Package 'lava'

November 18, 2014

Type Package

Title Latent Variable Models

Version 1.3

Date 2014-11-18

Author Klaus K. Holst

Maintainer Klaus K. Holst <klaus@holst.it>

Description Estimation and simulation of latent variable models

URL http://lava.r-forge.r-project.org/

License GPL-3

LazyLoad yes

Depends R (>= 2.15)

Imports numDeriv

Suggests mets (>= 0.2.7), graph, Rgraphviz, igraph (>= 0.6), Matrix,KernSmooth, lme4, geepack, gof (>= 0.9), foreach, rgl,survival, testthat, ascii

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-18 15:18:44

R topics documented:

`lava-package` *Estimation and simulation of latent variable models*

Description

Framwork for estimating parameters and simulate data from Latent Variable Models.

Author(s)

Klaus K. Holst Maintainer: <k.k.holst@biostat.ku.dk>

Examples

```
lava()
```

addvar *Add variable to (model) object*

Description

Generic method for adding variables to model object

Usage

```
addvar(x, ...)
```

Arguments

x	Model object
...	Additional arguments

Author(s)

Klaus K. Holst

baptize *Label elements of object*

Description

Generic method for labeling elements of an object

Usage

```
baptize(x, ...)
```

Arguments

x	Object
...	Additional arguments

Author(s)

Klaus K. Holst

blockdiag *Combine matrices to block diagonal structure*

Description

Combine matrices to block diagonal structure

Usage

```
blockdiag(x, ..., pad = 0)
```

Arguments

x	Matrix
pad	Vyalue outside block-diagonal
...	Additional matrices

Author(s)

Klaus K. Holst

Examples

```
A <- diag(3)+1
blockdiag(A,A,A,pad=NA)
```

bmd *Longitudinal Bone Mineral Density Data (Wide format)*

Description

Bone Mineral Density Data consisting of 112 girls randomized to receive calcium og placebo. Longitudinal measurements of bone mineral density (g/cm^2) measured approximately every 6th month in 3 years.

Format

data.frame

Source

Vonesh & Chinchilli (1997), Table 5.4.1 on page 228.

See Also

calcium

bmidata *Data*

Description

Description

Format

data.frame

bootstrap *Generic bootstrap method*

Description

Generic method for calculating bootstrap statistics

Usage

```
bootstrap(x, ...)
```

Arguments

x	Model object
...	Additional arguments

Author(s)

Klaus K. Holst

See Also

bootstrap.lvm bootstrap.lvmfit

bootstrap.lvm *Calculate bootstrap estimates of a lvm object*

Description

Draws non-parametric bootstrap samples

Usage

```
## S3 method for class 'lvm'
bootstrap(x,R=100,data,fun=NULL,control=list(),
                         p, parametric=FALSE, bollenstine=FALSE,
                         constraints=TRUE,sd=FALSE,silent=FALSE,...)

## S3 method for class 'lvmfit'
bootstrap(x,R=100,data=model.frame(x),
                            control=list(start=coef(x)),
                            p=coef(x), parametric=FALSE, bollenstine=FALSE,
                            estimator=x$estimator,weight=Weight(x),...)
```

Arguments

x	lvm-object.
R	Number of bootstrap samples
data	The data to resample from
fun	Optional function of the (bootstrapped) model-fit defining the statistic of interest
control	Options to the optimization routine
p	Parameter vector of the null model for the parametric bootstrap
parametric	If TRUE a parametric bootstrap is calculated. If FALSE a non-parametric (row-sampling) bootstrap is computed.
bollenstine	Bollen-Stine transformation (non-parametric bootstrap) for bootstrap hypothesis testing.
constraints	Logical indicating whether non-linear parameter constraints should be included in the bootstrap procedure
sd	Logical indicating whether standard error estimates should be included in the bootstrap procedure
silent	Suppress messages
estimator	String definining estimator, e.g. 'gaussian' (see estimator)
weight	Optional weight matrix used by estimator
...	Additional arguments, e.g. choice of estimator.

Value

A bootstrap.lvm object.

Author(s)

Klaus K. Holst

See Also

confint.lvmfit

Examples

```
m <- lvm(y~x)
d <- sim(m,100)
e <- estimate(y~x, d)
## Not run:
B <- bootstrap(e,R=100)
B

## End(Not run)
```

---

brisa *Simulated data*

Description

Simulated data

Format

data.frame

Source

Simulated

By *Apply a Function to a Data Frame Split by Factors*

Description

Apply a Function to a Data Frame Split by Factors

Usage

```
By(x, INDICES, FUN, COLUMNS, array = FALSE, ...)
```

Arguments

x	Data frame
INDICES	Indices (vector or list of indices, vector of column names, or formula of column names)
FUN	A function to be applied to data frame subsets of 'data'.
COLUMNS	(Optional) subset of columns of x to work on
array	if TRUE an array/matrix is always returned
...	Additional arguments to lower-level functions

Details

Simple wrapper of the 'by' function

Author(s)

Klaus K. Holst

Examples

```
By(datasets::CO2,~Treatment+Type,colMeans,~conc)
By(datasets::CO2,~Treatment+Type,colMeans,~conc+uptake)
```

---

calcium *Longitudinal Bone Mineral Density Data*

Description

Bone Mineral Density Data consisting of 112 girls randomized to receive calcium og placebo. Longitudinal measurements of bone mineral density (g/cm^2) measured approximately every 6th month in 3 years.

Format

A data.frame containing 560 (incomplete) observations. The 'person' column defines the individual girls of the study with measurements at visiting times 'visit', and age in years 'age' at the time of visit. The bone mineral density variable is 'bmd' (g/cm^2).

Source

Vonesh & Chinchilli (1997), Table 5.4.1 on page 228.

cancel *Generic cancel method*

Description

Generic cancel method

Usage

```
cancel(x, ...)
```

Arguments

x	Object
...	Additioal arguments

Author(s)

Klaus K. Holst

children *Extract children or parent elements of object*

Description

Generic method for extracting children or parent elements of object (e.g. a graph)

Usage

```
children(object, ...)
```

Arguments

object	Object
...	Additional arguments

Author(s)

Klaus K. Holst

`click` *Identify points on plot*

Description

Extension of the `identify` function

Usage

```
## Default S3 method:
click(x, y=NULL, label=TRUE, n=length(x), pch=19, col="orange", cex=3, ...)
idplot(x,y,...,id=list())
```

Arguments

`x`	X coordinates
`y`	Y coordinates
`label`	Should labels be added?
`n`	Max number of inputs to expect
`pch`	Symbol
`col`	Colour
`cex`	Size
`id`	List of arguments parsed to `click` function
`...`	Additional arguments parsed to `plot` function

Details

For the usual 'X11' device the identification process is terminated by pressing any mouse button other than the first. For the 'quartz' device the process is terminated by pressing either the pop-up menu equivalent (usually second mouse button or 'Ctrl'-click) or the 'ESC' key.

Author(s)

Klaus K. Holst

See Also

`idplot`, `identify`

Examples

```
n <- 10; x <- seq(1:n); y <- runif(n)
plot(y ~ x); click(x,y)

data(iris)
l <- lm(Sepal.Length ~ Sepal.Width*Species,iris)
res <- plotConf(l,var2="Species")## ylim=c(6,8), xlim=c(2.5,3.3))
with(res, click(x,y))

with(iris, idplot(Sepal.Length,Petal.Length))
```

---

closed.testing *Closed testing procedure*

Description

Closed testing procedure

Usage

```
closed.testing(object, idx = seq_along(coef(object)), null = rep(0,
  length(idx)), ...)
```

Arguments

object	estimate object
idx	Index of parameters to adjust for multiple testing
null	Null hypothesis value
...	Additional arguments

Examples

```
m <- lvm()
regression(m, c(y1,y2,y3,y4,y5,y6,y7)~x) <- c(0,0.25,0,0.25,0.25,0,0)
regression(m, to=endogenous(m), from="u") <- 1
variance(m,endogenous(m)) <- 1
set.seed(2)
d <- sim(m,200)
l1 <- lm(y1~x,d)
l2 <- lm(y2~x,d)
l3 <- lm(y3~x,d)
l4 <- lm(y4~x,d)
l5 <- lm(y5~x,d)
l6 <- lm(y6~x,d)
l7 <- lm(y7~x,d)
```

```
(a <- merge(l1,l2,l3,l4,l5,l6,l7,subset=2))
p.correct(a)
as.vector(closed.testing(a))
```

`Col` *Generate a transparent RGB color*

Description

This function transforms a standard color (e.g. "red") into an transparent RGB-color (i.e. alpha-blend<1).

Usage

```
Col(col, alpha = 0.2, locate = 0)
```

Arguments

`col`	Color (numeric or character)
`alpha`	Degree of transparency (0,1)
`locate`	Choose colour (with mouse)

Details

This only works for certain graphics devices (Cairo-X11 (x11 as of R>=2.7), quartz, pdf, ...).

Value

A character vector with elements of 7 or 9 characters, '"\#"' followed by the red, blue, green and optionally alpha values in hexadecimal (after rescaling to '0 ... 255').

Author(s)

Klaus K. Holst

Examples

```
plot(runif(1000),cex=runif(1000,0,4),col=Col(c("darkblue","orange"),0.5),pch=16)
```

`colorbar` *Add color-bar to plot*

Description

Add color-bar to plot

Usage

```
colorbar(clut = Col(rev(rainbow(11, start = 0, end = 0.69)), 0.5),
  x.range = c(-0.5, 0.5), y.range = c(-0.1, 0.1), values = seq(clut),
  digits = 2, label.offset, srt = 45, cex = 0.5, border = NA,
  position = 1, direction = c("horizontal", "vertical"), ...)
```

Arguments

`clut`	Color look-up table
`x.range`	x range
`y.range`	y range
`values`	label values
`digits`	number of digits
`label.offset`	label offset
`srt`	rotation of labels
`cex`	text size
`border`	border of color bar rectangles
`position`	Label position left/bottom (1) or top/right (2) or no text (0)
`direction`	horizontal or vertical color bars
`...`	additional low level arguments (i.e. parsed to `text`)

Examples

```
## Not run:
plotNeuro(x,roi=R,mm=-18,range=5)
colorbar(clut=Col(rev(rainbow(11,start=0,end=0.69)),0.5),
         x=c(-40,40),y.range=c(84,90),values=c(-5:5))

colorbar(clut=Col(rev(rainbow(11,start=0,end=0.69)),0.5),
         x=c(-10,10),y.range=c(-100,50),values=c(-5:5),
         direction="vertical",border=1)

## End(Not run)
```

Combine *Report estimates across different models*

Description

Report estimates across different models

Usage

```
Combine(x, ...)
```

Arguments

x	list of model objects
...	additional arguments to lower level functions

Author(s)

Klaus K. Holst

Examples

```
data(serotonin)
m1 <- lm(cau ~ age*gene1 + age*gene2,data=serotonin)
m2 <- lm(cau ~ age + gene1,data=serotonin)
m3 <- lm(cau ~ age*gene2,data=serotonin)

Combine(list(A=m1,B=m2,C=m3),fun=function(x) c(R2=format(summary(x)$r.squared,digits=2)))
```

compare *Statistical tests Performs Likelihood-ratio, Wald and score tests*

Description

Statistical tests

Performs Likelihood-ratio, Wald and score tests

Usage

```
compare(object, ...)
```

Arguments

object	lvmfit-object
...	Additional arguments to low-level functions

Value

Matrix of test-statistics and p-values

Author(s)

Klaus K. Holst

See Also

modelsearch, equivalence

Examples

```
m <- lvm();
regression(m) <- c(y1,y2,y3) ~ eta; latent(m) <- ~eta
regression(m) <- eta ~ x
m2 <- regression(m, c(y3,eta) ~ x)
set.seed(1)
d <- sim(m,1000)
e <- estimate(m,d)
e2 <- estimate(m2,d)

compare(e)

compare(e,e2) ## LRT, H0: y3<-x=0
compare(e,scoretest=y3~x) ## Score-test, H0: y3~x=0
compare(e2,par=c("y3~x")) ## Wald-test, H0: y3~x=0

B <- diag(2); colnames(B) <- c("y2~eta","y3~eta")
compare(e2,contrast=B,null=c(1,1))

B <- rep(0,length(coef(e2))); B[1:3] <- 1
compare(e2,contrast=B)

compare(e,scoretest=list(y3~x,y2~x))
```

---

confband *Add Confidence limits bar to plot*

Description

Add Confidence limits bar to plot

Usage

```
confband(x, lower, upper, center = NULL, delta = 0.07, centermark = 0.03,
  pch, blank = TRUE, vert = TRUE, polygon = FALSE, step = FALSE, ...)
```

Arguments

x	Position (x-coordinate if vert=TRUE, y-coordinate otherwise)
lower	Lower limit (if NULL no limits is added, and only the center is drawn (if not NULL))
upper	Upper limit
center	Center point
delta	Length of limit bars
centermark	Length of center bar
pch	Center symbol (if missing a line is drawn)
blank	If TRUE a white ball is plotted before the center is added to the plot
vert	If TRUE a vertical bar is plotted. Otherwise a horizontal bar is used
polygon	If TRUE polygons are added between 'lower' and 'upper'.
step	Type of polygon (step-function or piecewise linear)
...	Additional low level arguments (e.g. col, lwd, lty,...)

