Package ‘simex’

February 20, 2015

Type Package
Version 1.5
Date 2013-03-31
Depends stats
Suggests mgcv, nlme
Title SIMEX- and MCSIMEX-Algorithm for measurement error models
Author Wolfgang Lederer <Wolfgang.Lederer@gmail.com>, Helmut Küchenhoff <Kuechenhoff@stat.uni-muenchen.de>
Maintainer Wolfgang Lederer <Wolfgang.Lederer@gmail.com>
Description Implementation of the SIMEX-Algorithm by Cook & Stefanski and MCSIMEX by Küchenhoff, Mwalili & Lesaffre
License GPL (>= 2)
Encoding latin1
LazyLoad yes
ZipData yes
NeedsCompilation no
Repository CRAN
Date/Publication 2013-05-06 07:30:27

R topics documented:

simex-package .................................................. 2
diag.block ..................................................... 4
mc.matrix ....................................................... 5
mcsimex .......................................................... 7
misclass .......................................................... 10
simex ............................................................. 12

Index 17
Description

Package simex is an implementation of the SIMEX–algorithm by Cook and Stephanski and the MCSIMEX–Algorithm by Küchenhof, Mwalili and Lesaffre.

Details

Package: simex
Type: Package
Version: 1.5
Date: 2013-03-31
License: GPL 2 or above
LazyLoad: yes

The package includes first of all the implementation for the SIMEX– and MCSIMEX–Algorithms. Jackknife and asymptotic variance estimation are implemented. Various methods and analytic tools are provided for a simple and fast access to the SIMEX– and MCSIMEX–Algorithm.

Functions simex() and mcsimex() can be used on models issued from \texttt{lm()}, \texttt{glm()} with asymptotic estimation. Models from \texttt{nls()}, \texttt{gam()} (package \texttt{mgcv}), \texttt{lme()} and \texttt{nlme()} (package \texttt{nlme}) can also be corrected with these algorithms, but without asymptotic estimations.

Author(s)

Wolfgang Lederer
Maintainer: Wolfgang Lederer,<wolfgang.lederer@gmail.com>

References


See Also

\texttt{simex}, \texttt{mcsimex}, \texttt{misclass}

and for functions generating the initial naive models: \texttt{lm}, \texttt{glm}, \texttt{nls}, \texttt{gam}, \texttt{lme}, \texttt{nlme}

Examples

# See example(simex) and example(mcsimex)
## Seed
set.seed(49494)
## simulating the measurement error standard deviations
sd_me1 <- 0.3
sd_me2 <- 0.4
temp <- runif(100, min = 0, max = 0.6)
sd_me_het1 <- sort(temp)
temp2 <- rnorm(100, sd = 0.1)
sd_me_het2 <- abs(sd_me_het1 + temp2)

## simulating the independent variables x (real and with measurement error):
x_real1 <- rnorm(100)
x_real2 <- rpois(100, lambda = 2)
x_real3 <- -4*x_real1 + runif(100, min = -2, max = 2)  # correlated to x_real

x_measured1 <- x_real1 + sd_me1 * rnorm(100)
x_measured2 <- x_real2 + sd_me2 * rnorm(100)
x_het1 <- x_real1 + sd_me_het1 * rnorm(100)
x_het2 <- x_real3 + sd_me_het2 * rnorm(100)

## calculating dependent variable y:
y1 <- x_real1 + rnorm(100, sd = 0.05)
y2 <- x_real1 + 2*x_real2 + rnorm(100, sd = 0.08)
y3 <- x_real1 + 2*x_real3 + rnorm(100, sd = 0.08)

### one variable with homoscedastic measurement error
(model_real <- lm(y1 ~ x_real1))
(model_naiv <- lm(y1 ~ x_measured1, x = TRUE))

(model_simex1 <- simex(model_naiv, SIMEXvariable = "x_measured1", measurement.error = sd_me1))
plot(model_simex1)

### two variables with homoscedastic measurement errors
(model_real2 <- lm(y2 ~ x_real1 + x_real2))
(model_naiv2 <- lm(y2 ~ x_measured1 + x_measured2, x = TRUE))

(model_simex2 <- simex(model_naiv2, SIMEXvariable = c("x_measured1", "x_measured2"),
measurement.error = cbind(sd_me1, sd_me2)))
plot(model_simex2)

