Package 'GJRM'

June 21, 2023

Type Package

Version 0.2-6.4

Author Giampiero Marra <giampiero.marra@ucl.ac.uk> and Rosalba Radice <rosalba.radice@city.ac.uk>

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

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, endogeneity, non-random sample selection or partial observability.

Depends R (>= 3.2.1), mgcv

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

Enhances sp

LazyLoad yes

License GPL (>= 2)

URL https://www.ucl.ac.uk/statistics/people/giampieromarra

Repository CRAN

Date/Publication 2023-06-21 18:50:02 UTC

NeedsCompilation no

R topics documented:

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 (survival models with mixed censoring and excess hazards), Kiron Das (Galambos copula), Eva Cantoni and William Aeberhard (robust gamlss), Alessia Eletti and Danilo Petti (copula survival model with general censoring scheme).

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

Submitted papers (developments available from release 0.2-5):

Marra G., Radice R., Zimmer D. (2022), A Unifying Switching Regime Regression Framework with Applications in Health Economics.

Key methodological references (ordered by year of publication):

Marra G., Fasiolo M., Radice R., Winkelmann R. (in press), A Flexible Copula Regression Model with Bernoulli and Tweedie Margins for Estimating the Effect of Spending on Mental Health. *Health Economics*.

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.

Marra G., Radice R., Zimmer D. (2020), Estimating the Binary Endogenous Effect of Insurance on Doctor Visits by Copula-Based Regression Additive Models. *Journal of the Royal Statistical Society Series C*, 69(4), 953-971.

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.

Gomes M., Radice R., Camarena-Brenes J., Marra G. (2019), Copula Selection Models for Non-Gaussian Outcomes that Are Missing Not at Random. *Statistics in Medicine*, 38(3), 480-496.

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.

Wojtys M., Marra G., Radice R. (2018), Copula Based Generalized Additive Models for Location, Scale and Shape with Non-Random Sample Selection. *Computational Statistics and Data Analysis*, 127, 1-14.

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.

Marra G., Wyszynski K. (2016), Semi-Parametric Copula Sample Selection Models for Count Responses. *Computational Statistics and Data Analysis*, 104, 110-129.

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

AT *Average treatment effect of a binary/continuous/discrete endogenous variable*

Description

AT can be used to calculate the treatment effect of a binary/continuous/discrete endogenous predictor/treatment, with corresponding interval obtained using posterior simulation.

Usage

```
AT(x, nm.end, eq = NULL, E = TRUE, treat = TRUE, type = "joint", ind = NULL,
   percentage = FALSE, n.sim = 100, prob.lev = 0.05, length.out = NULL,
   hd.plot = FALSE, te.plot = FALSE,
   main = "Histogram and Kernel Density of Simulated Average Effects",
   xlab = "Simulated Average Effects", ...)
```

Arguments

x	A fitted gjrm object as produced by the respective fitting function.
nm.end	Name of the endogenous variable.
eq	Number of equation containing the endogenous variable. This is only used for trivariate models.
E	If TRUE then AT calculates the sample ATE. If FALSE then it calculates the sample AT for the treated individuals only.
treat	If TRUE then AT calculates the AT using the treated only. If FALSE then it calculates the effect on the control group. This only makes sense if E = FALSE.
type	This argument can take three values: "naive" (the effect is calculated ignoring the presence of observed and unobserved confounders), "univariate" (the effect is obtained from the univariate model which neglects the presence of unobserved confounders) and "joint" (the effect is obtained from the simultaneous model which accounts for observed and unobserved confounders).
ind	Binary logical variable. It can be used to calculate the AT for a subset of the data. Note that it does not make sense to use ind when some observations are excluded from the AT calculation (e.g., when using E = FALSE).
percentage	Only for the Roy model, when TRUE it provides results in terms of percentage.
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.

length.out	Ddesired length of the sequence to be used when calculating the effect that a continuous/discrete treatment has on a binary outcome.
hd.plot	If TRUE then a plot of the histogram and kernel density estimate of the simulated average effects is produced. This can only be produced when when binary responses are used.
te.plot	For the case of continuous/discrete endogenous variable and binary outcome, if TRUE then a plot showing the treatment effects that the binary outcome is equal to 1 for each incremental value of the endogenous variable and respective intervals 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

AT measures the 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.

AT can also calculate the effect that a continuous/discrete 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.AT	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>

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)
```

Arguments

x	gjrm object.

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>

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*

Description

gamlss fits flexible univariate regression models with several continuous and discrete distributions, and types of covariate effects. The purpose of this function was only to provide, in some cases, starting values for the simultaneous models in the package, 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. The distributions implemented here have been parametrised according to Rigby and Stasinopoulos (2005).

Usage

```
gamlss(formula, data = list(), weights = NULL, subset = NULL,
       margin = "N", surv = FALSE, cens = NULL, type.cens = "R", upperB = NULL,
       robust = FALSE, rc = 3, lB = NULL, uB = NULL, infl.fac = 1,
       rinit = 1, rmax = 100, iterlimsp = 50, tolsp = 1e-07,
       gc.l = FALSE, parscale, extra.regI = "t", gev.par = -0.25,
       chunk.size = 10000, k.tvc = 0, knots = NULL,
       informative = "no", inform.cov = NULL, margin2 = "PH",
       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	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), typically the environment from which gamlss is called.
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.
margin	Possible distributions are normal ("N"), 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

	distribution), Poisson ("PO"), zero truncated Poisson ("ZTP"), negative binomial - type I ("NBI"), negative binomial - type II ("NBII"), Poisson inverse Gaussian ("PIG"), generalised extreme value link function ("GEVlink", this is used for binary responses and is more stable and faster than the R package bgeva).
surv	If TRUE then a survival model is fitted. Here margin can be "PH" (generalised proportional hazards), "PO" (generalised proportional odds), "probit" (generalised probit).
cens	This is required when surv = TRUE. 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".
upperB	Variable name of right/upper bound when type.cens = "I" or type.cens = "mixed" and interval censoring is present.
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.
extra.regI	If "t" then regularization as from trust is applied to the information matrix if needed. If different from "t" then extra regularization is applied via the options "pC" (pivoted Choleski - this will only work when the information matrix is semi-positive or positive definite) and "sED" (symmetric eigen-decomposition).
gev.par	GEV link parameter.
chunk.size	This is used for discrete robust models.
k.tvc	Experimental. Only used for tvc ps smoothers when using survival 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.