Author(s)

Klaus K. Holst

See Also

confband

Examples

```
plot(0,0,type="n",xlab="",ylab="")
confband(0.5,-0.5,0.5,0,col="darkblue")
confband(0.8,-0.5,0.5,0,col="darkred",vert=FALSE,pch=1,cex=1.5)

set.seed(1)
K <- 20
est <- rnorm(K); est[c(3:4,10:12)] <- NA
se <- runif(K,0.2,0.4)
x <- cbind(est,est-2*se,est+2*se,runif(K,0.5,2))
rownames(x) <- unlist(lapply(letters[seq(K)],function(x) paste(rep(x,4),collapse="")))
rownames(x)[which(is.na(est))] <- ""
signif <- sign(x[,2])==sign(x[,3])
forestplot(x,text.right=FALSE)
forestplot(x[,-4],sep=c(2,15),col=signif+1,box1=TRUE,delta=0.2,pch=16,cex=1.5)
##'
z <- seq(10)
zu <- c(z[-1],10)
plot(z,type="n")
confband(z,zu,rep(0,length(z)),col=Col("darkblue"),polygon=TRUE,step=TRUE)
confband(z,zu,zu-2,col=Col("darkred"),polygon=TRUE,step=TRUE)
##'
z <- seq(0,1,length.out=100)
```

```
plot(z,z,type="n")
confband(z,z,z^2,polygon="TRUE",col=Col("darkblue"))
```

`confint.lvmfit` *Calculate confidence limits for parameters*

Description

Calculate Wald og Likelihood based (profile likelihood) confidence intervals

Usage

```
## S3 method for class 'lvmfit'
confint(object, parm = seq_len(length(coef(object))),
  level = 0.95, profile = FALSE, curve = FALSE, n = 20,
  interval = NULL, lower = TRUE, upper = TRUE, ...)
```

Arguments

`object`	`lvm`-object.
`parm`	Index of which parameters to calculate confidence limits for.
`level`	Confidence level
`profile`	Logical expression defining whether to calculate confidence limits via the profile log likelihood
`curve`	if FALSE and profile is TRUE, confidence limits are returned. Otherwise, the profile curve is returned.
`n`	Number of points to evaluate profile log-likelihood in over the interval defined by `interval`
`interval`	Interval over which the profiling is done
`lower`	If FALSE the lower limit will not be estimated (profile intervals only)
`upper`	If FALSE the upper limit will not be estimated (profile intervals only)
`...`	Additional arguments to be passed to the low level functions

Details

Calculates either Wald confidence limits:

$$\hat{\theta} \pm z_{\alpha/2} * \hat{\sigma}_{\hat{\theta}}$$

or profile likelihood confidence limits, defined as the set of value τ:

$$logLik(\hat{\theta}_\tau, \tau) - logLik(\hat{\theta}) < q_\alpha/2$$

where q_α is the α fractile of the χ^2_1 distribution, and $\hat{\theta}_\tau$ are obtained by maximizing the log-likelihood with tau being fixed.

Value

A 2xp matrix with columns of lower and upper confidence limits

Author(s)

Klaus K. Holst

See Also

bootstrap{lvm}

Examples

```
m <- lvm(y~x)
d <- sim(m,100)
e <- estimate(y~x, d)
confint(e,3,profile=TRUE)
confint(e,3)
## Not run:
B <- bootstrap(e,R=100)
B

## End(Not run)
```

---

constrain<- *Add non-linear constraints to latent variable model*

Description

Add non-linear constraints to latent variable model

Usage

```
## Default S3 replacement method:
constrain(x,par,args,...) <- value

## S3 replacement method for class 'multigroup'
constrain(x,par,k=1,...) <- value

constraints(object,data=model.frame(object),vcov=object$vcov,level=0.95,
                       p=pars.default(object),k,idx,...)
```

Arguments

x	lvm-object
value	Real function taking args as a vector argument

par	Name of new parameter. Alternatively a formula with lhs specifying the new parameter and the rhs defining the names of the parameters or variable names defining the new parameter (overruling the args argument).
args	Vector of variables names or parameter names that are used in defining par
k	For multigroup models this argument specifies which group to add/extract the constraint
object	lvm-object
data	Data-row from which possible non-linear constraints should be calculated
vcov	Variance matrix of parameter estimates
level	Level of confidence limits
p	Parameter vector
idx	Index indicating which constraints to extract
...	Additional arguments to be passed to the low level functions

Details

Add non-linear parameter constraints as well as non-linear associations between covariates and latent or observed variables in the model (non-linear regression).

As an example we will specify the follow multiple regression model:

$$E(Y|X_1, X_2) = \alpha + \beta_1 X_1 + \beta_2 X_2$$

$$V(Y|X_1, X_2) = v$$

which is defined (with the appropiate parameter labels) as

```
m <- lvm(y ~ f(x,beta1) + f(x,beta2))
intercept(m) <- y ~ f(alpha)
covariance(m) <- y ~ f(v)
```

The somewhat strained parameter constraint

$$v = \frac{(beta1 - beta2)^2}{alpha}$$

can then specified as

```
constrain(m,v ~ beta1 + beta2 + alpha) <- function(x) (x[1]-x[2])^2/x[3]
```

A subset of the arguments args can be covariates in the model, allowing the specification of non-linear regression models. As an example the non-linear regression model

$$E(Y \mid X) = \nu + \Phi(\alpha + \beta X)$$

where Φ denotes the standard normal cumulative distribution function, can be defined as

```
m <- lvm(y ~ f(x,0)) # No linear effect of x
```

Next we add three new parameters using the parameter assigment function:

```
parameter(m) <- ~nu+alpha+beta
```

The intercept of Y is defined as mu

```
intercept(m) <- y ~ f(mu)
```

And finally the newly added intercept parameter mu is defined as the appropiate non-linear function of α, ν and β:

```
constrain(m, mu ~ x + alpha + nu) <- function(x) pnorm(x[1]*x[2])+x[3]
```

The constraints function can be used to show the estimated non-linear parameter constraints of an estimated model object (lvmfit or multigroupfit). Calling constrain with no additional arguments beyound x will return a list of the functions and parameter names defining the non-linear restrictions.

The gradient function can optionally be added as an attribute grad to the return value of the function defined by value. In this case the analytical derivatives will be calculated via the chain rule when evaluating the corresponding score function of the log-likelihood. If the gradient attribute is omitted the chain rule will be applied on a numeric approximation of the gradient.

Value

A lvm object.

Author(s)

Klaus K. Holst

See Also

regression, intercept, covariance

Examples

```
##############################
### Non-linear parameter constraints 1
##############################
m <- lvm(y ~ f(x1,gamma)+f(x2,beta))
covariance(m) <- y ~ f(v)
d <- sim(m,100)
m1 <- m; constrain(m1,beta ~ v) <- function(x) x^2
## Define slope of x2 to be the square of the residual variance of y
## Estimate both restricted and unrestricted model
e <- estimate(m,d,control=list(method="NR"))
e1 <- estimate(m1,d)
p1 <- coef(e1)
p1 <- c(p1[1:2],p1[3]^2,p1[3])
## Likelihood of unrestricted model evaluated in MLE of restricted model
logLik(e,p1)
## Likelihood of restricted model (MLE)
logLik(e1)

##############################
### Non-linear regression
##############################
```

```
## Simulate data
m <- lvm(c(y1,y2)~f(x,0)+f(eta,1))
latent(m) <- ~eta
covariance(m,~y1+y2) <- "v"
intercept(m,~y1+y2) <- "mu"
covariance(m,~eta) <- "zeta"
intercept(m,~eta) <- 0
set.seed(1)
d <- sim(m,100,p=c(v=0.01,zeta=0.01))[,manifest(m)]
d <- transform(d,
               y1=y1+2*pnorm(2*x),
               y2=y2+2*pnorm(2*x))

## Specify model and estimate parameters
constrain(m, mu ~ x + alpha + nu + gamma) <- function(x) x[4]*pnorm(x[3]+x[1]*x[2])

e <- estimate(m,d,control=list(trace=1,constrain=TRUE))
constraints(e,data=d)
## Plot model-fit
plot(y1~x,d,pch=16); points(y2~x,d,pch=16,col="gray")
x0 <- seq(-4,4,length.out=100)
lines(x0,coef(e)["nu"] + coef(e)["gamma"]*pnorm(coef(e)["alpha"]*x0))


##############################
### Multigroup model
##############################
### Define two models
m1 <- lvm(y ~ f(x,beta)+f(z,beta2))
m2 <- lvm(y ~ f(x,psi) + z)
### And simulate data from them
d1 <- sim(m1,500)
d2 <- sim(m2,500)
### Add 'non'-linear parameter constraint
constrain(m2,psi ~ beta2) <- function(x) x
## Add parameter beta2 to model 2, now beta2 exists in both models
parameter(m2) <- ~ beta2

ee <- estimate(list(m1,m2),list(d1,d2),control=list(method="NR"))
summary(ee)

m3 <- lvm(y ~ f(x,beta)+f(z,beta2))
m4 <- lvm(y ~ f(x,beta2) + z)
e2 <- estimate(list(m3,m4),list(d1,d2),control=list(method="NR"))
```

`correlation` *Generic method for extracting correlation coefficients of model object*

Description

Generic correlation method

Usage

```
correlation(x, ...)
```

Arguments

x	Object
...	Additional arguments

Author(s)

Klaus K. Holst

---

covariance *Add covariance structure to Latent Variable Model*

Description

Define covariances between residual terms in a lvm-object.

Usage

```
## S3 replacement method for class 'lvm'
covariance(object, var1=NULL, var2=NULL, constrain=FALSE, pairwise=FALSE,...) <- value
```

Arguments

object	lvm-object
var1	Vector of variables names (or formula)
var2	Vector of variables names (or formula) defining pairwise covariance between var1 and var2)
constrain	Define non-linear parameter constraints to ensure positive definite structure
pairwise	If TRUE and var2 is omitted then pairwise correlation is added between all variables in var1
...	Additional arguments to be passed to the low level functions
value	List of parameter values or (if var1 is unspecified)

Details

The covariance function is used to specify correlation structure between residual terms of a latent variable model, using a formula syntax.

For instance, a multivariate model with three response variables,

$$Y_1 = \mu_1 + \epsilon_1$$

$$Y_2 = \mu_2 + \epsilon_2$$

$$Y_3 = \mu_3 + \epsilon_3$$

can be specified as

```
m <- lvm(~y1+y2+y3)
```

Pr. default the two variables are assumed to be independent. To add a covariance parameter $r = cov(\epsilon_1, \epsilon_2)$, we execute the following code

```
covariance(m) <- y1 ~ f(y2,r)
```

The special function `f` and its second argument could be omitted thus assigning an unique parameter the covariance between `y1` and `y2`.

Similarily the marginal variance of the two response variables can be fixed to be identical ($var(Y_i) = v$) via

```
covariance(m) <- c(y1,y2,y3) ~ f(v)
```

To specify a completely unstructured covariance structure, we can call

```
covariance(m) <- ~y1+y2+y3
```

All the parameter values of the linear constraints can be given as the right handside expression of the assigment function `covariance<-` if the first (and possibly second) argument is defined as well. E.g:

```
covariance(m,y1~y1+y2) <- list("a1","b1")
covariance(m,~y2+y3) <- list("a2",2)
```

Defines

$$var(\epsilon_1) = a1$$

$$var(\epsilon_2) = a2$$

$$var(\epsilon_3) = 2$$

$$cov(\epsilon_1, \epsilon_2) = b1$$

Parameter constraints can be cleared by fixing the relevant parameters to `NA` (see also the `regression` method).