### one variable with increasing heteroscedastic measurement error
(model_real1 <- lm(y1 ~ x_het1, x = TRUE))
(model_simex1 <- simex(model_naiv1, SIMEXvariable = "x_het1", measurement.error = sd_me_het1, asymptotic = FALSE))
plot(model_simex1)
## Not run:
### two correlated variables with heteroscedastic measurement errors
```r
(mod_real3 <- lm(y3 ~ x_real1 + x_real3))
```
```r
(mod_simex2 <- simex(mod_naiv2, SIMEXvariable = c("x_het1", "x_het2"),
measurement.error = cbind(sd_me_het1, sd_me_het2), asymptotic = FALSE))
plot(mod_simex2)
```
### two variables, one with homoscedastic, one with heteroscedastic measurement error
```r
(mod_real3 <- lm(y2 ~ x_measured1 + x_het2, x = TRUE))
```
```r
(mod_simex3 <- simex(mod_naiv3, SIMEXvariable = c("x_measured1", "x_het2"),
measurement.error = cbind(sd_me1, sd_me_het2), asymptotic = FALSE))
```
### glm: two variables, one with homoscedastic, one with heteroscedastic measurement error
```r
t <- x_real1 + 2*x_real2
g <- 1 / (1 + exp(-t))
u <- runif(100)
ybin <- as.numeric(u < g)
```
```r
(logit_real <- glm(ybin ~ x_real1 + x_real2, family = binomial))
```
```r
(logit_naiv <- glm(ybin ~ x_measured1 + x_het2, x = TRUE, family = binomial))
```
```r
(logit_simex <- simex(logit_naiv, SIMEXvariable = c("x_measured1", "x_het2"),
measurement.error = cbind(sd_me1, sd_me_het2), asymptotic = FALSE))
summary(logit_simex)
print(logit_simex)
plot(logit_simex)
```
## End(Not run)

---

### diag.block

*Constructs a block diagonal matrix*

**Description**

The function takes a list and constructs a block diagonal matrix with the elements of the list on the diagonal. If `d` is not a list then `d` will be repeated `n` times and written on the diagonal (a wrapper for `kronecker()`)

**Usage**

```r
diag.block(d, n)
```
mc.matrix

Arguments

- **d**: a list of matrices or vectors, or a matrix or vector
- **n**: number of repetitions

Value

returns a matrix with the elements of the list or the repetitions of the supplied matrix or vector on the diagonal.

Author(s)

Wolfgang Lederer, <wolfgang.lederer@gmail.com>

See Also

diag, kronecker

Examples

```r
a <- matrix(rep(1, 4), nrow = 2)
b <- matrix(rep(2, 6), nrow = 2)
e <- c(3, 3, 3, 3)
f <- t(e)
d <- list(a, b, e, f)
diag.block(d)
diag.block(a, 3)
```

Description

Empirical misclassification matrices to the power of lambda may not exist for small values of lambda. These functions provide methods to estimate the nearest version of the misclassification matrix that satisfies the conditions a misclassification matrix has to fulfill, and to check it (existence for exponents smaller than 1).

Usage

```r
build.mc.matrix(mc.matrix, method = "series", tuning = sqrt(.Machine$double.eps),
                 diag.cor = FALSE, tol = .Machine$double.eps, max.iter = 100)
check.mc.matrix(mc.matrix, tol = .Machine$double.eps)
```
Arguments

- `mc.matrix`: an empirical misclassification matrix
- `method`: method used to estimate the generator for the misclassification matrix (see Details)
- `tuning`: security parameter for numerical reasons
- `diag.cor`: should corrections be subtracted from the diagonal or from all values corresponding to the size?
- `tol`: tolerance level for series method for convergence
- `max.iter`: maximal number of iterations for the serie method to converge

Details

Method "series" constructs a generator via the series

\[(Pi - I) - (Pi - I)^2/2 + (Pi - I)^3/3 - \ldots\]

Method "log" constructs the generator via taking the log of the misclassification matrix.

Small negative off-diagonal values are corrected and set to \((0 + \text{tuning})\).

The amount used to correct for negative values is added to the diagonal element if \(\text{diag.cor} = \text{TRUE}\) and distributed among all values if \(\text{diag.cor} = \text{FALSE}\).

Method "jlt" uses the method described by Jarro et al. (see Israel et al.).

Value

- `build.mc.matrix()` returns a misclassification matrix that is the closest estimate for a working misclassification matrix.
- `check.mc.matrix()` returns a vector of logicals.

Note

Does not always work! So check properly.