`margin2` In the informative survival case, the margin for the censored equation can be different from that of the survival equation.

`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/survival 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 some extra regularisation (see `extra.regI`) and/or 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

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.

Rigby R.A., Stasinopoulos D.M. (2005). Generalized additive models for location, scale and shape (with discussion). *Journal of the Royal Statistical Society, Series C*, 54(3), 507-554.

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)
post.check(out)

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

library(GJRM)

########################################
## 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,
              surv = TRUE, margin = "PH", cens = delta)
post.check(out)
summary(out)
AIC(out)
```

```
BIC(out)
plot(out, eq = 1, scale = 0, pages = 1)
hazsurv.plot(out, newdata = data.frame(z1 = 0, z2 = 0), shade = TRUE,
             n.sim = 1000, baseline = TRUE)
hazsurv.plot(out, type = "hazard", newdata = data.frame(z1 = 0, z2 = 0),
             shade = TRUE, n.sim = 1000, baseline = TRUE)

out1 <- 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

# note that the Weibull is implemented as AFT
# as using the PH parametrisation makes
# computation unstable
out2 <- gamlss(list(u ~ z1 + s(z2) ), data = dataSim, surv = TRUE,
               margin = "WEI", cens = delta)

####################################
## 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, surv = TRUE,
              margin = "PO", cens = delta)
post.check(out)
```

```
summary(out)
AIC(out)
BIC(out)
plot(out, eq = 1, scale = 0)
hazsurv.plot(out, newdata = data.frame(z = 0), shade = TRUE, n.sim = 1000,
             baseline = TRUE)
hazsurv.plot(out, type = "hazard", 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, surv = TRUE, margin = "PH",
              cens = cens, type.cen = "mixed", upperB = "u2")

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)

