Package: eivtools (via r-universe)

September 6, 2024

Type Package

Title Measurement Error Modeling Tools

Version 0.1-9

Date 2020-04-01

Depends R (>= 3.4.0), stats, grDevices, graphics, utils, R2jags

Author J.R. Lockwood

Maintainer J.R. Lockwood <jr@duolingo.com>

Description This includes functions for analysis with error-prone covariates, including deconvolution, latent regression and errors-in-variables regression. It implements methods by Rabe-Hesketh et al. (2003) <doi:10.1191/1471082x03st056oa>, Lockwood and McCaffrey (2014) <doi:10.3102/1076998613509405>, and Lockwood and McCaffrey (2017) <doi:10.1007/s11336-017-9556-y>, among others.

License GPL (>= 2) | file LICENSE

LazyData true

Repository https://jrlockwood.r-universe.dev

RemoteUrl https://github.com/jrlockwood/eivtools

RemoteRef HEAD

RemoteSha b8d8d6230280d22593cc9e6bc8039bba9d913a1c

Contents

deconv_npmle *Nonparametric MLE deconvolution under heteroskedastic normal error*

Description

Implements a version of the Rabe-Hesketh et al. (2003) algorithm for computing the nonparametric MLE of a univariate latent variable distribution from observed error-prone measures. Allows for normal heteroskedastic measurement error with variance that depends on the latent variable, such as with estimates of latent ability from item response theory models.

Usage

```
deconv_npmle(W, csem,
gridspec = list(fixed=FALSE, xmin=-5, xmax=5, numpoints=2000),
lambda = 0.0005, lltol = 1e-7, psmall = 0.00005,
discrete = FALSE, quietly = FALSE)
```

Arguments

`W` Vector of observed measures, where `W[i]` is assumed to be an error-prone measure of a latent variable `X[i]`

`csem` A function of a single variable `x` that returns the conditional standard deviation of `W` given `X=x`. It needs to be able to accept a vector input and return a vector of the same length as the input.

`gridspec` A named list specifying the grid over which the NPMLE will be computed. It must have a logical component `fixed` indicating if the NPMLE grid is fixed (TRUE), or if the final grid is chosen by optimization over a candidate grid (FALSE). The default is FALSE. The remaining components of `gridspec` specify the grid in one of two mutually exclusive ways. In the first way, `gridspec` must contain elements `xmin` providing the minimum grid value, `xmax` providing the maximum grid value, and `numpoints` providing the desired number of points. In this case, the grid will be `numpoints` equally-spaced values ranging from `xmin` to `xmax`. In the second way, `gridspec` must contain an element `grid`, a numeric vector providing the actual desired grid. It must be an arbitrary sequence of increasing numeric values with no missing values.

`lambda` Step size, on probability scale, in Rabe-Hesketh et al. (2003) algorithm. See reference for details.

`lltol` Algorithm stops when the improvement to the log likelihood does not exceed `lltol`.

`psmall` If a mass point in the estimated latent distribution evolves to have probability less than `psmall`, it gets dropped.

`discrete` Not currently implemented.

`quietly` If `FALSE` (the default), prints iteration-by-iteration progress.

Details

The assumed model is `W = X + U` where the conditional distribution of `U` given `X = x` is assumed to be normal with mean zero and standard deviation `csem(x)`. The function uses `W` to estimate a discrete latent distribution for `X` that maximizes the likelihood of the observed data. The function optimizes the mass points (among a grid of candidate values) and the associated probabilities.

In the special case of homoskedastic error, the function `csem` must be defined such that when passed a vector of length `n`, it returns a vector of length `n` where each element is a common constant.

The default values of `xmin` and `xmin` in `gridspec` are generally appropriate only for a latent variable on a standardized scale with mean zero and variance one, and should be set to appropriate values given the scale of `W`.

Value

A list with elements

- gridspec: Information about the initial grid
- .history: Iteration-by-iteration evolution of the estimated distribution, if `gridspec$fixed` is FALSE. Otherwise it is an empty list
- px: A dataframe providing the final NPMLE distribution. There are as many rows as there are mass points in the estimated distribution; fields described below
- reliability: An estimate of the reliability of `W`, equal to the estimated variance of `X` divided by the sample variance of `W`
- simex_varfuncs: A dataframe with as many rows as there are unique values of `W`, providing estimated plug-in variance functions to use for SIMEX data generation with latent heteroskedastic error as described in Lockwood and McCaffrey (forthcoming); see references. Fields described below

The fields of `px` are:

- x: Location of mass point
- csem: Value of function `csem` at mass point
- p: probability at mass point
- ll: log likelihood at solution
- ex: Estimate of mean of latent distribution
- varx: Estimate of variance of latent distribution

The fields of `simex_varfuncs` are:

- W: Unique observed values `w` of `W`
- gW: The square of `csem` evaluated at `W = w`
- gEXW: The square of `csem` evaluated at `E[X | W=w]`, the conditional mean of `X` given `W=w`
- EgXW: The conditional mean of the square of `csem` of `X` given `W=w`, equal to `E[g(X) | W=w]`

Author(s)

J.R. Lockwood `<jrlockwood@ets.org>`

References

Lockwood J.R. and McCaffrey D.F. (2014). “Correcting for test score measurement error in ANCOVA models for estimating treatment effects,” *Journal of Educational and Behavioral Statistics* 39(1):22-52.