The function `covariance` (called without additional arguments) can be used to inspect the covariance constraints of a `lvm`-object.

Value

A `lvm`-object

Author(s)

Klaus K. Holst

See Also

regression<-, intercept<-, constrain<- parameter<-, latent<-, cancel<-, kill<-

Examples

```
m <- lvm()
### Define covariance between residuals terms of y1 and y2
covariance(m) <- y1~y2
covariance(m) <- c(y1,y2)~f(v) ## Same marginal variance
covariance(m) ## Examine covariance structure
```

---

curereg *Regression model for binomial data with unkown group of immortals*

Description

Regression model for binomial data with unkown group of immortals

Usage

```
curereg(formula, cureformula = ~1, data, family = binomial(),
  offset = NULL, start, var = "hessian", ...)
```

Arguments

formula	Formula specifying
cureformula	Formula for model of disease prevalence
data	data frame
family	Distribution family (see the help page family)
offset	Optional offset
start	Optional starting values
var	Type of variance (robust, expected, hessian, outer)
...	Additional arguments to lower level functions

Author(s)

Klaus K. Holst

Examples

```
## Simulation
n <- 2e3
x <- runif(n,0,20)
age <- runif(n,10,30)
z0 <- rnorm(n,mean=-1+0.05*age)
z <- cut(z0,breaks=c(-Inf,-1,0,1,Inf))
p0 <- lava:::expit(model.matrix(~z+age) %*% c(-.4, -.4, 0.2, 2, -0.05))
y <- (runif(n)<lava:::tigol(-1+0.25*x-0*age))*1
u <- runif(n)<p0
y[u==0] <- 0
d <- data.frame(y=y,x=x,u=u*1,z=z,age=age)
head(d)

## Estimation
e0 <- curereg(y~x*z,~1+z+age,data=d)
e <- curereg(y~x,~1+z+age,data=d)
compare(e,e0)
e
PD(e0,intercept=c(1,3),slope=c(2,6))

B <- rbind(c(1,0,0,0,20),
           c(1,1,0,0,20),
           c(1,0,1,0,20),
           c(1,0,0,1,20))
prev <- summary(e,pr.contrast=B)$prevalence

x <- seq(0,100,length.out=100)
newdata <- expand.grid(x=x,age=20,z=levels(d$z))
fit <- predict(e,newdata=newdata)
plot(0,0,type="n",xlim=c(0,101),ylim=c(0,1),xlab="x",ylab="Probability(Event)")
count <- 0
for (i in levels(newdata$z)) {
  count <- count+1
  lines(x,fit[which(newdata$z==i)],col="darkblue",lty=count)
}
abline(h=prev[3:4,1],lty=3:4,col="gray")
abline(h=prev[3:4,2],lty=3:4,col="lightgray")
abline(h=prev[3:4,3],lty=3:4,col="lightgray")
legend("topleft",levels(d$z),col="darkblue",lty=seq_len(length(levels(d$z))))
```

---

devcoords *Returns device-coordinates and plot-region*

Description

Returns device-coordinates and plot-region

Usage

```
devcoords()
```

Value

A `list` with elements

`dev.x1`	Device: Left x-coordinate
`dev.x2`	Device: Right x-coordinate
`dev.y1`	Device Bottom y-coordinate
`dev.y2`	Device Top y-coordinate
`fig.x1`	Plot: Left x-coordinate
`fig.x2`	Plot: Right x-coordinate
`fig.y1`	Plot: Bottom y-coordinate
`fig.y2`	Plot: Top y-coordinate

Author(s)

Klaus K. Holst

`dsort` *Sort data frame*

Description

Sort data according to columns in data frame

Usage

```
dsort(data, ...)
```

Arguments

`data`	Data frame
`...`	Arguments to order

Value

data.frame

Examples

```
data(hubble)
dsort(hubble, "sigma")
dsort(hubble, hubble$sigma)
dsort(hubble,sigma,v)
```

equivalence *Identify candidates of equivalent models*

Description

Identifies candidates of equivalent models

Usage

```
equivalence(x, rel, tol = 0.001, k = 1, omitrel = TRUE, ...)
```

Arguments

x	lvmfit-object
rel	Formula or character-vector specifying two variables to omit from the model and subsequently search for possible equivalent models
tol	Define two models as empirical equivalent if the absolute difference in score test is less than tol
k	Number of parameters to test simultaneously. For equivalence the number of additional associations to be added instead of rel.
omitrel	if k greater than 1, this boolean defines wether to omit candidates containing rel from the output
...	Additional arguments to be passed to the low level functions

Author(s)

Klaus K. Holst

See Also

compare, modelsearch

estimate.default *Estimation of functional of parameters*

Description

Estimation of functional of parameters. Wald tests, robust standard errors, cluster robust standard errors, LRT (when f is not a function)...

Usage

```
## Default S3 method:
estimate(x = NULL, f = NULL, data, id, iddata,
  stack = TRUE, average = FALSE, subset, score.deriv, level = 0.95,
  iid = TRUE, type = c("robust", "df", "mbn"), keep, contrast, null, vcov,
  coef, print = NULL, labels, ...)
```

Arguments

x	model object (glm, lvmfit, ...)
f	transformation of model parameters and (optionally) data, or contrast matrix (or vector)
data	data.frame
id	(optional) id-variable corresponding to iid decomposition of model parameters.
iddata	(optional) id-variable for 'data'
stack	If TRUE (default) the i.i.d. decomposition is automatically stacked according to 'id'
average	If TRUE averages are calculated
subset	(optional) subset of data.frame on which to condition (logical expression or variable name)
score.deriv	(optional) derivative of mean score function
level	Level of confidence limits
iid	If TRUE (default) the iid decompositions are also returned (extract with iid method)
type	Type of small-sample correction
keep	(optional) Index of parameters to keep
contrast	(optional) Contrast matrix for final Wald test
null	(optional) Null hypothesis to test
vcov	(optional) covariance matrix of parameter estimates (e.g. Wald-test)
coef	(optional) parameter coefficient
print	(optional) print function
labels	(optional) names of coefficients
...	additional arguments to lower level functions

Examples

```
## Simulation from logistic regression model
m <- lvm(y~x+z);
distribution(m,y~x) <- binomial.lvm("logit")
d <- sim(m,1000)
g <- glm(y~z+x,data=d,family=binomial())
g0 <- glm(y~1,data=d,family=binomial())
```

```
## LRT
estimate(g,g0)

## Plain estimates (robust standard errors)
estimate(g)

## Testing contrasts
estimate(g,null=0)
estimate(g,rbind(c(1,1,0),c(1,0,2)))
estimate(g,rbind(c(1,1,0),c(1,0,2)),null=c(1,2))
estimate(g,2:3) ## same as rbind(c(0,1,0),c(0,0,1))

## Transformations
estimate(g,function(p) p[1]+p[2])

## Multiple parameters
e <- estimate(g,function(p) c(p[1]+p[2],p[1]*p[2]))
e
vcov(e)

## Label new parameters
estimate(g,function(p) list("a1"=p[1]+p[2],"b1"=p[1]*p[2]))

## Multiple group
m <- lvm(y~x)
m <- baptize(m)
d2 <- d1 <- sim(m,50)
e <- estimate(list(m,m),list(d1,d2))
estimate(e) ## Wrong
estimate(e,id=rep(seq(nrow(d1)),2))
estimate(lm(y~x,d1))

## Marginalize
f <- function(p,data)
  list(p0=lava:::expit(p[1] + p[3]*data[,"z"]),
       p1=lava:::expit(p[1] + p[2] + p[3]*data[,"z"]))
e <- estimate(g, f, average=TRUE)
e
estimate(e,diff)
estimate(e,cbind(1,1))

## Clusters and subset (conditional marginal effects)
d$id <- rep(seq(nrow(d)/4),each=4)
estimate(g,function(p,data)
         list(p0=lava:::expit(p[1] + p["z"]*data[,"z"])),
         subset=d$z>0, id=d$id, average=TRUE)

## More examples with clusters:
m <- lvm(c(y1,y2,y3)~u+x)
d <- sim(m,10)
l1 <- glm(y1~x,data=d)
l2 <- glm(y2~x,data=d)
l3 <- glm(y3~x,data=d)
```

```
## Some random id-numbers
id1 <- c(1,1,4,1,3,1,2,3,4,5)
id2 <- c(1,2,3,4,5,6,7,8,1,1)
id3 <- seq(10)

## Un-stacked and stacked i.i.d. decomposition
iid(estimate(l1,id=id1,stack=FALSE))
iid(estimate(l1,id=id1))

## Combined i.i.d. decomposition
e1 <- estimate(l1,id=id1)
e2 <- estimate(l2,id=id2)
e3 <- estimate(l3,id=id3)
(a2 <- merge(e1,e2,e3))

## Same:
iid(a1 <- merge(l1,l2,l3,id=list(id1,id2,id3)))

iid(merge(l1,l2,l3,id=TRUE)) # one-to-one (same clusters)
iid(merge(l1,l2,l3,id=FALSE)) # independence
```

`estimate.lvm` *Estimation of parameters in a Latent Variable Model (lvm)*

Description

Estimate parameters. MLE, IV or user-defined estimator.

Usage

```
## S3 method for class 'lvm'
estimate(x, data = parent.frame(), estimator = "gaussian",
  control = list(), missing = FALSE, weight, weightname, weight2, cluster,
  fix, index = TRUE, graph = FALSE, silent = lava.options()$silent,
  quick = FALSE, param, ...)
```

Arguments

`x`	`lvm`-object
`data`	`data.frame`
`estimator`	String defining the estimator (see details below)
`control`	control/optimization parameters (see details below)
`missing`	Logical variable indiciating how to treat missing data. Setting to FALSE leads to complete case analysis. In the other case likelihood based inference is obtained by integrating out the missing data under assumption the assumption that data is missing at random (MAR).

weight	Optional weights to used by the chosen estimator.
weightname	Weight names (variable names of the model) in case weight was given as a vector of column names of data
weight2	Optional additional dataset used by the chosen estimator.
cluster	Vector (or name of column in data) that identifies correlated groups of observations in the data leading to variance estimates based on a sandwich estimator
fix	Logical variable indicating whether parameter restriction automatically should be imposed (e.g. intercepts of latent variables set to 0 and at least one regression parameter of each measurement model fixed to ensure identifiability.)
index	For internal use only
graph	For internal use only
silent	Logical argument indicating whether information should be printed during estimation
quick	If TRUE the parameter estimates are calculated but all additional information such as standard errors are skipped
param	set parametrization (see help(lava.options))
...	Additional arguments to be passed to the low level functions

Details

A list of parameters controlling the estimation and optimization procedures is parsed via the control argument. By default Maximum Likelihood is used assuming multivariate normal distributed measurement errors. A list with one or more of the following elements is expected:

start: Starting value. The order of the parameters can be shown by calling coef (with mean=TRUE) on the lvm-object or with plot(..., labels=TRUE). Note that this requires a check that it is actual the model being estimated, as estimate might add additional restriction to the model, e.g. through the fix and exo.fix arguments. The lvm-object of a fitted model can be extracted with the Model-function.

starterfun: Starter-function with syntax function(lvm, S, mu). Three builtin functions are available: startvalues, startvalues0, startvalues1, ...

estimator: String defining which estimator to use (Defaults to "gaussian")

meanstructure Logical variable indicating whether to fit model with meanstructure.

method: String pointing to alternative optimizer (e.g. optim to use simulated annealing).

control: Parameters passed to the optimizer (default stats::nlminb).

tol: Tolerance of optimization constraints on lower limit of variance parameters.

Value

A lvmfit-object.

Author(s)

Klaus K. Holst

See Also

score, information, ...