hazsurv.plot(out, eq = 1, newdata = ndf, type = "surv")
hazsurv.plot(out, eq = 1, newdata = ndf, type = "hazard", 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,
    copula = "N", copula2 = "N", margins, model, dof = 3, dof2 = 3, ordinal = FALSE,
     surv = FALSE, cens1 = NULL, cens2 = NULL, cens3 = NULL, dep.cens = FALSE,
     upperBt1 = NULL, upperBt2 = NULL,
     gamlssfit = FALSE, fp = FALSE, infl.fac = 1,
     rinit = 1, rmax = 100,
     iterlimsp = 50, tolsp = 1e-07,
     gc.l = FALSE, parscale, extra.regI = "t",
     k1.tvc = 0, k2.tvc = 0, 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)
```

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	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), typically the environment from which gjrm is called.
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.

`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 the two or three margins. Possible distributions are normal ("N"), 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 distribution), Poisson ("PO"), zero truncated Poisson ("ZTP"), negative binomial - type I ("NBI"), negative binomial - type II ("NBII"), Poisson inverse Gaussian ("PIG"). If the responses are binary then possible link functions are "probit", "logit", "cloglog". For survival models, the margins can be proportional hazars ("PH"), odds ("PO") or "probit".

`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. "ROY" is for the Roy 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.

`ordinal` If `TRUE` then the ordinal model is employed.

`surv` If `TRUE` then a bivariate survival model is fitted.

`cens1` Censoring indicator for the first equation. This is required when `surv = TRUE`. 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 surv = TRUE, dep.cens = TRUE and administrative censoring is present.
dep.cens	If TRUE then the dependence censored model is employed.
upperBt1, upperBt2	Variable names of right/upper bounds when interval censoring is present.
gamlssfit	If gamlssfit = TRUE then gamlss 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.
extra.regI	If "t" then regularization as from trust is applied to the information matrix if needed. If different from "t" then extra regularization is applied via the options "pC" (pivoted Choleski - this will only work when the information matrix is semi-positive or positive definite) and "sED" (symmetric eigen-decomposition).
k1.tvc, k2.tvc	Experimental. Only used for tvc ps smoothers when using survival models.
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.

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 some extra regularisation (see `extra.regI`) and/or rescaling (see `parscale`). Using `gamlssfit = 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, `gamlssfit = 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 `gamlssfit = FALSE` and `gamlssfit = 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`, `VuongClarke`, `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 - red lines are true curves

x2 <- sort(x2)
f1.x2 <- f1(x2)[order(x2)] - mean(f1(x2))
f2.x2 <- f2(x2)[order(x2)] - mean(f2(x2))
f3.x3 <- rep(0, length(x3))

par(mfrow=c(2,2),mar=c(4.5,4.5,2,2))
plot(out, eq = 1, select = 1, seWithMean = TRUE, scale = 0)
lines(x2, f1.x2, col = "red")
plot(out, eq = 1, select = 2, seWithMean = TRUE, scale = 0)
lines(x3, f3.x3, col = "red")
plot(out, eq = 2, select = 1, seWithMean = TRUE, scale = 0)
lines(x2, f2.x2, col = "red")
plot(out, eq = 2, select = 2, seWithMean = TRUE, scale = 0)
lines(x3, f3.x3, col = "red")


## 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 = 1, seWithMean = TRUE)
plot(outD, eq = 2, seWithMean = TRUE)
plot(outD, eq = 3, seWithMean = TRUE)
graphics.off()

############
## Example 2
############
## Generate data with one endogenous variable
## and exclusion restriction
```

```
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)
AIC(out); BIC(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)
AIC(out); BIC(out)

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

gt.bpm(out)
```

```
#
## treatment effect, risk ratio and odds ratio with CIs

mb(y1, y2, model = "B")
AT(out, nm.end = "y1", hd.plot = TRUE)
RR(out, nm.end = "y1")
OR(out, nm.end = "y1")
AT(out, nm.end = "y1", type = "univariate")
re.imp <- imputeCounter(out, m = 10, "y1")
re.imp$AT

## 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)
AT(outC, nm.end = "y1")
re.imp <- imputeCounter(outC, m = 10, "y1")
re.imp$AT

## 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)
AT(outJ, "y1")
re.imp <- imputeCounter(outJ, m = 10, "y1")
re.imp$AT

VuongClarke(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

#
#################
## See also ?meps
#################

############
## 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, hence fit a bivariate model

#
## 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
## 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, hd.plot = TRUE)
prev(out, type = "univariate", hd.plot = TRUE)

## 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)


#######################
# Impute using Mice
#######################

library(mice)

ys <- dataSim$ys

dataSim$yo[dataSim$ys == FALSE] <- NA
dataSim <- dataSim[, -c(1:2)]

imp <- mice(dataSim, m = 1, method = c("copulaSS", "", "", ""),
            mice.formula = outC$mice.formula, margins = outC$margins,
            copula = outC$BivD, maxit = 1)
```

```
comp.yo <- dataSim$yo
comp.yo[ys == 0] <- imp$imp$yo[[1]]
mean(comp.yo)

#
################
## See also ?hiv
################

############
## Example 4
############
## Generate data with partial observability

set.seed(0)

n <- 10000

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
# (more computationally intensive)

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 estimated and true functions

x2 <- sort(x2); f1.x2 <- f1(x2)[order(x2)] - mean(f1(x2))
plot(out, eq = 1, scale = 0); lines(x2, f1.x2, col = "red")

#
################
## See also ?war
################

## End(Not run)

## Not run:

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

library(GJRM)

############
## 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)
post.check(out)
summary(out)
AIC(out)
BIC(out)
jc.probs(out, 1.4, 2.3, intervals = TRUE)[1:4,]

#############
## 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

rc <- resp.check(y2, margin = "N", print.par = TRUE, loglik = TRUE)
AIC(rc); BIC(rc)

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

## 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"), gamlssfit = TRUE,
            model = "B")
conv.check(out)
summary(out)
post.check(out)
jc.probs(out, 1, 1.5, intervals = TRUE)[1:4,]
AT(out, nm.end = "y1")
AT(out, nm.end = "y1", type = "univariate")

#
#

############
## 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)
post.check(out)
AT(out, nm.end = "y1")
AT(out, nm.end = "y1", type = "univariate")
RR(out, nm.end = "y1", rr.plot = TRUE)
RR(out, nm.end = "y1", type = "univariate")
OR(out, nm.end = "y1", or.plot = TRUE)
OR(out, nm.end = "y1", type = "univariate")

#
#

#############
## 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, surv = TRUE,
            margins = c("PH", "PO"),
            cens1 = delta1, cens2 = delta2, model = "B")

# PH margin fit can also be compared with cox.ph from mgcv

conv.check(out)
res <- post.check(out)
```

```
## martingale residuals
mr1 <- out$cens1 - res$qr1
mr2 <- out$cens2 - res$qr2

# can be plotted against covariates
# obs index, survival time, rank order of
# surv times

# to determine func form, one may use
# res from null model against covariate

# to test for PH, use:
# library(survival)
# fit <- coxph(Surv(u1, delta1) ~ z1 + z2, data = dataSim)
# temp <- cox.zph(fit)
# print(temp)
# plot(temp, resid = FALSE)


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

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

jc.probs(out, type = "joint", intervals = TRUE)[1:5,]

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

jc.probs(out, type = "joint", newdata = newd0, intervals = TRUE)
jc.probs(out, type = "joint", newdata = newd1, intervals = TRUE)

out1 <- gjrm(list(eq1, eq2, eq3), data = dataSim, surv = TRUE,
                 margins = c("PH", "PO"),
                 cens1 = delta1, cens2 = delta2, gamlssfit = TRUE,
                 model = "B")


########################################
## Joint continuous and survival outcomes
```

```
########################################
# work in progress
#
# 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, surv = TRUE,
#                   margins = c("N", "PO"),
#                   cens1 = NULL, cens2 = delta2,
#                   gamlssfit = TRUE, model = "B")
#
# conv.check(out3)
# post.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)
#
# jc.probs(out3, y1 = 0.6, type = "joint", newdata = newd, intervals = TRUE)

## End(Not run)

## Not run:

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

library(GJRM)

############
## 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)

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

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

out  <- gjrm(f.l, data = dataSim, model = "T",
             margins = c("probit","logit","cloglog"))
out1 <- gjrm(f.l, data = dataSim, Chol = TRUE, model = "T",
             margins = c("probit","logit","cloglog"))
conv.check(out)
summary(out)
plot(out, eq = 1)
plot(out, eq = 2)
AIC(out)
BIC(out)

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 = c("probit", "probit", "probit"))

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 = c("probit", "probit", "probit"))

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 = c("probit", "probit", "probit"))

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 = c("probit", "probit", "probit"))

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 = c("probit", "probit", "probit"))

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 = c("probit", "probit", "probit"))


jcres1 <- jc.probs(out2, 1, 1, 1, type = "joint", cond = 0,
                   intervals = TRUE, n.sim = 100)

nw <- data.frame( x1 = 0, x2 = seq(0, 1, length.out = 100) )

jcres2 <- jc.probs(out2, 1, 1, 1, newdata = nw, type = "joint",
                   cond = 0, intervals = TRUE, n.sim = 100)

##############
## 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, type = "univariate")
prev(out, type = "naive")

## End(Not run)

## Not run:

####################################################
####################################################
####################################################
# JOINT MODELS WITH BINARY AND CONTINUOUS MARGINS #
# WITH SAMPLE SELECTION                            #
####################################################
####################################################
####################################################

library(GJRM)

#############################################################################
## Generate data
```

```
## Correlation between the two equations and covariate correlation 0.5
## Sample size 2000
#########################################################################
#############
## 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)
post.check(out)
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)
post.check(out)
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)


## IMPUTE MISSING VALUES

n.m <- 10
res <- imputeSS(out, n.m)
bet <- NA

for(i in 1:n.m){

dataSim$yo[dataSim$ys == 0] <- res[[i]]

outg <- gamlss(list(yo ~ bi + s(x1)), data = dataSim)
bet[i] <- coef(outg)["bi"]
print(i)
}

mean(bet)

##


## 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)
post.check(out)
summary(out)
summary(out$sigma)
summary(out$theta)

jc.probs(out, 0, 0.3, intervals = TRUE)[1:4,]

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

## IMPUTE MISSING VALUES

n.m <- 10
res <- imputeSS(outC0, n.m)

###############
# or using MICE
###############

library(mice)

ys <- dataSim$ys

dataSim$yo[dataSim$ys == FALSE] <- NA
dataSim <- dataSim[, -1]

imp <- mice(dataSim, m = 1, method = c("copulaSS", "", "", ""),
            mice.formula = outC0$mice.formula, margins = outC0$margins,
            copula = outC0$BivD, maxit = 1)

comp.yo <- dataSim$yo
comp.yo[ys == 0] <- imp$imp$yo[[1]]
mean(comp.yo)


#
#
#########################################################
## example using Gumbel copula and normal-gamma margins
#########################################################
#############
## Example 12
#############
```

```
set.seed(1)

y  <- exp(-0.68 - 1.5*bi + f21(x1) + u[, 2])
yo <- y*(ys > 0)

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


out <- gjrm(list(ys ~ bi + s(x1) + s(x2),
                 yo ~ bi + s(x1)),
                 data = dataSim, copula = "G0",
                 margins = c("probit", "GA"),
                 model = "BSS")
conv.check(out)
post.check(out)
summary(out)


ATE <- NA
n.m <- 10
res <- imputeSS(out, n.m)

for(i in 1:n.m){

dataSim$yo[dataSim$ys == 0] <- res[[i]]

outg <- gamlss(list(yo ~ bi + s(x1)), margin = "GA", data = dataSim)

out$gamlss <- outg
ATE[i] <- AT(out, nm.end = "bi", type = "univariate")$res[2]

print(i)

}

AT(out, nm.end = "bi")
mean(ATE)

## End(Not run)
```

