Package ‘nmm’

January 7, 2021

Type Package
Title Nonlinear Multivariate Models
Version 0.9
Description Estimates a subset of nonlinear multivariate models (NMM):

system of nonlinear regressions (SNR), logit, and a joint model of SNR and logit.

'nmm' uniquely accounts for correlations between the error terms from nonlinear regressions
and the probabilities from logit models.

It also enables a very flexible design of logit: alternative-specific indirect utilities,
individual-specific choice set and number of actual choices.

Imports Rdpack, AER, mlogit, Hmisc, stats, gsubfn, abind, tidyr, plyr,
dplyr
Depends R (>= 4.0), systemfit, DEoptim, data.table, magrittr, maxLik
RdMacros Rdpack
License GPL (>= 2)
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
Suggests knitr, rmarkdown, recipes, Stat2Data
NeedsCompilation no
Author Simona Jokubauskaite [aut, cre],
Reinhard Hoessinger [aut],
Friedrich Leisch [aut]
Maintainer Simona Jokubauskaite <rteam.prog@gmail.com>
Repository CRAN
Date/Publication 2021-01-07 11:20:03 UTC

R topics documented:

  addInter ......................................................... 2
  add_variable ................................................... 3
addInter

**Description**

addInter add interactions into continuous equations.

**Usage**

addInter(eqcont, par_c, intv, inter_parl)
**add_variable**

**Arguments**

- `eqcont` Vector of strings containing equations.
- `par_c` Names of coefficients.
- `intv` Vector of integers corresponding to coefficients to which interactions should be added.
- `inter_parl` Names of new coefficients (interactions).

**Value**

list: 1 - expressions of errors, equations, parameters to estimate

**Examples**

```r
eq_c <- c("Tw ~ tw*w + phi*Tc", "Tf1 ~ (1+w)^tw + phi^3*Tc")
parl <- c("tw", "phi")
intv <- c(1,0)
inter_parl <- c('yttw','yyphi')
res <- addInter(eq_c, parl, intv, inter_parl)
```

---

**add_variable**

**Description**

add_variable adds columns to the data matrix

**Usage**

```r
add_variable(data = data, dname = "chc", weights = NULL)
```

**Arguments**

- `data` data.frame, if dname="chc" columns "chc_i" has to be in the data.
- `dname` if dname="chc" (dummy for chosen alternative) dummy for the choice alternative added, if "weights" weights added
- `weights` Matrix with weights to be added to the data

**Value**

data.frame

**Examples**

```r
chc <- c(1,2,1,4,3,1,4)
data <- data.frame(choice=chc, x=rnorm(length(chc)), y=rnorm(length(chc)))
add_variable(data, dname="chc")
ww <- c(1,1,1,2,2,2,3)
add_variable(data, dname="weights", weights=ww)
```
AICc

Adjusted Akaike's Information Criterion.

Description

Calculates adjusted and Bayesian Information Criterion for nmm object

Usage

AICc(object, ..., k = 2)

## S3 method for class 'nmm'
AICc(object, ..., k = 2)

## Default S3 method:
AICc(object, ..., k = 2)

## S3 method for class 'nmm'
BIC(object, ..., k = 2)

Arguments

object Fitted nmm model.

... Not used.

k Multiplication factor.

Value

a numeric value with the corresponding AIC, AICc, BIC.

Examples

library(systemfit)
data( ppine , package="systemfit")
hg.formula <- hg = exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev)
dg.formula <- dg = exp( d0 + d1*log(dbh) + d2*hg + d3*cr)
labels <- list( "height.growth", "diameter.growth" )
model <- list( hg.formula, dg.formula )
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001,
d0=-0.5, d1=0.009, d2=0.25, d3=0.005)
model.sur <- nlsystemfit( "SUR", model, start.values, data=ppine, eqnlabels=labels )
eq_c <- as.character(c(hg.formula, dg.formula))
par1 <- c(paste0("h", 0:3),paste0("d", 0:3))
res <- nmm(ppine, eq_c=eq_c, start_v=start.values, par_c=par1,
  eq_type = "cont", best_method = FALSE)
aa <- in2nmm(res, model.sur$B)
AICc(res)
AICc(aa)
cond_expr returns moments of conditional multivariate normal distribution \( X|Y \) (last variable is dependent). Only expression for \( X|Y \). Requires installation of Maxima software.

**Description**

cond_expr returns moments of conditional multivariate normal distribution \( X|Y \) (last variable is dependent). Only expression for \( X|Y \). Requires installation of Maxima software.

**Usage**

cond_expr(neq, sdv, mv, nconteq = neq - 1, tex = FALSE)

**Arguments**

- **neq**: Number of equations/variables.
- **sdv**: Vector of standard deviation of normally distributed variables, e.g. \( c(NA, NA, NA, 1) \) \( NA \) - unknown, any number - know.
- **mv**: Vector of means of normally distributed variables, e.g. \( rep(0, 4) \).
- **nconteq**: Number of continuous equations.
- **tex**: i if TRUE TeX expressions from wxMaxima are returned.

**Value**

List of strings. First element is an expression of conditional mean and covariance. The second element is a TeX formula.

**Examples**

# this means that \( E[y_3|y_1,y_2] \) and \( V[y_3|y_1,y_2] \) will be returned
# all continuous w/ unknown means
## Not run:
res <- cond_expr(neq=3)
# 3 continuous w/ unknown means and the last one with mean 0 and sd 1, c1c2c3
res <- cond_expr(neq=4, sdv=c(NA, NA, NA, 1), mv=c(NA, NA, NA, 0))
# 2 continuous w/ unknown means and 2 discrete with mean 0 and sd 1, d1c1c2c3d2
res <- cond_expr(neq=4, sdv=c(NA, NA, 1, 1), mv=c(NA, NA, 0, 0), nconteq=2)

## End(Not run)
Description

Calculate RMSE, MAPE, R\(^2\) and adjusted R\(^2\)

Usage

```r
cont_stats(
  x,
  which = c("all", "RMSE", "MAPE", "Rx2", "Rx2adj"),
  only_total = FALSE
)
```

Arguments

- `x`: Fitted `nmm` model.
- `which`: What to calculate. Options: "all", "RMSE", "MAPE", "Rx2", "Rx2adj".
- `only_total`: If TRUE, calculate statistics only for totals.

Value

Matrix with Goodness of fit measures

Examples

```r
library(systemfit)
data(ppine, package="systemfit")
hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr)
labels <- list( "height.growth", "diameter.growth" )
model <- list( hg.formula, dg.formula )
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001,
                  d0=-0.5, d1=0.009, d2=0.25, d3=0.005)
model.sur <- nlsystemfit( "SUR", model, start.values, data=ppine, eqnlabels=labels )
eq_c <- as.character(c(hg.formula, dg.formula))
par_l <- c(paste0("h", 0:3), paste0("d", 0:3))
res <- nmm(ppine, eq_c=eq_c, start_v=start.values, par_c=par_l,
          eq_type = "cont", best_method = FALSE)
cont_stats(res, which = "all")
```
**convert_attr2exp**

Converts symbolic attribute of derivative into expression object.

