Package 'lavaSearch2'

October 5, 2018

Type Package

Title Tools for Model Specification in the Latent Variable Framework

Version 1.4

Date 2018-10-05

Maintainer Brice Ozenne <brice.ozenne@orange.fr>

URL https://github.com/bozenne/lavaSearch2

BugReports https://github.com/bozenne/lavaSearch2/issues

Description Tools for model specification in the latent variable framework
(add-on to the 'lava' package). The package contains three main functionalities:
Wald tests/F-tests with improved control of the type 1 error in small samples,
adjustment for multiple comparisons when searching for local dependencies,
and adjustment for multiple comparisons when doing inference for multiple latent variable models.

License GPL-3

VignetteBuilder R.rsp

Depends R (>= 2.10), lava, ggplot2

Imports doParallel, MASS, Matrix, methods, multcomp, mvtnorm, nlme,
parallel, reshape2, sandwich, stats, utils

Suggests clubSandwich, data.table, foreach, lava.tobit, lme4,
lmerTest, numDeriv, pbapply, pbkrtest, R.rsp, riskRegression,
survival, testthat

RoxygenNote 6.1.0.9000

NeedsCompilation no

Author Brice Ozenne [aut, cre] (<https://orcid.org/0000-0001-9694-2956>)

Repository CRAN

Date/Publication 2018-10-05 09:20:03 UTC

R topics documented:

addLink *Add a New Link Between Two Variables in a LVM*

Description

Generic interface to add links to lvm objects.

Usage

```
addLink(object, ...)

## S3 method for class 'lvm'
addLink(object, var1, var2, covariance,
  all.vars = lava::vars(object), warnings = FALSE, ...)

## S3 method for class 'lvm.reduced'
addLink(object, ...)
```

Arguments

object	a lvm object.
...	[internal] only used by the generic method and from addLink.lvm.reduced to addLink.lvm.
var1	[character or formula] the exogenous variable of the new link or a formula describing the link to be added to the lvm.
var2	[character] the endogenous variable of the new link. Disregarded if the argument var1 is a formula.
covariance	[logical] is the link is bidirectional? Ignored if one of the variables non-stochastic (e.g. exogenous variables).
all.vars	[internal] a character vector containing all the variables of the lvm object.
warnings	[logical] Should a warning be displayed when no link is added?

Details

The argument all.vars is useful for lvm.reduce object where the command vars(object) does not return all variables. The command vars(object, xlp = TRUE) must be used instead.

Arguments var1 and var2 are passed to initVarlink.

Examples

```
library(lava)
set.seed(10)

m <- lvm()
regression(m) <- c(y1,y2,y3)~u
regression(m) <- u~x1+x2
```

```
latent(m) <- ~u
m2 <- m

addLink(m, x1 ~ y1, covariance = FALSE)
addLink(m, y1 ~ x1, covariance = FALSE)
coef(addLink(m, y1 ~ y2, covariance = TRUE))

addLink(m2, "x1", "y1", covariance = FALSE)
addLink(m2, "y1", "x1", covariance = FALSE)
newM <- addLink(m, "y1", "y2", covariance = TRUE)
coef(newM)
```

`autoplot.intDensTri` *2D-display of the Domain Used to Compute the Integral*

Description

2D-display of the domain used to compute the integral.

Usage

```
## S3 method for class 'intDensTri'
autoplot(object, coord.plot = c("x", "y1"),
  plot = TRUE, ...)
```

Arguments

`object`	output of the function `intDensTri`.
`coord.plot`	[character vector] the x and y coordinates. Can be `"x"`, `"y1"` to `"yd"`, `"z"` if `zmin` was specified when calling `intDensTri`.
`plot`	[logical] should the plot be displayed?
`...`	[internal] Only used by the generic method.

Value

A `ggplot` object.

See Also

`intDensTri`

autoplot_calibrateType1 *Graphical Display of the Bias or Type 1 Error*

Description

Graphical display of the bias or type 1 error for the output of `calibrateType1`.

Usage

```
## S3 method for class 'calibrateType1'
autoplot(object, type = "bias", plot = TRUE,
  color.threshold = "red", type.bias = "absolute", alpha = 0.05,
  nrow.legend = NULL, name2label = NULL, color = NULL,
  keep.method = NULL, ...)
```

Arguments

`object`	output of `calibrateType1`.
`type`	[character] if type equals `"bias"` the bias will be displayed. Otherwise if it equals `"type1error"` the type 1 error will be displayed.
`plot`	[logical] should the plot be displayed?
`color.threshold`	[character] the color for the line representing the expected value(s).
`type.bias`	[character] if type.bias equals `"absolute"` the absolute bias will be used. Otherwise if it equals `"relative"` the relative bias will be used. Only relevant when type equals `"bias"`.
`alpha`	[numeric, 0-1] the significance threshold to consider. Only relevant when type equals `"type1error"`.
`nrow.legend`	[integer, >0] the number of rows for the legend. Only relevant when type equals `"type1error"`.
`name2label`	[named character vector] the label for the legend. The vector should contain the method names (see details). Only relevant when type equals `"type1error"`.
`color`	[character vector] a vector of colours to be used to color the lines. Only relevant when type equals `"type1error"`.
`keep.method`	[character vector] the methods names for which the type 1 error should be displayed. Only relevant when type equals `"type1error"`.
`...`	[internal] Only used by the generic method.

Details

Method names:

- `p.Ztest`

- p.Satt
- p.KR
- p.robustZtest
- p.robustSatt
- p.robustKR

Value

An list containing:

- plot: a ggplot object.
- data: the dataset used to generate the ggplot object.

`autplot-modelsearch2` *Display the Value of a Coefficient across the Steps.*

Description

Display the value of a coefficient across the steps.

Usage

```
## S3 method for class 'modelsearch2'
autoplot(object, param, ci = TRUE,
  step = 0:nStep(object), plot = TRUE, add.0 = TRUE, ...)
```

Arguments

object	a modelsearch2 object.
param	[character vector] the name of the coefficient(s) to be displayed.
ci	[logical] should the confidence intervals of the coefficient(s) be displayed.
step	[integer >0] the steps at which the coefficient value should be displayed.
plot	[logical] should the graph be displayed?
add.0	[logical] should an horizontal line representing no effect be displayed?
...	[internal] only used by the generic method.

Value

A list containing

- plot: a ggplot object.
- data: the data used to generate the ggplot object.

Examples

```
## Not run:
mSim <- lvm(Y~G+X1+X2+X3+X4+X5)
addvar(mSim) <- ~Z1+Z2

set.seed(10)
df.data <- lava::sim(mSim, 1e2)

mBase <- lvm(Y~G)
addvar(mBase) <- ~X1+X2+X3+X4+X5+Z1+Z2
e.lvm <- estimate(mBase, data = df.data)
res <- modelsearch2(e.lvm, method.p.adjust = "holm", alpha = 0.05)
autoplot(res, param = "Y~G")
autoplot(res, param = c("Y","Y~G"))

## End(Not run)
```

`calcDistMax` *Adjust the p.values Using the Quantiles of the Max Statistic*

Description

Adjust the p.values using the quantiles of the max statistic.

Usage

```
calcDistMaxIntegral(statistic, iid, df, iid.previous = NULL,
  quantile.previous = NULL,
  quantile.compute = lava.options()$search.calc.quantile.int, alpha,
  cpus = 1, cl = NULL, trace)

calcDistMaxBootstrap(statistic, iid, iid.previous = NULL,
  quantile.previous = NULL, method, alpha, cpus = 1, cl = NULL,
  n.sim, trace, n.repmax = 100)
```

Arguments

`statistic`	[numeric vector] the observed Wald statistic. Each statistic correspond to a null hypothesis (i.e. a coefficient) that one wish to test.
`iid`	[matrix] zero-mean iid decomposition of the coefficient used to compute the statistic.
`df`	[numeric] the degree of freedom defining the multivariate Student's t distribution. If `NULL` the multivariate Gaussian distribution will be used instead.
`iid.previous`	[matrix, EXPERIMENTAL] zero-mean iid decomposition of previously tested coefficient.

quantile.previous
: [numeric, EXPERIMENTAL] rejection quantiles of the previously tested hypotheses. If not NULL the values should correspond the variable in to the first column(s) of the argument iid.previous.

quantile.compute
: [logical] should the rejection quantile be computed?

alpha
: [numeric 0-1] the significance cutoff for the p-values. When the p-value is below, the corresponding link will be retained.

cpus
: [integer >0] the number of processors to use. If greater than 1, the computation of the p-value relative to each test is performed in parallel.

cl
: [cluster] a parallel socket cluster generated by parallel::makeCluster that has been registered using registerDoParallel.

trace
: [logical] should the execution of the function be traced?

method
: [character] the method used to compute the p-values. See the output of lava.options()$search.calcMa for the possible values.

n.sim
: [integer >0] the number of bootstrap simulations used to compute each p-values. Disregarded when the p-values are computed using numerical integration.

n.repmax
: [integer >0] the maximum number of rejection for each bootstrap sample before switching to a new bootstrap sample. Only relevant when conditioning on a previous test. Disregarded when the p-values are computed using numerical integration.

Value

A list containing

- p.adjust: the adjusted p-values.
- z: the rejection threshold.
- Sigma: the correlation matrix between the test statistic.
- correctedLevel: the alpha level corrected for conditioning on previous tests.

Examples

```
library(mvtnorm)

set.seed(10)
n <- 100
p <- 4
link <- letters[1:p]
n.sim <- 1e3 # number of bootstrap simulations

#### test - not conditional ####
X.iid <- rmvnorm(n, mean = rep(0,p), sigma = diag(1,p))
colnames(X.iid) <- link
statistic <- setNames(1:p,link)
```

```
r1 <- calcDistMaxIntegral(statistic = statistic, iid = X.iid,
            trace = FALSE, alpha = 0.05, df = 1e6)

r3 <- calcDistMaxBootstrap(statistic = statistic, iid = X.iid,
            method = "residual",
            trace = FALSE, alpha = 0.05, n.sim = n.sim)

r4 <- calcDistMaxBootstrap(statistic = statistic, iid = X.iid,
            method = "wild",
            trace = FALSE, alpha = 0.05, n.sim = n.sim)

rbind(integration = c(r1$p.adjust, quantile = r1$z),
      bootResidual = c(r3$p.adjust, quantile = r3$z),
      bootWild    = c(r4$p.adjust, quantile = r4$z))

#### test - conditional ####
## Not run:
Z.iid <- rmvnorm(n, mean = rep(0,p+1), sigma = diag(1,p+1))
seqQuantile <- qmvnorm(p = 0.95, delta = rep(0,p+1), sigma = diag(1,p+1),
                   tail = "both.tails")$quantile

r1c <- calcDistMaxIntegral(statistic = statistic, iid = X.iid,
            iid.previous = Z.iid, quantile.previous =  seqQuantile,
            trace = FALSE, alpha = 0.05, df = NULL)

r3c <- calcDistMaxBootstrap(statistic = statistic, iid = X.iid,
            iid.previous = Z.iid, quantile.previous =  seqQuantile, method = "residual",
            trace = FALSE, alpha = 0.05, n.sim = n.sim)

r4c <- calcDistMaxBootstrap(statistic = statistic, iid = X.iid,
            iid.previous = Z.iid, quantile.previous =  seqQuantile, method = "wild",
            trace = FALSE, alpha = 0.05, n.sim = n.sim)

rbind(integration = c(r1c$p.adjust, quantile = r1c$z),
      bootResidual = c(r3c$p.adjust, quantile = r3c$z),
      bootWild    = c(r4c$p.adjust, quantile = r4c$z))

## End(Not run)
```

`calcType1postSelection` *Compute the Type 1 Error After Selection*

Description

Compute the type 1 error after selection.

