Package 'pim'

May 9, 2026

Type Package

Title Fit Probabilistic Index Models

Version 2.0.4

Date 2025-01-07

Author Joris Meys [aut, cre],
Jan De Neve [aut],
Nick Sabbe [aut],
Gustavo Guimaraes de Castro Amorim [aut]

Maintainer Joris Meys <Joris.Meys@UGent.be>

Description Fit a probabilistic index model as described in
Thas et al, 2012: <doi:10.1111/j.1467-9868.2011.01020.x>. The interface to the
modeling function has changed in this new version. The old version is
still available at R-Forge.

Depends R (>= 3.0)

Imports methods, utils, stats4, nleqslv, BB

License GPL (>= 2)

URL https://github.com/CenterForStatistics-UGent/pim

BugReports https://github.com/CenterForStatistics-UGent/pim/issues

Collate 'CreateScoreFun.R' 'DysData.R' 'EngelData.R' 'Estimators.R'
'pim.poset-class.R' 'pim.environment-class.R' 'pim-package.R'
'pim.formula-class.R' 'pim-class.R' 'pim.summary-class.R'
'Extract.pim.summary.R' 'FEVData.R' 'Getters.R' 'Getters_pim.R'
'Getters_pim.formula.R' 'InternalFunctions.R'
'InternalObjects.R' 'LR.R' 'MHData.R' 'P.R' 'SUData.R'
'add.poset.R' 'as.data.frame.pim.environment.R'
'as.matrix.pim.summary.R' 'coef.R' 'confint.pim.R'
'create.poset.R' 'formula.R' 'has.intercept.R' 'is.complete.R'
'make.posfun.R' 'model.matrix.pim.R' 'new.pim.R'
'new.pim.env.R' 'new.pim.formula.R' 'new.pim.poset.R' 'nobs.R'
'penv.R' 'pim.R' 'pim.fit.R' 'pimdata.R' 'poset.R' 'print.R'
'response.R' 'sandwich.estimator.R' 'summary.R' 'vcov.R'
'vcov.estimators.R' 'vcov.internal.R' 'zzz.R'

Suggests testthat, MASS

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2025-01-08 11:30:02 UTC

Contents

`pim-package` *Probabilistic Index Models*

Description

Fit a probabilistic index model. Note that this version is NOT compatible with the previous version used in the original publications on probabilistic index models. If you want to try out the original code, please install the package pimold from R-Forge. You can install the old package using:

Details

```
install.packages('pimold', repos = 'https://R-Forge.R-project.org')
```

Author(s)

Joris Meys <Joris.Meys@UGent.be> Jan De Neve <Jan.DeNeve@UGent.be> original package and engine code by Nick Sabbe.

References

https://r-forge.r-project.org/projects/pim/

See Also

Useful links:

- https://github.com/CenterForStatistics-UGent/pim
- Report bugs at https://github.com/CenterForStatistics-UGent/pim/issues

.make.posfun *Create a poset function*

Description

This function creates a poset function from a poset. The function is not exported and shouldn't be called by the user.

Usage

```
.make.posfun(poset)
```

Arguments

poset	a vector with the columns as indices

Value

A function that takes a single vector as argument, and that returns the vector with the poset vector applied to it.

add.poset *Add a poset to a pim.environment object*

Description

This function adds a poset to a pim.environment object.

Usage

```
add.poset(x, ...)

## S4 method for signature 'pim.environment'
add.poset(x, overwrite = FALSE, ...)
```

Arguments

x	a pim.environment object
...	further parameters passed to new.pim.poset.
overwrite	a logical value indicating whether the poset should be overwritten if it's already present. Defaults to FALSE to avoid problems.

Value

The object with a (new) poset attached.

Warning

Although it might be tempting to pass the argument nobs to new.pim.poset, you shouldn't. The necessary information is taken from the respective slot in the pim.environment object.

If you provide a matrix or a list as value for the argument compare, note that you can easily create a poset that doesn't use all the observations. This might or might not be your intention. If the poset you try to create contains indices that go beyond the number of observations, you will get errors.

See Also

new.pim.poset for the possible values of the arguments compare and nobs.

Examples

```
data(DysData)
Dysenv <- new.pim.env(DysData)
Dysenv
DysenvAll <- add.poset(Dysenv, overwrite = TRUE,
                       compare = 'all', nobs = nobs(DysData))
compare(Dysenv)
compare(DysenvAll)
```

---

as.data.frame *Convert a pim.environment to a data frame*

Description

This function extracts all data from a pim.environment and returns it as a data frame. Note that this is the original data frame, not the one with pseudo observations.

Usage

```
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'pim.environment'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	a pim.environment object
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
optional	logical. if TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional
...	additional arguments to be passed to or from methods, including stringsAsFactors. For more information, see the function as.data.frame from the base package.

Details

TO DO: Insert link to how to get pseudo observations out.

Value

a data frame.

Examples

```
# Create a pim environment
data("DysData")
Dys <- new.pim.env(DysData)
str(as.data.frame(Dys))
```

---

as.matrix.pim.summary *Convert a pim.summary object to a matrix*

Description

This function converts a summary object to a matrix so values can be extracted from it.

Usage

```
as.matrix(x, ...)

## S4 method for signature 'pim.summary'
as.matrix(x, ...)

## S4 method for signature 'pim'
as.matrix(x, ...)
```

Arguments

x	a pim.summary object
...	additional arguments to be passed to or from methods. This one is ignored.

Value

a matrix with the estimate, standard error, Z value and probability for every coefficient.

classes *Extract information from pim.environment and pim.poset objects*

Description

These functions serve to extract the information contained in the objects of class pim.environment and pim.poset.

Usage

```
classes(x)

## S4 method for signature 'pim.environment'
classes(x)

## S4 method for signature 'pim.environment'
names(x)

## S4 method for signature 'pim.poset'
names(x)

compare(x)

## S4 method for signature 'pim.environment'
compare(x)

## S4 method for signature 'pim.poset'
compare(x)

model(object, ...)