Lockwood J.R. and McCaffrey D.F. (2017). “Simulation-extrapolation with latent heteroskedastic variance,” *Psychometrika* 82(3):717-736.

Rabe-Hesketh S., Pickles A. and Skrondal A. (2003). “Correcting for covariate measurement error in logistic regression using nonparametric maximum likelihood estimation,” *Statistical Modelling* 3:215-232.

See Also

testscores,eivreg

Examples

```
data(testscores)

## get the unique values of the lag 1 math score and CSEM
## values and approximate the CSEM function using approxfun()
tmp <- unique(testscores[,c("math_lag1","math_lag1_csem")])
print(tmp <- tmp[order(tmp$math_lag1),])

.csem <- approxfun(tmp$math_lag1, tmp$math_lag1_csem, rule=2:2)
plot(tmp$math_lag1, tmp$math_lag1_csem)
lines(tmp$math_lag1, .csem(tmp$math_lag1), col="blue")

## get NPMLE distribution of latent lag 1 math achievement
m <- deconv_npmle(W        = testscores$math_lag1,
                  csem     = .csem,
                  gridspec = list(fixed = FALSE,
                                  xmin = min(testscores$math_lag1),
                                  xmax = max(testscores$math_lag1),
                                  numpoints = 10000),
                  quietly  = TRUE)

print(m$px)

## estimated mean is approximately the mean of W, but
## the estimated variance is less than the variance of W,
## as it should be
print(c(empirical = mean(testscores$math_lag1),
        estimated = m$px$ex[1]))

print(c(empirical = var(testscores$math_lag1),
        estimated = m$px$varx[1]))

## estimated reliability of W:
print(m$reliability)

## if implementing SIMEX, simex_varfuncs provides plug-in
## options to use for the heteroskedastic error variance
## of each observed W
print(m$simex_varfuncs)

## simple "value-added" estimates of school effects on math,
## adjusting for measurement error in the lag 1 math score.
testscores$schoolid <- factor(testscores$schoolid)

meiv <- eivreg(math ~ math_lag1 + sped + frl + schoolid,
         data = testscores,
         reliability = c(math_lag1 = m$reliability),
         contrasts = list(schoolid = "contr.sum"))

print(summary(meiv))

## alternative deconvolution with fixed grid
m <- deconv_npmle(W        = testscores$math_lag1,
                  csem     = .csem,
                  gridspec = list(fixed = TRUE,
                                  xmin = min(testscores$math_lag1),
                                  xmax = max(testscores$math_lag1),
                                  numpoints = 40),
                  quietly  = TRUE)

print(m$px)
```

eivreg *Errors-in-variables (EIV) linear regression*

Description

Fits errors-in-variables (EIV) linear regression given specified reliabilities, or a specified variance/covariance matrix for the measurement errors. For either case, it computes robust standard error estimates that allow for weighting and/or clustering.

Usage

```
eivreg(formula, data, subset, weights, na.action, method = "qr",
model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = FALSE,
contrasts = NULL, reliability = NULL, Sigma_error = NULL,
cluster_varname = NULL, df_adj = FALSE, stderr = TRUE, offset,
...)
```

Arguments

`formula`, `data`, `subset`, `weights`, `na.action`, `method`, `model`, `x`, `y`, `qr`
: See documentation for `lm`.

`singular.ok`, `contrasts`, `offset`, `...`
: See documentation for `lm`.

`reliability` Named numeric vector giving the reliability for each error-prone covariate. If left `NULL`, `Sigma_error` must be specified.

`Sigma_error` Named numeric matrix giving the variance/covariance matrix of the measurement errors for the error-prone covariate(s). If left `NULL`, `reliability` must be specified.

`cluster_varname`
A character variable providing the name of a variable in `data` that will be used as a clustering variable for robust standard error computation.

`df_adj` Logical (default FALSE); if TRUE, the estimated variance/covariance matrix of the regression parameters is multiplied by `N/(N-p)`, where `N` is the number of observations used in the model fit and `p` is the number of regression parameters (including an intercept, if any).

`stderr` Logical (default TRUE); if FALSE, does not compute estimated variance/covariance matrix of the regression parameters.