Usage

```
calcType1postSelection(level, mu, Sigma, quantile.previous, distribution,
  df, n = 10, correct = TRUE, ...)
```

Arguments

`level`	[numeric 0-1] expected coverage.
`mu`	[numeric vector] the expectation of the joint distribution of the test statistics
`Sigma`	[matrix] the variance-covariance of the joint distribution of the test statistics.
`quantile.previous`	[numeric] significance quantile used at the previous step.
`distribution`	[character] distribution of the test statistics. Can be `"pmvnorm"` (normal distribution) or `"pvmt"` (Student's t distribution)
`df`	[integer > 0] the degree of freedom of the joint Student's t distribution. Only used when `distribution="pvmt"`.
`n`	[integer > 0] number of points for the numerical integration
`correct`	[logical] if true, correct the level to account for previous testings.
`...`	arguments passed to `intDensTri`.

Details

The number of tests at the current step (i.e. after selection) is assumed to be one less than the number of tests at the previous step (i.e. before selection).

Arguments `mu` and `Sigma` must contain the moments for the vector of test statistics before and after selection (in that order).

Value

numeric the type 1 error.

Author(s)

Brice Ozenne

Examples

```
library(mvtnorm)
n <- 350

#### only 2 tests
Sigma <- rbind(c(1,0,0),c(0,1,1),c(0,1,1))
z2 <- qmvnorm(0.95, mean = rep(0,2), sigma = Sigma[1:2,1:2], tail = "both.tails")$quantile

## no selection since strong effect
mu <- c(10,0,0)
calcType1postSelection(0.95, quantile.previous = z2, distribution = "gaussian",
                       mu = mu, Sigma = Sigma, correct = TRUE)

## strong selection
## Not run:
mu <- c(0,0,0)
levelC <- calcType1postSelection(0.95, quantile.previous = z2, distribution = "gaussian",
```

```
                        mu = mu, Sigma = Sigma)
print(levelC) # more liberal than without selection
calcType1postSelection(levelC, quantile.previous = z2, distribution = "gaussian",
                        mu = mu, Sigma = Sigma, correct = FALSE)

## End(Not run)

#### 3 tests
Sigma <- diag(1,5,5)
Sigma[4,2] <- 1
Sigma[2,4] <- 1
Sigma[5,3] <- 1
Sigma[3,5] <- 1

z2 <- qmvnorm(0.95, mean = mu[1:3], sigma = Sigma[1:3,1:3], tails = "both.tails")$quantile

## no selection since strong effect
## Not run:
mu <- c(10,0,0,0,0)
calcType1postSelection(0.95, quantile.previous = z2, distribution = "gaussian",
                        mu = mu, Sigma = Sigma, correct = TRUE)

## strong selection
mu <- c(0,0,0,0,0)
levelC <- calcType1postSelection(0.95, quantile.previous = z2,
                        mu = mu, Sigma = Sigma, distribution = "gaussian")
calcType1postSelection(levelC, quantile.previous = z2, distribution = "gaussian",
                        mu = mu, Sigma = Sigma, correct = FALSE)

## End(Not run)
```

`calibrateType1` *Simulation Study Assessing Bias and Type 1 Error*

Description

Perform a simulation study over one or several sample size to assess the bias of the estimate and the type 1 error of the Wald test and robust Wald test

Usage

```
calibrateType1(object, null, n.rep, ...)

## S3 method for class 'lvm'
calibrateType1(object, null, n.rep, n, F.test = FALSE,
  cluster = NULL, generative.object = NULL, generative.coef = NULL,
  true.coef = NULL, n.true = 1e+06, round.true = 2,
  bootstrap = FALSE, n.bootstrap = 1000, checkType1 = FALSE,
```

```
  checkType2 = FALSE, dir.save = NULL, label.file = NULL,
  seed = NULL, cpus = 1, trace = 2, ...)

## S3 method for class 'lvmfit'
calibrateType1(object, null, n.rep, F.test = FALSE,
  bootstrap = FALSE, n.bootstrap = 1000, seed = NULL, trace = 2,
  cpus = 1, ...)
```

Arguments

object	a lvm object defining the model to be fitted.
null	[character vector] names of the coefficient whose value will be tested against 0.
n.rep	[integer, >0] number of simulations per sample size.
...	[internal] Only used by the generic method.
n	[integer vector, >0] sample size(s) considered in the simulation study.
F.test	[logical] should a multivariate Wald test be perform testing simultaneously all the null hypotheses?
cluster	[integer vector] the grouping variable relative to which the observations are iid. Will be passed to lava::estimate.
generative.object	[lvm] object defining the statistical model generating the data.
generative.coef	[name numeric vector] values for the parameters of the generative model. Can also be NULL: in such a case the coefficients are set to default values decided by lava (usually 0 or 1).
true.coef	[name numeric vector] expected values for the parameters of the fitted model.
n.true	[integer, >0] sample size at which the estimated coefficients will be a reliable approximation of the true coefficients.
round.true	[integer, >0] the number of decimal places to be used for the true value of the coefficients. No rounding is done if NULL.
bootstrap	[logical] should bootstrap resampling be performed?
n.bootstrap	[integer, >0] the number of bootstrap sample to be used for each bootstrap.
checkType1	[logical] returns an error if the coefficients associated to the null hypotheses do not equal 0.
checkType2	[logical] returns an error if the coefficients associated to the null hypotheses equal 0.
dir.save	[character] path to the directory were the results should be exported. Can also be NULL: in such a case the results are not exported.
label.file	[character] element to include in the file name.
seed	[integer, >0] seed value that will be set at the beginning of the simulation to enable eproducibility of the results. Can also be NULL: in such a case no seed is set.
cpus	[integer >0] the number of processors to use. If greater than 1, the simulations are performed in parallel.
trace	[integer] should the execution of the function be trace. Can be 0, 1 or 2.

Value

An object of class `calibrateType1`.

Author(s)

Brice Ozenne

See Also

`link{autoplot.calibrateType1}` for a graphical display of the bias or of the type 1 error.

Examples

```
#### simulate data ####
m.Sim <- lvm(c(Y1[mu1:sigma]~1*eta,
               Y2[mu2:sigma]~1*eta,
               Y3[mu3:sigma]~1*eta,
               eta~beta1*Group+beta2*Gender))
latent(m.Sim) <- ~eta
categorical(m.Sim, labels = c("M","F")) <- ~Gender

d <- lava::sim(m.Sim, 1e2)


#### calibrate type 1 error on the estimated model ####
m <- lvm(Y1~eta,
         Y2~eta,
         Y3~eta,
         eta~Group+Gender)
e <- lava::estimate(m, data = d)

## Not run:
res <- calibrateType1(e, null = "eta~Group", n.rep = 100)
res <- calibrateType1(e, null = "eta~Group", n.rep = 100, cpus = 4)

## End(Not run)
summary(res)
```

`checkData` *Check that Validity of the Dataset*

Description

Check whether the dataset can be used to fit the `lvm` object.

Usage

```
checkData(object, data, trace)

## S3 method for class 'lvm'
checkData(object, data, trace = TRUE)
```

Arguments

object	a lvm object.
data	[data.frame] the dataset used to obtain the object.
trace	[logical] when TRUE, the outcome of the check will be displayed.

Value

Invisible TRUE or FALSE.

Examples

```
m <- lvm()
regression(m) <- c(y1,y2,y3)~u
regression(m) <- u~x
latent(m) <- ~u

d <- lava::sim(m,1e2)

try(checkData(m, data = d)) # return an error

checkData(m, data = d[,-4])

try(checkData(m, data = d[,-(3:4)])) # return an error
```

---

coefByType *Extract the Coefficient by Type*

Description

Extract specific types of coefficient from a lvm object: covariance coefficient(s) (coefCov), extra parameter(s) (coefExtra), position in the list of models for each coefficient (coefIndexModel), intercept coefficient(s) (coefIntercept), coefficient(s) that are used as reference (coefRef), regression coefficient(s) (coefReg), variance coefficient(s) (coefVar).

Usage

```
coefCov(object, value, keep.var, ...)

## S3 method for class 'lvm'
coefCov(object, value = FALSE, keep.var = FALSE, ...)

## S3 method for class 'lvmfit'
coefCov(object, value = FALSE, keep.var = FALSE, ...)

## S3 method for class 'multigroup'
coefCov(object, value = FALSE, keep.var = FALSE,
  ...)

coefExtra(object, value, ...)

## S3 method for class 'lvm'
coefExtra(object, value = FALSE, ...)

## S3 method for class 'lvmfit'
coefExtra(object, value = FALSE, ...)

## S3 method for class 'multigroup'
coefExtra(object, value = FALSE, ...)

coefIndexModel(object, ...)

## S3 method for class 'lvm'
coefIndexModel(object, ...)

## S3 method for class 'lvmfit'
coefIndexModel(object, ...)

## S3 method for class 'multigroup'
coefIndexModel(object, ...)

## S3 method for class 'multigroupfit'
coefIndexModel(object, ...)

coefIntercept(object, value, ...)

## S3 method for class 'lvm'
coefIntercept(object, value = FALSE, ...)

## S3 method for class 'lvmfit'
coefIntercept(object, value = FALSE, ...)

## S3 method for class 'multigroup'
coefIntercept(object, value = FALSE, ...)
```

```
coefRef(object, value, ...)

## S3 method for class 'lvmfit'
coefRef(object, value = FALSE, ...)

coefReg(object, value, ...)

## S3 method for class 'lvm'
coefReg(object, value = FALSE, ...)

## S3 method for class 'lvmfit'
coefReg(object, value = FALSE, ...)

## S3 method for class 'multigroup'
coefReg(object, value = FALSE, ...)

coefVar(object, value, ...)

## S3 method for class 'lvm'
coefVar(object, value = FALSE, ...)

## S3 method for class 'lvmfit'
coefVar(object, value = FALSE, ...)

## S3 method for class 'multigroup'
coefVar(object, value = FALSE, ...)
```

Arguments

object	a lvm model or a fitted lvm model
value	should the name of the coefficient be returned? Else return the coefficients
keep.var	should the variance coefficients be returned?
...	arguments to be passed to

Value

A vector containing the names of the positions of the coefficients.