## S4 method for signature 'pim'
model(object)

## S4 method for signature 'pim.summary'
model(object)

link(object, ...)

## S4 method for signature 'pim'
link(object)

## S4 method for signature 'pim.summary'
link(object)
```

Arguments

x	an object of class pim.environment or pim.poset

object	an object of class pim or pim.summary
...	arguments passed to and from other methods.

Value

classes(): A named vector with the classes of the data contained in the pim.environment

names(): For an object of class pim.environment the names of the variables in the object. For an object of class pim.poset, the name of the poset functions inside the environment

compare(): A character value indicating how the comparison is defined in a pim.poset object, or the poset-slot of a pim.environment object respectively.

model(): a character value that displays the type of model (difference, marginal, regular or customized)

link(): a character value that displays the type of link (difference, marginal, regular or customized)

See Also

nobs, poset, is.complete, pim.environment-class, pim.poset-class, pim-class, pim.summary-class

Examples

```
data(DysData)
DysPimEnv <- new.pim.env(DysData,poset=TRUE)
classes(DysPimEnv)
names(DysPimEnv)
compare(DysPimEnv)

themodel <- pim(SPC_D2 ~ Chemo, data = DysData, model = 'difference')
model(themodel)
thesummary <- summary(themodel)
model(thesummary)
```

---

coef *Extract the coefficients from a pim or pim.summary object*

Description

This function works like coef from the stats package. It extracts the coefficients from the objects.

Usage

```
coef(object, ...)

## S4 method for signature 'pim'
coef(object, ...)

## S4 method for signature 'pim.summary'
coef(object, ...)
```

Arguments

object	a pim or pim.summary object
...	currently ignored.

Value

a named vector with the coefficients.

Examples

```
data("FEVData")
Model <- pim(FEV~ Age + Smoke*Sex , data=FEVData)
coef(Model)
summ <- summary(Model)
coef(summ)
```

---

confint.pim *Calculate Wald confidence intervals around the coefficients of a PIM*

Description

This function returns Wald confidence intervals around the coefficients of a fitted pim object

Usage

```
confint(object, parm, level = 0.95, ...)

## S4 method for signature 'pim'
confint(object, parm, level = 0.95, ...)

## S4 method for signature 'pim.summary'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	a pim or pim.summary object
parm	a specification of which parameters are to be given confidence intervals. Either a vector of numbers or a vector of names. If missing, all parameters are considered
level	The confidence level required.
...	extra arguments to methods

create.poset *Create a poset*

Description

This function creates a poset for use in a pim model based on a number of observations and a comparison type. This function is called from `new.pim.poset` and returns a list that can be used as a value for its argument `compare`.

Usage

```
create.poset(compare = c("unique", "all"), n)
```

Arguments

`compare`	a character value, either 'unique' or 'all'
`n`	an single integer value indicating how many observations there are in the model.

Value

A named list with 2 elements, called "L" and "R", containing the selection indices for the left hand and right hand side of a pim.

Examples

```
create.poset(n=10)
create.poset('all',n=4)
```

CreateScoreFun *Create a score function for use in a pim.*

Description

This function creates a suitable score function for the fitting process of a probabilistic index model.

Usage

```
CreateScoreFun(Z, Y, link = c("probit", "logit", "identity"), W = NULL)
```

Arguments

`Z`	the model matrix of pseudo-observations
`Y`	a vector with the response of the pseudo-observations
`link`	a character vector indicating the link function to be used.
`W`	a vector with weights.

Value

a function used for estimating the coefficients by the estimator functions.

NOTE

This function is not exported.

DysData *This is the Dysphagia data*

Description

This is the Dysphagia data

Details

These are the columns and their meanings

- `out` Outcome: a factor with values 1 to 4, indicating the outcome
- `Chemo` Whether the patient underwent chemotherapy ("ja" is yes, "nee" is no)
- `SNP_XRCC1__77` Genotype of this SNP. A factor with three levels: "TT", "TC" and "CC"
- `SPC_D2` Dose of radiation that reached 2
- `SNP_XRCC1__77TC` 1 if `SNP_XRCC1__77` is "TC", 0 otherwise

EngelData *This is the engel data*

Description

This is the engel data

Details

These are the columns and their meanings

- `foodexp` Food expenditure (FE)
- `income` Household income (HI)

estimators *Estimator functions for probabilistic index models*

Description

This page documents different possibilities for solving the score function of a probabilistic index model or pim. All functions mentioned on this page, are essentially wrappers around different solver functions.

Usage

```
estimator.nleqslv(
  x,
  y,
  start = rep(0, ncol(x)),
  link = "logit",
  construct = NULL,
  ...
)

estimator.glm(x, y, start = rep(0, ncol(x)), link = "logit", ...)

estimator.BB(
  x,
  y,
  start = rep(0, ncol(x)),
  link = "logit",
  construct = NULL,
  method = c(1, 2, 3),
  control = list(NM = c(FALSE, TRUE)),
  ...
)
```

Arguments

x	a model matrix for the respective pim model. See also model.matrix.
y	a vector with the response for the respective pim model.
start	a vector as long as there are columns in x, containing the starting values for the algorithm
link	a character vector describing the link function. This link function is used to adapt the calculation depending on the link used in the fitting process.
construct	a function that creates the score function used by either nleqslv or BBsolve for numerical optimization. See Details. The estimator estimator.glm doesn't allow for specification of your own score function.
...	extra arguments passed down to the actual solver function. See details.

`method` A vector of integers specifying which Barzilai-Borwein steplengths should be used in a consecutive manner. The methods will be used in the order specified. More information on the help page of `BBsolve`.

`control` a list with extra controlling parameters for `BBsolve`. See the help page of `BBsolve` for more information.

Details

All functions share the same three arguments, being the design matrix `x`, the response vector `y` and the start values for the estimating function. If you follow the same principles, you can write your own wrapper function for any solver function of your choice.