Details

Theory

The EIV estimator applies when one wishes to estimate the parameters of a linear regression of Y on (X, Z), but covariates (W, Z) are instead observed, where $W = X + U$ for mean zero measurement error U. Additional assumptions are required about U for consistent estimation; see references for details.

The standard EIV estimator of the regression coefficients is $(Q'Q - S)^{-1}Q'Y$, where Q is the design matrix formed from (W, Z) and S is a matrix that adjusts $Q'Q$ to account for elements that are distorted due to measurement error. The value of S depends on whether `reliability` or `Sigma_error` is specified. When `Sigma_error` is specified, S is known. When `reliability` is specified, S must be estimated using the marginal variances of the observed error-prone covariates.

The estimated regression coefficients are solutions to a system of estimating equations, and both the system of equations and the solutions depend on whether `reliability` or `Sigma_error` is specified. For each of these two cases, standard errors for the estimated regression coefficients are computed using standard results from M-estimation; see references. For either case, adjustments for clustering are provided if specified.

Syntax Details

Exactly one of `reliability` or `Sigma_error` must be specified in the call. `Sigma_error` need not be diagonal in the case of correlated measurement error across multiple error-prone covariates.

Error-prone variables must be included as linear main effects only; the current version of the code does not allow interactions among error-prone covariates, interactions of error-prone covariates with error-free covariates, or nonlinear functions of error-prone covariates. The error-prone covariates cannot be specified with any construction involving `I()`.

The current version does not allow `singular.ok=TRUE`.

It is strongly encouraged to use the `data` argument to pass a dataframe containing all variables to be used in the regression, rather than using a matrix on the right hand side of the regression formula. In addition, if `cluster_varname` is specified, everything including the clustering variable must be passed as `data`.

If `weights` is specified, a weighted version of the EIV estimator is computed using operations analogous to weighted least squares in linear regression, and a standard error for this weighted estimator is computed. Weights must be positive and will be normalized inside the function to sum to the number of observations used to fit the model. Cases with missing weights will get dropped just like cases with missing covariates.

Different software packages that compute robust standard errors make different choices about degrees-of-freedom adjustments intended to improve small-sample coverage properties. The `df_adj` argument will inflate the estimated variance/covariance matrix of the estimated regression coefficients by `N/(N-p)`; see Wooldridge (2002, p. 57). In addition, if `cluster_varname` is specified, the estimated variance/covariance matrix will be inflated by `M/(M-1)` where `M` is the number of unique clusters present in the estimation sample.

Value

An list object of class `eivlm` with the following components:

`coefficients` Estimated regression coefficients from EIV model.

`residuals` Residuals from fitted EIV model.

`rank` Column rank of regression design matrix.

`fitted.values` Fitted values from EIV model.

`N` Number of observations used in fitted model.

`Sigma_error` The measurement error covariance matrix, if supplied.

`reliability` The vector of reliabilities, if supplied.

`relnames` The names of the error-prone covariates.

`XpX_adj` The cross-product matrix of the regression, adjusted for measurement error.

`varYXZ` The maximum likelihood estimate of the covariance matrix of the outcome Y, the latent covariates X and the observed, error-free covariates Z.

`latent_resvar` A degrees-of-freedom adjusted estimate of the residual variance of the latent regression. NOTE: this not an estimate of the residual variance of the regression on the observed covariates (W, Z), but rather an estimate of the residual variance of the regression on (X, Z).

`vcov` The estimated variance/covariance matrix of the regression coefficients.

`cluster_varname`, `cluster_values`, `cluster_num`

If `cluster_varname` is specified, it is returned in the object, along with `cluster_values` providing the actual values of the clustering variable for the cases used in the fitted model, and `cluster_num`, the number of unique such clusters.

`OTHER` The object also includes components `assign`, `df.residual`, `xlevels`, `call`, `terms`, `model` and other optional components such as `weights`, depending on the call; see `lm`. In addition, the object includes components `unadj_coefficients`, `unadj_fitted.values`, `unadj_residuals`, `unadj_effects`, and `unadj_qr` that are computed from the unadjusted regression model that ignores measurement error; see `lm`.

Author(s)

J.R. Lockwood <`jrlockwood@ets.org`> modified the `lm` function to adapt it for EIV regression.

References

Carroll R.J, Ruppert D., Stefanski L.A. and Crainiceanu C.M. (2006). *Measurement Error in Nonlinear Models: A Modern Perspective* (2nd edition). London: Chapman & Hall.

Fuller W. (2006). *Measurement Error Models* (2nd edition). New York: John Wiley & Sons.

Stefanksi L.A. and Boos D.B. (2002). "The calculus of M-estimation," *The American Statistician* 56(1):29-38.