Examples

```
#### regression ####
m <- lvm(Y~X1+X2)
e <- estimate(m, lava::sim(m, 1e2))

coefCov(m)
coefCov(m, value = TRUE)

coefCov(m, keep.var = TRUE)
```

```
coefCov(m, value = TRUE, keep.var = TRUE)

coefIndexModel(m)
coefIndexModel(e)

coefIntercept(m)
coefIntercept(m, value = TRUE)

coefReg(m)
coefReg(m, value = TRUE)

#### LVM ####
m <- lvm()
regression(m) <- c(y1,y2,y3)~u
regression(m) <- u~x1+x2
latent(m) <- ~u
covariance(m) <- y1~y2

m.Sim <- m
categorical(m.Sim, labels = c("a","b","c")) <- ~x2
e <- estimate(m, lava::sim(m.Sim, 1e2))

coefCov(m)
coefCov(m, value = TRUE)

coefCov(m, keep.var = TRUE)
coefCov(m, value = TRUE, keep.var = TRUE)

coefExtra(m)

coefIndexModel(m)
coefIndexModel(e)

## additional categorical variable
categorical(m, labels = as.character(1:3)) <- "X1"

coefExtra(m)
coefExtra(m, value = TRUE)

## additional categorical variable
categorical(m, labels = as.character(1:3)) <- "x1"

coefIntercept(m)
coefIntercept(m, value = TRUE)
coefIntercept(e)

coefReg(e, value = TRUE)

#### multigroup ####
m <- lvm(Y~X1+X2)
eG <- estimate(list(m,m), list(lava::sim(m, 1e2), lava::sim(m, 1e2)))

coefIndexModel(eG)
```

`coefType` *Extract the Type of Each Coefficient*

Description

Extract the type of each coefficient of a `lvm` object.

Usage

```
coefType(object, as.lava, ...)

## S3 method for class 'lvm'
coefType(object, as.lava = TRUE, data = NULL, ...)

## S3 method for class 'lvmfit'
coefType(object, as.lava = TRUE, ...)

## S3 method for class 'multigroup'
coefType(object, as.lava = TRUE, ...)
```

Arguments

`object`	a `lvm` or `lvmfit` object.
`as.lava`	[logical] export the type of coefficients mimicking `lava:::coef`.
`...`	arguments to be passed to `lava::coef`
`data`	[data.frame, optional] the dataset. Help to identify the categorical variables.

Details

A lvm can be written as a measurement model:

$$Y_i = \nu + \Lambda\eta_i + KX_i + \epsilon_i$$

and a structural model:

$$\eta_i = \alpha + B\eta_i + \Gamma X_i + \zeta_i$$

where Ψ is the variance covariance matrix of the residuals ζ
and Σ is the variance covariance matrix of the residuals ϵ.

`coefType` either returns the Latin/Greek letter corresponding to the coefficients or it groups them:

- intercept: ν and α.
- regression: Λ, K, B, and Γ.
- covariance: extra-diagonal terms of Σ and Ψ.

- variance: diagonal of Σ and Ψ.

A link denotes a relationship between two variables. The coefficient are used to represent the strength of the association between two variable, i.e. the strength of a link. A coefficient may corresponds to the strength of one or several link.

Value

`coefType` returns a `data.frame` when `as.lava=FALSE`:

- name: name of the link
- Y: outcome variable
- X: regression variable in the design matrix (could be a transformation of the original variables, e.g. dichotomization).
- data: original variable
- type: type of link
- value: if TRUE, the value of the link is set and not estimated.
- marginal: if TRUE, the value of the link does not impact the estimation.
- detail: a more detailed description of the type of link (see the details section)
- lava: name of the coefficient in lava

When `as.lava=TRUE`, `coefType` returns a named vector containing the type of each coefficient.

Examples

```
#### regression ####
m <- lvm(Y~X1+X2)
e <- estimate(m, lava::sim(m, 1e2))

coefType(m)
coefType(e)

#### LVM ####
m <- lvm()
regression(m) <- c(y1,y2,y3)~u
regression(m) <- u~x1+x2
latent(m) <- ~u
covariance(m) <- y1~y2

m.Sim <- m
categorical(m.Sim, labels = c("a","b","c")) <- ~x2
e <- estimate(m, lava::sim(m.Sim, 1e2))

coefType(m)
coefType(e)

## additional categorical variables
categorical(m, labels = as.character(1:3)) <- "X1"

coefType(m, as.lava = FALSE)
```

```
#### LVM with constrains ####
m <- lvm(c(Y1~0+1*eta1,Y2~0+1*eta1,Y3~0+1*eta1,
          Z1~0+1*eta2,Z2~0+1*eta2,Z3~0+1*eta2))
latent(m) <- ~eta1 + eta2
e <- estimate(m, lava::sim(m,1e2))

coefType(m)
coefType(e)

#### multigroup ####
m <- lvm(Y~X1+X2)
eG <- estimate(list(m,m), list(lava::sim(m, 1e2), lava::sim(m, 1e2)))
coefType(eG)
```

`compare2` *Test Linear Hypotheses with small sample correction*

Description

Test Linear Hypotheses using a multivariate Wald statistic. Similar to `lava::compare` but with small sample correction.

Usage

```
compare2(object, df, bias.correct, ...)

## S3 method for class 'lm'
compare2(object, df = TRUE, bias.correct = TRUE, ...)

## S3 method for class 'gls'
compare2(object, df = TRUE, bias.correct = TRUE,
  cluster = NULL, ...)

## S3 method for class 'lme'
compare2(object, df = TRUE, bias.correct = TRUE, ...)

## S3 method for class 'lvmfit'
compare2(object, df = TRUE, bias.correct = TRUE,
  cluster = NULL, ...)

## S3 method for class 'lm2'
compare2(object, ...)

## S3 method for class 'gls2'
compare2(object, ...)
```

```
## S3 method for class 'lme2'
compare2(object, ...)

## S3 method for class 'lvmfit2'
compare2(object, ...)

.compare2(object, par = NULL, contrast = NULL, null = NULL,
  rhs = NULL, robust = FALSE, cluster = NULL, df = TRUE,
  as.lava = TRUE, F.test = TRUE, level = 0.95)
```

Arguments

object	an object that inherits from lm/gls/lme/lvmfit.
df	[logical] should the degree of freedoms of the Wald statistic be computed using the Satterthwaite correction? Otherwise the degree of freedoms are set to Inf, i.e. a normal distribution is used instead of a Student's t distribution when computing the p-values.
bias.correct	[logical] should the standard errors of the coefficients be corrected for small sample bias? Argument passed to sCorrect.
...	[internal] only used by the generic method.
cluster	[integer vector] the grouping variable relative to which the observations are iid.
par	[vector of characters] expression defining the linear hypotheses to be tested. See the examples section.
contrast	[matrix] a contrast matrix defining the left hand side of the linear hypotheses to be tested.
null, rhs	[vector] the right hand side of the linear hypotheses to be tested.
robust	[logical] should the robust standard errors be used instead of the model based standard errors?
as.lava	[logical] should the output be similar to the one return by lava::compare?
F.test	[logical] should a joint test be performed?
level	[numeric 0-1] the confidence level of the confidence interval.

Details

The par argument or the arguments contrast and null (or equivalenty rhs) specify the set of linear hypotheses to be tested. They can be written:

$$contrast * \theta = null$$

where θ is the vector of the model coefficients.
The par argument must contain expression(s) involving the model coefficients. For example "beta = 0" or c("-5*beta + alpha = 3","-alpha") are valid expressions if alpha and beta belong to the set of model coefficients. A contrast matrix and the right hand side will be generated inside the function.

When directly specified, the contrast matrix must contain as many columns as there are coefficients in the model (mean and variance coefficients). Each hypothesis correspond to a row in the contrast

matrix.

The null vector should contain as many elements as there are row in the contrast matrix.

Argument rhs and null are equivalent. This redondance enable compatibility between `lava::compare`, `compare2`, `multcomp::glht`, and `glht2`.

Value

If `as.lava=TRUE` an object of class `htest`. Otherwise a `data.frame` object.

See Also

`createContrast` to create contrast matrices.
`sCorrect` to pre-compute quantities for the small sample correction.

Examples

```
#### simulate data ####
set.seed(10)
mSim <- lvm(Y~0.1*X1+0.2*X2)
categorical(mSim, labels = c("a","b","c")) <- ~X1
transform(mSim, Id~Y) <- function(x){1:NROW(x)}
df.data <- lava::sim(mSim, 1e2)

#### with lm ####
## direct use of compare2
e.lm <- lm(Y~X1+X2, data = df.data)
anova(e.lm)
compare2(e.lm, par = c("X1b=0","X1c=0"))

## or first compute the derivative of the information matrix
sCorrect(e.lm) <- TRUE

## and define the contrast matrix
C <- createContrast(e.lm, par = c("X1b=0","X1c=0"), add.variance = TRUE)

## run compare2
compare2(e.lm, contrast = C$contrast, null = C$null)
compare2(e.lm, contrast = C$contrast, null = C$null, robust = TRUE)

#### with gls ####
library(nlme)
e.gls <- gls(Y~X1+X2, data = df.data, method = "ML")

## first compute the derivative of the information matrix
sCorrect(e.gls, cluster = 1:NROW(df.data)) <- TRUE

compare2(e.gls, par = c("5*X1b+2*X2 = 0","(Intercept) = 0"))

#### with lvm ####
```

```
m <- lvm(Y~X1+X2)
e.lvm <- estimate(m, df.data)

compare2(e.lvm, par = c("-Y","Y~X1b+Y~X1c"))
```

`createContrast` *Create Contrast matrix*

Description

Returns a contrast matrix corresponding an object. The contrast matrix will contains the hypotheses in rows and the model coefficients in columns.

Usage

```
createContrast(object, ...)

## S3 method for class 'character'
createContrast(object, name.param,
  diff.first = FALSE, add.rowname = TRUE, rowname.rhs = TRUE, ...)

## S3 method for class 'lm'
createContrast(object, par, add.variance, ...)

## S3 method for class 'gls'
createContrast(object, par, add.variance, ...)

## S3 method for class 'lme'
createContrast(object, par, add.variance, ...)

## S3 method for class 'lvmfit'
createContrast(object, par = NULL, var.test = NULL,
  ...)

## S3 method for class 'list'
createContrast(object, par = NULL, add.variance = NULL,
  var.test = NULL, ...)

## S3 method for class 'mmm'
createContrast(object, par = NULL, add.variance = NULL,
  var.test = NULL, ...)
```

Arguments

`object`	a `ls.lvmfit` object.
`...`	[internal] Only used by the generic method.
`name.param`	[internal] the names of all the model coefficients.

diff.first	[logical] should the contrasts between the first and any of the other coefficients define the null hypotheses.
add.rowname	[internal] should a name be defined for each hypothesis.
rowname.rhs	should the right hand side of the null hypothesis be added to the name.
par	[vector of characters] expression defining the linear hypotheses to be tested. See the examples section.
add.variance	[logical] should the variance coefficients be considered as model coefficients? Required for lm, gls, and lme models.
var.test	[character] a regular expression that is used to identify the coefficients to be tested using grep. Each coefficient will be tested in a separate hypothesis. When this argument is used, the argument par is disregarded.

Details

One can initialize an empty contrast matrix setting the argumentpar to character(0).

When using multcomp::glht one should set the argument add.variance to FALSE.
When using lavaSearch2::glht2 one should set the argument add.variance to TRUE.