The solvers `estimator.nleqslv` and `estimator.BBsolve` allow for specification of your own score function as well. For this, you have the possibility to provide a constructor function that takes three arguments

x The model matrix

y the vector with pseudo-observations

link a character vector specifying the link

The function should return a function that can be used in either `nleqslv` or `BBsolve`. If you don't specify this constructor function, the package will use the constructor function `CreateScoreFun` to provide the score function.

Value

a list with following elements:

`coef` the estimated coefficients

WARNING

If you specify your own score function without changing the estimators for the variance-covariance matrix, this vcov matrix will be blatantly wrong!!!!!

See Also

`nleqslv`, `glm.fit`, `BBsolve` for more information on the fitting algorithms.

`vcov.estimators`, `pim.fit` and `pim` for more information on the fitting process

Examples

```
# This is a reimplementation of the identity link
myconstruct <- function(x,y,link){
  # this function is returned
 function(beta){
   xb <- as.vector(x %*% beta)
   colSums(x * (y - xb))
  }
}
```

```
data(ChickWeight)
themodel <- pim(weight ~ Diet, data = ChickWeight,
construct = myconstruct)

# compare coefficients to
themodel2 <- pim(weight ~ Diet, data = ChickWeight,
                link = "identity")
coef(themodel)
coef(themodel2)

# Note that this example uses a wrong estimate for the variance-covariance matrix
# You have to specify the correct vcov estimator as well
```

`Extract.pim.summary` *Extract method for pim.summary objects*

Description

This method allows to extract data directly from a `pim.summary` object. It's exactly the same as extracting from `as.matrix(thesummary)`.

Usage

```
## S4 method for signature 'pim.summary'
x[i, j, drop = TRUE]
```

Arguments

x	a `pim.summary` object to extract information from.
i	indices specifying row to extract or replace. They are used in the same way as you woul after transforming x to a matrix.
j	see i, but for columns
drop	see `Extract`

Value

the selected matrix

Examples

```
data(FEVData)
Model <- pim(FEV~ Smoke*Sex , data=FEVData)

thesummary <- summary(Model)
thesummary[,2:3]
thesummary["Sex"]
```

FEVData — *This is the Childhood respiratory disease data*

Description

This is the Childhood respiratory disease data

Details

These are the columns and their meanings

- `FEV` Forced Expiratory Volme (FE)
- `Age` Age of the child
- `Height` Height of the child
- `Sex` Gender of the child (1 for boys, 0 for girls)
- `Smoke` 1 if the child smokes, 0 otherwise

formula — *Extract the formula from a pim or pim.formula object*

Description

This function allows you to extract a formula from a `pim` or a `pim.formula` object. In the latter case, you extract the original formula.

Usage

```
formula(x, ...)

## S4 method for signature 'pim'
formula(x, orig = FALSE, ...)

## S4 method for signature 'pim.formula'
formula(x, ...)
```

Arguments

`x`	a `pim` or `pim.formula` object
`...`	arguments passed to other methods
`orig`	a locigal value indicating whether the original formula (`TRUE`) or the `pim.formula` object (`FALSE`) should be returned. Defaults to `FALSE`

Details

This function is based on formula from the stats package. It creates a generic and can hence be used more or less in the same way. Yet, as the pim package is dependent on the correct binding between the formula objects and different environments, it is advised not to change the environments tied to the formulas and to use this function only to extract the desired information.

Value

a pim.formula if x is a pim object and orig = TRUE. Otherwise a formula object.

See Also

pim.formula-class and pim-class for more information on the classes.

Examples

```
data("DysData")
themodel <- pim(SPC_D2 ~ Chemo, data = DysData)

thepimform <- formula(themodel)
formula(thepimform)
formula(themodel, orig = TRUE)
```

---

has.intercept *Check whether formula has an explicit intercept*

Description

This function checks whether an intercept is present in a formula of some form. It works for a formula, a terms.object a pim.formula object or a character vector representing a formula.

Usage

```
has.intercept(x)

## S4 method for signature 'character'
has.intercept(x)

## S4 method for signature 'formula'
has.intercept(x)

## S4 method for signature 'terms'
has.intercept(x)

## S4 method for signature 'pim.formula'
has.intercept(x)
```

```
## S4 method for signature 'pim'
has.intercept(x)
```

Arguments

x	either a formula, pim.formula, terms.object or a character vector representing a formula.

Details

In case of a terms.object, this function only checks whether the intercept attribute is larger than 0. In all other cases, the function checks whether it can find a + 1 somewhere in the formula, indicating that an intercept has to be fit in a pim.

Value

a single logical value

WARNING

This function will return FALSE for a standard formula that is used in the context of a marginal model. Keep in mind that when specifying model = 'marginal' in a call to pim, the model will contain an intercept regardless of the outcome of has.intercept

Note

This function is meant to be used in the context of a pim call. Although the function should work for standard formulas as well, correct results are not guaranteed when used outside a pim context.

Examples

```
data("FEVData")
# Create the "model frame"
FEVenv <- new.pim.env(FEVData, compare="unique")
# create the formula and bind it to the pim.environment.
FEVform <- new.pim.formula(
  Age ~ I(L(Height) - R(Height))  ,
  FEVenv
)
has.intercept(FEVform)
FEVform2 <- new.pim.formula(Age ~ Height + 1, FEVData)
has.intercept(FEVform2)
```

has.specials *Extract information from a pim.formula object*

Description

This group of functions provides an easy way to extract the extra information saved in a pim.formula object. Take a look at the help page of pim.formula for more information.

Usage

```
has.specials(x)

## S4 method for signature 'pim.formula'
has.specials(x)

terms(x, ...)

## S4 method for signature 'pim.formula'
terms(x)

lhs(x)

## S4 method for signature 'pim.formula'
lhs(x)
```

Arguments

x	an object of the class pim.formula
...	arguments passed to other methods

Value

has.specials(): a single TRUE or FALSE value indicating whether the formula right-hand side contains any special functions.

terms(): the terms object of the pim.formula object

lhs(): an object of class call containing the left hand side of the formula as used in the pim.