Author(s)

Wolfgang Lederer, <wolfgang.lederer@gmail.com>

References


See Also

`mcsimex`, `misclass`, `diag.block`
Examples

\begin{verbatim}
Pi <- matrix(data = c(0.989, 0.01, 0.001, 0.17, 0.829, 0.001, 0.001, 0.18, 0.819),
nrow = 3, byrow = FALSE)
check.mc.matrix(list(Pi))
check.mc.matrix(list(build.mc.matrix(Pi)))
build.mc.matrix(Pi)

Pi3 <- matrix(c(0.8, 0.2, 0, 0, 0, 0.8, 0.1, 0.1, 0, 0.1, 0.8, 0.1, 0, 0, 0.3, 0.7),
nrow = 4)
check.mc.matrix(list(Pi3))
build.mc.matrix(Pi3)
check.mc.matrix(list(build.mc.matrix(Pi3)))

P1 <- matrix(c(1, 0, 0, 1), nrow = 2)
P2 <- matrix(c(0.8, 0.15, 0, 0.2, 0.7, 0.2, 0, 0.15, 0.8), nrow = 3, byrow = TRUE)
P3 <- matrix(c(0.4, 0.6, 0.6, 0.4), nrow = 2)
mc.matrix <- list(P1, P2, P3)
check.mc.matrix(mc.matrix) # TRUE FALSE FALSE
\end{verbatim}

---

mcsimex

Misclassification in models using MCSIMEX

Description

Implementation of the misclassification MCSIMEX algorithm as described by Küchenhoff, Mwalili and Lesaffre.

Usage

\begin{verbatim}
mcsimex(model, SIMEXvariable, mc.matrix, lambda = c(0.5, 1, 1.5, 2),
   B = 100, fitting.method = "quadratic", jackknife.estimation = "quadratic",
   asymptotic = TRUE)

## S3 method for class 'mcsimex'
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'mcsimex'
summary(object, ...)
## S3 method for class 'mcsimex'
plot(x, xlab = expression((1 + lambda)), ylab = colnames(b[, -1]),
   ask = FALSE, show = rep(TRUE, NCOL(b) - 1), ...)
## S3 method for class 'mcsimex'
predict(object, newdata, ...)
## S3 method for class 'mcsimex'
refit(object, fitting.method = "quadratic",
   jackknife.estimation = "quadratic", asymptotic = TRUE, ...)
\end{verbatim}
Arguments

- **model**: the naive model, the misclassified variable must be a factor
- **SIMEXvariable**: vector of names of the variables for which the MCSIMEX-method should be applied
- **mc.matrix**: if one variable is misclassified it can be a matrix. If more than one variable is misclassified it must be a list of the misclassification matrices, names must match with the SIMEXvariable names, column- and row-names must match with the factor levels. If a special misclassification is desired, the name of a function can be specified (see details)
- **lambda**: vector of exponents for the misclassification matrix (without 0)
- **B**: number of iterations for each lambda
- **fitting.method**: linear, quadratic and loglinear are implemented (first 4 letters are enough)
- **jackknife.estimate**: specifying the extrapolation method for jackknife variance estimation. Can be set to FALSE if it should not be performed
- **asymptotic**: logical, indicating if asymptotic variance estimation should be done, the option x = TRUE must be enabled in the naive model
- **x**: object of class 'mcsimex'
- **digits**: number of digits to be printed
- **object**: object of class 'mcsimex'
- **xlab**: optional name for the X-Axis
- **ylab**: vector containing the names for the Y-Axis
- **ask**: logical. If TRUE, the user is asked for input, before a new figure is drawn
- **show**: vector of logicals indicating for which variables a plot should be produced
- **newdata**: optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used
- **...**: arguments passed to other functions

Details

- if **mc.matrix** is a function the first argument of that function must be the whole dataset used in the naive model, the second argument must be the exponent (lambda) for the misclassification. The function must return a data.frame containing the misclassified SIMEXvariable. An example can be found below.
- Asymptotic variance estimation is only implemented for **lm** and **glm**
- The loglinear fit has the form $g(\lambda, \text{GAMMA}) = \exp(\gamma_0 + \gamma_1 \times \lambda)$. It is realized via the **log()** function. To avoid negative values the minimum +1 of the dataset is added and after the prediction later substracted $\exp(\text{predict(...)} - \text{min(data)} - 1$.
- The 'log2' fit is fitted via the **nls()** function for direct fitting of the model $y \sim \exp(\gamma_0 + \gamma_1 \times \lambda)$. As starting values the results of a LS-fit to a linear model with a log transformed response are used.
- If **nls** does not converge, the model with the starting values is returned.
- **refit()** refits the object with a different extrapolation function.
Value