Value

A list containing

- contrast [matrix] a contrast matrix corresponding to the left hand side of the linear hypotheses.
- null [vector] the right hand side of the linear hypotheses.
- Q [integer] the rank of the contrast matrix.
- ls.contrast [list, optional] the contrast matrix corresponding to each submodel. Only present when the argument object is a list of models.

Examples

```
## Simulate data
mSim <- lvm(X ~ Age + Treatment,
            Y ~ Gender + Treatment,
            c(Z1,Z2,Z3) ~ eta, eta ~ treatment,
            Age[40:5]~1)
latent(mSim) <- ~eta
categorical(mSim, labels = c("placebo","SSRI")) <- ~Treatment
categorical(mSim, labels = c("male","female")) <- ~Gender
n <- 1e2
set.seed(10)
df.data <- lava::sim(mSim,n)

## Estimate separate models
lmX <- estimate(lvm(X ~ -1 + Age + Treatment), data = df.data)
lmY <- estimate(lvm(Y ~ -1 + Gender + Treatment), data = df.data)
lvmZ <- estimate(lvm(c(Z1,Z2,Z3) ~ -1 + 1*eta, eta ~ -1 + Treatment),
```

```
                data = df.data)

## Contrast matrix for a given model
createContrast(lmX, par = "X~Age")
createContrast(lmX, par = c("X~Age=0","X~Age+5*X~TreatmentSSRI=0"))
createContrast(lmX, par = character(0))

## Contrast matrix for the join model
ls.lvm <- list(X = lmX, Y = lmY, Z = lvmZ)
createContrast(ls.lvm, var.test = "Treatment", add.variance = FALSE)
createContrast(ls.lvm, par = character(0), add.variance = FALSE)

## Contrast for multigroup models
m <- lvm(Y~Age+Treatment)
e <- estimate(list(m,m), data = split(df.data, df.data$Gender))
createContrast(e, par = "1@Y~TreatmentSSRI - 2@Y~TreatmentSSRI = 0")
createContrast(e, par = "2@Y~TreatmentSSRI - 1@Y~TreatmentSSRI = 0")
```

`dfSigma` *Degree of Freedom for the Chi-Square Test*

Description

Computation of the degrees of freedom of the chi-squared distribution relative to the model-based variance

Usage

```
dfSigma(contrast, vcov, dVcov, keep.param)
```

Arguments

`contrast`	[numeric vector] the linear combination of parameters to test
`vcov`	[numeric matrix] the variance-covariance matrix of the parameters.
`dVcov`	[numeric array] the first derivative of the variance-covariance matrix of the parameters.
`keep.param`	[character vector] the name of the parameters with non-zero first derivative of their variance parameter.

dfSigmaRobust *Degree of Freedom for the Robust Chi-Square Test*

Description

Computation of the degrees of freedom of the chi-squared distribution relative to the robust-based variance

Usage

```
dfSigmaRobust(contrast, vcov, score)
```

Arguments

contrast	[numeric vector] the linear combination of parameters to test
vcov	[numeric matrix] the variance-covariance matrix of the parameters.
score	[numeric matrix] the individual score for each parameter.

Details

When contrast is the identity matrix, this function compute the moments of the sandwich estimator and the degrees of freedom of the approximate t-test as described in (Pan, 2002) section 2 and 3.1.

References

Wei Pan and Melanie M. Wall, Small-sample adjustments in using the sandwich variance estiamtor in generalized estimating equations. Statistics in medicine (2002) 21:1429-1441.

estfun.lvmfit *Extract Empirical Estimating Functions (lvmfit Object)*

Description

Extract the empirical estimating functions of a lvmfit object. This function is for internal use but need to be public to enable its use by `multcomp::glht`.

Usage

```
## S3 method for class 'lvmfit'
estfun(x, ...)
```

Arguments

x	an `lvmfit` object.
...	arguments passed to methods.

Details

This function enables to use the glht function with lvmfit object. Otherwise when calling multcomp:::vcov.mmm then sandwich::sandwich and then sandwich::meat, sandwich::meat will complain that estfun is not defined for lvmfit objects.

Examples

```
library(multcomp)

#### generative model ####
mSim <- lvm(X ~ Age + 0.5*Treatment,
            Y ~ Gender + 0.25*Treatment,
            c(Z1,Z2,Z3) ~ eta, eta ~ 0.75*treatment,
            Age[40:5]~1)
latent(mSim) <- ~eta
categorical(mSim, labels = c("placebo","SSRI")) <- ~Treatment
categorical(mSim, labels = c("male","female")) <- ~Gender

#### simulate data ####
n <- 5e1
set.seed(10)
df.data <- lava::sim(mSim, n = n, latent = FALSE)

#### fit separate models ####
lmX <- lm(X ~ Age + Treatment, data = df.data)
lvmY <- estimate(lvm(Y ~ Gender + Treatment), data = df.data)
lvmZ <- estimate(lvm(c(Z1,Z2,Z3) ~ eta, eta ~ Treatment),
                 data = df.data)

#### create mmm object ####
e.mmm <- mmm(X = lmX, Y = lvmY, Z = lvmZ)

#### create contrast matrix ####
resC <- createContrast(e.mmm, var.test = "Treatment", add.variance = FALSE)

#### adjust for multiple comparisons ####
e.glht <- glht(e.mmm, linfct = resC$mlf)
summary(e.glht)
```

---

extractData *Extract Data From a Model*

Description

Extract data from a model using nlme::getData, riskRegression::coxDesign or model.frame.. If it fails it will try to extract it by its name according to model$call$data.

Usage

```
extractData(object, design.matrix, as.data.frame, envir)

## S3 method for class 'lm'
extractData(object, design.matrix = FALSE,
  as.data.frame = TRUE, envir = environment())

## S3 method for class 'coxph'
extractData(object, design.matrix = FALSE,
  as.data.frame = TRUE, envir = environment())

## S3 method for class 'cph'
extractData(object, design.matrix = FALSE,
  as.data.frame = TRUE, envir = environment())

## S3 method for class 'lvmfit'
extractData(object, design.matrix = FALSE,
  as.data.frame = TRUE, envir = environment())

## S3 method for class 'gls'
extractData(object, design.matrix = FALSE,
  as.data.frame = TRUE, envir = environment())

## S3 method for class 'lme'
extractData(object, design.matrix = FALSE,
  as.data.frame = TRUE, envir = environment())
```

Arguments

object	the fitted model.
design.matrix	[logical] should the data be extracted after transformation (e.g. conversion of categorical variables to dummy variables)? Otherwise the original data will be returned.
as.data.frame	[logical] should the output be converted into a data.frame object?
envir	[environment] the environment from which to search the data.

Value

a dataset.

Examples

```
set.seed(10)
n <- 101

#### linear regression ####
Y1 <- rnorm(n, mean = 0)
Y2 <- rnorm(n, mean = 0.3)
```

```
Id <- findInterval(runif(n), seq(0.1,1,0.1))
data.df <- rbind(data.frame(Y=Y1,G="1",Id = Id),
           data.frame(Y=Y2,G="2",Id = Id)
           )
m.lm <- lm(Y ~ G, data = data.df)
a <- extractData(m.lm, design.matrix = TRUE)
b <- extractData(m.lm, design.matrix = FALSE)

library(nlme)
m.gls <- gls(Y ~ G, weights = varIdent(form = ~ 1|Id), data = data.df)
c <- extractData(m.gls)
m.lme <- lme(Y ~ G, random = ~ 1|Id, data = data.df)
d <- extractData(m.lme)

library(lava)
e.lvm <- estimate(lvm(Y ~ G), data = data.df)
e <- extractData(e.lvm)
e <- extractData(e.lvm, design.matrix = TRUE)

#### survival ####
library(survival)

## Not run:
  library(riskRegression) ## needs version >=1.4.3
  dt.surv <- sampleData(n, outcome = "survival")
  m.cox <- coxph(Surv(time, event) ~ X1 + X2, data = dt.surv, x = TRUE, y = TRUE)
  f <- extractData(m.cox, design.matrix = FALSE)
  f <- extractData(m.cox, design.matrix = TRUE)
  m.cox <- coxph(Surv(time, event) ~ strata(X1) + X2, data = dt.surv, x = TRUE, y = TRUE)
  f <- extractData(m.cox, design.matrix = TRUE)

## End(Not run)

#### nested fuuctions ####
fct1 <- function(m){
   fct2(m)
}
fct2 <- function(m){
   extractData(m)
}
g <- fct1(m.gls)
```

---

findNewLink *Find all New Links Between Variables*

Description

Find all new links between variables (adapted from lava::modelsearch).

Usage

```
findNewLink(object, ...)

## S3 method for class 'lvm'
findNewLink(object, data = NULL, type = "both",
  exclude.var = NULL, rm.latent_latent = FALSE, rm.endo_endo = FALSE,
  rm.latent_endo = FALSE, output = "names", ...)
```

Arguments

object	a lvm object.
...	[internal] only used by the generic method.
data	[optional] a dataset used to identify the categorical variables when not specified in the lvm object.
type	[character vector] the type of links to be considered: "regression", "covariance", or "both", .
exclude.var	[character vector] all links related to these variables will be ignore.
rm.latent_latent	[logical] should the links relating two latent variables be ignored?
rm.endo_endo	[logical] should the links relating two endogenous variables be ignored?
rm.latent_endo	[logical] should the links relating one endogenous variable and one latent variable be ignored?
output	[character] Specify "names" to return the names of the variables to link or specify "index" to return their position.

Value

A list containing:

- M.links: a matrix with two columns indicating (by name or position) the exogenous and endogenous variable corresponding to each link.
- links: the name of the additional possible links
- directional: a logical vector indicating for each link whether the link is unidirectional (TRUE, i.e. regression link) or bidirectional (FALSE, i.e. covariance link).

Examples

```
library(lava)

m <- lvm()
regression(m) <- c(y1,y2,y3)~u
categorical(m,labels=c("M","F","MF")) <- ~X1
findNewLink(m, rm.endo = FALSE)
findNewLink(m, rm.endo = TRUE)
findNewLink(m, exclude.var = "X1")

regression(m) <- u~x1+x2
```

```
latent(m) <- ~u

findNewLink(m, rm.endo = FALSE)
findNewLink(m, rm.endo = TRUE)
findNewLink(m, rm.endo = TRUE, output = "index")
findNewLink(m, type = "covariance")
findNewLink(m, type = "regression")
```

---

getNewLink *Extract the Links that Have Been Found by the modelsearch2.*

Description

Extract the links that have been found relevant by modelsearch2.

Usage

```
getNewLink(object, step)

## S3 method for class 'modelsearch2'
getNewLink(object, step = 1:nStep(object))
```

Arguments

object	a modelsearch2 object.
step	[logical] which test should be extracted?

Value

A character vector.

Examples

```
## Not run:
mSim <- lvm(Y~G+X1+X2)
addvar(mSim) <- ~Z1+Z2+Z3+Z4+Z5+Z6

set.seed(10)
df.data <- lava::sim(mSim, 1e2)

mBase <- lvm(Y~G)
addvar(mBase) <- ~X1+X2+Z1+Z2+Z3+Z4+Z5+Z6
e.lvm <- estimate(mBase, data = df.data)
res <- modelsearch2(e.lvm, method.p.adjust = "holm")
getNewLink(res)

## End(Not run)
```

getNewModel *Extract the Model that Has Been Retains by the modelsearch2.*

Description

Extract the model that has been retained by modelsearch2.