Wooldridge J. (2002). *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

See Also

lm, summary.eivlm, deconv_npmle

Examples

```
set.seed(1001)
## simulate data with covariates x1, x2 and z.
.n    <- 1000
.d    <- data.frame(x1 = rnorm(.n))
.d$x2 <- sqrt(0.5)*.d$x1 + rnorm(.n, sd=sqrt(0.5))
.d$z  <- as.numeric(.d$x1 + .d$x2 > 0)

## generate outcome
## true regression parameters are c(2,1,1,-1)
.d$y  <- 2.0 + 1.0*.d$x1 + 1.0*.d$x2 - 1.0*.d$z + rnorm(.n)

## generate error-prone covariates w1 and w2
Sigma_error <- diag(c(0.20, 0.30))
dimnames(Sigma_error) <- list(c("w1","w2"), c("w1","w2"))
.d$w1 <- .d$x1 + rnorm(.n, sd = sqrt(Sigma_error["w1","w1"]))
.d$w2 <- .d$x2 + rnorm(.n, sd = sqrt(Sigma_error["w2","w2"]))

## fit EIV regression specifying known measurement error covariance matrix
.mod1 <- eivreg(y ~ w1 + w2 + z, data = .d, Sigma_error = Sigma_error)
print(class(.mod1))
.tmp <- summary(.mod1)
print(class(.tmp))
print(.tmp)

## fit EIV regression specifying known reliabilities.  Note that
## point estimator is slightly different from .mod1 because
## the correction matrix S must be estimated when the reliability
## is known.
.lambda <- c(1,1) / (c(1,1) + diag(Sigma_error))
.mod2 <- eivreg(y ~ w1 + w2 + z, data = .d, reliability = .lambda)
print(summary(.mod2))
```

`get_bugs_wishart_scalemat`

Compute a BUGS-compliant scale matrix for a Wishart prior distribution for precision matrix that is consistent with target variances.

Description

Computes a scale matrix in the BUGS parameterization that corresponds to a minimally-informative Wishart prior distribution for a precision matrix, with the property that the medians of the diagonals of the implied prior distribution for the corresponding covariance matrix are approximately equal to specified target variances.

Usage

```
get_bugs_wishart_scalemat(target, nsim=25000, reltol = 0.05, quietly=TRUE)
```

Arguments

`target`	A `p`-dimensional vector of target variances. These are the variances that you would like your BUGS Wishart prior distribution to support.
`nsim`	Number of Monte-Carlo simulations used to set target scale matrix. Default is 25,000.
`reltol`	Relative tolerance determining when the algorithm stops trying to find a better scale matrix. Default is 0.05.
`quietly`	If `quietly` is `FALSE`, prints iterative and final diagnostic information. Default is `TRUE`.

Details

When using WinBUGS/OpenBUGS/JAGS, it is often necessary to provide a Wishart prior distribution for the precision matrix of a `p`-dimensional random vector. It is common to use a Wishart distribution with `p+1` degrees of freedom in this case. The question is what scale matrix to use. The BUGS languages parameterize the Wishart distribution such that if a precision matrix `M` is given the prior distribution `M` ~ `dwish(S,p+1)` for a `pxp` scale matrix `S` and `p+1` degrees of freedom, the expected value of `M` is `p+1` times the inverse of `S`.

The current function determines a diagonal scale matrix `S` such that the implied prior distribution for the inverse of `M`, the variance/covariance matrix of the random vector, under the distribution `M` ~ `dwish(S,p+1)` in the BUGS parameterization, has medians of the diagonal elements approximately equal to some target variances specified by `target`. It iteratively tries values of `S` via Monte Carlo simulation to select a value of `S` with the desired property.

The value of `reltol` determines how close the match must be. Larger values of `nsim` and smaller values of `reltol` will lead to smaller Monte Carlo error in the estimate scale matrix.

Value

A list with elements

- bugs.df: Degrees of freedom to use for Wishart prior distribution in BUGS, equal to `p+1` where `p` is the dimension of `target`.
- bugs.scalemat: Scale matrix to use for Wishart prior distribution in BUGS.
- varsum: Summary of prior distribution of implied variances; medians should approximately equal `target`.
- corsum: Summary of prior distribution of implied correlations.

Author(s)

J.R. Lockwood <jrlockwood@ets.org>

Examples

```
tmp <- get_bugs_wishart_scalemat(target = c(10,4,4,8), nsim = 30000,
reltol = 0.02, quietly=FALSE)
print(tmp)

## if you now model precison matrix M ~ dwish(tmp$bugs.scalemat,
## tmp$bugs.df) in a BUGS language, this will imply a prior distribution
## for the inverse of M that has medians of the diagonal elements
## approximately equal to 'target'
```

lr_ancova *Latent Regression for Group Effects with Latent Heteroskedastic Error Variance*

Description

Uses the `jags` function in R2jags to fit a latent-variable GLM with error-prone covariates that may have heteroskedastic normal measurement error with variance that is a function of the latent variable, such as commonly occurs with test scores computed using item-response-theory (IRT) models.