**Description**

**convert_attr2exp** converts symbolic attribute of derivative into expression object.

**Usage**

```r
convert_attr2exp(obj)
```

**Arguments**

- **obj**
  
  Symbolic expression of gradient or hessian

**Value**

Combine expression of derivatives

**Examples**

```r
eq1 <- parse(text="2*(log(sin(x)/log(x)))+x^4*log(x)+cos(y+x)"

tt1 <- deriv(eq1, c("x", "y"), hessian=TRUE)
r1 <- convert_attr2exp(extract_attr_deriv(tt1, "grad"))
r2 <- convert_attr2exp(extract_attr_deriv(tt1, "hessian"))
```

---

**dat4cond_mean_cov_expr**

Log-likelihood expressions for cont. equations plus 1 discrete

**Description**

Log-likelihood expressions for cont. equations plus 1 discrete

**Usage**

```r
data(dat4cond_mean_cov_expr)
```

**Format**

An object of class list of length 4.

**Examples**

```r
data(dat4cond_mean_cov_expr)
```
Example dataset

Description
Data "MathPlacement" taken from Stat2Data package.

Usage
data(dataM)

Format
A data frame containing:

- **Student**: Identification number for each student
- **Gender**: 0=Female, 1=Male
- **PSATM**: PSAT score in Math
- **SATM**: SAT score in Math
- **ACTM**: ACT Score in Math
- **Rank**: Adjusted rank in HS class
- **Size**: Number of students in HS class
- **GPAadj**: Adjusted GPA
- **PlcmntScore**: Score on math placement exam
- **Recommends**: Recommended course: R0 R01 R1 R12 R2 R3 R4 R6 R8
- **Course**: Actual course taken
- **Grade**: Course grade
- **RecTaken**: 1=recommended course, 0=otherwise
- **TooHigh**: 1=took course above recommended, 0=otherwise
- **TooLow**: 1=took course below recommended, 0=otherwise
- **CourseSuccess**: 1=B or better grade, 0=grade below B
- **DR_Course**: according to recommendations, which level of course was taken: alow - lower, bnorm - recommended, chigh - higher

Details
Code for data modifications can be found in the example section.
Examples

```r
data(dataM)
library(magrittr)
library(dplyr)
if (requireNamespace("recipes", quietly = TRUE)&requireNamespace("Stat2Data", quietly = TRUE)){
data("MathPlacement", package="Stat2Data")
head(MathPlacement)
library(recipes)
# As some of the data is missing, k-nearest neighbors (knn) imputation is
# used to fill the gaps. This is done with recipes package and function
# step_knnimpute.
dataM <- recipe(~ ., data = MathPlacement) %>%
  step_knnimpute(everything()) %>% prep() %>% juice()
# Afterwards we create a categorical variable that will show whether a
# student took a course which was too high, too low, the recommended one or
# something else happened:
dataM %<>% mutate(Student = 1:n(), DR_Course = case_when(
  TooHigh == 1 ~ "chigh",
  TooLow == 1 ~ "alow",
  RecTaken == 1 ~ "bnormal",
  TRUE ~"dother"
  ))
# We remove observations with ambiguous course status:
dataM %<>% filter(DR_Course!="dother")
dataM %>% select(DR_Course) %>% table %>% t
}
```

**datmaxle**  
*Log-likelihood expressions for cont. equations*

**Description**  
Log-likelihood expressions for cont. equations

**Usage**  
```r
data(datmaxle)
```

**Format**  
An object of class list of length 4.

**Examples**  
```r
data(datmaxle)
```
**datmlsem**

*Log-likelihood expressions for cont. equations sem*

**Description**

Log-likelihood expressions for cont. equations sem

**Usage**

```r
data(datmlsem)
```

**Format**

An object of class list of length 4.

**Examples**

```r
data(datmlsem)
```

---

**diagnostics**

*Goodness of fit measures for both parts*

**Description**

Calculation RMSE, misclassification and other goodness of fit measures.

**Usage**

```r
diagnostics(
  x,
  xdigit = 4,
  which = "all",
  only_total = FALSE,
  cPseudoR = TRUE,
  cRs = TRUE
)
```

**Arguments**

- `x` Fitted nmm model.
- `xdigit` rounding number
- `which` What to calculate. Options: "all", "RMSE", "MAPE", "Rx2", "Rx2adj".
- `only_total` If TRUE, calculate statistics only for the whole system
- `cPseudoR` If TRUE, calculate pseudo R^2s.
- `cRs` Include "AIC", "AICc", "BIC"
Value

matrix with goodness of fit measures. attribute corr holds empirical variance-covariance matrix.

Examples

```
library(systemfit)
data(ppine, package="systemfit")
hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*ttht^2 + h3*elev)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr)
labels <- list( "height.growth", "diameter.growth" )
model <- list( hg.formula, dg.formula )
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001,
d0=-0.5, d1=0.009, d2=0.25, d3=0.005)
model.sur <- nlsystemfit( "SUR", model, start.values, data=ppine, eqnlabels=labels )

res <- nmm(ppine, eq_c=eq_c, start_v=start.values, par_c=parl, eq_type = "cont", best_method = FALSE)
ressur <- in2nmm(res, new_coef=model.sur$b)
diagnostics(res)
diagnostics(ressur)
```

# example discrete

```
library(mlogit)
data("Fishing", package = "mlogit")
Fish <- mlogit.data(Fishing, varying = c(2:9), shape = "wide", choice = "mode")
## a pure "conditional" model
mres <- summary(mlogit(mode ~ price + catch, data = Fish))
data <- prepare_data(Fish %>% data.frame %>% dplyr::select(-idx),
choice="alt", dummy="mode", PeID="chid", mode_spec_var = c("price", "catch"),
type="long")
eq_d <- c("a1 + p1 * price_1 + p2 * catch_2", "a2 + p1 * price_2 + p2 * catch_2",
"a3 + p1 * price_3 + p2 * catch_3", "a4 + p1 * price_4 + p2 * catch_4")
par_d <- c(paste0("a", 1:4), paste0("p", 1:2))
res <- nmm(data, eq_d=eq_d, par_d=par_d, eq_type="disc")
ncoef <- mres$coefficients
names(ncoef) <- par_d[-1]
resdisc <- in2nmm(res, new_coef = ncoef)
a <- diagnostics(res, xdigit=2)
a2 <- diagnostics(resdisc)
attributes(a2)$corr
```

```r
```
expr_ll_norm

Log-likelihood expressions for cont. equations

Description

Log-likelihood expressions for cont. equations
Usage

data(expr_ll_norm)

Format

An object of class list of length 8.

Examples

data(expr_ll_norm)

expr_ll_norm_v2

Another Log-likelihood expressions for cont. equations version 2

Description

Another Log-likelihood expressions for cont. equations version 2

Usage

data(expr_ll_norm_v2)

Format

An object of class list of length 8.

Examples

data(expr_ll_norm_v2)

extract_attr_deriv

extract_attr_deriv converts attributes(hessian/gradient) of deriv() into a matrix of character strings.

Description

extract_attr_deriv converts attributes(hessian/gradient) of deriv() into a matrix of character strings.

Usage

extract_attr_deriv(ex, attribute)

Arguments

ex Expression of derivative. Results of deriv().
attribute "grad" for gradient or "hessian" for the Hessian matrix.
**formula2string**

**Value**

Returns a matrix of character strings.