Usage

```
getNewModel(object, step)

## S3 method for class 'modelsearch2'
getNewModel(object, step = nStep(object))
```

Arguments

object	a modelsearch2 object.
step	[integer >=0] the step at which the model should be extracted. 0 returns the initial model, i.e. before adding any links.

Value

A lvmfit object.

Examples

```
## Not run:
mSim <- lvm(Y~G+X1+X2)
addvar(mSim) <- ~Z1+Z2+Z3+Z4+Z5+Z6

set.seed(10)
df.data <- lava::sim(mSim, 1e2)

mBase <- lvm(Y~G)
addvar(mBase) <- ~X1+X2+Z1+Z2+Z3+Z4+Z5+Z6
e.lvm <- estimate(mBase, data = df.data)
res <- modelsearch2(e.lvm, method.p.adjust = "holm")
getNewModel(res)

## End(Not run)
```

getStep *Extract one Step From the Sequential Procedure*

Description

Extract one step from the sequential procedure.

Usage

```
getStep(object, step, slot)

## S3 method for class 'modelsearch2'
getStep(object, step = nStep(object),
  slot = NULL)
```

Arguments

object	a modelsearch2 object
step	[integer >0] which test should be extracted?
slot	[character] the element from the modelsearch2 object that should be extracted.

Examples

```
## Not run:
mSim <- lvm(Y~G+X1+X2)
addvar(mSim) <- ~Z1+Z2+Z3+Z4+Z5+Z6
df.data <- lava::sim(mSim, 1e2)

mBase <- lvm(Y~G)
addvar(mBase) <- ~X1+X2+Z1+Z2+Z3+Z4+Z5+Z6
e.lvm <- estimate(mBase, data = df.data)
res <- modelsearch2(e.lvm, method.p.adjust = "holm")

getStep(res)
getStep(res, slot = "sequenceTest")
getStep(res, step = 1)

## End(Not run)
```

getVarCov2 *Reconstruct the Conditional Variance Covariance Matrix*

Description

Reconstruct the conditional variance covariance matrix from a nlme or lvm model. Only compatible with specific correlation and variance structure.

Usage

```
getVarCov2(object, ...)

## S3 method for class 'gls'
getVarCov2(object, data = NULL, cluster, ...)

## S3 method for class 'lme'
getVarCov2(object, data = NULL, cluster, ...)

## S3 method for class 'lvmfit'
getVarCov2(object, ...)
```

Arguments

object	a gls or lme object
...	[internal] only used by the generic method.
data	[data.frame] the data set.
cluster	[integer vector] the grouping variable relative to which the observations are iid.

Details

The conditional variance covariance matrix for gls model is of the form:

$$\Sigma = \begin{matrix} \sigma^2 & \sigma^2\sigma_2\rho_{1,2} & \sigma^2\sigma_3\rho_{1,3} \\ . & \sigma^2\sigma_2^2 & \sigma^2\sigma_3\rho_{1,3} \\ . & . & \sigma^2\sigma_3^2 \end{matrix}$$

Value

A list containing the residual variance-covariance matrix in the element Omega.

Examples

```
## simulate data
library(nlme)
n <- 5e1
mSim <- lvm(c(Y1~1*eta,Y2~1*eta,Y3~1*eta,eta~G))
latent(mSim) <- ~eta
transform(mSim,Id~Y1) <- function(x){1:NROW(x)}
set.seed(10)
dW <- lava::sim(mSim,n,latent = FALSE)
dW <- dW[order(dW$Id),,drop=FALSE]
dL <- reshape2::melt(dW,id.vars = c("G","Id"), variable.name = "time")
dL <- dL[order(dL$Id),,drop=FALSE]
dL$Z1 <- rnorm(NROW(dL))
dL$time.num <- as.numeric(as.factor(dL$time))

#### iid model ####
```

```
e1.gls <- nlme::gls(Y1 ~ G, data = dW, method = "ML")
getVarCov2(e1.gls, cluster = 1:n)$Omega

#### heteroschedasticity ####
dW$group <- rbinom(n, size = 1, prob = 1/2)
dW$repetition <- as.numeric(as.factor(dW$group))
e2a.gls <- nlme::gls(Y1 ~ G, data = dW, method = "ML",
                     weights = varIdent(form =~ repetition|group))
getVarCov2(e2a.gls, cluster = 1:n)$Omega


e2b.gls <- nlme::gls(value ~ 0+time + time:G,
                   weight = varIdent(form = ~ time.num|time),
                   data = dL, method = "ML")
getVarCov2(e2b.gls, cluster = "Id")$Omega

#### compound symmetry ####
e3.gls <- nlme::gls(value ~ time + G,
                   correlation = corCompSymm(form = ~1| Id),
                   data = dL, method = "ML")
getVarCov2(e3.gls)$Omega

#### unstructured ####
e4.gls <- nlme::gls(value ~ time,
                    correlation = corSymm(form = ~time.num| Id),
                    weight = varIdent(form = ~ 1|time),
                    data = dL, method = "ML")
getVarCov2(e4.gls)$Omega

#### lvm model ####
m <- lvm(c(Y1~1*eta,Y2~1*eta,Y3~1*eta,eta~G))
latent(m) <- ~eta
e <- estimate(m, dW)
getVarCov2(e)
```

---

glht2 *General Linear Hypothesis*

Description

Test general linear hypotheses and across latent variable models with small sample corrections.

Usage

```
glht2(model, linfct, rhs, bias.correct, df, robust, cluster)

## S3 method for class 'lvmfit'
glht2(model, linfct, rhs = 0, bias.correct = TRUE,
  df = TRUE, robust = FALSE, cluster = NULL)
```

```
## S3 method for class 'mmm'
glht2(model, linfct, rhs = 0, bias.correct = TRUE,
  df = TRUE, robust = FALSE, cluster = NULL)
```

Arguments

model	a lvmfit or mmm object. The mmm object can only contain lm/gls/lme/lvmfit objects.
linfct	[matrix or vector of character] the linear hypotheses to be tested. Same as the argument par of createContrast.
rhs	[vector] the right hand side of the linear hypotheses to be tested.
bias.correct	[logical] should the standard errors of the coefficients be corrected for small sample bias?
df	[logical] should the degree of freedoms of the Wald statistic be computed using the Satterthwaite correction?
robust	[logical] should robust standard error be used? Otherwise rescale the influence function with the standard error obtained from the information matrix.
cluster	[integer vector] the grouping variable relative to which the observations are iid.

Details

Whenever the argument linfct is not a matrix, it is passed to the function createContrast to generate the contrast matrix and, if not specified, rhs.

Since only one degree of freedom can be specify in a glht object and it must be an integer, the degree of freedom of the denominator of an F test simultaneously testing all hypotheses is retained, after rounding.

Argument rhs and null are equivalent. This redondance enable compatibility between lava::compare, compare2, multcomp::glht, and glht2.

Value

A glht object.

See Also

createContrast to create contrast matrices.
sCorrect to pre-compute quantities for the small sample correction.

Examples

```
library(multcomp)

## Simulate data
mSim <- lvm(c(Y1,Y2,Y3)~ beta * eta, Z1 ~ E, Z2 ~ E, Age[40:5]~1)
latent(mSim) <- "eta"
set.seed(10)
n <- 1e2

df.data <- lava::sim(mSim, n, latent = FALSE, p = c(beta = 1))

#### Inference on a single model ####
e.lvm <- estimate(lvm(Y1~E), data = df.data)
summary(glht2(e.lvm, linfct = c("Y1~E + Y1","Y1")))

#### Inference on separate models ####
## fit separate models
lmX <- lm(Z1 ~ E, data = df.data)
lvmY <- estimate(lvm(Z2 ~ E + Age), data = df.data)
lvmZ <- estimate(lvm(c(Y1,Y2,Y3) ~ eta, eta ~ E),
                 data = df.data)

#### create mmm object ####
e.mmm <- mmm(X = lmX, Y = lvmY, Z = lvmZ)

#### create contrast matrix ####
resC <- createContrast(e.mmm, var.test = "E", add.variance = TRUE)

#### adjust for multiple comparisons ####
e.glht2 <- glht2(e.mmm, linfct = resC$contrast)
summary(e.glht2)
```

---

iid2 *Extract corrected i.i.d. decomposition*

Description

Extract corrected i.i.d. decomposition from a gaussian linear model.

Usage

```
iid2(object, ...)

## S3 method for class 'lm'
iid2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'gls'
```

```
iid2(object, cluster = NULL, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lme'
iid2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lvmfit'
iid2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lm2'
iid2(object, cluster = NULL, param = NULL, data = NULL,
  robust = TRUE, ...)

## S3 method for class 'gls2'
iid2(object, cluster = NULL, param = NULL,
  data = NULL, robust = TRUE, ...)

## S3 method for class 'lme2'
iid2(object, cluster = NULL, param = NULL,
  data = NULL, robust = TRUE, ...)

## S3 method for class 'lvmfit2'
iid2(object, cluster = NULL, data = NULL, ...)
```

Arguments

object	a linear model or a latent variable model
...	arguments to be passed to sCorrect.
param	[named numeric vector] the fitted parameters.
data	[data.frame] the data set.
bias.correct	[logical] should the standard errors of the coefficients be corrected for small sample bias? Only relevant if the sCorrect function has not yet be applied to the object.
cluster	[integer vector] the grouping variable relative to which the observations are iid.
robust	[logical] if FALSE, the i.i.d. decomposition is rescaled such its the squared sum equals the model-based standard error (instead of the robust standard error).

Details

If argument p or data is not null, then the small sample size correction is recomputed to correct the influence function.

Value

A matrix containing the 1st order influence function relative to each sample (in rows) and each model coefficient (in columns).

See Also

sCorrect to obtain lm2, gls2, lme2, or lvmfit2 objects.

Examples

```
n <- 5e1
p <- 3
X.name <- paste0("X",1:p)
link.lvm <- paste0("Y~",X.name)
formula.lvm <- as.formula(paste0("Y~",paste0(X.name,collapse="+")))

m <- lvm(formula.lvm)
distribution(m,~Id) <- sequence.lvm(0)
set.seed(10)
d <- sim(m,n)

## linear model
e.lm <- lm(formula.lvm,data=d)
iid.tempo <- iid2(e.lm, bias.correct = FALSE)
range(iid.tempo[,1:4]-iid(e.lm))

## latent variable model
e.lvm <- estimate(lvm(formula.lvm),data=d)
iid.tempo <- iid2(e.lvm, bias.correct = FALSE)
range(iid.tempo-iid(e.lvm))
## difference due to the use of the observed info matrix vs. the expected one.

## rescale i.i.d using model-based standard error
iid.tempo <- iid2(e.lvm, robust = FALSE, bias.correct = FALSE)
diag(crossprod(iid.tempo))-diag(vcov(e.lvm))
```

---

iidJack *Jackknife iid Decomposition from Model Object*

Description

Extract iid decomposition (i.e. influence function) from model object.