See Also

the class pim.formula-class

response for extracting the pseudoresponse variable, model.matrix for extracting the design matrix of pseudo-observations, formula for extracting the pim.formula and penv for extracting the pim environment.

Examples

```
data("FEVData")
# Create the "model frame"
FEVenv <- new.pim.env(FEVData, compare="unique")

# create the formula and bind it to the pim.environment.
FEVform <- new.pim.formula(
  Age ~ I(L(Height) - R(Height))  ,
  FEVenv
)
lhs(FEVform)
has.specials(FEVform)
penv(FEVform)

FEVform2 <- new.pim.formula(
  FEV ~ Height*Sex,
  FEVenv
)

has.specials(FEVform2)
terms(FEVform2)
```

---

is.complete *Check whether a pim environment is complete*

Description

Objects of class pim.environment can be created with or without a poset. To check whether an object has a poset included, you use the function is.complete

Usage

```
is.complete(x)
```

Arguments

x	an object of class pim.environment

Value

a single value TRUE or FALSE

Note

This function is not written as an S4 method. Might be rewritten to S4 later on.

Examples

```
# the constructor returns an empty environment without poset
is.complete(new.pim.env())

# Constructing a pim environment with a poset
data("FEVData")
FEVenv <- new.pim.env(FEVData, compare="unique")
is.complete(FEVenv)
```

L *Specify the left hand and right hand side of an expression used in pims*

Description

These functions allow you to specify the left hand side and right hand side of a term in a pim model. The user should only use this functions within a formula using the pim function. Use in a different context will return an error.

Usage

```
L(x)

R(x)

PP(x)
```

Arguments

x	any vector specified in a formula

Details

These specific functions are actually not used by the function pim. pim calls the internal function .make.posfun to create the actual functions L and R to work with the specified posets of the model of interest.

The actual functions used by pim are saved in a specific environment, a pim.environment, which resides in the pim-class object returned by pim. This way of working is chosen in order to avoid unnecessary copying of data.

The function PP serves simply as short for R(x) - L(x). If used outside the context of a pim model, it will generate multiple warnings (see section warning).

Value

a vector with the pseudo-observations for x, based on the poset used to create the function. If used in a wrong context (i.e. not in a call to pim), it returns x unchanged and throws a warning.

warning

These functions serve only as placeholder. During the fitting process of a pim, they get updated to include the posets (the indices that determine which observations are compared) Note that this makes the functions behave fundamentally different from what you would expect R. The result of these functions depends on the context in which they are called.

Examples

```
## Not run:
pim(income~(L(foodexp) - R(foodexp)), data=Engeldata)
L(1:10) # Gives a warning

## End(Not run)
```

`MHData` *This is the Mental health data*

Description

This is the Mental health data

Details

These are the columns and their meanings

- `mental` Mental impairment (MI)
- `ses` Socioeconomic status
- `life` Life index (LI)

`model.matrix.pim` *Create a model matrix for a probabilistic index model*

Description

This function creates a model matrix for use in a probabilistic index model. This model matrix can be passed to `pim.fit`.

Usage

```
model.matrix(object, ...)

## S4 method for signature 'pim'
model.matrix(object, data, ...)

## S4 method for signature 'pim.formula'
model.matrix(
  object,
  data,
  model = c("difference", "marginal", "regular", "customized"),
  ...
)
```

Arguments

object	a pim.formula object that contains the formula necessary for constructing the model matrix.
...	extra arguments passed to or from other methods. This is currently only implemented in concordance with the generic model.matrix function.
data	an optional argument specifying the data frame for which the model matrix should be constructed. See also model.matrix) in the stats package.
model	a single character value with possible values "difference" (the default), "marginal", "regular" or "customized". See also pim.

Value

a design matrix for a pim model

Examples

```
data("FEVData")
# Create the "model frame"
FEVenv <- new.pim.env(FEVData, compare="unique")
# This includes the poset
pos <- poset(FEVenv, as.list=TRUE)

# create the formula and bind it to the pim.environment.
FEVform <- new.pim.formula(
  Age ~ I(L(Height) - R(Height))  ,
  FEVenv
)

# Use this formula object to construct the model matrix
# use the default model ( difference )
MM <- model.matrix(FEVform)

# Use this formula object to construct the pseudo response
Y <- response(FEVform)
```

```
# Now pim.fit can do what it does
res <- pim.fit(MM,Y, estim = "estimator.glm", penv=FEVenv)
```

`new.pim` *Create an object of class pim*

Description

This function is the constructor for an object of class pim. It is nothing but a placeholder for `new("pim", ...)`. This function is not exported.

Usage

```
new.pim(...)
```

Arguments

`...`	Data to include in the new object. See `new`

Value

an object of class `pim`. See `pim-class`

`new.pim.env` *Constructor for a pim.environment*

Description

This functions serves as a constructor for an object of the class `pim.environment`. In most cases, calling this function directly is not necessary.

Usage

```
new.pim.env(data, ...)

## S4 method for signature 'missing'
new.pim.env(data, ...)

## S4 method for signature 'environment'
new.pim.env(
  data,
  compare = "unique",
  env = parent.frame(),
  vars = NULL,
```

```
  classes = NULL,
  ...
)

## S4 method for signature 'list'
new.pim.env(data, compare = "unique", vars = NULL, ...)

## S4 method for signature 'data.frame'
new.pim.env(data, compare = "unique", vars = NULL, ...)

## S4 method for signature 'ANY'
new.pim.env(data, ...)
```

Arguments

data	a data frame, a list or an environment containing the data for a probabilistic index model.
...	extra parameters for construction of the poset, like the argument compare from new.pim.poset.
compare	a character vector, matrix or list that defines how the set of pseudo observations (poset) should be constructed. if set to NULL, no poset is constructed. See also new.pim.poset for more information on how to specify a custom poset.
env	an environment that is the parent environment of the object.
vars	An optional character vector with the names of the variables that should be included in the pim environment. Note that the variable names should be found in the object passed to argument data.
classes	An optional character vector with the classes of the variables in the environment, given in the same order as the argument data.names.