Usage

```
lr_ancova(outcome_model, Y, W, Z, G, varfuncs, plotfile=NULL,
seed=12345, modelfileonly=FALSE, scalemat=NULL, blockprior=TRUE, ...)
```

Arguments

`outcome_model` A character string indicating the outcome model. Valid values are currently 'normal', 'normalME', 'poisson', 'bernoulli_probit', and 'bernoulli_logit'.

`Y` A numeric vector of outcome values. Missing values are allowed.

`W` A numeric matrix of error-prone covariates. Missing values are allowed, though no column of `W` may be entirely missing.

`Z` A numeric matrix of error-free covariates. Missing values are not allowed. First column must be a vector of 1s to serve as a model intercept because effects of groups in `G` are parameterized with a sum-to-zero constraint. See Details for additional information.

`G` A numeric or factor vector indicating group memberships of units. Missing values not allowed.

`varfuncs` A list with as many components as there are error-prone covariates, equal to the number of columns of `W`. For each `i` from `1` to `ncol(W)`, `varfuncs[[i]]` is itself a list summarizing the known information about the measurement error in the variable `W[,i]`. See Details.

`plotfile` Character string providing full path to a PDF file that will store some diagnostic plots regarding the variance functions. Default is NULL and will be assigned to a file in a temporary directory and the name of file will be returned.

`seed` An integer that will be passed to `set.seed` so that Monte Carlo results can be reproduced.

`modelfileonly` If TRUE, function will return a link to a file that contains the JAGS model code, but will not actually fit the model. Default is FALSE.

`scalemat` When there are multiple error-prone covariates, the specification of the Bayesian model as implemented in JAGS requires a scale matrix for a Wishart prior distribution for a precision matrix. The default is NULL, in which case the function will set a value of `scalemat`; see Details. If the user wishes to pass a `scalemat` it must be a positive-definite symmetric matrix of dimension `ncol(W)`.

`blockprior` If TRUE (the default), specifies JAGS code to encourage updating regression model parameters as a block to improve MCMC mixing.

`...` Additional arguments to `jags`.

Details

Theory

The outcome Y is assumed to depend on (X, Z, G) where X is a vector of latent variables, Z is a vector of observed, error-free variables, and G is a grouping variable. For example, one may be interested in the effects of some intervention where G indicates groupings of units that received different treatments, and the variables (X, Z) are potential confounders. This function addresses the case where X is unobserved, and error-prone proxies W are instead observed. It is assumed that $W = X + U$ for mean-zero, normally-distributed measurement error U, and that $Var(U)$ may be a function $g(X)$ of X. Such error structures commonly arise with the use of test scores computed using item-response-theory (IRT) models. Details on these issues and other model assumptions are provided in the references. The model is a generalization of errors-in-variables linear regression.

The model assumes that the outcome Y depends on (X, Z, G) through a linear function of these predictors, and parameters for this linear function are estimated. The conditional distribution of Y given these predictors that is assumed by the model depends on `outcome_model`. If `outcome_model` is `normal`, the conditional distribution of Y is assumed to be normal, and the model also estimates a residual variance for Y given the covariates. If `outcome_model` is `normalME`, it is assumed that there is a latent variable (call it `Yl`) that follows the same conditional distribution as when `outcome_model` is `normal`, and then Y measures `Yl` with normal measurement error and the known information about this error is passed as the last component of `varfuncs`. In this way, the `lr_ancova` can support models with heteroskedastic measurement error in both the predictors and the outcome. If `outcome_model` is `poisson`, Y must consists of non-negative integers and a log link is assumed. If `outcome_model` is `bernoulli_logit`, Y must take on values of 0 and 1, and a logit link is assumed. Finally, if `outcome_model` is `bernoulli_probit`, Y must take on values of 0 and 1, and a probit link is assumed.

The model assumes that the conditional distribution of X given (Z, G) is normal with a mean vector that depends on (Z, G) and a covariance matrix that is assumed not to depend on (Z, G). Both the regression parameters and the residual covariance matrix of this conditional distribution are estimated.

All parameters of the model involving (Y, X, Z, G) are estimated using the observed data (Y, W, Z, G) using assumptions and information about the distribution of the measurement errors U. The structure assumed here is that measurement errors are independent across units and across dimensions of X, and that the conditional distribution of U given (Y, X, Z, G) is a normal distribution with mean zero and variance $g(X)$. The function g must be specified and can be constant. Additional discussions of this class of error functions are provided in the references, and details about how information about g is conveyed to this function are provided below.

Syntax Details

Note that this function requires the R2jags package, which in turn requires JAGS to be installed on your system.

The function will check that the only column of `Z` that is in the span of the columns of the design matrix implied by the grouping variable `G` is the first column, corresponding to an intercept. The effects of `G` are parameterized with a sum-to-zero constraint, so that the effect of each group is expressed relative to the average of all group effects.