**Examples**

```r
eq <- parse(text="2*(log(sin(x)/log(x)))+x^4*log(x)+cos(y+x)"

tt <- deriv(eq, c("x", "y"), hessian=TRUE)
g <- tt%>%extract_attr_deriv(. attribute = "grad")
h <- tt%>%extract_attr_deriv(. attribute = "hessian")
```

---

**formula2string**


**Description**


**Usage**

`formula2string(x)`

**Arguments**

- `x` String with square brackets.

**Value**

String without square brackets.

**Examples**

```r
xx <- "par[1]*3+par[2]*par+rhol[1,2]+sigma1[2]"
formula2string(xx)
```
**f_create**

`f_create` creates functions for log-likelihood of different models.

**Description**

`f_create` creates functions for log-likelihood of different models.

**Usage**

```r
f_create(
  mn,
  data,
  fixed = 0,
  cheqs0 = NULL,
  separate = FALSE,
  probt = NULL,
  tformula = NULL,
  hessian = NULL,
  transform = TRUE,
  sume = NULL
)
```

**Arguments**

- `mn` Expression, can be a list of equations.
- `data` Name of the data frame with which the function will be evaluated.
- `fixed` Integer, which parameter is fixed to be 0.
- `cheqs0` If continuous are supplied, include the expressions of errors.
- `separate` if TRUE, separate log-likelihood for each equations is produced.
- `probt` Expressions of un-simplified probabilities with quantile transformation(qnorm(ifelse(P))).
- `tformula` unsimplified log(P)
- `hessian` Adds lines to check the Hessian, hessian should be the name of hessian function.
- `transform` if TRUE, adds lines to check conditional means
- `sume` Expression of summed likelihoods.

**Value**

Function.
Examples

```r
eq_d <- c("ASC1 * 1 + B11_dur * dur_1", "ASC2 * 1 + B12_dur * dur_2",
"ASC3 * 1 + B13_dur * dur_3 + B20_cost * cost_3 + B53_parkman * PbAvl_3",
"ASC4 * 1 + B14_dur * dur_4 + B20_cost * cost_4 + B34_serv * servIdx_4 + B44_stop * stopUs1R1_4")
parl <- c(paste0("ASC", 1:4), paste0("B1", 1:4, ",_dur"), "B20_cost", "B53_parkman", "B34_serv", "B44_stop")
obj <- get_par(parl, eq_d)
ffor <- obj$cheqs0
res <- MNlogitf(ffor, separateEmm=FALSE, transform=FALSE)
ff <- f_create(res$formula, data="data", fixed=1)
```

**get_npar**

*Get number of parameters or vector of parameters in supplied equations. Extracts the number of parameters used in equations. Parameters are given as par[1], ..., par[n].*

### Description

*get_npar* Get number of parameters or vector of parameters in supplied equations. Extracts the number of parameters used in equations. Parameters are given as par[1], ..., par[n].

### Usage

```r
get_npar(x, values = FALSE)
```

### Arguments

- **x** List of strings.
- **values** if TRUE returns the character vector with parameters (par[i]).

### Value

Number of parameters or vector with parameters (in form par[i]).

### Examples

```r
eq_d <- c("ASC1 * 1 + B11_dur * dur_1", "ASC2 * 1 + B12_dur * dur_2",
"ASC3 * 1 + B13_dur * dur_3 + B20_cost * cost_3 + B53_parkman * PbAvl_3",
"ASC4 * 1 + B14_dur * dur_4 + B20_cost * cost_4 + B34_serv * servIdx_4 + B44_stop * stopUs1R1_4")
disc_par <- get_par(parl, eq_d)
get_npar(disc_par$cheqs0)
get_npar(disc_par$cheqs0, values=TRUE)
```
get_par  

get_par replaces names of parameters with par[i].

Description  

get_par replaces names of parameters with par[i].

Usage  

get_par(par, object)

Arguments  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>par</td>
<td>Names of parameters, vector of character strings.</td>
</tr>
<tr>
<td>object</td>
<td>List consisting formulas.</td>
</tr>
</tbody>
</table>

Value  

A list object consisting of equations for errors with par[i] (cheqs0), the original list with formulas (eqlab), vector of parameters (parld, same as par), vector of parameters in form of par[i] (parn), exogenous variables (exog), endogenous variables (endog, in case of discrete ""), number of parameters in each equation (neq_par).

Examples  

```r
# System of Non-linear Regressions
eq_c <- c("hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev)",
"dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr))")
par_c <- c(paste0("h", 0:3),paste0("d", 0:3))
para_cont <- get_par(par_c, eq_c)
# Indirect utility functions for discrete choice:
eq_d <- c("a1 + p1 * price_1 + p2 * catch_2", "a2 + p1 * price_2 + p2 * catch_2",
"a3 + p1 * price_3 + p2 * catch_3", "a4 + p1 * price_4 + p2 * catch_4")
par_d <- c(paste0("a", 1:4), paste0("p", 1:2))
disc_par <- get_par(par_d, eq_d)
```

get_start  

get_start get starting values for discrete or continuous choice model.

Description  

get_start get starting values for discrete or continuous choice model.
get_start

Usage

get_start(
  eq_c = NULL,
  eq_d = NULL,
  data = NULL,
  part = "joint",
  datan = "data",
  fixed_term = FALSE,
  weight_paths = TRUE,
  weight_paths_cont = FALSE,
  data_weight = NULL,
  par_c = NULL,
  par_d = NULL,
  best_method = FALSE,
  startvals = NULL,
  DEoptim_run = FALSE,
  hessian = NULL,
  transform = TRUE,
  MNtypef = "logit",
  pardogit = NULL,
  opt_method = "BFGS",
  numerical_deriv = FALSE
)

Arguments

eq_c                Continuous equations errors.
eq_d                Discrete equations.
data                data.frame is used in the optimization.
part                Type of estimation: "joint", "cont", "disc".
datan               Name of data.frame used in the optimization.
fixed_term          if TRUE, includes fixed term in log-likelihood.
weight_paths        if TRUE, weights paths of the whole system.
weight_paths_cont   if TRUE, weight paths only in continuous part.
data_weight          data.frame with weights for continuous and discrete equations, same dim as data.
par_c                Names of parameters in continuous equations.
par_d                Names of parameters in discrete equations.
best_method          if TRUE, try all possible optimization methods and choose the one with the smallest likelihood.
startvals            Starting values, can be also NULL.
DEoptim_run          if TRUE, runs DEoptim for the optimization.
hessian              Name of hessian function.
transform if TRUE, quantile transformation is applied.
MNTtypef "dogit" or "logit".
pardogit "dogit" parameters.
opt_method optimization method to use.
umerical_deriv if TRUE, numerical derivatives are calculated in nmm function

Value
Starting values for discrete or continuous blocks.