Details

This function is called during the preparation of the model matrix for a pim. The resulting object is used to evaluate the formula of a pim, and stores information on how this is done.

Note that the parent of the environment is actually the pim.poset object in the poset slot. The parent you set using the env argument, is the parent of the pim.poset object. This ensures that when a formula is evaluated in the pim.environment it will use a suitable search path to find all functions and objects.

Value

an object of the class pim.environment

Examples

```
new.pim.env() # Creates an empty object

# Starting from a data frame
data(DysData)
```

```
env1 <- new.pim.env(DysData)

env2 <- new.pim.env(DysData, compare=NULL)
poset(env2)
env3 <- new.pim.env(DysData, compare="all")
poset(env3)


data(FEVData)
env4 <- new.pim.env(FEVData, vars=c('Age','Sex'))
ls(env4)
```

`new.pim.formula` *Constructor for pim.formula*

Description

This function reworks a formula to a pim.formula for use in a probabilistic index model. This function is only meant to be used internally, but is exported. It should be used only in concordance with `model.matrix.pim`

Usage

```
new.pim.formula(formula, data, ...)

## S4 method for signature 'formula,pim.environment'
new.pim.formula(formula, data, ...)

## S4 method for signature 'formula,ANY'
new.pim.formula(formula, data, ...)
```

Arguments

`formula`	a formula object
`data`	either a `pim.environment` object containing the data for the pim, or an object that can be converted to a `pim.environment` by `new.pim.env`
`...`	extra arguments to `new.pim.env`

Details

It is the constructor to be used for a `pim.formula` object, and should only be used in conjunction with `model.matrix.pim` and `pim.fit` as shown in the examples.

Value

a `pim.formula` object.

See Also

pim.formula-class for more information on the class itself. PO, L and R for some functions that can be used in a pim.formula

#' @examples data("FEVData") # Create the "model frame" FEVenv <- new.pim.env(FEVData, compare="unique") # This includes the poset pos <- poset(FEVenv, as.list=TRUE)

create the formula and bind it to the pim.environment. FEVform <- new.pim.formula(Age ~ I(L(Height) - R(Height)) , FEVenv)

Use this formula object to construct the model matrix # use the default model (difference) MM <- model.matrix(FEVform)

Use this formula object to construct the pseudo response Y <- response(FEVform)

Now pim.fit can do what it does res <- pim.fit(MM,Y, estim = "estimator.glm", penv=FEVenv)

---

new.pim.poset *Create a pim.poset environment*

Description

This function allows you to create a pim.poset environment that can be added to a pim.environment object. You can use this function to create a custom poset, but in general it's safer to use the relevant arguments of the pim function. That way more safety checks are carried out.

Usage

```
new.pim.poset(compare, nobs, parent = parent.frame(), ...)

## S4 method for signature 'character,numeric'
new.pim.poset(compare, nobs, parent = parent.frame(), ...)

## S4 method for signature 'matrix,numeric'
new.pim.poset(compare, nobs, parent = parent.frame(), ...)

## S4 method for signature 'list,numeric'
new.pim.poset(compare, nobs, parent, comp.value = "custom", ...)

## S4 method for signature 'matrix,missing'
new.pim.poset(compare, nobs, parent = parent.frame(), ...)

## S4 method for signature 'list,missing'
new.pim.poset(compare, nobs, parent = parent.frame(), ...)

## S4 method for signature 'missing,numeric'
new.pim.poset(compare, nobs, parent = parent.frame(), ...)
```

Arguments

compare	A character value, matrix or list indicating how the poset should be constructed. Defaults to the default value of create.poset. See Details section for more information.
nobs	An integer value determining the number of observations this poset is created for. If compare is not a character value, the number of observations
parent	An optional environment that serves as the parent for the pim.poset environment. By default this is the environment from which the function is called. Note that for a correct functioning, the parent environment should be set to the pim.environment this object is part of. This is done automatically by the function add.poset.
...	arguments passed to other methods.
comp.value	a character value to be used as value for the compare slot of the object. Defaults to 'custom' and should be left at the default without a very good reason to change it.

Details

A poset (or pseudo observation set) in the context of probabilistic index models is a set of indices that determines which observations are compared with one another. It is used to construct the pseudo-observations on which the model is fitted. You can think of a poset as a "pseudo-observation set".

The most convenient way to use this function, is by specifying a character value for the argument compare. The value "unique" creates a poset in such a way that only unique combinations of two observations are used in the model. The value "all" creates all possible L-R combinations between the observations.

If you want to define the poset yourself, you can pass either a matrix or a list with 2 elements as value for the argument compare. Columns of the matrix or elements of the list should either be named "L" and "R", or be unnamed. When unnamed, the function takes the first column/element as the left poset, and the second column/element as the right poset. If the (col)names are anything else but "L" and "R", these names are ignored and the first column is seen as "L".

Value

an pim.poset object that can be used to replace the poset in a pim environment.

Note

You can omit the argument compare if you supply a value for nobs. You can also omit the argument nobs if you provide a matrix or list as value for compare. The function will try to deduct the number of observations from the highest index value present in the matrix/list

You can't omit both arguments together though, as the function needs at least some information on the number of observations the poset is designed for.

Warning

Changing the value of comp.value by hand might result in errors or a wrongly fitted model. The argument exists for internal purposes and possible extensions later on, but should not be used.

See Also

add.poset for more information on how to adapt the poset of a pim.environment object.