---

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. (in press) 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>

`hazsurv.plot` *Hazard and survival plots*

Description

This function produces either a survival or hazard plot.

Usage

```
hazsurv.plot(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.out = 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` The plot to produce, either `"surv"`, `"hazard"` or `"cumhaz"`. In the excess hazard setting these are, respectively, the net survival, the excess hazard and the cumulative excess hazard.

`t.range` Time variable range to be considered for plotting. 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 to be considered for plotting. This could 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.out` 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 a plot or set of plots.

Author(s)

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

`hiv` *HIV Zambian data*

Description

HIV Zambian data by region, together with polygons describing the regions' shapes.

Usage

```
data(hiv)
data(hiv.polys)
```

Format

`hiv` is a 6416 row data frame with the following columns

hivconsent binary variable indicating consent to test for HIV.

hiv binary variable indicating whether an individual is HIV positive.

age age in years.

education years of education.

region code identifying region, and matching `names(hiv.polys)`. It can take nine possible values: 1 central, 2 copperbelt, 3 eastern, 4 luapula, 5 lusaka, 6 northwestern, 7 northern, 8 southern, 9 western.

marital never married, currently married, formerly married.

std had a sexually transmitted disease.

highhiv had high risk sex.

condom used condom during last intercourse.

aidscare equal to 1 if would care for an HIV-infected relative.

knowsdiedofaids equal to 1 if know someone who died of HIV.

evertestedHIV equal to 1 if previously tested for HIV.

smoke smoker.

ethnicity bemba, lunda (luapula), lala, ushi, lamba, tonga, luvale, lunda (northwestern), mbunda, kaonde, lozi, chewa, nsenga, ngoni, mambwe, namwanga, tumbuka, other.

language English, Bemba, Lozi, Nyanja, Tonga, other.

interviewerID interviewer identifier.

sw survey weights.

`hiv.polys` contains the polygons defining the areas in the format described below.

Details

The data frame `hiv` relates to the regions whose boundaries are coded in `hiv.polys`. `hiv.polys[[i]]` is a 2 column matrix, containing the vertices of the polygons defining the boundary of the ith region. `names(hiv.polys)` matches `hiv$region` (order unimportant).

Source

The data have been produced as described in:

McGovern M.E., Barnighausen T., Marra G. and Radice R. (2015), On the Assumption of Joint Normality in Selection Models: A Copula Approach Applied to Estimating HIV Prevalence. *Epidemiology*, 26(2), 229-237.

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.

Examples

```
## Not run:

#########################################################
#########################################################

library("GJRM")

data("hiv", package = "GJRM")
data("hiv.polys", package = "GJRM")

#########################################################
#########################################################
## The stuff below is useful if the user wishes to employ
## a Markov Random Field (MRF) smoother. It provides
## the instructions to set up polygons automatically
## and the dataset variable needed to fit a model with
## MRF.
#########################################################
#########################################################
#
# ## hiv.polys was already created and
# ## made available via the call
# ## data("hiv.polys", package = "GJRM")
# ## hiv.polys was created using the code below
#
# obj <- readRDS("ZMB_adm1.rds")
# ## RDS Zambian Level 1 file obtained from
# ## http://www.gadm.org.
#
# pol <- polys.setup(obj)
#
# hiv.polys <- pol$polys
# name <- cbind(names(hiv.polys), pol$names1)
# name
#
## last step was to create a factor variable with range
```

```
## range(name[,1]) where the numerical values were linked
## to the regions in name[, 2]. This is what was done in
## the hiv dataset; see hiv$region. Specifically,
## the procedure used was
##
# reg <- NULL
#
# for(i in 1:dim(hiv)[1]){
#
# if(hiv$region[i] == "Central")       reg[i] <- 1
# if(hiv$region[i] == "Copperbelt")    reg[i] <- 2
# if(hiv$region[i] == "Eastern")       reg[i] <- 3
# if(hiv$region[i] == "Luapula")       reg[i] <- 4
# if(hiv$region[i] == "Lusaka")        reg[i] <- 5
# if(hiv$region[i] == "North-Western") reg[i] <- 6
# if(hiv$region[i] == "Northern")      reg[i] <- 7
# if(hiv$region[i] == "Southern")      reg[i] <- 8
# if(hiv$region[i] == "Western")       reg[i] <- 9
#
# }
#
# hiv$region <- as.factor(reg)
#
#
############################################################
############################################################

xt <- list(polys = hiv.polys)