Examples

# Example of discrete choice model
data("TravelMode", package = "AER")
eq_d <- c("ASC1 * 1 + B2_t * travel_1 + B3_v * vcost_1",  
          "ASC2 * 1 + B2_t * travel_2 + B3_v * vcost_2",  
          "ASC3 * 1 + B2_t * travel_3 + B3_v * vcost_3",  
          "ASC4 * 1 + B2_t * travel_4 + B3_v * vcost_4")
parl <- c(paste0("ASC", 1:4), "B2_t", "B3_v")
obj <- get_par(parl, eq_d)

mode_spec_var <- c("wait", "vcost", "travel", "gcost")
data <- TravelMode
data$wait <- as.numeric(data$wait)
data[data$wait==0,"wait"] <- 0.000001 # add a small number to 0
data$travel <- as.numeric(data$travel)
data[data$travel==0,"travel"] <- 0.000001
data$vcost <- as.numeric(data$vcost)
data[data$vcost==0,"vcost"] <- 0.000001
data <- prepare_data(data, choice="mode", dummy="choice", PeID="individual", WeID="",  
type="long", mode_spec_var =mode_spec_var, wc=FALSE)
stv <- get_start(eq_d=eq_d, data=data, datan="data", part="disc", par_d = parl,  
transform = FALSE)

# Example of continuous model

data("CreditCard", package="AER")
cdat <- CreditCard
cdat$income2 <- cdat$income^2
cdat$d_selfemp <- as.numeric(cdat$selfemp)
eq_c <- c("expenditure ~ b1*age + b2*income + b3*income2",  
          "income ~ a1*age + a2*d_selfemp + a3*dependents + a4*majorcards")
parl <- c(paste0("b", 1:3), paste0("a", 1:4))
para_cont <- get_par(parl, eq_c)
cheqs0 <- para_cont$cheqs0

stv <- get_start(eq_c = eq_c, data=cdat, datan="cdat", part="cont", par_c=parl)
**grad_hess_eval**

forms function of gradient and Hessian of log-likelihood produced by f_create.

### Description

grad_hess_eval forms function of gradient and Hessian of log-likelihood produced by f_create.

### Usage

```r
grad_hess_eval(mn, parnl, hessian = FALSE, fixed = 0, data = "", cheqs0 = NULL)
```

### Arguments

- **mn**: Expression, can be a list of equations.
- **parnl**: Names of parameters.
- **hessian**: if TRUE, returns hessian function, otherwise gradient.
- **fixed**: Integer, which parameter is fixed to be 0.
- **data**: Name of the data frame with which the function will be evaluated.
- **cheqs0**: If continuous are supplied, include the expressions of errors.

### Value

A function for evaluation of gradient or Hessian.

### Examples

```r
eq_d <- c("ASC1 * 1 + B11_dur * dur_1", "ASC2 * 1 + B12_dur * dur_2", "ASC3 * 1 + B13_dur * dur_3 + B20_cost * cost_3 + B53_parkman * PbAvl_3", "ASC4 * 1 + B14_dur * dur_4 + B20_cost * cost_4 + B34_serv * servIdx_4 + B44_stop * stopUs1R1_4")
parl <- c(paste0("ASC", 1:4), paste0("B1", 1:4, ",dur"), "B20_cost", "B53_parkman", "B34_serv", "B44_stop")
disc_par <- get_par(parl, eq_d)
ffor <- disc_par$cheqs0
parld <- disc_par$parld
res <- MNLogitf(ffor, separate=FALSE, transform=FALSE)
parnl <- paste0("par", 1:length(parld))
gf <- grad_hess_eval (res, parnl, data="data", fixed=1)
hf <- grad_hess_eval (res, parnl, data="data", fixed=1, hessian=TRUE)
```
in2nmm

Convert some estimation results into \texttt{nmm} object.

Description

\texttt{in2nmm} convert some estimation results into \texttt{nmm} object.

Usage

\texttt{in2nmm(to, new_coef)}

Arguments

\texttt{to} \hspace{1cm} \texttt{nmm} object.

\texttt{new_coef} \hspace{1cm} New coefficients.

Value

\texttt{nmm} object.

Examples

```
# example continuous nonlinear
library(systemfit)
data( ppine , package="systemfit")
hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr)
labels <- list( "height.growth", "diameter.growth" )
model <- list( hg.formula, dg.formula )
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001,
d0=-0.5, d1=0.009, d2=0.25, d3=0.005)
model.sur <- nlsystemfit("SUR", model, start.values, data=ppine, eqnlabels=labels )
eq_c <- as.character(c(hg.formula, dg.formula))
parl <- c(paste0("h", 0:3),paste0("d", 0:3))
res <- nmm(ppine, eq_c=eq_c, start_v=start.values, par_c=parl, eq_type = "cont",
best_method = FALSE)
aa <- in2nmm(res, model.sur$b)
summary(res, new_coef=model.sur$b, type="robust")
summary(aa, type="robust")
summary(res, type="robust")
```
Description

Log-likelihood(LL) with supplied coefficients.

Usage

```r
## S3 method for class 'nnm'
logLik(
  object,
  new_coef = NULL,
  separatenmm = FALSE,
  transform = FALSE,
  methodopt = "NA",
  ...
)
```

Arguments

- `object`: Object of class `nnm`.
- `new_coef`: "New" coefficients for which LL should be calculated.
- `separatenmm`: if TRUE, returns separate LL for each equation.
- `transform`: if TRUE, do quantile transformation (normal quantiles).
- `methodopt`: "NA" means that automatic algorithm was used in `maxLik`, if equal to "BHHH" will return LL for each individual.
- `...`: some methods for this generic function require additional arguments.

Value

Returns log-likelihood.

Examples

```r
#example continuous nonlinear
library(systemfit)
data( ppine , package="systemfit")
hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr)
labels <- list( "height.growth", "diameter.growth" )
model <- list( hg.formula, dg.formula )
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001,
  d0=-0.5, d1=0.009, d2=0.25, d3=0.005)
model.sur <- nlsystemfit( "SUR", model, start.values, data=ppine, eqnlabels=labels )
eq_c <- as.character(c(hg.formula, dg.formula))
par1 <- c(paste0("h", 0:3),paste0("d", 0:3))
```
library(mlogit)
data("Fishing", package = "mlogit")
Fish <- mlogit.data(Fishing, varying = c(2:9), shape = "wide", choice = "mode")
## a pure "conditional" model
mres <- summary(mlogit(mode ~ price + catch, data = Fish))
data <- prepare_data(Fish %>% data.frame %>% dplyr::select(-idx),
                    choice="alt", dummy="mode", PeID="chid", mode_spec_var = c("price", "catch"),
type="long")
eq_d <- c("a1 + p1 * price_1 + p2 * catch_2", "a2 + p1 * price_2 + p2 * catch_2",
          "a3 + p1 * price_3 + p2 * catch_3", "a4 + p1 * price_4 + p2 * catch_4")
par_d <- c(paste0("a", 1:4), paste0("p", 1:2))
res <- nmm(data, eq_d=eq_d, eq_type="disc", fixed_term=FALSE, par_d=par_d,
           best_method=FALSE)
logLik(res)
logLik(res, new_coef=res$estimate)
logLik(res, new_coef=mres$coefficients)

MAEDtimeExpenditure  Time-use and expenditure dataset

Description

Data gathered in Austria in 2015 according to Mobility-Activity-Expenditure-Dairy (MAED), which reported all trips, activities (time use) and expenditures of 737 persons over a whole week

Usage

data(MAEDtimeExpenditure)

Format

A data frame containing:

- **PeID**  individual index
- **PeGenF**  gender of the individual
- **PeAge**  age in years
- **PeEduc**  education level
- **PeEmploy**  employment state
- **HhCh**  type of household: with children or without children
- **w**  hourly wage rate, EUR/h
- **I**  income not realted to work, EUR/week
**Details**

Time and expenditure data correspond to weekly totals. Time in hours and expenditure in EUR.