Examples

```
dd <- read.table(header=TRUE,
text="x1 x2 x3
1.0 2.0 1.4
2.1 4.1 4.0
3.1 3.4 7.0
4.2 6.1 3.5
5.3 5.2 2.3
1.1 1.6 2.9")
e <- estimate(lvm(c(x1,x2,x3)~u),dd)

## Simulation example
m <- lvm(list(y~v1+v2+v3+v4,c(v1,v2,v3,v4)~x))
covariance(m) <- v1~v2+v3+v4
dd <- sim(m,10000) ## Simulate 10000 observations from model
e <- estimate(m, dd) ## Estimate parameters
e

## Using just sufficient statistics
n <- nrow(dd)
e0 <- estimate(m,data=list(S=cov(dd)*(n-1)/n,mu=colMeans(dd),n=n))

## Multiple group analysis
m <- lvm()
regression(m) <- c(y1,y2,y3)~u
regression(m) <- u~x
d1 <- sim(m,100,p=c("u,u"=1,"u~x"=1))
d2 <- sim(m,100,p=c("u,u"=2,"u~x"=-1))

mm <- baptize(m)
regression(mm,u~x) <- NA
covariance(mm,~u) <- NA
intercept(mm,~u) <- NA
ee <- estimate(list(mm,mm),list(d1,d2))

## Missing data
d0 <- makemissing(d1,cols=1:2)
e0 <- estimate(m,d0,missing=TRUE)
e0
```

---

eventTime *Add an observed event time outcome to a latent variable model.*

Description

For example, if the model 'm' includes latent event time variables are called 'T1' and 'T2' and 'C' is the end of follow-up (right censored), then one can specify

Usage

```
eventTime(object, formula, eventName, ...)
```

Arguments

object	Model object
formula	Formula (see details)
eventName	Event names
...	Additional arguments to lower levels functions

Details

`eventTime(object=m,formula=ObsTime~min(T1=a,T2=b,C=0,"ObsEvent"))`

when data are simulated from the model one gets 2 new columns:

- "ObsTime": the smallest of T1, T2 and C - "ObsEvent": 'a' if T1 is smallest, 'b' if T2 is smallest and '0' if C is smallest

Note that "ObsEvent" and "ObsTime" are names specified by the user.

Author(s)

Thomas A. Gerds

Examples

```
# Right censored survival data without covariates
m0 <- lvm()
distribution(m0,"eventtime") <- coxWeibull.lvm(scale=1/100,shape=2)
distribution(m0,"censtime") <- coxExponential.lvm(rate=10)
m0 <- eventTime(m0,time~min(eventtime=1,censtime=0),"status")
sim(m0,10)

# Alternative specification of the right censored survival outcome
## eventTime(m,"Status") <- ~min(eventtime=1,censtime=0)

# Cox regression:
# lava implements two different parametrizations of the same
# Weibull regression model. The first specifies
# the effects of covariates as proportional hazard ratios
# and works as follows:
m <- lvm()
distribution(m,"eventtime") <- coxWeibull.lvm(scale=1/100,shape=2)
distribution(m,"censtime") <- coxWeibull.lvm(scale=1/100,shape=2)
m <- eventTime(m,time~min(eventtime=1,censtime=0),"status")
distribution(m,"sex") <- binomial.lvm(p=0.4)
distribution(m,"sbp") <- normal.lvm(mean=120,sd=20)
regression(m,from="sex",to="eventtime") <- 0.4
regression(m,from="sbp",to="eventtime") <- -0.01
sim(m,6)
# The parameters can be recovered using a Cox regression
```

```
# routine or a Weibull regression model. E.g.,
## Not run:
    set.seed(18)
    d <- sim(m,1000)
    library(survival)
    coxph(Surv(time,status)~sex+sbp,data=d)

    sr <- survreg(Surv(time,status)~sex+sbp,data=d)
    library(SurvRegCensCov)
    ConvertWeibull(sr)


## End(Not run)

# The second parametrization is an accelerated failure time
# regression model and uses the function weibull.lvm instead
# of coxWeibull.lvm to specify the event time distributions.
# Here is an example:

ma <- lvm()
distribution(ma,"eventtime") <- weibull.lvm(scale=3,shape=0.7)
distribution(ma,"censtime") <- weibull.lvm(scale=2,shape=0.7)
ma <- eventTime(ma,time~min(eventtime=1,censtime=0),"status")
distribution(ma,"sex") <- binomial.lvm(p=0.4)
distribution(ma,"sbp") <- normal.lvm(mean=120,sd=20)
regression(ma,from="sex",to="eventtime") <- 0.7
regression(ma,from="sbp",to="eventtime") <- -0.008
set.seed(17)
sim(ma,6)
# The regression coefficients of the AFT model
# can be tranformed into log(hazard ratios):
#  coef.coxWeibull = - coef.weibull / shape.weibull
## Not run:
    set.seed(17)
    da <- sim(ma,1000)
    library(survival)
    fa <- coxph(Surv(time,status)~sex+sbp,data=da)
    coef(fa)
    c(0.7,-0.008)/0.7

## End(Not run)


# The Weibull parameters are related as follows:
# shape.coxWeibull = 1/shape.weibull
# scale.coxWeibull = exp(-scale.weibull/shape.weibull)
# scale.AFT = log(scale.coxWeibull) / shape.coxWeibull
# Thus, the following are equivalent parametrizations
# which produce exactly the same random numbers:

model.aft <- lvm()
distribution(model.aft,"eventtime") <- weibull.lvm(scale=-log(1/100)/2,shape=0.5)
distribution(model.aft,"censtime") <- weibull.lvm(scale=-log(1/100)/2,shape=0.5)
```

```
set.seed(17)
sim(model.aft,6)

model.cox <- lvm()
distribution(model.cox,"eventtime") <- coxWeibull.lvm(scale=1/100,shape=2)
distribution(model.cox,"censtime") <- coxWeibull.lvm(scale=1/100,shape=2)
set.seed(17)
sim(model.cox,6)

# The minimum of multiple latent times one of them still
# being a censoring time, yield
# right censored competing risks data

mc <- lvm()
distribution(mc,~X2) <- binomial.lvm()
regression(mc) <- T1~f(X1,-.5)+f(X2,0.3)
regression(mc) <- T2~f(X2,0.6)
distribution(mc,~T1) <- coxWeibull.lvm(scale=1/100)
distribution(mc,~T2) <- coxWeibull.lvm(scale=1/100)
distribution(mc,~C) <- coxWeibull.lvm(scale=1/100)
mc <- eventTime(mc,time~min(T1=1,T2=2,C=0),"event")
sim(mc,6)
```

Expand *Create a Data Frame from All Combinations of Factors*

Description

Create a Data Frame from All Combinations of Factors

Usage

```
Expand(x, ...)
```

Arguments

x	Data.frame
...	vectors, factors or a list containing these

Details

Simple wrapper of the 'expand.grid' function. If x is a table then a data frame is returned with one row pr individual observation.

Author(s)

Klaus K. Holst

Examples

```
dd <- Expand(iris, Sepal.Length=2:8, Species=c("virginica","setosa"))
summary(dd)

T <- with(warpbreaks, table(wool, tension))
Expand(T)
```

`getMplus` *Read Mplus output*

Description

Read Mplus output files

Usage

```
getMplus(infile = "template.out", coef = TRUE, ...)
```

Arguments

`infile`	Mplus output file
`coef`	Coefficients only
`...`	additional arguments to lower level functions

Author(s)

Klaus K. Holst

See Also

getSAS

`getSAS` *Read SAS output*

Description

Run SAS code like in the following:

Usage

```
getSAS(infile, entry = "Parameter Estimates", ...)
```

Arguments

infile	file (csv file generated by ODS)
entry	Name of entry to capture
...	additional arguments to lower level functions

Details

ODS CSVALL BODY="myest.csv"; proc nlmixed data=aj qpoints=2 dampstep=0.5; ... run; ODS CSVALL Close;

and read results into R with:

```
getsas("myest.csv","Parameter Estimates")
```

Author(s)

Klaus K. Holst

See Also

getMplus

---

gof *Extract model summaries and GOF statistics for model object*

Description

Calculates various GOF statistics for model object including global chi-squared test statistic and AIC. Extract model-specific mean and variance structure, residuals and various predicitions.

Usage

```
gof(object, ...)

## S3 method for class 'lvmfit'
gof(object, chisq=FALSE, level=0.90, rmsea.threshold=0.05,all=FALSE,...)

moments(x,...)

## S3 method for class 'lvm'
moments(x, p, debug=FALSE, conditional=FALSE, data=NULL, ...)

## S3 method for class 'lvmfit'
logLik(object, p=coef(object),
                      data=model.frame(object),
                      model=object$estimator,
                      weight=Weight(object),
                      weight2=object$data$weight2,
```

```
                          ...)

## S3 method for class 'lvmfit'
score(x, data=model.frame(x), p=pars(x), model=x$estimator,
                  weight=Weight(x), weight2=x$data$weight2, ...)

## S3 method for class 'lvmfit'
information(x,p=pars(x),n=x$data$n,data=model.frame(x),
               model=x$estimator,weight=Weight(x), weight2=x$data$weight2, ...)
```

Arguments

object	Model object
x	Model object
p	Parameter vector used to calculate statistics
data	Data.frame to use
weight2	Optional second data.frame (only for censored observations)
weight	Optional weight matrix
n	Number of observations
conditional	If TRUE the conditional moments given the covariates are calculated. Otherwise the joint moments are calculated
model	String defining estimator, e.g. "gaussian" (see estimate)
debug	Debugging only
chisq	Boolean indicating whether to calculate chi-squared goodness-of-fit (always TRUE for estimator='gaussian')
level	Level of confidence limits for RMSEA
rmsea.threshold	Which probability to calculate, Pr(RMSEA<rmsea.treshold)
all	Calculate all (ad hoc) FIT indices: TLI, CFI, NFI, SRMR, ...
...	Additional arguments to be passed to the low level functions

Value

A htest-object.

Author(s)

Klaus K. Holst

Examples

```
m <- lvm(list(y~v1+v2+v3+v4,c(v1,v2,v3,v4)~x))
set.seed(1)
dd <- sim(m,1000)
e <- estimate(m, dd)
gof(e,all=TRUE,rmsea.threshold=0.05,level=0.9)
```

```
set.seed(1)
m <- lvm(list(c(y1,y2,y3)~u,y1~x)); latent(m) <- ~u
regression(m,c(y2,y3)~u) <- "b"
d <- sim(m,1000)
e <- estimate(m,d)
rsq(e)
##'
rr <- rsq(e,TRUE)
rr
estimate(rr,contrast=rbind(c(1,-1,0),c(1,0,-1),c(0,1,-1)))
```

`Graph` *Extract graph*

Description

Extract or replace graph object

Usage

```
Graph(x, ...)

Graph(x, ...) <- value
```

Arguments

`x`	Model object
`value`	New `graphNEL` object
`...`	Additional arguments to be passed to the low level functions

Author(s)

Klaus K. Holst

See Also

`Model`

Examples

```
m <- lvm(y~x)
Graph(m)
```

hubble *Hubble data*

Description

Velocity (v) and distance (D) measures of 36 Type Ia super-novae from the Hubble Space Telescope

Format

data.frame

Source

Freedman, W. L., et al. 2001, AstroPhysicalJournal, 553, 47.

hubble2 *Hubble data*

Description

Hubble data

Format

data.frame

See Also

hubble

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

Description

Extract i.i.d. decomposition (influence function) from model object

Usage

```
iid(x,...)

## Default S3 method:
iid(x,bread,id,...)
```

Arguments

x	model object
...	additional arguments
id	id/cluster variable (optional)
bread	(optional) Inverse of derivative of mean score function

Examples

```
m <- lvm(y~x+z)
distribution(m, ~y+z) <- binomial.lvm("logit")
d <- sim(m,1e3)
g <- glm(y~x+z,data=d,family=binomial)
crossprod(iid(g))
```

---

images *Organize several image calls (for visualizing categorical data)*

Description

Visualize categorical by group variable

Usage

```
images(x, group, ncol = 2, byrow = TRUE, colorbar = 1,
  colorbar.space = 0.1, label.offset = 0.02, order = TRUE,
  colorbar.border = 0, main, rowcol = FALSE, plotfun = NULL, axis1, axis2,
  mar, col = list(c("#EFF3FF", "#BDD7E7", "#6BAED6", "#2171B5"), c("#FEE5D9",
  "#FCAE91", "#FB6A4A", "#CB181D"), c("#EDF8E9", "#BAE4B3", "#74C476",
  "#238B45"), c("#FEEDDE", "#FDBE85", "#FD8D3C", "#D94701")), ...)
```

Arguments

x	data.frame or matrix
group	group variable
ncol	number of columns in layout
byrow	organize by row if TRUE
colorbar	Add color bar
colorbar.space	Space around color bar
label.offset	label offset
order	order
colorbar.border	Add border around color bar
main	Main title

rowcol	switch rows and columns
plotfun	Alternative plot function (instead of 'image')
axis1	Axis 1
axis2	Axis 2
mar	Margins
col	Colours
...	Additional arguments to lower level graphics functions

Author(s)

Klaus Holst

Examples

```
X <- matrix(rbinom(400,3,0.5),20)
group <- rep(1:4,each=5)
images(X,colorbar=0,zlim=c(0,3))
images(X,group=group,zlim=c(0,3))
## Not run:
images(X,group=group,col=list(RColorBrewer::brewer.pal(4,"Purples"),
                              RColorBrewer::brewer.pal(4,"Greys"),
                              RColorBrewer::brewer.pal(4,"YlGn"),
                           RColorBrewer::brewer.pal(4,"PuBuGn")),colorbar=2,zlim=c(0,3))

## End(Not run)
images(list(X,X,X,X),group=group,zlim=c(0,3))
images(list(X,X,X,X),ncol=1,group=group,zlim=c(0,3))
images(list(X,X),group,axis2=c(FALSE,FALSE),axis1=c(FALSE,FALSE),
      mar=list(c(0,0,0,0),c(0,0,0,0)),yaxs="i",xaxs="i",zlim=c(0,3))
```

---

indoorenv *Data*

Description

Description

Format

data.frame

Source

Simulated

`intercept` *Fix mean parameters in 'lvm'-object*

Description

Define linear constraints on intercept parameters in a `lvm`-object.