# neighbourhood structure info for MRF
# to use in model specification

############################################################
# Bivariate brobit model with non-random sample selection
############################################################

sel.eq <- hivconsent ~ s(age) + s(education) + s(wealth) +
                       s(region, bs = "mrf", xt = xt, k = 7) +
                       marital + std + age1sex_cat + highhiv +
                       partner + condom + aidscare +
                       knowsdiedofaids + evertestedHIV +
                       smoke + religion + ethnicity +
                       language + s(interviewerID, bs = "re")

out.eq <- hiv        ~ s(age) + s(education) + s(wealth) +
                       s(region, bs = "mrf", xt = xt, k = 7) +
                       marital + std + age1sex_cat + highhiv +
                       partner + condom + aidscare +
                       knowsdiedofaids + evertestedHIV +
                       smoke + religion + ethnicity +
                       language

theta.eq <-          ~ s(region, bs = "mrf", xt = xt, k = 7)
```

```
fl <- list(sel.eq, out.eq, theta.eq)

# the above model specification is fairly
# complex and it serves to illustrate the
# flexibility of the modelling approach

bss <- gjrm(fl, data = hiv, copula = "J90", model = "BSS",
            margins = c("probit", "probit"))

conv.check(bss)

set.seed(1)
sb <- summary(bss)
sb

plot(bss, eq = 1, seWithMean = TRUE, scheme = 1,
     scale = 0, pages = 1, jit = TRUE)

plot(bss, eq = 2, seWithMean = TRUE, scheme = 1,
     scale = 0, pages = 1, jit = TRUE)


prev(bss, sw = hiv$sw, type = "naive")

set.seed(1)
prev(bss, sw = hiv$sw, type = "univariate")

prev(bss, sw = hiv$sw)


lr <- length(hiv.polys)
prevBYreg  <- matrix(NA, lr, 2)
thetaBYreg <- NA

for(i in 1:lr) {
prevBYreg[i,1] <- prev(bss, sw = hiv$sw, ind = hiv$region==i,
                       type = "univariate")$res[2]
prevBYreg[i,2] <- prev(bss, sw = hiv$sw, ind = hiv$region==i)$res[2]
thetaBYreg[i]  <- bss$theta[hiv$region==i][1]
}


zlim <- range(prevBYreg)  # to establish a common prevalence range

par(mfrow = c(1, 3), cex.axis = 1.3)

polys.map(hiv.polys, prevBYreg[,1], zlim = zlim, lab = "",
          cex.lab = 1.5, cex.main = 1.5,
          main = "HIV - Imputation Model")

polys.map(hiv.polys, prevBYreg[,2], zlim = zlim, cex.main = 1.5,
          main = "HIV - Selection Model")
```

```
polys.map(hiv.polys, thetaBYreg, rev.col = FALSE, cex.main = 1.7,
          main = expression(paste("Copula parameter (",hat(theta),")")))

sb$CItheta[1,]

## End(Not run)

#
```

`imputeCounter` *Imputation of Counterfactual*

Description

`imputeCounter` imputes counterfactual missing values for a gjrm model object.

Usage

```
imputeCounter(x, m = 10, nm.end)
```

Arguments

`x`	A fitted `gjrm` object.
`m`	Number of imputed response vectors.
`nm.end`	Name endogenous variable.

Details

This function generates m sets of imputed values for the outcome of interest under a fitted joint causal model. The algorithm draws parameters from the posterior distribution of the model which are then used to obtain simulated responses (from the posterior predictive distribution of the missing values) via a rejection algorithm. The bound for acceptance/rejection is obtained via a trust region optimisation.

The imputed values are used to create m complete imputed datasets and perform complete data analysis and inference about the parameters of interest using any summary statistics.

Value

It returns a list containing m imputed response vectors.

Author(s)

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

References

Robert C. and Casella G. (2004). Monte Carlo Statistical Methods. New York: Springer-Verlag.

Ripley B. D. (1987) Stochastic Simulation. New York: John Wiley & Sons, Inc.

See Also

gjrm

Examples

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

`imputeSS` *Missing values' imputation*

Description

`imputeSS` imputes missing values for a gjrm model object.

Usage

```
imputeSS(x, m = 10)
```

Arguments

`x`	A fitted `gjrm` object.
`m`	Number of imputed response vectors.

Details

This function generates m sets of imputed values for the outcome of interest under a fitted copulaSampleSel model. The algorithm draws parameters from the posterior distribution of copulaSampleSel which are then used to obtain simulated responses (from the posterior predictive distribution of the missing values) via a rejection algorithm. The bound for acceptance/rejection is obtained via a trust region optimisation.

The imputed values are used to create m complete imputed datasets and perform complete data analysis and inference about the parameters of interest using function `gamlss()` within this package.

Value

It returns a list containing m imputed response vectors.

Author(s)

Authors: Jose Camarena, Giampiero Marra and Rosalba Radice

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

References

Robert C. and Casella G. (2004). Monte Carlo Statistical Methods. New York: Springer-Verlag.

Ripley B. D. (1987) Stochastic Simulation. New York: John Wiley & Sons, Inc.

See Also

gjrm

Examples

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

---

jc.probs *Joint or conditional probabilities from a fitted joint model*

Description

jc.probs can be used to calculate the joint or conditional probabilities from a fitted joint model with intervals obtained using posterior simulation.

Usage

```
jc.probs(x, y1, y2, y3 = NULL, newdata, type = "joint", cond = 0,
      intervals = FALSE, n.sim = 100, prob.lev = 0.05, min.pr = 1e-323, max.pr = 1,
         cumul = "no")
```

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 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.
type	This argument can take two: "joint" (the probabilities are calculated from the fitted joint model) and "independence" (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.
`min.pr, max.pr`	Allowed minimum and maximum for estimated probabities.
`cumul`	Only used for discrete and continuous margins' case.

Details

This function calculates joint or conditional probabilities from a fitted joint model or a model assuming independence, with intervals obtained using 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`

`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

---

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 AT or prev when type = "naive", and is different from what obtained by AT 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
```