For more on data collection and description see (Aschauer et al. 2018) and (Aschauer et al. 2019).

A variant of this dataset was used in: (Schmid et al. 2019),(Jokubauskaite et al. 2019) and (Hoessinger et al. 2020).

To get the full dataset please contact r.hoessinger@boku.ac.at.

**References**


Examples

data(MAEDtimeExpenditure)

Description

Data gathered in Austria in 2015 according to Mobility-Activity-Expenditure-Dairy (MAED), which reported all trips, activities (time use) and expenditures of 737 persons over a whole week.

Usage

data(MAEDtravel)

Format

A dataframe containing:

- **PeID** individual index
- **PeGenF** gender of the individual
- **PeAge** age in years
- **PeEduc** education level
- **PeEmploy** employment state
- **HhCh** type of household: with children or without children
- **WeID** trip index
- **choice** chosen mode: 1 - walk, 2 - bike, 3 - car, 4 - public transport (PT)
- **dist** trip distance, km
- **avl_1** availability dummy for mode 1, walk
- **avl_2** availability dummy for mode 2, bike
- **avl_3** availability dummy for mode 3, car
- **avl_4** availability dummy for mode 4, PT
- **chc_1** choice dummy for mode 1, walk
- **chc_2** choice dummy for mode 2, bike
- **chc_3** choice dummy for mode 3, car
- **chc_4** choice dummy for mode 4, PT
- **cost_3** cost of car mode
- **cost_4** cost of PT mode
- **dur_1** trip duration with mode 1, minutes
- **dur_2** trip duration with mode 2, minutes
- **dur_3** trip duration with mode 3, minutes
**Details**

For more on data collection and description see (Aschauer et al. 2019) and (Aschauer et al. 2018). A variant of this dataset was used in: (Schmid et al. 2019), (Jokubauskaite et al. 2019) and (Hoessinger et al. 2020).

To get the full dataset please contact r.hoessinger@boku.ac.at.

Transport modes available: walk, bike, car, public transport (PT). The inertia variable \( \text{int}_i \) is a dummy, which is equal to one if the mode chosen by a person for a trip at the start of the current tour is the same as the one chosen in the previous tour made for the same purpose, and zero otherwise. Variables for trip purpose (leis, work, oth) were created using the effect coding.

**References**


**Examples**

data(MAEDtravel)

```r
maxle
```

*maxle returns expression of log-likelihood (LL) of joint normal distribution.*

**Description**

*maxle* returns expression of log-likelihood (LL) of joint normal distribution.

**Usage**

```r
maxle(cheqs0, fixed_term = TRUE)
```

**Arguments**

- `cheqs0` Strings defining equations of errors. Systems of Nonlinear Regressions (SNR) variant.
- `fixed_term` if TRUE fixed term -(k/2)*log(2*pi) (k number of equations) is included

**Value**

List with LL expressions of joint normal distribution, first element is string with expression for derivative calculations, the second - string for evaluation.

**Examples**

```r
# normal distribution
eq_c <- c("Tw ~ (((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) -(1 + (PH))) +
sqrt((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 *(1 + (PH) +
(tw)) *(-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w -Ec/w))))/(2 * (1 + (PH) +
(tw)))", "Tf1 ~ (th1) * (ta - (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
sqrt((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 *(1 + (PH) +
(tw)) *(-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w -Ec/w))))/(2 * (1 + (PH) +
(tw))) -Ec + 2) - 1", "Ef1 ~ (ph1)/(PH) * (w * (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
sqrt((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 *(1 + (PH) +
(tw)) *(-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w -Ec/w))))/(2 * (1 + (PH) +
(tw))) -Ec + 2) - 1")
parl <- c("tw","PH","th1","ph1")
```
maxle_p

para_cont <- get_par(parl, eq_c)
cheqs0 <- para_cont$cheqs0
res <- maxle(cheqs0=cheqs0)

### Description

`maxle_p` returns expression of partitioned log-likelihood. 
\[ f(y_1, y_2, \ldots, y_n) = f(y_1)f(y_2|y_1)f(y_3|y_2y_1)\ldots f(y_n|y_1\ldots y_{n-1}) \]

### Usage

`maxle_p(cheqs0, fixed_term = TRUE, version2 = TRUE)`

### Arguments

- **cheqs0**: Strings defining equations of errors.
- **fixed_term**: if `TRUE` fixed term \(-\frac{k}{2}\log(2\pi)\) (k number of equations) is included
- **version2**: another formulation of log-likelihood

### Value

List. First element is expression of joint distribution for derivatives, second for evaluation, third latex, fourth marginal distributions for each variable.

### Examples

```r
# joint normal distribution
eq_c <- c("Tw ~ (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
            sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) +
            (tw)) * (-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w -Ec/w)))/((2 * (1 + (PH) +
            (tw))))",
    "Tf1 ~ (th1) * (ta - (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
            sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) +
            (tw)) * (-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w)))/(2 * (1 + (PH) +
            (tw)))) -Tc + 2) - 1",
    "Ef1 ~ (ph1)/(PH) * (w * (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
            sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) +
            (tw)) * (-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w)))/(2 * (1 + (PH) +
            (tw)))) -Ec + 2) - 1")
parl <- c("tw", "PH", "th1", "ph1")
para_cont <- get_par(parl, eq_c)
cheqs0 <- para_cont$cheqs0
res <- maxle_p(cheqs0=cheqs0)
```
**mlsem**

*mlsem* returns expression of log-likelihood for joint normal distribution, for maximum likelihood (ML), Simultaneous Equations Models (SEM) variant.

**Description**

*mlsem* returns expression of log-likelihood for joint normal distribution, for maximum likelihood (ML), Simultaneous Equations Models (SEM) variant.

**Usage**

`mlsem(cheqs0, fixed_term = TRUE)`

**Arguments**

- `cheqs0`: Strings defining equations of errors.
- `fixed_term`: if TRUE fixed term -(k/2)*log(2*pi) (k number of equations) is included

**Value**

List with LL expressions of joint normal distribution, first element is string with expression for derivative calculations, the second - string for evaluation.

**Examples**

```r
# normal distribution
eq_c <- c("Tw ~ (((PH) + (tw)) * (ta - Tc) + Ec/w * (1 + (tw)) + sqrt((Ec/w *(1 + (tw)) + (ta - Tc) * ((PH) + (tw)))^2 - 4 * Ec/w * (ta -Tc) * (tw) * (1 + (PH) + (tw))))/(2 * (1 + (PH) + (tw))))",
"Tf1 ~ (th1) *(Tw - Tc)",
"Ef1 ~ (ph1)/(PH) * (w * Tw - Ec)")
parl <- c("tw","PH","th1","ph1")
para_cont <- get_par(parl, eq_c)
cheqs0 <- para_cont$cheqs0
res <- mlsem(cheqs0, fixed_term=FALSE)
```

---

**MNlogitf**

*MNlogitf* or *MNdogitf* returns log-likelihood (LL) expression for discrete equations of "logit" or "dogit" model.