An object of class ‘mcsimex’ which contains:

- **coefficients**: corrected coefficients of the MCSIMEX model,
- **SIMEX.estimates**: the MCSIMEX-estimates of the coefficients for each lambda,
- **lambda**: the values of lambda,
- **model**: the naive model,
- **mc.matrix**: the misclassification matrix,
- **B**: the number of iterations,
- **extrapolation**: the model object of the extrapolation step,
- **fitting.method**: the fitting method used in the extrapolation step,
- **SIMEXvariable**: name of the SIMEXvariables,
- **call**: the function call,
- **variance.jackknife**: the jackknife variance estimates,
- **extrapolation.variance**: the model object of the variance extrapolation,
- **variance.jackknife.lambda**: the data set for the extrapolation,
- **variance.asymptotic**: the asymptotic variance estimates,
- **theta**: all estimated coefficients for each lambda and B,

Author(s)

Wolfgang Lederer, <wolfgang.lederer@gmail.com>

References


See Also

misclass, simex
Examples

```r
x <- rnorm(200, 0, 1.142)
z <- rnorm(200, 0, 2)
y <- factor(rbinom(200, 1, (1 / (1 + exp(-1 * (-2 + 1.5 * x - 0.5 * z)))))
Pi <- matrix(data = c(0.9, 0.1, 0.3, 0.7), nrow = 2, byrow = FALSE)
dimnames(Pi) <- list(levels(y), levels(y))
ystar <- misclass(data.frame(y), list(y = Pi), k = 1)[, 1]
naiive.model <- glm(y ~ x + z, family = binomial, x = TRUE, y = TRUE)
truemodel <- glm(y ~ x + z, family = binomial)
simex.model <- mcsimex(naiive.model, mc.matrix = Pi, SIMEXvariable = "ystar")

op <- par(mfrow = c(2, 3))
invisible(lapply(simex.model$theta, boxplot, notch = TRUE, outline = FALSE, names = c(0.5, 1, 1.5, 2)))
plot(simex.model)

simex.model2 <- refit(simex.model, "line")
plot(simex.model2)
par(op)
```

# example for a function which can be supplied to the function mcsimex()
# "ystar" is the variable which is to be misclassified
# using the example above
## Not run:
my.misclass <- function (datas, k) {
  ystar <- datas$"ystar"
  p1 <- matrix(data = c(0.75, 0.25, 0.25, 0.75), nrow = 2, byrow = FALSE)
  colnames(p1) <- levels(ystar)
 rownames(p1) <- levels(ystar)
  p0 <- matrix(data = c(0.8, 0.2, 0.2, 0.8), nrow = 2, byrow = FALSE)
  colnames(p0) <- levels(ystar)
  rownames(p0) <- levels(ystar)
  ystar[datas$x < 0] <-
    misclass(data.frame(ystar = ystar[datas$x < 0]), list(ystar = p1), k = k)[, 1]
  ystar[datas$x > 0] <-
    misclass(data.frame(ystar = ystar[datas$x > 0]), list(ystar = p0), k = k)[, 1]
  ystar <- factor(ystar)
  return(data.frame(ystar))
}
simex.model.differential <- mcsimex(naiive.model, mc.matrix = "my.misclass", SIMEXvariable = "ystar")
## End(Not run)

misclass

Generates misclassified data

Description

Takes a data.frame and produces misclassified data. Probabilities for the missclassification are given in the mc.matrix.
misclass

Usage

misclass(data.org, mc.matrix, k)

Arguments

- **data.org** data.frame containing the factor variables. Must be factors
- **mc.matrix** a list of matrices giving the probabilities for the misclassification. Names of the list must correspond to the variable names in data.org. The colnames must be named according to the factor levels
- **k** the exponent for the misclassification matrix

Value

A data.frame containing the misclassified variables

Author(s)