Usage

```
iidJack(object, ...)

## Default S3 method:
iidJack(object, data = NULL, grouping = NULL,
  cpus = 1, keep.warnings = TRUE, keep.error = TRUE, cl = NULL,
  trace = TRUE, ...)
```

Arguments

object	a object containing the model.
...	[internal] only used by the generic method.
data	[data.frame] dataset used to perform the jackknife.
grouping	[vector] variable defining cluster of observations that will be simultaneously removed by the jackknife.
cpus	[integer >0] the number of processors to use. If greater than 1, the fit of the model and the computation of the influence function for each jackknife sample is performed in parallel.
keep.warnings	[logical] keep warning messages obtained when estimating the model with the jackknife samples.
keep.error	[logical]keep error messages obtained when estimating the model with the jackknife samples.
cl	[cluster] a parallel socket cluster generated by `parallel::makeCluster` that has been registered using `registerDoParallel`.
trace	[logical] should a progress bar be used to trace the execution of the function

Value

A matrix with in row the samples and in columns the parameters.

Examples

```
n <- 20

#### glm ####
set.seed(10)
m <- lvm(y~x+z)
distribution(m, ~y+z) <- binomial.lvm("logit")
d <- lava::sim(m,n)
g <- glm(y~x+z,data=d,family="binomial")
iid1 <- iidJack(g, cpus = 1)
iid2 <- lava::iid(g)
quantile(iid1-iid2)
vcov(g)
colSums(iid2^2)
colSums(iid1^2)

#### Cox model ####
library(survival)
data(Melanoma, package = "riskRegression")
m <- coxph(Surv(time,status==1)~ici+age, data = Melanoma, x = TRUE, y = TRUE)

## Not run:
## require riskRegression > 1.4.3
if(utils::packageVersion("riskRegression") > "1.4.3"){
library(riskRegression)
iid1 <- iidJack(m)
```

```
iid2 <- iidCox(m)$IFbeta

apply(iid1,2,sd)

print(iid2)

apply(iid2,2,sd)
  }

## End(Not run)

#### LVM ####
set.seed(10)

mSim <- lvm(c(Y1,Y2,Y3,Y4,Y5) ~ 1*eta)
latent(mSim) <- ~eta
categorical(mSim, K=2) <- ~G
transform(mSim, Id ~ eta) <- function(x){1:NROW(x)}
dW <- lava::sim(mSim, n, latent = FALSE)
dL <- reshape2::melt(dW, id.vars = c("G","Id"),
                     variable.name = "time", value.name = "Y")
dL$time <- gsub("Y","",dL$time)

m1 <- lvm(c(Y1,Y2,Y3,Y4,Y5) ~ 1*eta)
latent(m1) <- ~eta
regression(m1) <- eta ~ G
e <- estimate(m1, data = dW)
## Not run:
iid1 <- iidJack(e)
iid2 <- iid(e)
attr(iid2, "bread") <- NULL

apply(iid1,2,sd)
apply(iid2,2,sd)
quantile(iid2 - iid1)

## End(Not run)

library(nlme)
e2 <- lme(Y~G+time, random = ~1|Id, weights = varIdent(form =~ 1|Id), data = dL)
e2 <- lme(Y~G, random = ~1|Id, data = dL)
## Not run:
iid3 <- iidJack(e2)
apply(iid3,2,sd)

## End(Not run)
```

`initVarLink` *Normalize var1 and var2*

Description

Convert var1 and var2 from formula or covariance to character.

Usage

```
initVarLink(var1, var2, rep.var1 = FALSE, format = "list",
  Slink = c(lava.options()$symbols[1], "~"),
  Scov = lava.options()$symbols[2])

initVarLinks(var1, format = "list", ...)
```

Arguments

var1	[character or formula] the exogenous variable of the new link or a formula describing the link.
var2	[character] the endogenous variable of the new link. Disregarded if the argument var1 is a formula.
rep.var1	[logical] should var1 be duplicated to match var2 length. Only active if format = "list".
format	[character] should the name of the variable be returned (format = "list"), a vector of character formula (format = "txt.formula"), or a list of formula (format = "formula").
Slink	[character] the symbol for regression link.
Scov	[character] the symbol for covariance link.
...	argument to be passed to initVarLink.

Value

See argument format.

Examples

```
initVarLink(y ~ x1)
initVarLink("y ~ x1")
initVarLink(y ~ x1 + x2)
initVarLink("y ~ x1 + x2")
initVarLink(y ~ x1 + x2, rep.var1 = TRUE)
initVarLink(y ~ x1 + x2, rep.var1 = TRUE, format = "formula")
initVarLink(y ~ x1 + x2, rep.var1 = TRUE, format = "txt.formula")
initVarLink("y", "x1", format = "formula")

initVarLink("y ~ x1:0|1")

initVarLinks(y ~ x1)
initVarLinks("y ~ x1")
initVarLinks(c("y ~ x1","y~ x2"))
initVarLinks(c(y ~ x1,y ~ x2))
initVarLinks(c("y ~ x1","y ~ x2"), format = "formula")
initVarLinks(c(y ~ x1,y ~ x2), format = "formula")
```

```
initVarLinks(c("y ~ x1","y~ x2"), format = "txt.formula")
initVarLinks(c(y ~ x1,y ~ x2), format = "txt.formula")
```

`intDensTri` *Integrate a Gaussian/Student Density over a Triangle*

Description

Consider a univariate random variable X, two multivariate random variables Y and Z, and t1 and t2 two real numbers. This function can compute either P[|X|>t1,|X]>|Y1|,...,|X]>|Ypl] if zmin is not specified, P[|Z1|<t2,...,|Zq|<t2,|X|>t1,|X]>|Y1|,...,|X]>|Ypl] if zmin is specified.

Usage

```
intDensTri(mu, Sigma, df, n, x.min, z.max = NULL, type = "double",
  proba.min = 1e-06, prune = NULL, distribution = "pmvnorm")
```

Arguments

`mu`	[numeric vector] the expectation of the joint distribution.
`Sigma`	[matrix] the variance-covariance of the joint distribution.
`df`	[integer > 0] the degree of freedom of the joint Student's t distribution. Only used when `distribution="pvmt"`.
`n`	[integer > 0] number of points for the numerical integration.
`x.min`	[numeric] the minimum value along the x axis.
`z.max`	[numeric vector, optional] the maximum value along the z axis. Define the dimension of Z.
`type`	[character] the type of mesh to be used. Can be `\"raw\"`, `\"double\"`, or `\"fine\"`.
`proba.min`	[numeric 0-1] the probability used to find the maximum value along the x axis. Only used if `prune` is not specified.
`prune`	[integer >0] number of standard deviations after which the domain ends along the x axis.
`distribution`	[character] type of joint distribution. Can be `"pmvnorm"` (normal distribution) or `"pvmt"` (Student's t distribution)

Details

Argument `type`:

- `\"raw\"`: mesh with points inside the domain
- `\"double\"`: mesh with points outside the domain
- `\"fine\"`: mesh with points inside the domain plus additional rectangles trying to fill the missing domain.

Argument `Sigma` and `mu`: define the mean and variance-covariance of the random variables X, Y, Z (in this order). The length of the argument `z.max` is used to define the dimension of Z. The dimension of X is always 1.

Value

A numeric.

Examples

```
library(mvtnorm)

p <- 2
Sigma <- diag(p)
mu <- rep(0, p)

## bivariate normal distribution
z2 <- qmvt(0.975, mean = mu, sigma = Sigma, df = 1e3)$quantile

# compute integral
intDensTri(mu = mu, Sigma = Sigma, n=5, x.min=0, type = "fine")$value-1/2
intDensTri(mu = mu, Sigma = Sigma, n=30, x.min=0, type = "raw")$value-1/2
intDensTri(mu = mu, Sigma = Sigma, n=50, x.min=0, type = "raw")$value-1/2

intDensTri(mu = mu, Sigma = Sigma, df = 5, n=5, x.min=0, distribution = "pmvt")$value-1/2
res <- intDensTri(mu = mu, Sigma = Sigma, df = 5, n=10, x.min=0, distribution = "pmvt")
res$value-1/2
ggplot2::autoplot(res)

## trivariate normal distribution
## Not run:
p <- 3
Sigma <- diag(p)
mu <- rep(0, p)

res2 <- intDensTri(mu = mu, Sigma = Sigma, n=5, x.min = 0, z.max = 10)
ggplot2::autoplot(res2)
ggplot2::autoplot(res2, coord.plot = c("x","z1"))
res2

## End(Not run)

#### when the distribution is far from 0
## Not run:
eq1 <- intDensTri(mu = c(10,0), Sigma = diag(1,2),
                  x.min = 2, n=10)
eq1$value-1
ggplot2::autoplot(eq1)

eq2 <- intDensTri(mu = c(10,0,0), Sigma = diag(1,3),
                  x.min=2, z.max = 10, type = "raw",
                  n=10)
ggplot2::autoplot(eq2, coord.plot = c("y1","z1"))
eq2$value-1

## more variables
p <- 5
```

```
Sigma <- diag(p)
mu <- rep(0, p)

res2 <- intDensTri(mu = mu, Sigma = Sigma, n=5, x.min = 1, z.max = c(2,2))
res2$grid

## End(Not run)
```

---

lavaSearch2 *Tools for Model Specification in the Latent Variable Framework*

Description

The package contains three main functionalities:

- compare2, summary2: Wald tests/robust Wald tests/F-tests/robust F-tests with improved control of the type 1 error in small samples.
- glht2: adjustment for multiple comparisons when doing inference for multiple latent variable models.
- modelsearch2: searching for local dependencies with adjustment for multiple comparisons.

It contains other useful functions such as:

- calibrateType1: simulation study of the type 1 error of Wald tests.
- createContrast: user-friendly function generating a contrast matrix.
- getVarCov2: reconstruct the conditional variance covariance matrix.
- iidJack: extract the jackknife iid decomposition.

Details

The latent variable models (LVM) considered in this package can be written as a measurement model:

$$Y_i = \nu + \eta_i \Lambda + X_i K + \epsilon_i$$

and a structural model:

$$\eta_i = \alpha + \eta_i B + X_i \Gamma + \zeta_i$$

where Σ is the variance covariance matrix of the residuals ϵ, and Ψ is the variance covariance matrix of the residuals ζ.

The corresponding conditional mean is:

$$\mu_i(\theta) = E[Y_i|X_i] = \nu + (\alpha + X_i\Gamma)(1-B)^{-1}\Lambda + X_i K$$

$$\Omega(\theta) = Var[Y_i|X_i] = \Lambda^t (1-B)^{-t} \Psi (1-B)^{-1}\Lambda + \Sigma$$

The package aims to provides tool for testing linear hypotheses on the model coefficients ν, Λ, K, Σ, α, B, Γ, Ψ. Searching for local dependency enable to test whether the proposed model is too simplistic and if so to identify which additional coefficients should be added to the model.

Limitations

'lavaSearch2' has been design for Gaussian latent variable models. This means that it may not work / give valid results:

- in presence of censored or binary outcomes.
- with stratified models (i.e. object of class `multigroup`).

`leverage2` *Extract Leverage Values*

Description

Extract leverage values from a Gaussian linear model.