---

meps *MEPS data*

Description

2008 MEPS data.

Usage

```
data(meps)
```

Format

meps is a 18592 row data frame with the following columns

bmi body mass index.

age age in years.

gender equal to 1 if male.

race levels: 2 white, 3 black, 4 native American, 5 others.

education years of education.

health levels: 5 excellent, 6 very good, 7 good, 8 fair, 9 poor.

limitation equal to 1 if health limits physical activity.

region levels: 2 northeast, 3 mid-west, 4 south, 5 west.

private equal to 1 if individual has private health insurance.

visits.hosp equal to 1 if at least one visit to hospital outpatient departments.

diabetes equal to 1 if diabetic.

hypertension equal to 1 if hypertensive.

hyperlipidemia equal to 1 if hyperlipidemic.

income income (000's).

Source

The data have been obtained from http://www.meps.ahrq.gov/.

Examples

```
## Not run:

#################################################
#################################################

library("GJRM")
data("meps", package = "GJRM")

################################################
# Bivariate brobit models with endogenous treatment
################################################

treat.eq <- private ~ s(bmi) + s(income) + s(age) + s(education) +
                      as.factor(health) + as.factor(race) +
                      as.factor(limitation) + as.factor(region) +
                      gender  + hypertension + hyperlipidemia + diabetes
out.eq <- visits.hosp ~ private + s(bmi) + s(income) + s(age) +
                        s(education) + as.factor(health) +
                        as.factor(race) + as.factor(limitation) +
                        as.factor(region) + gender + hypertension +
                        hyperlipidemia + diabetes
```

```
f.list <- list(treat.eq, out.eq)
mr     <- c("probit", "probit")
bpN    <- gjrm(f.list, data = meps, margins = mr,                  model = "B")
bpF    <- gjrm(f.list, data = meps, margins = mr, copula = "F",    model = "B")
bpC0   <- gjrm(f.list, data = meps, margins = mr, copula = "C0",   model = "B")
bpC180 <- gjrm(f.list, data = meps, margins = mr, copula = "C180", model = "B")
bpJ0   <- gjrm(f.list, data = meps, margins = mr, copula = "J0",   model = "B")
bpJ180 <- gjrm(f.list, data = meps, margins = mr, copula = "J180", model = "B")
bpG0   <- gjrm(f.list, data = meps, margins = mr, copula = "G0",   model = "B")
bpG180 <- gjrm(f.list, data = meps, margins = mr, copula = "G180", model = "B")

conv.check(bpJ0)

AIC(bpN, bpF, bpC0, bpC180, bpJ0, bpJ180, bpG0, bpG180)

set.seed(1)
summary(bpJ0)

#dev.copy(postscript, "contplot.eps")
#dev.off()

par(mfrow = c(2, 2), mar = c(4.5, 4.5, 2, 2),
    cex.axis = 1.6, cex.lab = 1.6)
plot(bpJ0, eq = 1, seWithMean = TRUE, scale = 0, shade = TRUE,
     pages = 1, jit = TRUE)

#dev.copy(postscript, "spline1.eps")
#dev.off()

par(mfrow = c(2, 2), mar = c(4.5, 4.5, 2, 2),
    cex.axis = 1.6, cex.lab = 1.6)
plot(bpJ0, eq = 2, seWithMean = TRUE, scale = 0, shade = TRUE,
     pages = 1, jit = TRUE)

#dev.copy(postscript, "spline2.eps")
#dev.off()

set.seed(1)
AT(bpJ0, nm.end = "private", hd.plot = TRUE, cex.axis = 1.5,
   cex.lab = 1.5, cex.main = 1.6)

#dev.copy(postscript, "hd.plotAT.eps")
#dev.off()

AT(bpJ0, nm.end = "private", type = "univariate")

AT(bpJ0, nm.end = "private", type = "naive")


## End(Not run)

#
```

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/discrete endogenous variable*

Description

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

Usage

```
OR(x, nm.end, E = TRUE, treat = TRUE, type = "joint", ind = NULL,
   n.sim = 100, prob.lev = 0.05, length.out = NULL, hd.plot = FALSE,
   or.plot = FALSE,
   main = "Histogram and Kernel Density of Simulated Odds Ratios",
   xlab = "Simulated Odds Ratios", ...)
```

Arguments

x	A fitted gjrm object.
nm.end	Name of the endogenous variable.
E	If TRUE then OR calculates the sample OR. If FALSE then it calculates the sample OR for the treated individuals only.
treat	If TRUE then OR calculates the OR using the treated only. If FALSE then it calculates the ratio using the control group. This only makes sense if E = FALSE.
type	This argument can take three values: "naive" (the effect is calculated ignoring the presence of observed and unobserved confounders), "univariate" (the effect is obtained from the univariate model which neglects the presence of unobserved confounders) and "joint" (the effect is obtained from the bivariate model which accounts for observed and unobserved confounders).

ind	Binary logical variable. It can be used to calculate the OR for a subset of the data. Note that it does not make sense to use ind when some observations are excluded from the OR calculation (e.g., when using E = FALSE).
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/discrete treatment has on a binary outcome.
hd.plot	If TRUE then a plot of the histogram and kernel density estimate of the simulated odds ratios is produced. This can only be produced when binary responses are used.
or.plot	For the case of continuous/discrete endogenous variable and binary outcome, if TRUE then a plot (on the log scale) showing the odd ratios that the binary outcome is equal to 1 for each incremental value of the endogenous variable and respective intervals 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

OR calculates the causal odds ratio for a binary/continuous/discrete 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/discrete 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,
   hd.plot = FALSE,
   main = "Histogram and Kernel Density 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.
hd.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 = TRUE, ...)
```

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 FALSE then coloring scheme is not 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
```

`post.check` *Diagnostic plots for discrete/continuous response margin*

Description

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

Usage

```
post.check(x, main = "Histogram and Density Estimate of Residuals",
           main2 = "Histogram and Density Estimate of Residuals",
           xlab = "Quantile Residuals", xlab2 = "Quantile Residuals",
           intervals = FALSE, n.sim = 100, prob.lev = 0.05, ...)
```

Arguments

`x`	A fitted `gjrm` object.
`main`	Title for the plot.
`main2`	Title for the plot in the second row. This comes into play only when fitting models with two non-binary margins.
`xlab`	Title for the x axis.
`xlab2`	Title for the x axis in the second row. As above.
`intervals`	If `TRUE` then intervals for the qqplots are produced.
`n.sim`	Number of replicate datasets used to simulate quantiles of the residual distribution.
`prob.lev`	Overall probability of the left and right tails of the probabilities' distributions used for interval calculations.
`...`	Other graphics parameters to pass on to plotting commands.