**Description**

*MNlogitf* or *MNdogitf* returns log-likelihood (LL) expression for discrete equations of "logit" or "dogit" model.
Usage

MNlogitf(ffor, transform = FALSE, separatennmm = FALSE, weight_disc = FALSE)

MNdogitf(
  ffor,
  transform = FALSE,
  separatennmm = FALSE,
  weight_disc = FALSE,
  pparadogit = NULL
)

Arguments

ffor   Discrete choice equations.
transform if TRUE, quantile transformation (normal) is applied.
separatenmm if TRUE, equation specific LL is calculated
weight_disc if TRUE, equations will include equation specific weights.
ppradogit "dogit" parameters only used in MNdogitf

Value

formula simplified LL for each equation or joint
probs probability expression unsimplified
expres LL string simplified
formulat unsimplified log(P), only if transform is TRUE
probt unsimplified P with quantile transformation qnorm(ifelse(P)), only if transform is TRUE
expreteso unsimplified pnorm((qnorm(P)-mean)/sd), only if transform is TRUE
probte qnorm(P)
tval If quantile transformation is applied
sume Denominator of logit probability

Functions

• MNlogitf: returns log-likelihood(LL) expression for discrete equations of "logit" model.
• MNdogitf: returns log-likelihood(LL) expression for discrete equations of "dogit" model.

Examples

eq_d <- c("ASC1 * 1 + B11_dur * dur_1", "ASC2 * 1 + B12_dur * dur_2",
  "ASC3 * 1 + B13_dur * dur_3 + B20_cost * cost_3 + B53_parkman * PbAv1_3",
  "ASC4 * 1 + B14_dur * dur_4 + B20_cost * cost_4 + B34_serv * servIdx_4 + B44_stop * stopUs1R1_4")
par1 <- c(paste0("ASC", 1:4), paste0("B1", 1:4, ".dur"), "B20_cost", "B53_parkman", "B34_serv", 
  "B44_stop")
disc_par <- get_par(par1, eq_d)
nmm <- Maximum likelihood estimation of nonlinear multivariate models (NMM).

Description

nmm, nmm_sigma and summary are the main functions used for the estimation of NMM.

- nmm - Maximum likelihood estimation of nonlinear multivariate models (NMM)
- nmm_sigma - Optimizes the covariance matrix
- summary - returns summary of nmm object with "normal", "robust" or "clustered" standard errors. With option new_coef one can supply new coefficients and test their significance.

Usage

nmm(
  data,
  eq_type = c("joint", "cont", "disc"),
  eq_d = NULL,
  eq_c = NULL,
  par_c = NULL,
  par_d = NULL,
  start_v = NULL,
  check_hess = TRUE,
  corrl = TRUE,
  weight_paths = TRUE,
  weight_paths_cont = FALSE,
  data_weight = NULL,
  estimate = TRUE,
  fixed_term = FALSE,
  best_method = FALSE,
  DEoptim_run = FALSE,
  hessian = "joint_hess",
  print_out = FALSE,
  diff_hessian = FALSE,
  bayesian_random = FALSE,
  DEoptim_run_main = FALSE,
  deconst = 2,
  numerical_deriv = FALSE,
  best_method4start = FALSE,
  eqsys = "sur",
  miterlim = 10000,
  opt_method = "BFGS",
)
nmm

try_last_DEoptim = TRUE,
transform = NULL,
MNtypef = "logit",
nmm_object = NULL
)

## S3 method for class 'nmm'
summary(object, type = "normal", new_coef = NULL, ...)
nmm_sigma(
    object,
    methodopt = "BFGS",
    try_1good = TRUE,
    try_DEoptim = FALSE,
    try_diff_method = FALSE,
    trace = FALSE,
    estimate = FALSE
)

Arguments

data data.frame used in the optimization.
eq_type Possible options "joint", "cont", "disc".
eq_d Discrete equations.
eq_c Continuous equations.
par_c Parameters from continuous equations.
par_d Parameters from discrete equations.
start_v Starting values for optimization. If NULL, starting values are found by get_start function.
check_hess If TRUE, check the Hessian.
corr If TRUE, correlation between blocks (continuous and discrete).
weight_paths If TRUE, weight according to the number of choices made by individual i will be added to the whole system.
weight_paths_cont If TRUE, if only to continuous part should be weighted.
data_weight Data weight matrix.
estimate If TRUE, estimation is performed.
fixed_term If TRUE, includes fixed term to continuous equation block.
best_method If TRUE, all optimizers are checked.
DEoptim_run If TRUE, runs DEoptim in generation of starting values.
hessian String, name of the Hessian function.
print_out If TRUE, prints out log-likelihood for each equation.
diff_hessian If TRUE, for changing hessian and gradient with weights.
bayesian_random

If TRUE, than par[1] is changed to par[1] to be used for optimization of random parameters in Bayesian estimation.

DEoptim_run_main

If TRUE, run DEoptim in the main optimization.

deconst

absolute value of lower and upper bound in DEoptim optimization.

numerical_deriv

If TRUE, uses numerical derivative instead of the analytical.

best_method4start

If TRUE, all optimizers are checked for starting values.

eqsys

"sur" or "sem".

miterlim

Number many iterations passed to maxLik function

opt_method

optimization method for maxLik.

try_last_DEoptim

If TRUE, in case of error in maxLik should DEoptim be run.

transform

if TRUE, quantile transformation is applied to discrete equations.

MNtypef

estimate "logit", or "dogit"

nmm_object

nmm object created by nmm function.

object

nmm object, for summary and nmm_sigma

type

Type of standard errors ("robust", "clustered", "normal"), for summary

new_coef

New coefficients that will be tested, for summary

... additional arguments affecting the summary produced, for summary

methodopt

optimizer from maxLik package, for nmm_sigma

try_1good

If TRUE, stops then first good values are found, for nmm_sigma

try_DEoptim

If TRUE, uses DEoptim for optimization, then maxLik optimizers produce errors, for nmm_sigma

try_diff_method

If TRUE, stops then first good values with Hessian check are found, for nmm_sigma

trace

If TRUE, trace of DEoptim is printed, for nmm_sigma

Value

nmm returns nmm object with estimated parameters, functions, and data.
nmm_sigma returns estimated parameters, functions, data.
summary returns summary of nmm object.

Examples

# estimation of System of Nonlinear Equations based on example from 'systemfit'
library(systemfit)
data( ppine , package="systemfit")
hg.formula <- hg ~ exp( h0 + h1*log(tht) + h2*tht^2 + h3*elev)
dg.formula <- dg ~ exp( d0 + d1*log(dbh) + d2*hg + d3*cr)
labels <- list( "height.growth", "diameter.growth" )
model <- list( hg.formula, dg.formula )
start.values <- c(h0=-0.5, h1=0.5, h2=-0.001, h3=0.0001,
d0=-0.5, d1=0.009, d2=0.25, d3=0.005)
model.sur <- nlsystemfit( "SUR", model, start.values, data=ppine, eqnlabels=labels )
res <- nmm(ppine, eq_c=eq_c, par_c=par_l, start_v = start.values,
eq_type = "cont", best_method = FALSE, numerical_deriv=TRUE)
summary(res)
res_sigma_cont <- nmm_sigma(res, estimate=TRUE) # Estimation of the Variance-Covariance matrix
summary(res_sigma_cont)