Examples

```
mypos <- new.pim.poset('unique',n=10) # creates empty environment
ls(mypos)
# Using the created poset functions L and R
# Note this is purely as illustration, this makes no sense
# in the context of a pim analysis.
mypos$L(1:10)
mypos$R(1:10)
```

---

nobs,pim.environment-method

Extract the number of observations

Description

This function extracts the number of observations in an object of class pim.environment, or the number of observations for which a pim.poset is constructed. If applied to a matrix or data.frame, it returns the number of rows. For any other object it does the same as length.

Usage

```
## S4 method for signature 'pim.environment'
nobs(object)

## S4 method for signature 'pim.poset'
nobs(object)

## S4 method for signature 'matrix'
nobs(object)

## S4 method for signature 'data.frame'
nobs(object)
```

Arguments

object	an object of the class pim.environment or pim.poset

Details

This package imports the generic nobs from the package stats4.

Value

In case the function is called on a `pim.environment` or a `pim.poset` object, an integer with the number of (foreseen) observations. If the pim.environment is empty, it returns `0`.

In all other cases, it returns the output of either `nrow` (for matrices and data.frames) or `length`.

P *Probability function*

Description

This functions transform a comparison or otherwise logical value to a numeric value for use in a pim.

Usage

```
P(x)

PO(x, y = NULL)
```

Arguments

x	for `P`, a logical value. For `PO` a numeric value.
y	a numeric value or `NULL`. If `NULL`, the function will try to calculate `PO(L(x),R(x))`, provided the functions `L` and `R` are defined correctly. This is the case when `PO` is used in the context of a probabilistic index model fitted with `pim`.

Details

These functions are constructed purely for notation. `P` is completely equivalent to `as.numeric`, apart from an extra control to check whether it actually makes sense to do so. The function `PO` is just short for `P(x < y) + 0.5*P(x == y)`

Value

A numeric value of 0, 0.5 or 1. 1 if x < y, 0.5 if x == y and 0 if x > y

See Also

`pim` and `pim.formula` for more information on how this is used inside a pim context.

Examples

```
# Check in pim
```

penv *Extract a pim environment from a model or formula*

Description

This function allows you to extract the pim.environment object from either a pim object or a pim.formula object.

Usage

```
penv(x)

## S4 method for signature 'pim.formula'
penv(x)

## S4 method for signature 'pim'
penv(x)
```

Arguments

x	either a pim or a pim.formula object

Value

In case of a pim object, the pim.environment contained therein. In case of a pim.formula object, the environment itself. See the help page pim.formula-class.

Examples

```
data("FEVData")
# Create the "model frame"
FEVenv <- new.pim.env(FEVData, compare="unique")

# create the formula and bind it to the pim.environment.
FEVform <- new.pim.formula(
  Age ~ I(L(Height) - R(Height))  ,
  FEVenv
)
theEnv <- penv(FEVform)
ls(theEnv)

themodel <- pim(Age ~ Height, FEVenv)
thePEnv <- penv(themodel)
thePEnv

ls(thePEnv)
# Note that this is a different environment, and that it only contains
# the variables in the formula, contrary to the environment created
# by new.pim.formula
```

pim *Fitting a Probabilistic Index Model*

Description

This function fits a probabilistic index model, also known as PIM. It can be used to fit standard PIMs, as well as many different flavours of models that can be reformulated as a pim. The most general models are implemented, but the flexible formula interface allows you to specify a wide variety of different models.

Usage

```
pim(
  formula,
  data,
  link = c("logit", "probit", "identity"),
  compare = if (model == "marginal") "all" else "unique",
  model = c("difference", "marginal", "regular", "customized"),
  na.action = getOption("na.action"),
  weights = NULL,
  keep.data = FALSE,
  ...
)
```

Arguments

formula	An object of class formula (or one that can be coerced to that class): A symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment that contains the variables in the model. Objects that can be coerced by as.data.frame can be used too.
link	a character vector with a single value that determines the used link function. Possible values are "logit", "probit" and "identity". The default is "logit".
compare	a character vector with a single value that describes how the model compares observations. It can take the values "unique" or "all". Alternatively you can pass a matrix with two columns. Each row represents the rownumbers in the original data frame that should be compared to eachother. See Details.
model	a single character value with possible values "difference" (the default), "marginal", "regular" or "customized". If the formula indicates a customized model (by the use of L() or R()), this parameter is set automatically to "customized". Currently, only the options "difference", "marginal" and "customized" are implemented.
na.action	the name of a function which indicates what should happen when the data contains NAs. The default is set by the na.action setting of options, and is na.fail when unset.

weights	Currently not implemented.
keep.data	a logical value indicating whether the model matrix should be saved in the object. Defaults to FALSE. See Details.
...	extra parameters sent to pim.fit

Details

PIMs are based on a set of pseudo-observations constructed from the comparison between a range of possible combinations of 2 observations. We call the set of pseudo observations *poset* in the context of this package.

By default, this poset takes every unique combination of 2 observations (compare = "unique"). You can either use a character value, or use a matrix or list to identify the set of observation pairs that have to be used as pseudo-observations. Note that the matrix and list should be either nameless, or have the (col)names 'L' and 'R'. If any other names are used, these are ignored and the first column/element is considered to be 'L'. See also new.pim.poset.

It's possible to store the model matrix and psuedo responses in the resulting object. By default this is not done (keep.data = FALSE) as this is less burden on the memory and the pim.formula object contains all information to reconstruct both the model matrix and the pseudo responses. If either the model matrix or the pseudo responses are needed for further calculations, setting keep.data to TRUE might reduce calculation time for these further calculations.

Value

An object of class pim. See pim-class for more information.

The enhanced formula interface

In case you want to fit a standard PIM, you can specify the model in mostly the same way as for lm. There's one important difference: a PIM has by default no intercept. To add an intercept, use + 1 in the formula.

Next to this, you can use the functions L and R in a formula to indicate which part of the poset you refer to. Remember a poset is essentially a matrix-like object with indices refering to the pseudo-observations. Using L() and R() you can define exactly how the pseudo-observations fit in the model. Keep in mind that any calculation done with these functions, has to be wrapped in a call to I(), just like you would do in any other formula interface.