Usage

```
## S3 replacement method for class 'lvm'
intercept(object, vars, ...) <- value
```

Arguments

`object`	`lvm`-object
`vars`	character vector of variable names
`value`	Vector (or list) of parameter values or labels (numeric or character) or a formula defining the linear constraints (see also the `regression` or `covariance` methods).
`...`	Additional arguments

Details

The `intercept` function is used to specify linear constraints on the intercept parameters of a latent variable model. As an example we look at the multivariate regression model

$$E(Y_1|X) = \alpha_1 + \beta_1 X$$

$$E(Y_2|X) = \alpha_2 + \beta_2 X$$

defined by the call

```
m <- lvm(c(y1,y2) ~ x)
```

To fix $\alpha_1 = \alpha_2$ we call

```
intercept(m) <- c(y1,y2) ~ f(mu)
```

Fixed parameters can be reset by fixing them to `NA`. For instance to free the parameter restriction of Y_1 and at the same time fixing $\alpha_2 = 2$, we call

```
intercept(m, ~y1+y2) <- list(NA,2)
```

Calling `intercept` with no additional arguments will return the current intercept restrictions of the `lvm`-object.

Value

A `lvm`-object

Note

Variables will be added to the model if not already present.

Author(s)

Klaus K. Holst

See Also

covariance<-, regression<-, constrain<-, parameter<-, latent<-, cancel<-, kill<-

Examples

```
## A multivariate model
m <- lvm(c(y1,y2) ~ f(x1,beta)+x2)
regression(m) <- y3 ~ f(x1,beta)
intercept(m) <- y1 ~ f(mu)
intercept(m, ~y2+y3) <- list(2,"mu")
intercept(m) ## Examine intercepts of model (NA translates to free/unique paramete##r)
```

---

kill *Remove variables from (model) object.*

Description

Generic method for removing elements of object

Usage

```
kill(x, ...) <- value
```

Arguments

x	Model object
value	Vector of variables or formula specifying which nodes to remove
...	additional arguments to lower level functions

Author(s)

Klaus K. Holst

See Also

cancel

Examples

```
m <- lvm()
addvar(m) <- ~y1+y2+x
covariance(m) <- y1~y2
regression(m) <- c(y1,y2) ~ x
### Cancel the covariance between the residuals of y1 and y2
cancel(m) <- y1~y2
### Remove y2 from the model
rmvar(m) <- ~y2
```

---

ksmooth2 *Plot/estimate surface*

Description

Plot/estimate surface

Usage

```
ksmooth2(x, data, h = NULL, xlab = NULL, ylab = NULL, zlab = "",
  gridsize = rep(51L, 2), ...)
```

Arguments

x	formula or data
data	data.frame
h	bandwidth
xlab	X label
ylab	Y label
zlab	Z label
gridsize	grid size of kernel smoother
...	Additional arguments to graphics routine (persp3d or persp)

Examples

```
ksmooth2(rmvn(1e4,sigma=diag(2)*.5+.5),c(-3.5,3.5),h=1)
ksmooth2(rmvn(1e4,sigma=diag(2)*.5+.5),c(-3.5,3.5),h=1,
        rgl=FALSE,theta=30)

ksmooth2(function(x,y) x^2+y^2, c(-20,20))
ksmooth2(function(x,y) x^2+y^2, xlim=c(-5,5), ylim=c(0,10))

f <- function(x,y) 1-sqrt(x^2+y^2)
surface(f,xlim=c(-1,1),alpha=0.9,aspect=c(1,1,0.75))
surface(f,xlim=c(-1,1),clut=heat.colors(128))
```

```
##play3d(spin3d(axis=c(0,0,1), rpm=8), duration=5)

ksmooth2(f,c(-1,1),rgl=FALSE,image=fields::image.plot)

surface(function(x) dmvn(x,sigma=diag(2)),c(-3,3),lit=FALSE,smooth=FALSE,box=FALSE,alpha=0.8)
```

`labels<-` *Define labels of graph*

Description

Alters labels of nodes and edges in the graph of a latent variable model

Usage

```
## Default S3 replacement method:
labels(object, ...) <- value
## S3 replacement method for class 'lvm'
edgelabels(object, to, ...) <- value
## Default S3 replacement method:
nodecolor(object, var=vars(object),
border, labcol, shape, lwd, ...) <- value
```

Arguments

`object`	`lvm`-object.
`value`	node label/edge label/color
`to`	Formula specifying outcomes and predictors defining relevant edges.
`...`	Additional arguments (`lwd`, `cex`, `col`, `labcol`), `border`.
`var`	Formula or character vector specifying the nodes/variables to alter.
`border`	Colors of borders
`labcol`	Text label colors
`shape`	Shape of node
`lwd`	Line width of border

Author(s)

Klaus K. Holst

Examples

```
m <- lvm(c(y,v)~x+z)
regression(m) <- c(v,x)~z
labels(m) <- c(y=expression(psi), z=expression(zeta))
nodecolor(m,~y+z+x,border=c("white","white","black"),
          labcol="white", lwd=c(1,1,5),
          lty=c(1,2)) <-  c("orange","indianred","lightgreen")
edgelabels(m,y~z+x, cex=c(2,1.5), col=c("orange","black"),labcol="darkblue",
           arrowhead=c("tee","dot"),
           lwd=c(3,1)) <- expression(phi,rho)
edgelabels(m,c(v,x)~z, labcol="red", cex=0.8,arrowhead="none") <- 2
plot(m,addstyle=FALSE)

m <- lvm(y~x)
labels(m) <- list(x="multiple\nlines")

op <- par(mfrow=c(1,2))
plot(m,plain=TRUE)
plot(m)
par(op)



d <- sim(m,100)
e <- estimate(m,d)
plot(e,type="sd")
```

`lava.options` *Set global options for* `lava`

Description

Extract and set global parameters of `lava`. In particular optimization parameters for the `estimate` function.

Usage

```
lava.options(...)
```

Arguments

`...`	Arguments

Details

- `param`: 'relative' (factor loading and variance of one endogenous variables in each measurement model are fixed to one), 'absolute' (mean and variance of latent variables are set to 0 and 1, respectively), 'hybrid' (intercept of latent variables is fixed to 0, and factor loading

of at least one endogenous variable in each measurement model is fixed to 1), 'none' (no constraints are added)

- `silent`: Set to `FALSE` to disable various output messages
- ...

see `control` parameter of the `estimate` function.

Value

`list` of parameters

Author(s)

Klaus K. Holst

Examples

```
## Not run:
lava.options(iter.max=100,silent=TRUE)

## End(Not run)
```

`lvm` *Initialize new latent variable model*

Description

Function that constructs a new latent variable model object

Usage

```
lvm(x = NULL, ..., silent = lava.options()$silent)
```

Arguments

`x`	Vector of variable names. Optional but gives control of the sequence of appearance of the variables. The argument can be given as a character vector or formula, e.g. `~y1+y2` is equivalent to `c("y1","y2")`. Alternatively the argument can be a formula specifying a linear model.
`silent`	Logical variable which indicates whether messages are turned on/off.
`...`	Additional arguments to be passed to the low level functions

Value

Returns an object of class `lvm`.

Author(s)

Klaus K. Holst

See Also

regression, covariance, intercept, ...

Examples

```
m <- lvm() # Empty model
m1 <- lvm(y~x) # Simple linear regression
m2 <- lvm(~y1+y2) # Model with two independent variables (argument)
m3 <- lvm(list(c(y1,y2,y3)~u,u~x+z)) # SEM with three items
```

---

makemissing *Create random missing data*

Description

Generates missing entries in data.frame/matrix

Usage

```
makemissing(data, p = 0.2, cols = 1:ncol(data), rowwise = FALSE,
  nafun = function(x) x)
```

Arguments

data	data.frame
p	Fraction of missing data in each column
cols	Which columns (name or index) to alter
rowwise	Should missing occur row-wise (either none or all selected columns are missing)
nafun	(Optional) function to be applied on data.frame before return (e.g. na.omit to return complete-cases only)

Value

data.frame

Author(s)

Klaus K. Holst

`Missing` *Missing value generator*

Description

Missing value generator

Usage

```
Missing(object, formula, Rformula, missing.name, suffix = "0", ...)
```

Arguments

`object`	`lvm`-object.
`formula`	The right hand side specifies the name of a latent variable which is not always observed. The left hand side specifies the name of a new variable which is equal to the latent variable but has missing values. If given as a string then this is used as the name of the latent (full-data) name, and the observed data name is 'missing.data'
`Rformula`	Missing data mechanism with left hand side specifying the name of the missing value indicator (may also just be given as a character instead of a formula)
`missing.name`	Name of observed data variable (only used if 'formula' was given as a character specifying the name of the full-data variable)
`suffix`	If missing.name is missing, then the name of the oberved data variable will be the name of the full-data variable + the suffix
`...`	Passed to binomial.lvm.

Details

This function adds a binary variable to a given `lvm` model and also a variable which is equal to the original variable where the binary variable is equal to zero

Value

lvm object

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```
library(lava)
set.seed(17)
m <- lvm(y0~x01+x02+x03)
m <- Missing(m,formula=x1~x01,Rformula=R1~0.3*x02+-0.7*x01,p=0.4)
sim(m,10)


m <- lvm(y~1)
m <- Missing(m,"y","r")
## same as
## m <- Missing(m,y~1,r~1)
sim(m,10)

## same as
m <- lvm(y~1)
Missing(m,"y") <- r~x
sim(m,10)

m <- lvm(y~1)
m <- Missing(m,"y","r",suffix=".")
## same as
## m <- Missing(m,"y","r",missing.name="y.")
## same as
## m <- Missing(m,y.~y,"r")
sim(m,10)
```

---

missingdata *Missing data example*

Description

Simulated data generated from model

$$E(Y_i \mid X) = X, \quad cov(Y_1, Y_2 \mid X) = 0.5$$

Format

list of data.frames

Details

The list contains four data sets 1) Complete data 2) MCAR 3) MAR 4) MNAR (missing mechanism depends on variable V correlated with Y1,Y2)

Source

Simulated

Examples

```
data(missingdata)
e0 <- estimate(lvm(c(y1,y2)~b*x,y1~~y2),missingdata[[1]]) ## No missing
e1 <- estimate(lvm(c(y1,y2)~b*x,y1~~y2),missingdata[[2]]) ## CC (MCAR)
e2 <- estimate(lvm(c(y1,y2)~b*x,y1~~y2),missingdata[[2]],missing=TRUE) ## MCAR
e3 <- estimate(lvm(c(y1,y2)~b*x,y1~~y2),missingdata[[3]]) ## CC (MAR)
e4 <- estimate(lvm(c(y1,y2)~b*x,y1~~y2),missingdata[[3]],missing=TRUE) ## MAR
```

Model *Extract model*

Description

Extract or replace model object

Usage

```
Model(x, ...)