# example discrete choice
library(mlogit)
data("Fishing", package = "mlogit")
Fish <- mlogit.data(Fishing, varying = c(2:9), shape = "wide", choice = "mode")
## a pure "conditional" model
mres <- summary(mlogit(mode ~ price + catch, data = Fish))
data <- prepare_data(Fish %>% data.frame %>% dplyr::select(-idx),
choice="alt", dummy="mode", PeID="chid", mode_spec_var = c("price", "catch"),
type="long")
eq_d <- c("a1 + p1 * price_1 + p2 * catch_2", "a2 + p1 * price_2 + p2 * catch_2",
"a3 + p1 * price_3 + p2 * catch_3", "a4 + p1 * price_4 + p2 * catch_4")
par_d <- c(paste0("a", 1:4), paste0("p", 1:2))
res <- nmm(data, eq_d=eq_d, par_d = par_d, eq_type="disc", fixed_term=FALSE,
best_method=FALSE)
summary(res)

# joint estimation mockup example
data(dataM)
dataMp <- dataM %>% data.frame %>% prepare_data(. , choice="DR_Course",
PeID = "Student")
eq_c <- c("PlcmtScore ~ exp(a0 + a1 * PSATM + a2 * Rank + a3 * Size)",
"ACTM ~ exp(c0 + c1 * GPAadj")
par_c <- c(paste0("a", 0:3), paste0("c", 0:1))
eq_d <- c("ASC1",
"ASC2 + b1_2 * SATM + b2_2 * PlcmtScore",
"ASC3 + b1_3 * SATM + b2_3 * PlcmtScore")
par_d <- c(paste0("ASC", 1:3), paste0("b", rep(1:2, rep(2,2)), ",", 2:3))

mmm_joint_res <- nmm(dataMp, eq_type = "joint", eq_d = eq_d,
par_d = par_d, eq_c = eq_c, par_c = par_c,
start_v = c(a0=3.394, a1=0.001, a2=-0.001, a3=0, c0=3.583, c1=-0.008,
ASC2=-1.452, ASC3=3.047, b1_2=0.145, b1_3=0.102, b2_2=-0.133, b2_3=-0.168))
summary(mmm_joint_res)
prepare_data

Description

prepare_data prepare data for the estimation.

Usage

prepare_data(
  data,
  choice = "",
  dummy = "",
  PeID = "",
  WeID = "",
  type = "",
  mode_spec_var = "",
  avl = TRUE,
  chc = TRUE,
  wc = TRUE,
  wd = TRUE,
  nc = 0,
  weights = NULL,
  weight_paths = FALSE,
  weight_paths_cont = FALSE,
  mode_factors = NULL
)

Arguments

data data.frame
choice Name of variable with modes.
dummy Name of variable indicating, if the mode was chosen.
PeID Name of variable with individual identification numbers.
WeID Name of variable with trip identification.
type Type of data. If "long", then modifications are done.
mode_spec_var Used if format "long", mode specific variables.
avl if TRUE, includes dummies for mode availability.
chc if TRUE, includes dummies for choice of mode.
wc if TRUE, creates weights 1 for continuous equations.
wd if TRUE, creates weights 1 for discrete equations.
nc Integer, number of continuous equations.
weights Data matrix with weights, column names have to be $wc_i$(continuous), $wd_i$(discrete).
weight_paths if TRUE, weight according to number of trips per person, discrete part.
weight_paths_cont if TRUE, weight continuous part.
mode_factors if choice is not factor or numeric, this is important to supply.
Value

data.frame used for modeling.

Examples

data("TravelMode", package = "AER")
mode_spec_var <- c("wait", "vcost", "travel", "gcost")
res <- prepare_data(TravelMode, choice="mode", dummy="choice", PeID="individual", WeID="",
type="long", mode_spec_var =mode_spec_var, nc=3)

pseudoR

pseudo R^2

Description

Calculates pseudo R^2 for discrete choice part.

Usage

pseudoR(
  x,
  which = c("all", "McFadden", "adjMcFadden", "Cox&Snell", "Nagelkerke"),
  only_total = FALSE
)

Arguments

  x          Fitted nmm model.
  which      which pseudo R^2 to calculate, options are: "all", "McFadden", "adjMcFadden", "Cox&Snell", "Nagelkerke".
  only_total If TRUE, compute R^2 only for the whole sample.

Value

matrix with goodness of fit measures

Examples

library(mlogit)
data("Fishing", package = "mlogit")
Fish <- mlogit.data(Fishing, varying = c(2:9), shape = "wide", choice = "mode")
## a pure "conditional" model
mres <- summary(mlogit(mode ~ price + catch, data = Fish))
data <- prepare_data(Fish %>% data.frame %>% dplyr::select(-idx),
  choice="alt", dummy="mode", PeID="chid", mode_spec_var = c("price", "catch"),
type="long")
eq_d <- c("a1 + p1 * price_1 + p2 * catch_2", "a2 + p1 * price_2 + p2 * catch_2",
  "a3 + p1 * price_3 + p2 * catch_3", "a4 + p1 * price_4 + p2 * catch_4")
par_d <- c(paste0("a", 1:4), paste0("p", 1:2))
res <- nmm(data, eq_d=eq_d, par_d = par_d, eq_type="disc", fixed_term=FALSE,
best_method=FALSE)
pseudoR(res, which = c("McFadden"))
ncf <- c(mres$coefficients)
names(ncf) <- par_d[-1]
mress <- in2nmm(res, new_coef = ncf)
pseudoR(mress, which = c("McFadden"))
pseudoR(mress)

Description
replace_par replaces text with other text.

Usage
replace_par(iter, repdat, biogu)

Arguments
iter Integer, which line of repdat is used, if 1 iteratively all will be replaced, if >1 only the i-th parameter.
repdat data.frame with columns "old" and "new"
biogu string which will be modified

Value
Modified string.

Examples
eq_c <- c("Tw ~ (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
  sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 *
  (1 + (PH) + (tw)) * ((-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w))/(2 *
  (1 + (PH) + (tw)))")",
  "Tf1 = (th1) * (ta - (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
  sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) +
  (tw)) * ((-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w))/(2 * (1 + (PH) +
  (tw)))) - Tc + 2) - 1",
  "Ef1 = (ph1)/(PH) * (w * (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
  sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) +
  (tw)) * ((-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w))/(2 * (1 + (PH) +
  (tw)))) - Ec + 2) - 1")
parl <- c("tw", "PH", "th1", "ph1")
parll <- paste0("par", 1:length(parl), "]")
repdat <- data.frame(old=parl, new=parll)
replace_par(1, repdat, eq_c)
replace_par_wrap

replace_par_wrap replace text with other text, wrapper of replace_par.

Description

replace_par_wrap replace text with other text, wrapper of replace_par.

Usage

replace_par_wrap(repdat, obj)

Arguments

repdat data.frame with columns "old" and "new"

obj character string or vector of strings that will be modified.

Value

Modified string/vector.

Examples

eq_c <- c("Tw ~ (((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 *
(1 + (PH) + (tw)) *(-1) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w)))/(2 *
(1 + (PH) + (tw)))",
"Tf1 ~ ((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) +
sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) +
(tw)) *(-1) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w)))/((2 * (1 + (PH) +
(tw))) - (ta - Tc + 2) - 1)",
"Ef1 ~ ((PH) * (w * (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) -
(1 + (PH))) + sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 -4 *
(1 + (PH) + (tw)) *(-1) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w)))/(2 *
(1 + (PH) + (tw))) - (Ec + 2) - 1)"
parl <- c("tw", "PH", "th1", "ph1")
parll <- paste0("par[", 1:length(parl), "]")
repdat <- data.frame(old=parl, new=parll)
replace_par(2, repdat, eq_c)
replace_par_wrap(repdat, eq_c)
Description

Produce function that calculate estimates of endogenous variables from the continuous block and probabilities from discrete part.