The `varfuncs` argument requires the most thinking. This argument is a list with as many elements as there are error-prone covariates, or one plus the number of error-prone covariates if `outcome_model` is `normalME`. In this latter case, the final element must be the error variance function for Y.

Each element of the list `varfuncs` is itself a list providing the measurement error information about one of the error-prone covariates (or the outcome, if `outcome_model` is `normalME`). For each `i`, `varfuncs[[i]]` must be a list following a particular structure. First, `varfuncs[[i]]$type` must be a character string taking one of four possible values: `constant`, `hknown`, `piecewise_linear` or `log_polynomial`. The case `constant` corresponds to the case of homoskedastic measurement error where $g(X)$ is constant, and the variance of this measurement error must be provided in `varfuncs[[i]]$vtab`. The case `hknown` corresponds to the case of heteroskedastic measurement error with a known value for each case, and these values must be provided as a vector in `varfuncs[[i]]$vtab`. Missing values in this vector are not allowed. The other two cases correspond to the case where the conditional measurement error variance $g(X)$ is a nontrivial function of X. In both of these cases, `varfuncs[[i]]$vtab` must be a matrix or data frame with exactly two columns and `K` rows, where the first column provides values `x[1],...,x[K]` of X

and the second column provides values `g(x[1]),...,g(x[K])`. That is, the function $g(X)$ is conveyed via a lookup table. The value of `K` is selected by the user. Larger values of `K` will make the approximation to $g(X)$ more accurate but will cause the model estimation to proceed more slowly. How the values in the lookup table are used to approximate $g(X)$ more generally depends whether `varfuncs[[i]]$type` is `piecewise_linear` or `log_polynomial`. In the case of `piecewise_linear`, the values in the lookup table are linearly interpolated. In the case of `log_polynomial`, a polynomial of degree `varfuncs[[i]]$degree` is fitted to the logs of the values of `g(x[1]),...,g(x[K])`, and the fitted model is used to build a smooth approximation to the function $g(X)$. The default value of `varfuncs[[i]]$degree` if it is not specified is 6. For either the piecewise linear or log polynomial approximations, the function `g(X)`g(X) is set to `g(x[1])` for values of `x` smaller than `x[1]`, and is set of `g(x[K])` for values of `x` larger than `x[K]`. Diagnostic plots of the approximate variance functions saved in PDF file whose location is returned by `lr_ancova`. The Examples section provides examples that will be helpful in specifying `varfuncs`.

When there are two or more error-prone covariates, the model estimates a residual variance/covariance matrix of X given (Z, G). Because the model is fit in a Bayesian framework, a prior distribution is required for this matrix. We are using JAGS and specify a prior distribution for the inverse of the residual variance/covariance matrix using a Wishart distribution. The degrees of freedom parameter of this distribution is set to one plus `ncol(W)` to be minimally informative. The scale matrix of this distribution can be set by passing an appropriate matrix via the `scalemat` argument. If `scalemat` is NULL, the function specifies a diagonal scale matrix that attempts to make the prior medians of the unknown residual variances approximately equal to the residual variances obtained by regressing components of W on (Z, G). See `get_bugs_wishart_scalemat`. Such variances will be somewhat inflated due to measurement error in W but the prior variance of the Wishart distribution is sufficiently large that this lack of alignment should be minimally consequential in most applications. The value of `scalemat` used in the estimation is returned by the function, and users can start with the default and then pass alternative values via the `scalemat` argument for sensitivity analyses if desired.

Value

A object of class `rjags`, with additional information specific to this context. The additional information is stored as a list called `lr_ancova_extras` with the following components:

`model.location` Path to file containing JAGS model code.

`plot.location` Path to file containing diagnostic plots regarding the variance functions.

`group.map` A dataframe mapping the original group labels in `G` to integer labels ranging from `1` to the number of unique elements of `G`. These are useful for mapping the group effects reported by JAGS back to the group labels.

`scalemat` The value of `scalemat` used in the estimation.

The parameters used in the JAGS model, and thus named in the model object, use naming conventions described here. The parameters in the linear function of (X, Z, G) that is related to Y are partitioned into `betaYXZ` and `betaYG`. In applications involving analysis of causal effects of groupings, the parameters `betaYG` will generally be of most interest. When `outcome_model` is `normal`, the residual standard deviation of Y given (X, Z, G) is also estimated and is called `sdYgivenXZG`. Similarly, when `outcome_model` is `normalME`, a residual standard deviation of the latent variable corresponding to Y given (X, Z, G) is also estimated and is also called `sdYgivenXZG`. Note in this case that the residual standard deviation of Y given its corresponding latent variable is assumed to be known and specified via `varfuncs`.