Model(x, ...) <- value
```

Arguments

x	Fitted model
value	New model object (e.g. lvm or multigroup)
...	Additional arguments to be passed to the low level functions

Value

Returns a model object (e.g. lvm or multigroup)

Author(s)

Klaus K. Holst

See Also

Graph

Examples

```
m <- lvm(y~x)
e <- estimate(m, sim(m,100))
Model(e)
```

`modelsearch` *Model searching*

Description

Performs Wald or score tests

Usage

```
modelsearch(x, k = 1, dir = "forward", ...)
```

Arguments

`x`	`lvmfit`-object
`k`	Number of parameters to test simultaneously. For `equivalence` the number of additional associations to be added instead of `rel`.
`dir`	Direction to do model search. "forward" := add associations/arrows to model/graph (score tests), "backward" := remove associations/arrows from model/graph (wald test)
`...`	Additional arguments to be passed to the low level functions

Value

Matrix of test-statistics and p-values

Author(s)

Klaus K. Holst

See Also

`compare`, `equivalence`

Examples

```
m <- lvm();
regression(m) <- c(y1,y2,y3) ~ eta; latent(m) <- ~eta
regression(m) <- eta ~ x
m0 <- m; regression(m0) <- y2 ~ x
dd <- sim(m0,100)[,manifest(m0)]
e <- estimate(m,dd);
modelsearch(e)
```

`multinomial` *Estimate probabilities in contingency table*

Description

Estimate probabilities in contingency table

Usage

```
multinomial(x, marginal = FALSE, transform, ...)
```

Arguments

`x`	Matrix or data.frame with observations (1 or 2 columns)
`marginal`	If TRUE the marginals are estimated
`transform`	Optional transformation of parameters (e.g., logit)
`...`	Additional arguments to lower-level functions

Author(s)

Klaus K. Holst

Examples

```
set.seed(1)
breaks <- c(-Inf,-1,0,Inf)
m <- lvm(); covariance(m,pairwise=TRUE) <- ~y1+y2+y3+y4
d <- transform(sim(m,5e2),
               z1=cut(y1,breaks=breaks),
               z2=cut(y2,breaks=breaks),
               z3=cut(y3,breaks=breaks),
               z4=cut(y4,breaks=breaks))

multinomial(d[,5])
(a1 <- multinomial(d[,5:6]))
(K1 <- kappa(a1)) ## Cohen's kappa

K2 <- kappa(d[,7:8])
## Testing difference K1-K2:
estimate(merge(K1,K2,id=TRUE),diff)

estimate(merge(K1,K2,id=FALSE),diff) ## Wrong std.err ignoring dependence
sqrt(vcov(K1)+vcov(K2))
##'
## Average of the two kappas:
estimate(merge(K1,K2,id=TRUE),function(x) mean(x))
estimate(merge(K1,K2,id=FALSE),function(x) mean(x)) ## Independence
```

```
## Goodman-Kruskal's gamma
m2 <- lvm(); covariance(m2) <- y1~y2
breaks1 <- c(-Inf,-1,0,Inf)
breaks2 <- c(-Inf,0,Inf)
d2 <- transform(sim(m2,5e2),
              z1=cut(y1,breaks=breaks1),
              z2=cut(y2,breaks=breaks2))

(g1 <- gkgamma(d2[,3:4]))
## same as
## Not run:
gkgamma(table(d2[,3:4]))
gkgamma(multinomial(d2[,3:4]))

## End(Not run)
```

`nldata` *Example data (nonlinear model)*

Description

Example data (nonlinear model)

Format

data.frame

Source

Simulated

`nsem` *Example SEM data (nonlinear)*

Description

Simulated data

Format

data.frame

Source

Simulated

ordreg *Univariate cumulative link regression models*

Description

Ordinal regression models

Usage

```
ordreg(formula, data = parent.frame(), offset, family = binomial("probit"),
  start, fast = FALSE, ...)
```

Arguments

formula	formula
data	data.frame
offset	offset
family	family (default proportional odds)
start	optional starting values
fast	If TRUE standard errors etc. will not be calculated
...	Additional arguments to lower level functions

Author(s)

Klaus K. Holst

Examples

```
m <- lvm(y~x)
ordinal(m,K=3) <- ~y
d <- sim(m,100)
e <- ordreg(y~x,d)
```

org *Convert object to ascii suitable for org-mode*

Description

Convert object to ascii suitable for org-mode

Usage

```
org(x, ..., ncol, include.rownames = TRUE, include.colnames = TRUE,
  header = TRUE, frame = "topbot", rownames = NULL, colnames = NULL,
  type = "org", tab = FALSE, margins = TRUE, print = TRUE, html, latex)
```

Arguments

x	R object
...	additional arguments to lower level functions
ncol	If x is a vector and ncol is given as argument, the resulting output will be a matrix with ncol columns
include.rownames	If FALSE row names are removed
include.colnames	If FALSE column names are removed
header	If TRUE the header is included
frame	Frame argument (see ascii)
rownames	Optional vector of row names
colnames	Optional vector of column names
type	Type argument (see ascii)
tab	Tabulate?
margins	Add margins to table?
print	print or return result
html	HTML prefix (added to ATTR_HTML)
latex	LaTeX prefix (added to ATTR_LaTeX)

Author(s)

Klaus K. Holst

---

parpos *Generic method for finding indeces of model parameters*

Description

Generic method for finding indeces of model parameters

Usage

```
parpos(x, ...)
```

Arguments

x	Model object
...	Additional arguments

Author(s)

Klaus K. Holst

partialcor *Calculate partial correlations*

Description

Calculate partial correlation coefficients and confidence limits via Fishers z-transform

Usage

```
partialcor(formula, data, level = 0.95)
```

Arguments

formula	formula speciying the covariates and optionally the outcomes to calculate partial correlation for
data	data.frame
level	Level of confidence limits

Value

A coefficient matrix

Author(s)

Klaus K. Holst

Examples

```
m <- lvm(c(y1,y2,y3)~x1+x2)
covariance(m) <- c(y1,y2,y3)~y1+y2+y3
d <- sim(m,500)
partialcor(~x1+x2,d)
```

path *Extract pathways in model graph*

Description

Extract all possible paths from one variable to another connected component in a latent variable model. In an estimated model the effect size is decomposed into direct, indirect and total effects including approximate standard errors.

Usage

```
## S3 method for class 'lvm'
 path(object, to = NULL, from, ...)
## S3 method for class 'lvmfit'
 effects(object, to, from, silent=FALSE, ...)
```

Arguments

object	Model object (lvm)
to	Outcome variable (string). Alternatively a formula specifying response and predictor in which case the argument from is ignored.
from	Response variable (string), not necessarily directly affected by to.
silent	Logical variable which indicates whether messages are turned on/off.
...	Additional arguments to be passed to the low level functions

Value

If object is of class lvmfit a list with the following elements is returned

idx	A list where each element defines a possible pathway via a integer vector indicating the index of the visited nodes.
V	A List of covariance matrices for each path.
coef	A list of parameters estimates for each path
path	A list where each element defines a possible pathway via a character vector naming the visited nodes in order.
edges	Description of 'comp2'

If object is of class lvm only the path element will be returned.

The effects method returns an object of class effects.

Note

For a lvmfit-object the parameters estimates and their corresponding covariance matrix are also returned. The effects-function additionally calculates the total and indirect effects with approximate standard errors

Author(s)

Klaus K. Holst

See Also

children, parents

Examples

```
m <- lvm(c(y1,y2,y3)~eta)
regression(m) <- y2~x1
latent(m) <- ~eta
regression(m) <- eta~x1+x2
d <- sim(m,500)
e <- estimate(m,d)

path(Model(e),y2~x1)
parents(Model(e), ~y2)
children(Model(e), ~x2)
children(Model(e), ~x2+eta)
effects(e,y2~x1)
```

PD *Dose response calculation for binomial regression models*

Description

Dose response calculation for binomial regression models

Usage

```
PD(model, intercept = 1, slope = 2, prob = NULL, x, level = 0.5,
  ci.level = 0.95, vcov, family, EB = NULL)
```

Arguments

model	Model object or vector of parameter estimates
intercept	Index of intercept parameters
slope	Index of intercept parameters
prob	Index of mixture parameters (only relevant for curereg models)
x	Optional weights length(x)=length(intercept)+length(slope)+length(prob)
level	Probability at which level to calculate dose
ci.level	Level of confidence limits
vcov	Optional estimate of variance matrix of parameter estimates
family	Optional distributional family argument
EB	Optional ratio of treatment effect and adverse effects used to find optimal dose (regret-function argument)

Author(s)

Klaus K. Holst

pdfconvert *Convert pdf to raster format*

Description

Convert PDF file to print quality png (default 300 dpi)

Usage

```
pdfconvert(files, dpi = 300, resolution = 1024, gsopt, format = "png",
  ...)
```

Arguments

files	Vector of (pdf-)filenames to process
dpi	DPI
resolution	Resolution of raster image file
gsopt	Optional ghostscript arguments
format	Raster format (e.g. png, jpg, tif, ...)
...	Additional arguments

Details

Access to ghostscript program 'gs' is needed

Author(s)

Klaus K. Holst

See Also

dev.copy2pdf, printdev

plot.lvm *Plot path diagram*

Description

Plot the path diagram of a SEM

Usage

```
## S3 method for class 'lvm'
plot(x, diag = FALSE, cor = TRUE, labels = FALSE,
  intercept = FALSE, addcolor = TRUE, plain = FALSE, cex,
  fontsize1 = 10, noplot = FALSE, graph = list(rankdir = "BT"),
  attrs = list(graph = graph), unexpr = FALSE, addstyle = TRUE,
  Rgraphviz = lava.options()$Rgraphviz, init = TRUE, layout = c("dot",
  "fdp", "circo", "twopi", "neato", "osage"),
  edgecolor = lava.options()$edgecolor, ...)
```

Arguments

x	Model object
diag	Logical argument indicating whether to visualize variance parameters (i.e. diagonal of variance matrix)
cor	Logical argument indicating whether to visualize correlation parameters
labels	Logical argument indiciating whether to add labels to plot (Unnamed parameters will be labeled p1,p2,...)
intercept	Logical argument indiciating whether to add intercept labels (current version: not used))
addcolor	Logical argument indiciating whether to add colors to plot (overrides nodecolor calls)
plain	if TRUE strip plot of colors and boxes
cex	Fontsize of node labels
fontsize1	Fontsize of edge labels
noplot	if TRUE then return graphNEL object only
graph	Graph attributes (Rgraphviz)
attrs	Attributes (Rgraphviz)
unexpr	if TRUE remove expressions from labels
addstyle	Logical argument indicating whether additional style should automatically be added to the plot (e.g. dashed lines to double-headed arrows)
Rgraphviz	if FALSE igraph is used for graphics
init	Reinitialize graph (for internal use)
layout	Graph layout (see Rgraphviz or igraph manual)
edgecolor	if TRUE plot style with colored edges
...	Additional arguments to be passed to the low level functions

Author(s)

Klaus K. Holst

Examples

```
m <- lvm(c(y1,y2) ~ eta)
regression(m) <- eta ~ z+x2
regression(m) <- c(eta,z) ~ x1
latent(m) <- ~eta
labels(m) <- c(y1=expression(y[scriptscriptstyle(1)]),
y2=expression(y[scriptscriptstyle(2)]),
x1=expression(x[scriptscriptstyle(1)]),
x2=expression(x[scriptscriptstyle(2)]),
eta=expression(eta))
edgelabels(m, eta ~ z+x1+x2, cex=2, lwd=3,
           col=c("orange","lightblue","lightblue")) <- expression(rho,phi,psi)
nodecolor(m, vars(m), border="white", labcol="darkblue") <- NA
nodecolor(m, ~y1+y2+z, labcol=c("white","white","black")) <- NA
plot(m,cex=1.5)

d <- sim(m,100)
e <- estimate(m,d)
plot(e)



m <- lvm(c(y1,y2) ~ eta)
regression(m) <- eta ~ z+x2
regression(m) <- c(eta,z) ~ x1
latent(m) <- ~eta
plot(lava:::beautify(m,edgecol=F))
```

---

plotConf *Plot regression lines*

Description

Plot regression line (with interactions) and partial residuals.

Usage