Wolfgang Lederer, <wolfgang.lederer@gmail.com>

See Also

mcsimex, build.mc.matrix, diag.block

Examples

```r
x1 <- factor(rbinom(100, 1, 0.5))
x2 <- factor(rbinom(100, 2, 0.5))
p1 <- matrix(c(1, 0, 0, 1), nrow = 2)
p2 <- matrix(c(0.8, 0.1, 0.1, 0.8, 0.1, 0.1, 0.1, 0.8), nrow = 3)

colnames(p1) <- levels(x1)
colnames(p2) <- levels(x2)

x <- data.frame(x1 = x1, x2 = x2)
mc.matrix <- list(x1 = p1, x2 = p2)

x.mc <- misclass(data.org = x, mc.matrix = mc.matrix, k = 1)

identical(x[, 1], x.mc[, 1]) # TRUE
identical(x[, 2], x.mc[, 2]) # FALSE
```
**Measurement error in models using SIMEX**

**Description**

Implementation of the SIMEX algorithm for measurement error models according to Cook and Stefanski

**Usage**

```r
simex(model, SIMEXvariable, measurement.error, lambda = c(0.5, 1, 1.5, 2),
      B = 100, fitting.method = "quadratic", jackknife.estimation = "quadratic",
      asymptotic = TRUE)
```

```r
## S3 method for class 'simex'
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'simex'
summary(object, ...)
## S3 method for class 'simex'
plot(x, xlab = expression((1 + lambda)), ylab = colnames(b[, -1]),
     ask = FALSE, show = rep(TRUE, NCOL(b) - 1), ...)
## S3 method for class 'simex'
predict(object, newdata, ...)
## S3 method for class 'simex'
refit(object, ...)
```

```r
refit(object, fitting.method = "quadratic",
      jackknife.estimation = "quadratic", asymptotic = TRUE, ...)
```

**Arguments**

- `model` the naive model
- `SIMEXvariable` character or vector of characters containing the names of the variables with measurement error
- `measurement.error` given standard deviations of measurement errors. In case of homoskedastic measurement error it is a matrix with dimension 1x`length(SIMEXvariable)`. In case of heteroskedastic error for at least one SIMEX variable it is a matrix of dimension `n`x`length(SIMEXvariable)`, where `n` is the sample size, see examples.
- `lambda` vector of lambdas for which the simulation step should be done (without 0)
- `B` number of iterations for each lambda
- `fitting.method` fitting method for the extrapolation. linear, quadratic, nonlinear are implemented. (first 4 letters are enough)
jackknife.estimation

specifying the extrapolation method for jackknife variance estimation. Can be set to FALSE if it should not be performed

asymptotic

logical, indicating if asymptotic variance estimation should be done, in the naive model the option x = TRUE has to be set

x

object of class 'simex'

digits

number of digits to be printed

object

object of class 'simex'

xlab

optional name for the X-Axis

ylab

vector containing the names for the Y-Axis

ask

logical. If TRUE, the user is asked for input, before a new figure is drawn

show

vector of logicals indicating for wich variables a plot should be produced

newdata

optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used

... arguments passed to other functions

Details

Nonlinear is implemented as described in Cook and Stefanski, but is numerically instable. It is not advisable to use this feature. If a nonlinear extrapolation is desired please use the refit(\) method.

Asymptotic is only implemented for naive models of class lm or glm with homoscedastic measurement error.

refit(\) refits the object with a different extrapolation function.

Value

An object of class 'simex' which contains:

coefficients the corrected coefficients of the SIMEX model,
SIMEX.estimates

the estimates for every lambda,

model

the naive model,

measurement.error

the known error standard deviations,

B

the number of iterations,

extrapolation

the model object of the extrapolation step,

fitting.method

the fitting method used in the extrapolation step,

residuals

the residuals of the main model,

fitted.values

the fitted values of the main model,

call

the function call,

variance.jackknife

the jackknife variance estimate,
extrapolation.variance

the model object of the variance extrapolation,

variance.jackknife.lambda

the data set for the extrapolation,

variance.asymptotic

the asymptotic variance estimates,

theta

the estimates for every B and lambda,

...