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

---

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

pred.mvt *Function to predict mean and variance of marginal distributions, as well as Kendall's tau*

Description

It takes a fitted gjrm object produced by gjrm() and produces predictions and respective intervals.

Usage

```
pred.mvt(x, eq, fun = "mean", n.sim = 100, prob.lev = 0.05, smooth.no = NULL, ...)
```

Arguments

x	A fitted gjrm object.
eq	The equation number.
fun	Either mean, variance or tau.
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.
smooth.no	Smooth number if the interest is in a particular smooth and not the additive predictor(s).
...	Other parameters.

Author(s)

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

See Also

gjrm

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 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 the delta method or posterior simulation.

Usage

```
prev(x, sw = NULL, type = "joint", ind = NULL, delta = FALSE,
     n.sim = 100, prob.lev = 0.05, hd.plot = FALSE,
     main = "Histogram and Kernel Density of Simulated Prevalences",
     xlab = "Simulated Prevalences", ...)
```

Arguments

`x`	A fitted `gjrm` object.
`sw`	Survey weights.
`type`	This argument can take three values: `"naive"` (the prevalence is calculated ignoring the presence of observed and unobserved confounders), `"univariate"` (the prevalence is obtained from the univariate probit/single imputation model which neglects the presence of unobserved confounders) and `"joint"` (the prevalence is obtained from the bivariate/trivariate model which accounts for observed and unobserved confounders).

ind	Binary logical variable. It can be used to calculate the prevalence for a subset of the data.
delta	If TRUE then the delta method is used for confidence interval calculations, otherwise Bayesian posterior simulation is employed.
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 prevalence distribution used for interval calculations.
hd.plot	If TRUE then a plot of the histogram and kernel density estimate of the simulated prevalences is produced. This can only be produced when delta = FALSE.
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

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 the delta method or 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	If delta = FALSE then it returns a 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.AT *Print an AT object*

Description

The print method for an AT object.

Usage

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

Arguments

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

Details

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

Author(s)

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

See Also

AT

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 object*

Description

The print method for a gjrm object.

Usage

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

Arguments

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

Details

print.gjrm 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

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.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>

resp.check *Plots for response variable*

Description

It produces a histogram of the response along with the estimated density from the assumed distribution as well as a normal Q-Q plot for the (randomised) normalised quantile response. It also provides the log-likelihood for AIC calculation, for instance.

Usage

```
resp.check(y, margin = "N", main = "Histogram and Density of Response",
           xlab = "Response", print.par = FALSE, plots = TRUE,
           loglik = FALSE, os = FALSE,
           intervals = FALSE, n.sim = 100, prob.lev = 0.05,
           i.f = FALSE,
           min.dn = 1e-40, min.pr = 1e-16, max.pr = 0.999999, ...)
```

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 ("PO"), zero truncated Poisson ("ZTP"), negative binomial - type I ("NBI"), negative binomial - type II ("NBII"), Poisson inverse Gaussian ("PIG").
main	Title for the plot.
xlab	Title for the x axis.

print.par	If TRUE then the estimated parameters used to construct the plots are returned.
plots	If FALSE then no plots are 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.
intervals	If TRUE then intervals for the qqplot are produced.
n.sim	Number of replicate datasets used to simulate quantiles of the residual distribution.
prob.lev	Overall probability of the left and right tails of the probabilities' distribution used for interval calculations.
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.
...	Other graphics parameters to pass on to plotting commands.

Details

Prior to fitting a model with discrete and/or continuous margins, the distributions for the responses may be chosen by looking at the histogram of the response along with the estimated density from the assumed distribution, and at 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.

The shapiro test can also be performed.

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)
```

Arguments

x	A fitted gjrm object.
B	Number of bootstrap replicates.

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.

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

RR *Causal risk ratio of a binary/continuous/discrete endogenous variable*

Description

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

Usage

```
RR(x, nm.end, E = TRUE, treat = TRUE, type = "joint", ind = NULL,
   n.sim = 100, prob.lev = 0.05, length.out = NULL, hd.plot = FALSE,
   rr.plot = FALSE,
   main = "Histogram and Kernel Density of Simulated Risk Ratios",
   xlab = "Simulated Risk Ratios", ...)
```

Arguments

x	A fitted gjrm object.
nm.end	Name of the endogenous variable.
E	If TRUE then RR calculates the sample RR. If FALSE then it calculates the sample RR for the treated individuals only.
treat	If TRUE then RR calculates the RR using the treated only. If FALSE then it calculates the ratio using the control group. This only makes sense if E = FALSE.
type	This argument can take three values: "naive" (the effect is calculated ignoring the presence of observed and unobserved confounders), "univariate" (the effect is obtained from the univariate model which neglects the presence of unobserved confounders) and "joint" (the effect is obtained from the bivariate model which accounts for observed and unobserved confounders).
ind	Binary logical variable. It can be used to calculate the RR for a subset of the data. Note that it does not make sense to use ind when some observations are excluded from the RR calculation (e.g., when using E = FALSE).
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/discrete treatment has on a binary outcome.
hd.plot	If TRUE then a plot of the histogram and kernel density estimate of the simulated risk ratios is produced. This can only be produced when binary responses are used.

rr.plot	For the case of continuous/discrete endogenous variable and binary outcome, if TRUE then a plot (on the log scale) showing the risk ratios that the binary outcome is equal to 1 for each incremental value of the endogenous variable and respective intervals 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

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/discrete 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/discrete 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>

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.
tau	Estimated Kendall's tau dependence measure between 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, CItau` Interval(s) for θ and Kendall's tau.

`CIsig1,CIsig2,CInu1,CInu2`
Intervals for distribution specific parameters

WARNINGS

Note that the summary output will also indeed provide the Kendall's tau and related interval. This is a valid measure of dependence for continuous margins but it may not for discrete margins, for instance. However, it is still displayed for the sake of keeping the printed output consistent with that of other models in the package. Also, it still provides an approximate measure of dependence under certan scenarios.

Author(s)

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

`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.
tau, CItau	Kendall's tau and respective intervals.

WARNINGS

Note that the summary output will also indeed provide the Kendall's tau and related interval. This is a valid measure of dependence for continuous margins but it is typically not for binary margins. However, it is still displayed for the sake of keeping the printed output consistent with that of other models in the package.

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

AT, 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>

vis.gjrm *Visualization function*

Description

It takes a fitted `gjrm` object produced by `gjrm()` and produces perspective or contour plot views of model predictions. This function is a wrapper of `vis.gam()` in `mgcv`. Please see the documentation of `vis.gam()` for full details.

Usage