```
plotConf(model, var1 = all.vars(formula(model))[2], var2 = NULL,
  data = NULL, ci.lty = 0, ci = TRUE, level = 0.95, pch = 16,
  lty = 1, lwd = 2, npoints = 100, xlim, col = NULL, colpt,
  alpha = 0.5, cex = 1, delta = 0.07, centermark = 0.03, jitter = 0.2,
  cidiff = FALSE, mean = TRUE, legend = ifelse(is.null(var1), FALSE,
  "topright"), trans = function(x) {     x }, partres = inherits(model,
  "lm"), partse = FALSE, labels, vcov, predictfun, ...)
```

Arguments

model	Model object (e.g. lm)
var1	predictor (Continuous or factor)
var2	Factor that interacts with var1
data	data.frame to use for prediction (model.frame is used as default)
ci.lty	Line type for confidence limits
ci	Boolean indicating wether to draw pointwise 95% confidence limits
level	Level of confidence limits (default 95%)
pch	Point type for partial residuals
lty	Line type for estimated regression lines
lwd	Line width for regression lines
npoints	Number of points used to plot curves
xlim	Range of x axis
col	Color (for each level in var2)
colpt	Color of partial residual points
alpha	Alpha level
cex	Point size
delta	For categorical var1
centermark	For categorical var1
jitter	For categorical var1
cidiff	For categorical var1
mean	For categorical var1
legend	Boolean (add legend)
trans	Transform estimates (e.g. exponential)
partres	Boolean indicating whether to plot partial residuals
partse	.
labels	Optional labels of var2
vcov	Optional variance estimates
predictfun	Optional predict-function used to calculate confidence limits and predictions
...	additional arguments to lower level functions

Value

list with following members:

x	Variable on the x-axis (var1)
y	Variable on the y-axis (partial residuals)
predict	Matrix with confidence limits and predicted values

Author(s)

Klaus K. Holst

See Also

termplot

Examples

```
n <- 100
x0 <- rnorm(n)
x1 <- seq(-3,3, length.out=n)
x2 <- factor(rep(c(1,2),each=n/2), labels=c("A","B"))
y <- 5 + 2*x0 + 0.5*x1 + -1*(x2=="B")*x1 + 0.5*(x2=="B") + rnorm(n, sd=0.25)
dd <- data.frame(y=y, x1=x1, x2=x2)
lm0 <- lm(y ~ x0 + x1*x2, dd)
plotConf(lm0, var1="x1", var2="x2")
### points(5+0.5*x1 -1*(x2=="B")*x1 + 0.5*(x2=="B") ~ x1, cex=2)

data(iris)
l <- lm(Sepal.Length ~ Sepal.Width*Species,iris)
plotConf(l,var2="Species")
```

---

predict.lvm *Prediction in structural equation models*

Description

Prediction in structural equation models

Usage

```
## S3 method for class 'lvm'
predict(object, x = NULL, residual = FALSE, p, data,
  path = FALSE, quick = is.null(x) & !(residual | path), ...)
```

Arguments

object	Model object
x	optional list of (endogenous) variables to condition on
residual	If true the residuals are predicted
p	Parameter vector
data	Data to use in prediction
path	Path prediction
quick	If TRUE the conditional mean and variance given covariates are returned (and all other calculations skipped)
...	Additional arguments to lower level function

Examples

```
m <- lvm(list(c(y1,y2,y3)~u,u~x)); latent(m) <- ~u
d <- sim(m,100)
e <- estimate(m,d)

## Conditional mean (and variance as attribute) given covariates
r <- predict(e)
## Best linear unbiased predictor (BLUP)
r <- predict(e,vars(e))
##  Conditional mean of y3 giving covariates and y1,y2
r <- predict(e,y3~y1+y2)
##  Conditional mean  gives covariates and y1
r <- predict(e,~y1+y2)
##  Predicted residuals (conditional on all observed variables)
r <- predict(e,vars(e),residual=TRUE)
```

Range.lvm *Define range constraints of parameters*

Description

Define range constraints of parameters

Usage

```
Range.lvm(a = 0, b = 1)
```

Arguments

a	Lower bound
b	Upper bound

Value

function

Author(s)

Klaus K. Holst

regression<- *Add regression association to latent variable model*

Description

Define regression association between variables in a lvm-object and define linear constraints between model equations.

Usage

```
## S3 method for class 'lvm'
regression(object = lvm(), to, from, fn = NA,
silent = lava.options()$silent, additive=TRUE, ...)
## S3 replacement method for class 'lvm'
regression(object, to=NULL, quick=FALSE, ...) <- value
```

Arguments

object	lvm-object.
value	A formula specifying the linear constraints or if to=NULL a list of parameter values.
to	Character vector of outcome(s) or formula object.
from	Character vector of predictor(s).
fn	Real function defining the functional form of predictors (for simulation only).
silent	Logical variable which indicates whether messages are turned on/off.
additive	If FALSE and predictor is categorical a non-additive effect is assumed
quick	Faster implementation without parameter constraints
...	Additional arguments to be passed to the low level functions

Details

The regression function is used to specify linear associations between variables of a latent variable model, and offers formula syntax resembling the model specification of e.g. lm.

For instance, to add the following linear regression model, to the lvm-object, m:

$$E(Y|X_1, X_2) = \beta_1 X_1 + \beta_2 X_2$$

We can write

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

Multivariate models can be specified by successive calls with regression, but multivariate formulas are also supported, e.g.

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

defines

$$E(Y_i|X_1, X_2) = \beta_{1i} X_1 + \beta_{2i} X_2$$

The special function, f, can be used in the model specification to specify linear constraints. E.g. to fix $\beta_1 = \beta_2$, we could write

```
regression(m) <- y ~ f(x1,beta) + f(x2,beta)
```

The second argument of f can also be a number (e.g. defining an offset) or be set to NA in order to clear any previously defined linear constraints.

Alternatively, a more straight forward notation can be used:

```
regression(m) <- y ~ beta*x1 + beta*x2
```

All the parameter values of the linear constraints can be given as the right handside expression of the assigment function regression<- (or regfix<-) if the first (and possibly second) argument is defined as well. E.g:

```
regression(m,y1~x1+x2) <- list("a1","b1")
```

defines $E(Y_1|X_1, X_2) = a1X_1 + b1X_2$. The rhs argument can be a mixture of character and numeric values (and NA's).

The function regression (called without additional arguments) can be used to inspect the linear constraints of a lvm-object.

For backward compatibility the "$"-symbol can be used to fix parameters at a given value. E.g. to add a linear relationship between y and x with slope 2 to the model m, we can write `regression(m,"y") <- "x$2"`. Similarily we can use the "@"-symbol to name parameters. E.g. in a multiple regression we can force the parameters to be equal: `regression(m,"y") <- c("x1@b","x2@b")`. Fixed parameters can be reset by fixing (with \$) them to NA.

Value

A lvm-object

Note

Variables will be added to the model if not already present.

Author(s)

Klaus K. Holst

See Also

intercept<-, covariance<-, constrain<-, parameter<-, latent<-, cancel<-, kill<-

Examples

```
m <- lvm() ## Initialize empty lvm-object
### E(y1|z,v) = beta1*z + beta2*v
regression(m) <- y1 ~ z + v
### E(y2|x,z,v) = beta*x + beta*z + 2*v + beta3*u
regression(m) <- y2 ~ f(x,beta) + f(z,beta)  + f(v,2) + u
### Clear restriction on association between y and
### fix slope coefficient of u to beta
regression(m, y2 ~ v+u) <- list(NA,"beta")
```

```
regression(m) ## Examine current linear parameter constraints

## ## A multivariate model, E(yi|x1,x2) = beta[1i]*x1 + beta[2i]*x2:
m2 <- lvm(c(y1,y2) ~ x1+x2)
```

`revdiag` *Create/extract 'reverse'-diagonal matrix*

Description

Create/extract 'reverse'-diagonal matrix

Usage

```
revdiag(x,...)

revdiag(x,...) <- value
```

Arguments

`x`	vector
`value`	For the assignment function the values to put in the diagonal
`...`	additional arguments to lower level functions

Author(s)

Klaus K. Holst

`scheffe` *Calculate simultaneous confidence limits by Scheffe's method*

Description

Function to compute the Scheffe corrected confidence interval for the regression line

Usage

```
scheffe(model, newdata = model.frame(model), conf.level = 0.95)
```

Arguments

`model`	Linear model
`newdata`	new data frame
`conf.level`	confidence level (0.95)

Examples

```
x <- rnorm(100)
d <- data.frame(y=rnorm(length(x),x),x=x)
l <- lm(y~x,d)
plot(y~x,d)
abline(l)
d0 <- data.frame(x=seq(-5,5,length.out=100))
d1 <- cbind(d0,predict(l,newdata=d0,interval="confidence"))
d2 <- cbind(d0,scheffe(l,d0))
lines(lwr~x,d1,lty=2,col="red")
lines(upr~x,d1,lty=2,col="red")
lines(lwr~x,d2,lty=2,col="blue")
lines(upr~x,d2,lty=2,col="blue")
```

---

semdata *Example SEM data*

Description

Simulated data

Format

data.frame

Source

Simulated

---

serotonin *Serotonin data*

Description

This simulated data mimics a PET imaging study where the 5-HT2A receptor and serotonin transporter (SERT) binding potential has been quantified into 8 different regions. The 5-HT2A cortical regions are considered high-binding regions measurements. These measurements can be regarded as proxy measures of the extra-cellular levels of serotonin in the brain

day	numeric	Scan day of the year
age	numeric	Age at baseline scan
mem	numeric	Memory performance score
depr	numeric	Depression (mild) status 500 days after baseline
gene1	numeric	Gene marker 1 (HTR2A)
gene2	numeric	Gene marker 2 (HTTTLPR)
cau	numeric	SERT binding, Caudate Nucleus

th	numeric	SERT binding, Thalamus
put	numeric	SERT binding, Putamen
mid	numeric	SERT binding, Midbrain
aci	numeric	5-HT2A binding, Anterior cingulate gyrus
pci	numeric	5-HT2A binding, Posterior cingulate gyrus
sfc	numeric	5-HT2A binding, Superior frontal cortex
par	numeric	5-HT2A binding, Parietal cortex

Format

data.frame

Source

Simulated

`serotonin2` *Data*

Description

Description

Format

data.frame

Source

Simulated

See Also

serotonin

sim *Simulate model*

Description

Simulate data from a general SEM model including non-linear effects and general link and distribution of variables.

Usage

```
## S3 method for class 'lvm'
sim(x, n = 100, p = NULL, normal = FALSE, cond = FALSE,
sigma = 1, rho = 0.5, X, unlink=FALSE, ...)
```

Arguments

x	Model object
n	Number of simulated values/individuals
p	Parameter value (optional)
normal	Logical indicating whether to simulate data from a multivariate normal distribution conditional on exogenous variables hence ignoring functional/distribution definition
cond	for internal use
sigma	Default residual variance (1)
rho	Default covariance parameter (0.5)
X	Optional matrix of covariates
unlink	Return Inverse link transformed data
...	Additional arguments to be passed to the low level functions

Author(s)

Klaus K. Holst

Examples

```
##################################################
## Logistic regression
##################################################
m <- lvm(y~x+z)
regression(m) <- x~z
distribution(m,~y+z) <- binomial.lvm("logit")
d <- sim(m,1e3)
head(d)

e <- estimate(m,d,estimator="glm")
e
```

```
## Simulate a few observation from estimated model
sim(e,n=5)

##################################################
## Poisson
##################################################
distribution(m,~y) <- poisson.lvm()
d <- sim(m,1e4,p=c(y=-1,"y~x"=2,z=1))
head(d)
estimate(m,d,estimator="glm")
mean(d$z); lava:::expit(1)

summary(lm(y~x,sim(lvm(y[1:2]~4*x),1e3)))