The regression parameters for the conditional distribution of X given (Z, G) are partitioned as `betaXZ` and `betaXG`. The residual variance/covariance matrix for X given (Z, G) is named `varXgivenXG`. Additional details on these parameters can be found by looking at the JAGS model file whose location is returned as noted above.

Author(s)

J.R. Lockwood <jrlockwood@ets.org>

References

Battauz, M. and Bellio, R. (2011). “Structural modeling of measurement error in generalized linear models with Rasch measures as covariates,” *Psychometrika* 76(1):40-56.

Lockwood J.R. and McCaffrey D.F. (2014). “Correcting for test score measurement error in ANCOVA models for estimating treatment effects,” *Journal of Educational and Behavioral Statistics* 39(1):22-52.

Lockwood J.R. and McCaffrey D.F. (2017). “Simulation-extrapolation with latent heteroskedastic variance,” *Psychometrika* 82(3):717-736.

Plummer, M. (2003). “JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling.” Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), Vienna, Austria.

Rabe-Hesketh S., Pickles A. and Skrondal A. (2003). “Correcting for covariate measurement error in logistic regression using nonparametric maximum likelihood estimation,” *Statistical Modelling* 3:215-232.

See Also

jags, get_bugs_wishart_scalemat

Examples

```
set.seed(3001)
cat("NOTE: this example uses MCMC and takes a little while to run\n")

## example of estimating school "value-added" effects on math test scores,
## adjusting for lag 1 math and ELA scores and accounting for the
## heteroskedastic measurement errors in those scores.
data(testscores)
print(length(unique(testscores$schoolid)))

## to help interpretation of model coefficients and school effects, standardize
## current and lag 1 test scores to have mean zero and variance 1.  Also adjust
## the conditional standard errors of measurement for the lag 1 scores.
testscores$math <- as.vector(scale(testscores$math))

testscores$math_lag1_csem <- testscores$math_lag1_csem / sd(testscores$math_lag1)
```

```
testscores$math_lag1      <- as.vector(scale(testscores$math_lag1))

testscores$lang_lag1_csem <- testscores$lang_lag1_csem / sd(testscores$lang_lag1)
testscores$lang_lag1      <- as.vector(scale(testscores$lang_lag1))

## create pieces needed to call lr_ancova.  Note that first column of Z
## must be an intercept.
outcome_model <- "normal"
Y             <- testscores$math
W             <- testscores[,c("math_lag1","lang_lag1")]
Z             <- cbind(1, testscores[,c("sped","frl")])
G             <- testscores$schoolid

## create varfuncs.  Need to be careful to pass conditional measurement error
## variances, which require squaring the CSEMs
varfuncs   <- list()

tmp        <- unique(testscores[,c("math_lag1","math_lag1_csem")])
names(tmp) <- c("x","gx")
tmp        <- tmp[order(tmp$x),]
tmp$gx     <- tmp$gx^2
varfuncs[[1]] <- list(type="log_polynomial", vtab=tmp)

tmp        <- unique(testscores[,c("lang_lag1","lang_lag1_csem")])
names(tmp) <- c("x","gx")
tmp        <- tmp[order(tmp$x),]
tmp$gx     <- tmp$gx^2
varfuncs[[2]] <- list(type="log_polynomial", vtab=tmp)

## fit the model.  NOTE: in practice, larger values of n.iter and n.burnin
## would typically be used; they are kept small here so that the example
## runs relatively quickly.
m1 <- lr_ancova(outcome_model, Y, W, Z, G, varfuncs, n.iter=300, n.burnin=100)

## you can check the approximation to the variance functions by looking at the
## PDF file:
print(m1$lr_ancova_extras$plot.location)

## and also can look at the JAGS model file:
print(m1$lr_ancova_extras$model.location)

## the model object is of class "rjags" and so inherits the appropriate methods,
## including print:
print(m1)
## betaXG, betaXZ, and varXgivenZG are for the conditional distribution of X
## given (Z,G).  betaYG, betaYXZ and sdYgivenXZG are for the conditional
## distribution of Y given (X,Z,G).
##
## the first two elements of betaYXZ are the coefficients for the two columns of
## X, whereas the following three elements are the coefficients for the three
## columns of Z.
##
## the school effects are in betaYG.  extract their posterior means and
## posterior standard deviations:
e <- m1$BUGSoutput$summary
e <- as.data.frame(e[grep("betaYG",rownames(e)),c("mean","sd")])
## check the sum-to-zero constraints:
print(sum(e$mean))
## put the actual school IDs onto "e"
e$schoolid <- m1$lr_ancova_extras$group.map$G
print(e)

## compare the school effect estimates to those from a simpler model that does
## not adjust for the lag 1 ELA score, and does not account for the measurement
## error in the lag 1 math score.  Use sum-to-zero contrasts and recover the
## estimate for the last school as negative the sum of the other estimates.
testscores$schid <- factor(testscores$schoolid)
m0 <- lm(math ~ math_lag1 + sped + frl + schid,
         data=testscores, contrasts=list(schid = "contr.sum"))
s  <- coef(m0)[grep("schid", names(coef(m0)))]
e$est_m0 <- c(s, -sum(s))

## Such estimates should have some amount of omitted variable bias, which
## should manifest as the differences between the "m0" and "m1" estimates
## being positively correlated with average prior achievement.
print(cor(tapply(testscores$math_lag1, testscores$schoolid, mean), e$est_m0 - e$mean))
print(cor(tapply(testscores$lang_lag1, testscores$schoolid, mean), e$est_m0 - e$mean))
```

model.matrix.eivlm — model.matrix *method for objects of class* eivlm.