You don't have to specify the model though. If you choose the option model = 'difference', every variable in the formula will be interpreted as I(R(x) - L(x)). If you use the option model = 'marginal', every variable will be interpreted as R(X).

If you don't specify any special function (i.e. L, R, P or PO), the lefthand side of the formula is defined as PO(y). The function PO calculates pseudo observations; it is 1 if the value of the dependent variable for the observation from the L-poset is smaller than, 0 if it is larger than and 0.5 if it is equal to the value for value from the R-poset (see also PO)

See Also

pim-class for more information on the returned object, pim.fit for more information on the fitting itself, pim-getters, coef, confint, vcov etc for how to extract information like coefficients, variance-covariance matrix, ..., summary for some tests on the coefficients.

Examples

```
data('FEVData')
# The most basic way to use the function
Model <- pim(FEV~ Smoke*Sex , data=FEVData)

# A model with intercept
# The argument xscalm is passed to nleqslv via pim.fit and estimator.nleqslv
# By constructing the estimator functions wisely, you can control most of
# the fitting process from the pim() function.
data('EngelData')
Model2 <- pim(foodexp ~ income + 1, data=EngelData,
   compare="all",
   xscalm = 'auto')

# A marginal model
# It makes sense to use the identity link in combination with the
# score estimator for the variance-covariance matrix
data('DysData')
Model3 <- pim(SPC_D2 ~ out, data = DysData,
  model = 'marginal', link = 'identity',
  vcov.estim = score.vcov)

# A Model using logical comparisons, this is also possible!
# Model the chance that both observations have a different
# outcome in function of whether they had a different Chemo treatment
Model6 <- pim(P(L(out) != R(out)) ~ I(L(Chemo) != R(Chemo)),
   data=DysData,
   compare="all")

# Implementation of the friedman test in the context of a pim
# warpbreaks data where we consider tension as a block
# To do so, you provide the argument compare with a custom
# set of comparisons
data(warpbreaks)
wb <- aggregate(warpbreaks$breaks,
                by = list(w = warpbreaks$wool,
                          t = warpbreaks$tension),
                FUN = mean)
comp <- expand.grid(1:nrow(wb), 1:nrow(wb))
comp <- comp[wb$t[comp[,1]] == wb$t[comp[,2]],] # only compare within blocks
m <- pim(x ~ w, data = wb, compare = comp, link = "identity",  vcov.estim = score.vcov)
summary(m)
friedman.test(x ~ w | t, data = wb)
## Not run:
# This illustrates how a standard model is actually built in a pim contex
Model4 <- pim(PO(L(Height),R(Height)) ~ I(R(Age) - L(Age)) + I(R(Sex) - L(Sex)),
data=FEVData,
estim = "estimator.BB")
# is the same as
Model5 <- pim(Height ~ Age + Sex, data = FEVData, estim = "estimator.BB")
summary(Model4)
summary(Model5)
```

```
## End(Not run)
```

`pim-class` *Class pim*

Description

This class contains the fitting information resulting from a call to `pim`.

Slots

`formula` The `pim.formula` object used in the fit

`coef` a numeric vector with the fitted coefficients

`vcov` a numeric matrix containing the variance-covariance matrix of the fitted coefficients

`penv` a `pim.environment` object containing the data used to fit this

`fitted` a numeric vector containing the raw fitted

`link` a character vector describing the used link function

`estimators` a list with the elements `coef` and `vcov`, containing either a character value with the name of the used estimator, or the function itself.

`model.matrix` If `keep.data` is set to `TRUE` while calling `pim` the original model matrix. Otherwise an empty matrix with 0 rows and columns.

`response` If `keep.data` is set to `TRUE` while calling `pim` the original response vector. Otherwise an empty numeric vector.

`keep.data` a logical value indicating whether the original data is kept in the object. This is set using the argument `keep.data` of the function `pim`. `model` a character value with the value "difference", "marginal", "regular" or "customized", indicating which type of pim model has been fitted.

`pim-getters` *Getters for slots of a pim object*

Description

Getters for slots of a pim object

Usage

```
keep.data(x)

fitted(object, ...)

## S4 method for signature 'pim'
fitted(object, ...)
```

Arguments

x	a pim object
object	a pim object
...	arguments passed to other methods. Currently ignored.

Value

keep.data(): a single logical value indicating whether the model matrix and pseudo responses were stored in the pim object.

fitted(): a numeric vector with the fitted values for the pseudo-observations.

Examples

```
data('FEVData')
themodel <- pim(FEV ~ Age + Height, data = FEVData)
keep.data(themodel)
fitted(themodel)
```

---

pim.environment-class *The pim.environment class*

Description

This S4 class inherits from the S3 class environment. The environment serves as a container to hold the data, poset and the poset related functions of a probabilistic index model generated by the function pim. The objects of this class behave much like an environment, but contain some extra slots with information on the objects inside the environment.

Slots

poset an environment of class pim.poset containing the poset-related functions (normally these are L and R). This environment has the object itself as parent.

data.names a character vector containing the names of the vectors that represent the data

nobs integer value indicating the number of observations in the environment

classes a _named_ list containing the classes of the objects inside the environment. Note that the value should be the one given by class.

is.complete a logical value indicating whether or not the poset was added before.

Note

This class is not exported, so it can't be extended as for now. Although it is possible to use the function new for creation of new instances, users are strongly advised to use the function new.pim.env in case they need to manually create a new instance of the class pim.environment.

pim.fit *Fitter function for a probabilistic index model*

Description

This is the basic computing engine called by pim to get the estimates for the coefficients and the variance- covariance matrices. This function currently only spits out these components using the sandwich estimators.