Author(s)

Wolfgang Lederer, <wolfgang.lederer@gmail.com>, Heidi Bold

References


See Also

`mcsimex` for discrete data with misclassification, `lm`, `glm`

Examples

```r
## Seed
set.seed(49494)

## simulating the measurement error standard deviations
sd_me <- 0.3
sd_me2 <- 0.4
temp <- runif(100, min = 0, max = 0.6)
sd_me_het1 <- sort(temp)
temp2 <- rnorm(100, sd = 0.1)
sd_me_het2 <- abs(sd_me_het1 + temp2)

## simulating the independent variables x (real and with measurement error):
x_real <- rnorm(100)
x_real2 <- rpois(100, lambda = 2)
x_real3 <- -4*x_real + runif(100, min = -10, max = 10)  # correlated to x_real
x_measured <- x_real + sd_me * rnorm(100)
x_measured2 <- x_real2 + sd_me2 * rnorm(100)
```
x_het1 <- x_real + sd_me_het1 * rnorm(100)
x_het2 <- x_real3 + sd_me_het2 * rnorm(100)

## calculating dependent variable y:
y <- x_real + rnorm(100, sd = 0.05)
y2 <- x_real + 2*x_real2 + rnorm(100, sd = 0.08)
y3 <- x_real + 2*x_real3 + rnorm(100, sd = 0.08)

### one variable with homoscedastic measurement error
(model_real <- lm(y ~ x_real))

(model_naiv <- lm(y ~ x_measured, x = TRUE))

(model_simex <- simex(model_naiv, SIMEXvariable = "x_measured", measurement.error = sd_me))
plot(model_simex)

### two variables with homoscedastic measurement errors
(model_real2 <- lm(y ~ x_real + x_real2))

(model_naiv2 <- lm(y2 ~ x_measured + x_measured2, x = TRUE))

(model_simex2 <- simex(model_naiv2, SIMEXvariable = c("x_measured","x_measured2"),
measurement.error = cbind(sd_me, sd_me2)))
plot(model_simex2)

### Not run:
### one variable with increasing heteroscedastic measurement error
model_real

(mod_naiv1 <- lm(y ~ x_het1, x = TRUE))

(mod_simex1 <- simex(mod_naiv1, SIMEXvariable = "x_het1", measurement.error = sd_me_het1, asymptotic = FALSE))
plot(mod_simex1)

### two correlated variables with heteroscedastic measurement errors
(model_realS <- lm(yS ~ x_real + x_realS))

(mod_naivR <- lm(yS ~ x_het1 + x_hetRL x = TRUE))

(mod_simexR <- simex(mod_naivR, SIMEXvariable = c("x_het1", "x_het2"),
measurement.error = cbind(sd_me_het1, sd_me_het2), asymptotic = FALSE))
plot(mod_simexR)

### two variables, one with homoscedastic, one with heteroscedastic measurement error
model_realR

(mod_naiv3 <- lm(y2 ~ x_measured + x_het2, x = TRUE))
(mod_simex3 <- simex(mod_naiv3, SIMEXvariable = c("x_measured", "x_het2"),
  measurement.error = cbind(sd_me, sd_me_het2), asymptotic = FALSE))

### glm: two variables, one with homoscedastic, one with heteroscedastic measurement error

t <- x_real + 2*x_real2 + rnorm(100, sd = 0.01)
g <- 1 / (1 + exp(t))

u <- runif(100)
ybin <- as.numeric(u < g)

(logit_real <- glm(ybin ~ x_real + x_real2, family = binomial))

(logit_naiv <- glm(ybin ~ x_measured + x_het2, x = TRUE, family = binomial))

(logit_simex <- simex(logit_naiv, SIMEXvariable = c("x_measured", "x_het2"),
  measurement.error = cbind(sd_me, sd_me_het2), asymptotic = FALSE))

summary(logit_simex)
print(logit_simex)
plot(logit_simex)

## End(Not run)
Index

*Topic datagen
  diag.block, 4
  misclass, 10

*Topic models
  mcsimex, 7
  simex, 12
  simex-package, 2

*Topic package
  simex-package, 2

*Topic regression
  mc.matrix, 5
  simex-package, 2

build.mc.matrix, 11
build.mc.matrix (mc.matrix), 5

ccheck.mc.matrix (mc.matrix), 5

diag, 5
  diag.block, 4, 6, 11

glm, 2, 14

kronecker, 5

lm, 2, 14

mc.matrix, 5
mcsimex, 2, 6, 7, 11, 14
misclass, 2, 6, 9, 10

plot.mcsimex (mcsimex), 7
plot.simex (simex), 12
predict.mcsimex (mcsimex), 7
predict.simex (simex), 12
print.mcsimex (mcsimex), 7
print.simex (simex), 12
print.summary.mcsimex (mcsimex), 7
print.summary.simex (simex), 12

refit (simex), 12
refit.mcsimex (mcsimex), 7
simex, 2, 9, 12
simex-package, 2
summary.mcsimex (mcsimex), 7
summary.simex (simex), 12