Usage

```
leverage2(object, ...)

## S3 method for class 'lm'
leverage2(object, param = NULL, data = NULL, ...)

## S3 method for class 'gls'
leverage2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lme'
leverage2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lvmfit'
leverage2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lm2'
leverage2(object, param = NULL, data = NULL, ...)

## S3 method for class 'gls2'
leverage2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lme2'
leverage2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lvmfit2'
leverage2(object, param = NULL, data = NULL, ...)
```

Arguments

`object`	a `lm2`, `gls2`, `lme2`, or `lvmfit2` object.
`...`	arguments to be passed to `sCorrect`.
`param`	[optional] the fitted parameters.
`data`	[optional] the data set.

Details

The leverage are defined as the partial derivative of the fitted values with respect to the observations.

$$leverage_i = \frac{\partial \hat{Y}_i}{\partial Y_i}$$

See Wei et al. (1998).

If argument p or data is not null, then the small sample size correction is recomputed to correct the residuals.

Value

a matrix containing the leverage relative to each sample (in rows) and each endogenous variable (in column).

References

Bo-Cheng Wei et al., Generalized Leverage and its applications (1998), Scandinavian Journal of Statistics 25:1:25-37.

See Also

sCorrect to obtain lm2, gls2, lme2, or lvmfit2 objects.

Examples

```
## simulate data
set.seed(10)
m <- lvm(Y1~eta,Y2~eta,Y3~eta)
latent(m) <- ~eta
d <- lava::sim(m,20, latent = FALSE)

## standard linear model
e.lm <- lm(Y1~Y2, data = d)

sCorrect(e.lm) <- TRUE
range(as.double(leverage2(e.lm)) - influence(e.lm)$hat)

## latent variable model
e.lvm <- estimate(m, data = d)
sCorrect(e.lvm) <- TRUE
leverage2(e.lvm)
```

`modelsearch2` *Data-driven Extension of a Latent Variable Model*

Description

Procedure adding relationship between variables that are supported by the data.

Usage

```
modelsearch2(object, link, data, method.p.adjust, type.information, alpha,
  nStep, na.omit, trace, cpus)

## S3 method for class 'lvmfit'
modelsearch2(object, link = NULL, data = NULL,
  method.p.adjust = "fastmax", type.information = "E", alpha = 0.05,
  nStep = NULL, na.omit = TRUE, trace = TRUE, cpus = 1)
```

Arguments

`object`	a `lvmfit` object.
`link`	[character, optional for `lvmfit` objects] the name of the additional relationships to consider when expanding the model. Should be a vector containing strings like "Y~X". See the details section.
`data`	[data.frame, optional] the dataset used to identify the model
`method.p.adjust`	[character] the method used to adjust the p.values for multiple comparisons. Can be any method that is valid for the `stats::p.adjust` function (e.g. `"fdr"`), or `"max"` or `"fastmax"`.
`type.information`	[character] the method used by `lava::information` to compute the information matrix.
`alpha`	[numeric 0-1] the significance cutoff for the p-values. When the p-value is below, the corresponding link will be added to the model and the search will continue. Otherwise the search will stop.
`nStep`	the maximum number of links that can be added to the model.
`na.omit`	should tests leading to NA for the test statistic be ignored. Otherwise this will stop the selection process.
`trace`	[logical] should the execution of the function be traced?
`cpus`	the number of cpus that can be used for the computations.

Details

method.p.adjust = "max" computes the p-values based on the distribution of the max statistic. This max statistic is the max of the square root of the score statistic. The p-value are computed integrating the multivariate normal distribution.

method.p.adjust = "fastmax" only compute the p-value for the largest statistic. It is faster than "max" and lead to identical results.

Value

A list containing:

- sequenceTest: the sequence of test that has been performed.
- sequenceModel: the sequence of models that has been obtained.
- sequenceQuantile: the sequence of rejection threshold. Optional.
- sequenceIID: the influence functions relative to each test. Optional.
- sequenceSigma: the covariance matrix relative to each test. Optional.
- initialModel: the model before the sequential search.
- statistic: the argument statistic.
- method.p.adjust: the argument method.p.adjust.
- alpha: [numeric 0-1] the significance cutoff for the p-values.
- cv: whether the procedure has converged.

Examples

```
## simulate data
mSim <- lvm()
regression(mSim) <- c(y1,y2,y3,y4)~u
regression(mSim) <- u~x1+x2
categorical(mSim,labels=c("A","B","C")) <- "x2"
latent(mSim) <- ~u
covariance(mSim) <- y1~y2
transform(mSim, Id~u) <- function(x){1:NROW(x)}
df.data <- lava::sim(mSim, n = 1e2, latent = FALSE)

## only identifiable extensions
m <- lvm(c(y1,y2,y3,y4)~u)
latent(m) <- ~u
addvar(m) <- ~x1+x2

e <- estimate(m, df.data)

## Not run:
resSearch <- modelsearch(e)
resSearch

resSearch2 <- modelsearch2(e, nStep = 2)
resSearch2
```

```
## End(Not run)


## some extensions are not identifiable
m <- lvm(c(y1,y2,y3)~u)
latent(m) <- ~u
addvar(m) <- ~x1+x2

e <- estimate(m, df.data)

## Not run:
resSearch <- modelsearch(e)
resSearch
resSearch2 <- modelsearch2(e)
resSearch2

## End(Not run)

## for instance
mNI <- lvm(c(y1,y2,y3)~u)
latent(mNI) <- ~u
covariance(mNI) <- y1~y2
## estimate(mNI, data = df.data)
## does not converge
```

---

nStep *Find the Number of Steps Performed During the Sequential Testing*

Description

Find the number of steps performed during the sequential testing.

Usage

```
nStep(object)

## S3 method for class 'modelsearch2'
nStep(object)
```

Arguments

object	a modelsearch2 object.

Value

an integer.

Examples

```
## Not run:
mSim <- lvm(Y~G+X1+X2)
addvar(mSim) <- ~Z1+Z2+Z3+Z4+Z5+Z6
df.data <- lava::sim(mSim, 1e2)

mBase <- lvm(Y~G)
addvar(mBase) <- ~X1+X2+Z1+Z2+Z3+Z4+Z5+Z6
e.lvm <- estimate(mBase, data = df.data)
res <- modelsearch2(e.lvm, method.p.adjust = "holm")
nStep(res)

## End(Not run)
```

`residuals2` *Extract Corrected Residuals*

Description

Extract correct residuals from a gaussian linear model.

Usage

```
residuals2(object, param, data, type)

## S3 method for class 'lm2'
residuals2(object, param = NULL, data = NULL,
  type = "response")

## S3 method for class 'gls2'
residuals2(object, param = NULL, data = NULL,
  type = "response")

## S3 method for class 'lme2'
residuals2(object, param = NULL, data = NULL,
  type = "response")

## S3 method for class 'lvmfit2'
residuals2(object, param = NULL, data = NULL,
  type = "response")
```

Arguments

`object`	a `lm2`, `gls2`, `lme2`, or `lvmfit2` object.
`param`	[named numeric vector] the fitted parameters.
`data`	[data.frame] the data set.
`type`	[character] the type of residual to extract: `"response"` for raw residuals, `"studentized"` for studentized residuals, `"normalized"` for normalized residuals.

Details

If argument p or data is not null, then the small sample size correction is recomputed to correct the residuals.

The raw residuals are defined by observation minus the fitted value:

$$\varepsilon = (Y_1 - \mu_1, ..., Y_m - \mu_m)$$

The studentized residuals divided the raw residuals relative to each endogenous variable by the modeled variance of the endogenous variable.

$$\varepsilon_{stud} = (\frac{Y_1 - \mu_1}{\sigma_1}, ..., \frac{Y_m - \mu_m}{\sigma_m})$$

The normalized residuals multiply the raw residuals by the inverse of the square root of the modeled residual variance covariance matrix.

$$\varepsilon_{norm} = \varepsilon \Omega^{-1/2}$$

Value

a matrix containing the residuals relative to each sample (in rows) and each endogenous variable (in column).

See Also

sCorrect to obtain lm2, gls2, lme2, or lvmfit2 objects.

Examples

```
## simulate data
set.seed(10)
m <- lvm(Y1~eta,Y2~eta,Y3~eta)
latent(m) <- ~eta
d <- lava::sim(m,20, latent = FALSE)

## standard linear model
e.lm <- lm(Y1~Y2, data = d)
sCorrect(e.lm) <- TRUE

sigma(e.lm)^2
mean(residuals(e.lm)^2)
mean(residuals2(e.lm)^2)

## latent variable model
e.lvm <- estimate(m, data = d)
sCorrect(e.lvm) <- TRUE
mean(residuals2(e.lvm)^2)
```

score2 *Extract The Individual Score*

Description

Extract The Individual Score from a gaussian linear model.

Usage

```
score2(object, ...)

## S3 method for class 'lm'
score2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'gls'
score2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lme'
score2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lvmfit'
score2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lm2'
score2(object, param = NULL, data = NULL, ...)

## S3 method for class 'gls2'
score2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lme2'
score2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lvmfit2'
score2(object, param = NULL, data = NULL, ...)
```

Arguments

object	a linear model or a latent variable model
...	arguments to be passed to sCorrect.
param	[optional] the fitted parameters.
data	[optional] the data set.

bias.correct [logical] should the standard errors of the coefficients be corrected for small sample bias? Only relevant if the sCorrect function has not yet be applied to the object.

Details

If argument p or data is not null, then the small sample size correction is recomputed to correct the influence function.

Value

A matrix containing the score relative to each sample (in rows) and each model coefficient (in columns).

See Also

sCorrect to obtain lm2, gls2, lme2, or lvmfit2 objects.

Examples

```
n <- 5e1
p <- 3
X.name <- paste0("X",1:p)
link.lvm <- paste0("Y~",X.name)
formula.lvm <- as.formula(paste0("Y~",paste0(X.name,collapse="+")))

m <- lvm(formula.lvm)
distribution(m,~Id) <- sequence.lvm(0)
set.seed(10)
d <- lava::sim(m,n)

## linear model
e.lm <- lm(formula.lvm,data=d)
score.tempo <- score2(e.lm, bias.correct = FALSE)
colMeans(score.tempo)

## latent variable model
e.lvm <- estimate(lvm(formula.lvm),data=d)
score.tempo <- score2(e.lvm, bias.correct = FALSE)
range(score.tempo-score(e.lvm, indiv = TRUE))
```

---

sCorrect *Satterthwaite Correction and Small Sample Correction*

Description

Correct the bias of the ML estimate of the variance and compute the first derivative of the information matrix.