Usage

stats_function(eq_c = NULL, eq_d = NULL, par_c = NULL, par_d = NULL, fixed = 0)

Arguments

- **eq_c**: continuous equations
- **eq_d**: discrete equations
- **par_c**: parameters from cont. eq
- **par_d**: parameters from disc. eq
- **fixed**: index of fixed parameter

Value

Returns functions.

Examples

```r
eq_d <- c("ASC1 * 1 + B11_dur * dur_1", "ASC2 * 1 + B12_dur * dur_2",
           "ASC3 * 1 + B13_dur * dur_3")
eq_c <- c("Tw ~ tw*w + ph1*Tc", "Tf1 ~ (1+w)^tw + ph1^3*Tc")
par1 <- c("tw", "ph1")
par_d <- c(aspect0("ASC", 1:3), aspect0("B1", 1:3, "_dur"))
stfunc <- stats_function(eq_c, eq_d, par1, par_d, fixed=3)
data <- matrix(runif(1000, min=0.001, max=50), ncol=8)
data <- data.frame(data)
names(data) <- c("dur_1", "dur_2", "dur_3", "w", "Tc", "avl_1", "avl_2", "avl_3")
parv <- c(0.5, 1, 1.5, 2, 1, -0.3, 0.2, -0.8)
methodopt <- "BHHH"
separatenm <- TRUE
env <- environment()
fnames <- c("prob_func", "cont_func")
sapply(1:(length(stfunc)-1), function(x) assign(fnames[x], stfunc[[x]], envir=env))
eval(parse(text=paste0("environment("fnames[1:(length(stfunc)-1)], ") &lt; env")
probs <- prob_func(parv)
apply(probs, 2, mean)
cont <- cont_func(parv)
apply(cont, 2, mean)
```
string2formula add square brackets to expressions. Reverse of formula2string. Convert \texttt{par[2]} <- \texttt{par2}, \texttt{sigma[2]} <- \texttt{sigma_2}, \texttt{sigma[2,3]} <- \texttt{sigma_2x2}

**Description**

string2formula add square brackets to expressions. Reverse of formula2string. Convert \texttt{par[2]} <- \texttt{par2}, \texttt{sigma[2]} <- \texttt{sigma_2}, \texttt{sigma[2,3]} <- \texttt{sigma_2x2}

**Usage**

\texttt{string2formula(x)}

**Arguments**

\texttt{x}  
String without square brackets.

**Value**

String with square brackets.

**Examples**

\begin{verbatim}
xx <- "par[1]*3+par[2]*par+rho1[1,2]+sigma1[2]"
xm <- formula2string(xx)
string2formula(xm)
\end{verbatim}

wxMaxima does symbolic computation in 'Maxima' Requires installation of Maxima software.

**Description**

wxMaxima does symbolic computation in 'Maxima' Requires installation of Maxima software.

**Usage**

\texttt{wxMaxima(obj, out, tex = FALSE)}

**Arguments**

\texttt{obj}  
Lines without printed output, that will be evaluated in \texttt{wxMaxima}.

\texttt{out}  
Lines with printed output, is end results of the function.

\texttt{tex}  
if \texttt{TRUE} TeX expression of the printed output.
Value

List of character strings.

Examples

#components(determinant, Sigma inverse, argument for exponent) for joint normal distribution
eq_c <- c("Tw ~ (((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) + 
            sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) + 
              (tw)) * ((-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w)))/((2 * (1 + (PH) + 
            (tw))))", 
        "Tf1 ~ (th1) * (ta - (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) + 
                  sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) + 
                    (tw)) + (-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w))/(2 * (1 + (PH) + 
                      (tw)))) - Tc + 2) - 1", 
        "Ef1 ~ (ph1)/(PH) * (w * (((((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH))) + 
                sqrt(((PH) + (tw)) * (ta - Tc + 2) + (1 + (tw)) * (Ec/w - 2/w) - (1 + (PH)))^2 - 4 * (1 + (PH) + 
                    (tw)) + (-PH) * (ta - Tc + 2) + (1 - (tw) * (ta - Tc + 2)) * (2/w - Ec/w))/(2 * (1 + (PH) + 
                      (tw)))) - Ec + 2) - 1")
parl <- c("tw","PH","th1","ph1")
para_cont <- get_par(parl, eq_c)
cheqs0 <- para_cont$cheqs0
npar <- get_npar(cheqs0)
neq <- length(cheqs0)
sdv <- rep(NA, neq)
mv <- rep(NA, neq)
sigma <- expand.grid(1:neq, 1:neq)%>%apply(., 1, function(x)paste0(x, collapse=’,’))%>%
sort
sigma <- matrix(sigma, neq, neq, byrow = TRUE)
spy <- paste0(sapply(1:neq, function(i) paste0(’[’,paste0(sigma[i,], collapse = ’,’), ‘]’))
#sigma
spy <- paste0(sapply(1:neq, function(i) paste0(’[’, paste0(sigma[i,], collapse = ’,’), ‘]’)),
            collapse = ’,’)
spy <- paste0(’Sigma : matrix(’, spy, ‘)’)
dets <- ‘dets : ratsimp(determinant(Sigma))’
invs <- ‘invs : ratsimp(invert(Sigma))’
expt <- ‘expt : ratsimp(transpose(Y) . invs . Y)’
obj <- c(spy, invs, dets, expt)
#grind function in Maxima returns an object that can be mathematically evaluated
out <- cc(’print(new)’, ’print(dets)’, ’print(invs)’, ’print(expt)’) 
## Not run:
rez <- wxMaxima(obj, out)
## End(Not run)
Index

* datasets
  - dat4cond_mean_cov_expr, 7
  - dataM, 8
  - datamxle, 9
  - datmlsem, 10
  - expr_ll_norm, 11
  - expr_ll_norm_v2, 12
  - MAEDtimeExpenditure, 22
  - MAEDtravel, 24

add_variable, 3
addInter, 2
AICc, 4
BIC.nmm (AICc), 4
cond_expr, 5
cont_stats, 6
convert_attr2exp, 7

dat4cond_mean_cov_expr, 7
dataM, 8
datamxle, 9
datmlsem, 10
diagnostics, 10
expr_ll_norm, 11
expr_ll_norm_v2, 12
extract_attr_deriv, 12

f_create, 14
formula2string, 13

get_npar, 15
get_par, 16
get_start, 16
grad_hess_eval, 19

in2nmm, 20

logLik.nmm, 21

MAEDtimeExpenditure, 22
MAEDtravel, 24
maxle, 26
maxle_p, 27
mlsem, 28
MNlogitf (MNlogitf), 28
MNlogitf, 28
nmm, 30
nmm_sigma (nmm), 30

prepare_data, 33
pseudoR, 35

replace_par, 36
replace_par_wrap, 37

stats_function, 38
string2formula, 39
summary.nmm (nmm), 30

wxMaxima, 39