internal Function*

Description

It provides score and Hessian for trivariate binary models.

Author(s)

Author: Panagiota Filippou

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

vuong.test *Vuong test*

Description

The Vuong test is likelihood-ratio-based tests that can be used for choosing between two non-nested models.

Usage

```
vuong.test(obj1, obj2, sig.lev = 0.05)
```

Arguments

obj1, obj2	Objects of the two fitted bivariate non-nested models.
sig.lev	Significance level used for testing.

Details

The Vuong test is a likelihood-ratio-based tests for model selection that use the Kullback-Leibler information criterion, and that can be employed for choosing between two bivariate models which are non-nested.

The null hypothesis is that the two models are equally close to the actual model, whereas the alternative is that one model is closer. The test follows asymptotically a standard normal distribution under the null. Assume that the critical region is $(-c, c)$, where c is typically set to 1.96. If the value of the test is higher than c then we reject the null hypothesis that the models are equivalent in favor of model obj1. Viceversa if the value is smaller than c. If the value falls in $[-c, c]$ then we cannot discriminate between the two competing models given the data.

Value

It returns a decision.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

References

Vuong Q.H. (1989), Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses. *Econometrica*, 57(2), 307-333.

Examples

```
## see examples for gjrm
```

working.comp *Internal Function*

Description

It efficiently calculates the working model quantities needed to implement the automatic multiple smoothing parameter estimation procedure by exploiting a result which leads to very fast and stable calculations.

Author(s)

Maintainer: Giampiero Marra <giampiero.marra@ucl.ac.uk>

Index