Usage

```
sCorrect(object, adjust.Omega, adjust.n, score, df, numeric.derivative,
  param, data, tol, n.iter, trace, ...)

## S3 method for class 'lm'
sCorrect(object, adjust.Omega = TRUE, adjust.n = TRUE,
  score = TRUE, df = TRUE, numeric.derivative = FALSE,
  param = NULL, data = NULL, tol = 1e-05, n.iter = 20, trace = 0,
  ...)

## S3 method for class 'lm2'
sCorrect(object, ...)

## S3 method for class 'gls'
sCorrect(object, adjust.Omega = TRUE, adjust.n = TRUE,
  score = TRUE, df = TRUE, numeric.derivative = FALSE,
  param = NULL, data = NULL, tol = 1e-05, n.iter = 20, trace = 0,
  cluster, ...)

## S3 method for class 'gls2'
sCorrect(object, ...)

## S3 method for class 'lme'
sCorrect(object, adjust.Omega = TRUE, adjust.n = TRUE,
  score = TRUE, df = TRUE, numeric.derivative = FALSE,
  param = NULL, data = NULL, tol = 1e-05, n.iter = 20, trace = 0,
  cluster, ...)

## S3 method for class 'lme2'
sCorrect(object, ...)

## S3 method for class 'lvmfit'
sCorrect(object, adjust.Omega = TRUE, adjust.n = TRUE,
  score = TRUE, df = TRUE, numeric.derivative = FALSE,
  param = NULL, data = NULL, tol = 1e-05, n.iter = 20, trace = 0,
  ...)

## S3 method for class 'lvmfit2'
sCorrect(object, ...)

sCorrect(x, ...) <- value

## S3 replacement method for class 'lm'
sCorrect(x, ...) <- value

## S3 replacement method for class 'lm2'
sCorrect(x, ...) <- value
```

```
## S3 replacement method for class 'gls'
sCorrect(x, ...) <- value

## S3 replacement method for class 'gls2'
sCorrect(x, ...) <- value

## S3 replacement method for class 'lme'
sCorrect(x, ...) <- value

## S3 replacement method for class 'lme2'
sCorrect(x, ...) <- value

## S3 replacement method for class 'lvmfit'
sCorrect(x, ...) <- value

## S3 replacement method for class 'lvmfit2'
sCorrect(x, ...) <- value
```

Arguments

object, x	a gls, lme, or lvm object.
adjust.Omega	[logical] should the standard errors of the coefficients be corrected for small sample bias?
adjust.n	[logical] should the correction for the degree of freedom be performed?
score	[internal] export the score.
df	[logical] should the degree of freedoms of the Wald statistic be computed using the Satterthwaite correction? Otherwise the degree of freedoms are set to Inf, i.e. a normal distribution is used instead of a Student's t distribution when computing the p-values.
numeric.derivative	[logical] should a numerical derivative be used to compute the first derivative of the information matrix? Otherwise an analytic formula is used.
param	[numeric vector, optional] the values of the parameters at which to perform the correction.
data	[data.frame, optional] the dataset relative to which the correction should be performed.
tol	[numeric >0] the minimum absolute difference between two estimation of the small sample bias. Below this value, the algorithm used to estimate the bias stop.
n.iter	[integer >0] the maximum number of iterations used to estimate the small sample bias of the residual variance-covariance matrix.
trace	[logical] should the execution of the function be traced.
...	[internal] only used by the generic method or by the <- methods.
cluster	[integer vector] the grouping variable relative to which the observations are iid. Only required for gls models with no correlation argument.
value	[logical] value for the arguments adjust.Omega and adjust.n.

Details

The argument `value` is equivalent to the argument `bias.correct` of the function `summary2`.

`setLink` *Set a Link to a Value*

Description

Generic interface to set a value to a link in a `lvm` object.

Usage

```
setLink(object, ...)

## S3 method for class 'lvm'
setLink(object, var1, var2, value, warnings = FALSE, ...)
```

Arguments

`object`	a `lvm` object.
`...`	[internal] only used by the generic method.
`var1`	[character or formula] the exogenous variable of the new link or a formula describing the link to be added to the lvm.
`var2`	[character] the endogenous variable of the new link. Disregarded if the argument `var1` is a formula.
`value`	[numeric] the value at which the link should be set.
`warnings`	[logical] should a warning be displayed if the link is not found in the `lvm` object.

Examples

```
library(lava)
set.seed(10)

m <- lvm()
regression(m) <- c(y1,y2,y3)~u
regression(m) <- u~x1+x2
latent(m) <- ~u
covariance(m) <- y1 ~ y2

m1 <- setLink(m, y3 ~ u, value = 1)
estimate(m1, lava::sim(m,1e2))
# m1 <- setLink(m, u ~ y3, value = 1)

m2 <- setLink(m, y1 ~ y2, value = 0.5)
estimate(m2, lava::sim(m,1e2))
```

summary.calibrateType1 *Display the Type 1 Error Rate*

Description

Display the type 1 error rate from the simulation results.

Usage

```
## S3 method for class 'calibrateType1'
summary(object, robust = FALSE,
  type = "type1error", alpha = 0.05, log.transform = TRUE,
  digits = 5, print = TRUE, ...)
```

Arguments

object	output of the calibrateType1 function.
robust	[character] should the results be displayed for both model-based and robust standard errors (TRUE), only model-based standard error (FALSE), or only robust standard error ("only")?
type	[character] should the type 1 error rate be diplayed ("type1error") or the bias ("bias").
alpha	[numeric, 0-1] the confidence levels.
log.transform	[logical] should the confidence intervals be computed on the logit scale.
digits	[integer >0] the number of decimal places to use when displaying the summary.
print	should the summary be printed in the terminal.
...	[internal] only used by the generic method.

summary.modelsearch2 *summary Method for modelsearch2 Objects*

Description

summary method for modelsearch2 objects.

Usage

```
## S3 method for class 'modelsearch2'
summary(object, print = TRUE, ...)
```

Arguments

object	output of the modelsearch2 function.
print	should the summary be printed in the terminal.
...	[internal] only used by the generic method.

Details

The column dp.Info contains the percentage of extended models (i.e. model with one additional link) for which the information matrix evaluated at the value of the parameters of the initial model is non positive definie.

---

summary2 *Summary with Small Sample Correction*

Description

Summary with small sample correction.

Usage

```
summary2(object, ...)

## S3 method for class 'lm'
summary2(object, df = TRUE, bias.correct = TRUE, ...)

## S3 method for class 'gls'
summary2(object, df = TRUE, bias.correct = TRUE,
  cluster = NULL, ...)

## S3 method for class 'lme'
summary2(object, df = TRUE, bias.correct = TRUE, ...)

## S3 method for class 'lvmfit'
summary2(object, df = TRUE, bias.correct = TRUE, ...)

## S3 method for class 'lm2'
summary2(object, digit = max(3, getOption("digit")),
  robust = FALSE, df = TRUE, ...)

## S3 method for class 'gls2'
summary2(object, digit = max(3, getOption("digit")),
  robust = FALSE, df = TRUE, ...)

## S3 method for class 'lme2'
summary2(object, digit = max(3, getOption("digit")),
  robust = FALSE, df = TRUE, ...)
```

```
## S3 method for class 'lvmfit2'
summary2(object, cluster = NULL, robust = FALSE,
  df = TRUE, ...)
```

Arguments

object	a gls, lme or lvm object.
...	arguments passed to the summary method of the object.
df	[logical] should the degree of freedoms of the Wald statistic be computed using the Satterthwaite correction? Otherwise the degree of freedoms are set to Inf, i.e. a normal distribution is used instead of a Student's t distribution when computing the p-values.
bias.correct	[logical] should the standard errors of the coefficients be corrected for small sample bias? See sCorrect for more details.
cluster	[integer vector] the grouping variable relative to which the observations are iid.
digit	[integer > 0] the number of decimal places to use when displaying the summary.
robust	[logical] should the robust standard errors be used instead of the model based standard errors?

Details

summary2 is the same as summary except that it first computes the small sample correction (but does not store it). So if summary2 is to be called several times, it is more efficient to pre-compute the quantities for the small sample correction using sCorrect and then call summary2.

See Also

sCorrect for more detail about the small sample correction.

Examples

```
m <- lvm(Y~X1+X2)
set.seed(10)
d <- lava::sim(m, 2e1)

## Gold standard
summary(lm(Y~X1+X2, d))$coef

## gls models
library(nlme)
e.gls <- gls(Y~X1+X2, data = d, method = "ML")
summary(e.gls)$tTable
sCorrect(e.gls, cluster = 1:NROW(d)) <- FALSE ## no small sample correction
summary2(e.gls)$tTable

sCorrect(e.gls, cluster = 1:NROW(d)) <- TRUE ## small sample correction
summary2(e.gls)$tTable
```

```
## lvm models
e.lvm <- estimate(m, data = d)
summary(e.lvm)$coef

sCorrect(e.lvm) <- FALSE ## no small sample correction
summary2(e.lvm)$coef

sCorrect(e.lvm) <- TRUE ## small sample correction
summary2(e.lvm)$coef
```

`tryWithWarnings` *Run an Expression and Catch Warnings and Errors*

Description

Similar to `try` but also returns warnings.

Usage

```
tryWithWarnings(expr)
```

Arguments

`expr`	the line of code to be evaluated

Details

from https://stackoverflow.com/questions/4948361/how-do-i-save-warnings-and-errors-as-output-from-a-function

Value

A list containing:

- value the result of the evaluation of the expression
- warnings warning(s) generated during the evaluation of the expression
- error error generated during the evaluation of the expression

Examples

```
FctTest <- function(x){
  return(log(x))
}
tryWithWarnings(FctTest(-1))
tryWithWarnings(FctTest(1))
tryWithWarnings(FctTest(xxxx))
```

vcov2 *Extract the Variance Covariance Matrix of the Model Parameters*

Description

Extract the variance covariance matrix of the model parameters from a Gaussian linear model.

Usage

```
vcov2(object, ...)

## S3 method for class 'lm'
vcov2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'gls'
vcov2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lme'
vcov2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lvmfit'
vcov2(object, param = NULL, data = NULL,
  bias.correct = TRUE, ...)

## S3 method for class 'lm2'
vcov2(object, param = NULL, data = NULL, ...)

## S3 method for class 'gls2'
vcov2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lme2'
vcov2(object, param = NULL, data = NULL, ...)

## S3 method for class 'lvmfit2'
vcov2(object, param = NULL, data = NULL, ...)
```

Arguments

object	a linear model or a latent variable model
...	arguments to be passed to sCorrect.
param	[optional] the fitted parameters.
data	[optional] the data set.

`bias.correct` [logical] should the standard errors of the coefficients be corrected for small sample bias? Only relevant if the `sCorrect` function has not yet be applied to the object.

Details

If argument `p` or `data` is not null, then the small sample size correction is recomputed to correct the influence function.

Value

A matrix.

See Also

`sCorrect` to obtain `lm2`, `gls2`, `lme2`, or `lvmfit2` objects.

Examples

```
n <- 5e1
p <- 3
X.name <- paste0("X",1:p)
link.lvm <- paste0("Y~",X.name)
formula.lvm <- as.formula(paste0("Y~",paste0(X.name,collapse="+")))

m <- lvm(formula.lvm)
distribution(m,~Id) <- sequence.lvm(0)
set.seed(10)
d <- lava::sim(m,n)

## linear model
e.lm <- lm(formula.lvm,data=d)
vcov.tempo <- vcov2(e.lm, bias.correct = TRUE)
vcov.tempo[rownames(vcov(e.lm)),colnames(vcov(e.lm))]/vcov(e.lm)

## latent variable model
e.lvm <- estimate(lvm(formula.lvm),data=d)
vcov.tempo <- vcov2(e.lvm, bias.correct = FALSE)
vcov.tempo/vcov(e.lvm)
```

Index