##################################################
### Gamma distribution
##################################################
m <- lvm(y~x)
distribution(m,~y+x) <- list(Gamma.lvm(shape=2),binomial.lvm())
intercept(m,~y) <- 0.5
d <- sim(m,1e4)
summary(g <- glm(y~x,family=Gamma(),data=d))
## Not run: MASS::gamma.shape(g)

args(lava::Gamma.lvm)
distribution(m,~y) <- Gamma.lvm(shape=2,log=TRUE)
sim(m,10,p=c(y=0.5))[,"y"]

##################################################
### Transform
##################################################

m <- lvm()
transform(m,xz~x+z) <- function(x) x[1]*(x[2]>0)
regression(m) <- y~x+z+xz
d <- sim(m,1e3)
summary(lm(y~x+z + x*I(z>0),d))


##################################################
### Non-random variables
##################################################
m <- lvm()
distribution(m,~x+z+v+w) <- list(sequence.lvm(0,5),## Seq. 0 to 5 by 1/n
                                 ones.lvm(),       ## Vector of ones
                                 ones.lvm(0.5),    ##  0.8n 0, 0.2n 1
                                 ones.lvm(interval=list(c(0.3,0.5),c(0.8,1))))
sim(m,10)


##################################################
### Cox model
### piecewise constant hazard
```

```
##################################################

m <- lvm(t~x)
rates <- c(1,0.5); cuts <- c(0,5)
## Constant rate: 1 in [0,5), 0.5 in [5,Inf)
distribution(m,~t) <- coxExponential.lvm(rate=rates,timecut=cuts)


## Not run:
    d <- sim(m,2e4,p=c("t~x"=0.1)); d$status <- TRUE
    plot(timereg::aalen(survival::Surv(t,status)~x,data=d,
                        resample.iid=0,robust=0),spec=1)
    L <- approxfun(c(cuts,max(d$t)),f=1,
                   cumsum(c(0,rates*diff(c(cuts,max(d$t))))),
                   method="linear")
    curve(L,0,100,add=TRUE,col="blue")

## End(Not run)


##################################################
### Cox model
### piecewise constant hazard, gamma frailty
##################################################

m <- lvm(y~x+z)
rates <- c(0.3,0.5); cuts <- c(0,5)
distribution(m,~y+z) <- list(coxExponential.lvm(rate=rates,timecut=cuts),
                             loggamma.lvm(rate=1,shape=1))
## Not run:
    d <- sim(m,2e4,p=c("y~x"=0,"y~z"=0)); d$status <- TRUE
    plot(timereg::aalen(survival::Surv(y,status)~x,data=d,
                        resample.iid=0,robust=0),spec=1)
    L <- approxfun(c(cuts,max(d$y)),f=1,
                   cumsum(c(0,rates*diff(c(cuts,max(d$y))))),
                   method="linear")
    curve(L,0,100,add=TRUE,col="blue")

## End(Not run)

## Equivalent via transform (here with Aalens additive hazard model)
m <- lvm(y~x)
distribution(m,~y) <- aalenExponential.lvm(rate=rates,timecut=cuts)
distribution(m,~z) <- Gamma.lvm(rate=1,shape=1)
transform(m,t~y+z) <- prod
sim(m,10)

## Shared frailty
m <- lvm(c(t1,t2)~x+z)
rates <- c(1,0.5); cuts <- c(0,5)
distribution(m,~y) <- aalenExponential.lvm(rate=rates,timecut=cuts)
distribution(m,~z) <- loggamma.lvm(rate=1,shape=1)
## Not run:
```

```
    mets::fast.reshape(sim(m,100),varying="t")

## End(Not run)
##'

################################################
### General multivariate distributions
################################################

## Not run:

m <- lvm()
distribution(m,~y1+y2,oratio=4) <- VGAM::rplack
ksmooth2(sim(m,1e4),rgl=FALSE,theta=-20,phi=25)

m <- lvm()
distribution(m,~z1+z2,"or1") <- VGAM::rplack
distribution(m,~y1+y2,"or2") <- VGAM::rplack
sim(m,10,p=c(or1=0.1,or2=4))

m <- lvm()
distribution(m,~y1+y2+y3,TRUE) <- function(n,...) rmvn(n,sigma=diag(3)+1)
var(sim(m,100))


## End(Not run)

################################################
### Categorical predictor
################################################

##library(mets)
m <- lvm()
## categorical(m,K=3) <- "v"
categorical(m,labels=c("A","B","C")) <- "v"

regression(m,additive=FALSE) <- y~v
## Not run:
  plot(y~v,sim(m,1000,p=c("y~v:2"=3)))

## End(Not run)

m <- lvm()
categorical(m,labels=c("A","B","C"),p=c(0.5,0.3)) <- "v"
regression(m,additive=FALSE,beta=c(0,2,-1)) <- y~v
## ## equivalent to:
## regression(m,y~v,additive=FALSE) <- c(0,2,-1)
regression(m,additive=FALSE,beta=c(0,4,-1)) <- z~v
table(sim(m,1e4)$v)
glm(y~v, data=sim(m,1e4))
glm(y~v, data=sim(m,1e4,p=c("y~v:1"=3)))
```

startvalues *For internal use*

Description

For internal use

Author(s)

Klaus K. Holst

subset.lvm *Extract subset of latent variable model*

Description

Extract measurement models or user-specified subset of model

Usage

```
## S3 method for class 'lvm'
subset(x, vars, ...)
```

Arguments

x	lvm-object.
vars	Character vector or formula specifying variables to include in subset.
...	Additional arguments to be passed to the low level functions

Value

A lvm-object.

Author(s)

Klaus K. Holst

Examples

```
m <- lvm(c(y1,y2)~x1+x2)
subset(m,~y1+x1)
```

`timedep` *Time-dependent parameters*

Description

Add time-varying covariate effects to model

Usage

```
timedep(object, formula, rate, timecut, type = "coxExponential.lvm", ...)
```

Arguments

`object`	Model
`formula`	Formula with rhs specifying time-varying covariates
`rate`	Optional rate parameters. If given as a vector this parameter is interpreted as the raw (baseline-)rates within each time interval defined by `timecut`. If given as a matrix the parameters are interpreted as log-rates (and log-rate-ratios for the time-varying covariates defined in the formula).
`timecut`	Time intervals
`type`	Type of model (default piecewise constant intensity)
`...`	Additional arguments to lower level functions

Author(s)

Klaus K. Holst

Examples

```
## Piecewise constant hazard
m <- lvm(y~1)
m <- timedep(m,y~1,timecut=c(0,5),rate=c(0.5,0.3))

## Not run:
d <- sim(m,1e4); d$status <- TRUE
dd <- mets::lifetable(Surv(y,status)~1,data=d,breaks=c(5));
exp(coef(glm(events ~ offset(log(atrisk)) + -1 + interval, dd, family=poisson)))

## End(Not run)


## Piecewise constant hazard and time-varying effect of z1
m <- lvm(y~1)
distribution(m,~z1) <- ones.lvm(0.5)
R <- log(cbind(c(0.2,0.7,0.9),c(0.5,0.3,0.3)))
m <- timedep(m,y~z1,timecut=c(0,3,5),rate=R)
```

```
## Not run:
d <- sim(m,1e4); d$status <- TRUE
dd <- mets::lifetable(Surv(y,status)~z1,data=d,breaks=c(3,5));
exp(coef(glm(events ~ offset(log(atrisk)) + -1 + interval+z1:interval, dd, family=poisson)))

## End(Not run)



## Explicit simulation of time-varying effects
m <- lvm(y~1)
distribution(m,~z1) <- ones.lvm(0.5)
distribution(m,~z2) <- binomial.lvm(p=0.5)
#variance(m,~m1+m2) <- 0
#regression(m,m1[m1:0] ~ z1) <- log(0.5)
#regression(m,m2[m2:0] ~ z1) <- log(0.3)
regression(m,m1 ~ z1,variance=0) <- log(0.5)
regression(m,m2 ~ z1,variance=0) <- log(0.3)
intercept(m,~m1+m2) <- c(-0.5,0)
m <- timedep(m,y~m1+m2,timecut=c(0,5))

## Not run:
d <- sim(m,1e5); d$status <- TRUE
dd <- mets::lifetable(Surv(y,status)~z1,data=d,breaks=c(5))
exp(coef(glm(events ~ offset(log(atrisk)) + -1 + interval + interval:z1, dd, family=poisson)))

## End(Not run)
```

`toformula` *Converts strings to formula*

Description

Converts a vector of predictors and a vector of responses (characters) i#nto a formula expression.

Usage

```
toformula(y = ".", x = ".")
```

Arguments

y	vector of predictors
x	vector of responses

Value

An object of class `formula`

Author(s)

Klaus K. Holst

See Also

as.formula,

Examples

```
toformula(c("age","gender"), "weight")
```

---

tr *Trace operator*

Description

Calculates the trace of a square matrix.

Usage

```
tr(A)
```

Arguments

A	Square numeric matrix

Value

numeric

Author(s)

Klaus K. Holst

See Also

crossprod, tcrossprod

Examples

```
tr(diag(1:5))
```

trim *Trim tring of (leading/trailing/all) white spaces*

Description

Trim tring of (leading/trailing/all) white spaces

Usage

```
trim(x, all = FALSE, ...)
```

Arguments

x	String
all	Trim all whitespaces?
...	additional arguments to lower level functions

Author(s)

Klaus K. Holst

twindata *Twin menarche data*

Description

Simulated data

id	numeric	Twin-pair id
zyg	character	Zygosity (MZ or DZ)
twinnum	numeric	Twin number (1 or 2)
agemena	numeric	Age at menarche (or censoring)
status	logical	Censoring status (observed:=T,censored:=F)
bw	numeric	Birth weight
msmoke	numeric	Did mother smoke? (yes:=1,no:=0)

Format

data.frame

Source

Simulated

vars *Extract variable names from latent variable model*

Description

Extract exogenous variables (predictors), endogenous variables (outcomes), latent variables (random effects), manifest (observed) variables from a lvm object.

Usage

```
vars(x,...)

endogenous(x,...)

exogenous(x,...)

manifest(x,...)

latent(x,...)

## S3 replacement method for class 'lvm'
exogenous(x,silent = FALSE, xfree = TRUE,...) <- value

## S3 method for class 'lvm'
exogenous(x,latent=FALSE,index=TRUE,...)

## S3 replacement method for class 'lvm'
latent(x,clear=FALSE,...) <- value
```

Arguments

x	lvm-object
latent	Logical defining whether latent variables without parents should be included in the result
index	For internal use only
clear	Logical indicating whether to add or remove latent variable status
silent	Suppress messages
xfree	For internal use only
value	Formula or character vector of variable names.
...	Additional arguments to be passed to the low level functions

Details

vars returns all variables of the lvm-object including manifest and latent variables. Similarily manifest and latent returns the observered resp. latent variables of the model. exogenous returns all manifest variables without parents, e.g. covariates in the model, however the argument latent=TRUE can be used to also include latent variables without parents in the result. Pr. default lava will not include the parameters of the exogenous variables in the optimisation routine during estimation (likelihood of the remaining observered variables conditional on the covariates), however this behaviour can be altered via the assignment function exogenous<- telling lava which subset of (valid) variables to condition on. Finally latent returns a vector with the names of the latent variables in x. The assigment function latent<- can be used to change the latent status of variables in the model.

Value

Vector of variable names.

Author(s)

Klaus K. Holst

See Also

endogenous, manifest, latent, exogenous, vars

Examples

```
g <- lvm(eta1 ~ x1+x2)
regression(g) <- c(y1,y2,y3) ~ eta1
latent(g) <- ~eta1
endogenous(g)
exogenous(g)
identical(latent(g), setdiff(vars(g),manifest(g)))
```

%+% *Concatenation operator*

Description

For matrices a block-diagonal matrix is created. For all other data types he operator is a wrapper of paste.

Usage

```
x %+% y
```

Arguments

x	First object
y	Second object of same class

Details

Concatenation operator

Author(s)

Klaus K. Holst

See Also

`blockdiag`, `paste`, `cat`,

Examples

```
matrix(rnorm(25),5)%+%matrix(rnorm(25),5)
"Hello "%+%" World"
```

%ni% *Matching operator (x not in y) oposed to the* %in%*-operator (x in y)*

Description

Matching operator

Usage

```
x %ni% y
```

Arguments

x	vector
y	vector of same type as x

Value

A logical vector.

Author(s)

Klaus K. Holst

See Also

`match`

Examples

```
1:10 %ni% c(1,5,10)
```

Index