Description

Extract model matrix from eivlm object. Analogous to model.matrix method for lm objects.

Usage

```
## S3 method for class 'eivlm'
model.matrix(object, ...)
```

Arguments

object	A model object of class eivlm.
...	See help for model.matrix.

Value

Design matrix used in EIV regression.

print.eivlm — print *method for objects of class* eivlm.

Description

Analogous to print method for lm objects.

Usage

```
## S3 method for class 'eivlm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	A model object of class eivlm.
digits, ...	See help for print.default.

print.summary.eivlm — print *method for objects of class* summary.eivlm.

Description

Similar to print method for summaries of lm objects, but provides additional information specific to the EIV regression. The summary method for objects of class eivlm returns an object of class summary.eivlm.

Usage

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

Arguments

x	A model object of class summary.eivlm.
digits, symbolic.cor, signif.stars, ...	See help for summary.lm.

See Also

See help for summary.eivlm for description of quantities relevant to summarizing eivlm objects.

summary.eivlm — summary *method for objects of class* eivlm.

Description

Computes summary quantities for a model of class `eivlm`. The computations include some quantities for the standard regression model (uncorrected for covariate measurement error), as well as quantities relevant to the EIV model.

Usage

```
## S3 method for class 'eivlm'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)
```

Arguments

`object` A model object of class `eivlm`.

`correlation, symbolic.cor, ...`
See help for `summary.lm`

Value

An list object of class `summary.eivlm` with components:

`residuals, fitted.values, N, latent_resvar, vcov, relnames, coefficients`
See `eivreg`.

`call, terms, aliased, df, coefficients`
See `summary.lm`.

`reliability` If `reliability` is specified in model fit.

`Sigma_error` If `Sigma_error` is specified in model fit.

`symbolic.cor` If `symbolic.cor` is specified in call to function.

`latent_R2` Maximum likelihood estimate of R-squared for regression of Y on (X,Z). NOTE: This is not the R-squared of the fitted regression of Y on (W,Z).

`latent_R2_dfadj`
Estimate of R-squared for regression of Y on (X,Z) adjusted by number of estimated regression parameters.

`OTHER` The object also includes components `unadj_residuals`, `unadj_coefficients`, `unadj_sigma`, `unadj_r.squared`, `unadj_adj.r.squared`, `unadj_fstatistic` and `unadj_cov.unscaled` that are computed from the "unadjusted" regression model that ignores measurement error; see `summary.lm`. Also, additional components will be included if either `weights` or `cluster_varname` is specified in the call to `eivlm`; see that help file for details.

Author(s)

J.R. Lockwood <`jrlockwood@ets.org`>

See Also

The model fitting function eivreg, summary.

Function coef will extract the matrix of coefficients with standard errors, t-statistics and p-values.

`testscores` *Example longitudinal test score data*

Description

Cohort of grade 6 students with mathematics test scores from a target school year, as well as mathematics and language test scores from the prior school year (grade 5). Measurement error in test scores quantified by conditional standard error of measurement (CSEM).

Usage

```
data(testscores)
```

Format

A data frame with 4853 observations and 10 fields:

`stuid` Unique identifier for each student (one record per student)

`schoolid` Unique identifier for each student's grade 6 school

`math` Grade 6 mathematics test score

`math_csem` CSEM for grade 6 mathematics test score

`math_lag1` Grade 5 mathematics test score

`math_lag1_csem` CSEM for grade 5 mathematics test score

`lang_lag1` Grade 5 language test score

`lang_lag1_csem` CSEM for grade 5 language test score

`sped` 1 = student designated as special education; 0 otherwise

`frl` 1 = student participates in Free and Reduced Price lunch program; 0 otherwise

Source

Anonymous

vcov.eivlm — vcov *method for objects of class* eivlm.

Description

Extract variance/covariance matrix of estimated parameters from eivlm model object. Analogous to vcov method for other models.

Usage

```
## S3 method for class 'eivlm'
vcov(object, ...)
```

Arguments

object	A model object of class eivlm.
...	Not currently implemented.

Value

Estimated variance/covariance matrix of estimated regression coefficients.

Index