```
vis.gjrm(x, eq, fun = NULL, ...)
```

Arguments

`x`	A fitted `gjrm` object.
`eq`	The equation number.
`fun`	Either mean or variance. If left as equal to NULL then predictions on the scale of the predictor will be produced.
`...`	Other graphics parameters to pass on to plotting commands, as described for `vis.gam()` in `mgcv`.

Value

The function generates plots.

Author(s)

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

See Also

`gjrm`

`VuongClarke` *Vuong and Clarke tests*

Description

The Vuong and Clarke tests are likelihood-ratio-based tests that can be used for choosing between two non-nested models.

Usage

```
VuongClarke(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 (1989) and Clarke (2007) tests are likelihood-ratio-based tests for model selection that use the Kullback-Leibler information criterion. The implemented tests can be used for choosing between two bivariate models which are non-nested.

In the Vuong test, 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.

In the Clarke test, 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 two decisions based on the tests and criteria discussed above.

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.

Vuong Q.H. (1989), Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses. *Econometrica*, 57(2), 307-333.

Examples

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

`war` *Civil war data*

Description

Civil war data from Fearon and Laitin (2003).

Usage

```
data(war)
```

Format

`war` is a 6326 row data frame with the following columns

onset equal to 1 for all country-years in which a civil war started.

instab equal to 1 if unstable government.

oil equal to 1 for oil exporter country.

warl equal to 1 if the country had a distinct civil war ongoing in the previous year.

gdpenl GDP per capita (measured as thousands of 1985 U.S. dollars) lagged one year.

ncontig equal to 1 for non-contiguous state.

nwstate equal to 1 for new state.

lpopl log(population size).

lmtnest log(mountainous).

ethfrac measure of ethnic fractionalization (calculated as the probability that two randomly drawn individuals from a country are not from the same ethnicity).

relfrac measure of religious fractionalization.

polity2l measure of political democracy (ranges from -10 to 10) lagged one year.

Source

Data are from:

Fearon J.D., Laitin D.D. (2003), Ethnicity, Insurgency, and Civil War. *The American Political Science Review*, 97, 75-90.

Examples

```
## Not run:

#########################################################
#########################################################

library("GJRM")

data("war", package = "GJRM")

###################################################
# Bivariate brobit model with partial observability
###################################################

reb.eq <- onset ~ instab + oil + warl + lpopl + lmtnest + ethfrac +
                  polity2l + s(gdpenl) + s(relfrac)
gov.eq <- onset ~ instab + oil + warl + ncontig + nwstate + s(gdpenl)

bpo <- gjrm(list(reb.eq, gov.eq), data = war, model = "BPO",
            margins = c("probit", "probit"))
conv.check(bpo)

# perhaps model is to complex

set.seed(1)
sbpo <- summary(bpo)
sbpo$theta; sbpo$CItheta

# let's exclude the correlation parameter in fitting

bpo0 <- gjrm(list(reb.eq, gov.eq), data = war, model = "BPO0",
             margins = c("probit", "probit"))
conv.check(bpo0)

summary(bpo0)


war.eq <- onset ~ instab + oil + warl +  ncontig + nwstate + lpopl +
                  lmtnest + ethfrac + polity2l + s(gdpenl) + s(relfrac)
Probit <- gam(war.eq, family = binomial(link = "probit"), data = war)
summary(Probit)


coef(Probit)[(which(names(coef(Probit)) == "s(gdpenl).9"))]

coef(bpo0)[(which(names(coef(bpo)) == "s(gdpenl).9"))]


probitW <- bpoW <- bpoReb <- bpoGov <- NA
gdp.grid <- seq(0, 8)
```

```
median.values <- data.frame(t(apply(war, 2, FUN = median)))

for (i in 1:length(gdp.grid)){

newd <- median.values; newd$gdpenl <- gdp.grid[i]
eta1 <- predict(bpo0, eq = 1, newd)
eta2 <- predict(bpo0, eq = 2, newd)
probitW[i] <- predict(Probit, newd, type = "response")
bpoW[i]    <- pnorm(eta1)*pnorm(eta2)
bpoReb[i]  <- pnorm(eta1)
bpoGov[i]  <- pnorm(eta2)

}


plot(gdp.grid, probitW, type = "l", ylim = c(0, 0.55), lwd = 2,
     col = "grey", xlab = "GDP per Capita (in thousands)",
     ylab = "Pr(Outcome)", main = "Probabilities for All Outcomes",
     cex.main = 1.5, cex.lab = 1.3, cex.axis = 1.3)
lines(gdp.grid, bpoW,   lwd = 2)
lines(gdp.grid, bpoReb, lwd = 2, lty = 2)
lines(gdp.grid, bpoGov, lwd = 2, lty = 3)

#dev.copy(postscript, "probWAR.eps", width = 8)
#dev.off()


## End(Not run)

#
```

`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