Usage

```
pim.fit(
  x,
  y,
  link = "logit",
  estim = "estimator.nleqslv",
  start = rep(0, ncol(x)),
  vcov.estim = "sandwich.vcov",
  weights = NULL,
  penv,
  ...
)
```

Arguments

x	a model matrix with as many rows as y.
y	a vector with the pseudo-responses
link	a character vector with a link function
estim	a character vector or a function indicating the solver to be used for estimating the coefficients. By default this is the function nleqslv. Other possibilities are given in the help page on estimators.
start	A numeric vector with the starting values for the fitting algorithm, if required.
vcov.estim	a function to determine the variance-covariance matrix. Possibilities are sandwich.vcov and link{score.vcov}. Defaults to sandwich.vcov
weights	currently not implemented
penv	An environment, pim.environment or pim.poset object containing the poset functions. Alternatively this can be a list of two numeric vectors, containing the poset indices for the left and right side of the pim.
...	Further arguments that need to be passed to the estimation function. The most relevant is construct, allowing you to write your own score function for the numerical optimization. See also estimators

Value

A list with the following elements

coefficients a numeric vector with the coefficients

vcov a numeric matrix with the variance-covarianc matrix for the coefficients

fitted a numeric vector with the raw fitted values

estim a list with two components named `coef` and `vcov` containing information on the used estimators for both.

See Also

`model.matrix` for how to construct a valid model matrix for a pim, `pim` for the general user interface

---

`pim.formula-class` *Class pim.formula*

Description

This class contains information on the formula passed in a call to `pim`. The object is used to create the model matrix of a pim (see `model.matrix`)

Details

Although a future version of this package will include the possibility to fit survival models, this is currently not implemented. If the `pim` function encounters special functions on the left-hand side (i.e. when `has.lhs.fun` is `TRUE`), the model won't be calculated.

The slot `penv` contains a reference to an environment In most cases, this will be the environment contained in a `pim.environment` object. Note though that the `pim.formula` object only contains a link to the environment. The extra slots contained in the `pim.environment` object are NOT copied to the `pim.formula`. Also keep in mind that the environment linked to the `pim.environment` object will continue to exist even after deleting the `pim.environment` itself, and this for as long as the `pim.formula` object exists.

This class is not exported and hence cannot be extended. It serves internal use in the pim package only.

Slots

`terms` a `terms.object` derived from the formula

`has.specials` a logical value indicating whether the right-hand side of the original formula contains special functions like `L` and `R`

`has.lhs.fun` a logical value indicating whether the left-hand side of the original formula contains special functions. These exclude the functions `P` and `PO` but include functions like `Surv`. See Details

`predictors` a character vector with the names of all the variables mentioned in the right-hand side of the formula.

`response` an character vector with the name of the response variable.

`lhs` a call with the processed left-hand side of the formula

`orig` a formula object with the original formula

`penv` an `environment` object to which the formula is related (i.e. the environment containing possible `L` and `R` function definitions.) See Details.

`has.intercept` a logical value indicating whether the formula has an explicit intercept (indicated by + 1)

Note

This class is not exported, so it can't be extended as for now. Although it is possible to use the function `new` for creation of new instances, users are strongly advised to use the function `new.pim.formula` in case they need to manually create a new instance of the class `pim.formula`.

---

pim.poset-class *The pim.poset class*

Description

The pim.poset class is an S4 class that inherits from `environment` and contains the poset functions for a `pim`. It's a class used internally and should not be adapted by the user. The correct interpretation of the formula is dependent on this object. The object mainly functions as a slot in object of class `pim.environment`.

Slots

`compare` a character value with the type of poset. This can take the values "unique", "all" and "custom".

`nobs` an integer value describing the number of observations for which this poset is meant to be used.

Note

The pim.poset class doesn't really make sense to be used on itself. It is part of the `pim.environment` class and shouldn't be used outside this context.

pim.summary-class *Class pim.summary*

Description

This class contains the summary information from a probabilistic index model , and is created by using the function summary on an object of the pim-class.

Details

The class pim.summary can be treated like a matrix to get out the coefficients, standard errors, Z values and/or p values.

Slots

formula contains an object of the class pim.formula containing the model fitted.

model a character vector describing the type of model. See also the argument model of the function pim

link a character value that contains the link. See also the argument link of the function pim

coef a numeric vector with the coefficients

se a numeric vector with the standard errors for the coefficients

zval a numeric vector containing the Z values for the coefficients, testing whether the coefficient differs significantly from 0.

pr a numeric vector containing the related p-values for the coefficients.

h0 a numeric value or a numeric vector containing the null hypothesis. See the argument at summary.pim

See Also

pim for more info on how to construct the model and summary.pim for the constructor.

pimdata *The data contained in the pim package*

Description

The pim package contains different datasets for use in examples and tests. Currently, you find the datasets DysData, EngelData and FEVData. More information can be found on the respective help pages.

Details

The data contained in the package has following structures

- EngelData: A single numeric predictor variable and a response
- FEVData: A data frame with a numeric response variable and 4 additional numeric predictor values.
- DysData: A dataframe with 3 factors and a numeric variable as predictors. The outcome is a factor with 4 levels.
- SNP_XRCC1__77 Genotype of this SNP. A factor with three levels: "TT", "TC" and "CC"
- SPC_D2 Dose of radiation that reached 2
- SNP_XRCC1__77TC 1 if SNP_XRCC1__77 is "TC", 0 otherwise

---

poset *Extract the poset as a matrix or list*

Description

This function allows you to extract the poset from either a pim.environment or a pim.poset object. The poset can be extracted as a matrix or a list.

Usage

```
poset(x, ...)

## S4 method for signature 'pim.environment'
poset(x, ...)

## S4 method for signature 'pim.poset'
poset(x, as.list = FALSE)

## S4 method for signature 'environment'
poset(x, as.list = FALSE)

## S4 method for signature 'pim'
poset(x, ...)

## S4 method for signature 'pim.formula'
poset(x, ...)
```

Arguments

x	an object of class pim.environment, pim.formula, pim or pim.poset, or an environment derived from either object.
...	arguments passed to other methods. Currently ignored.
as.list	a logical value indicating whether the poset should be returned as list or as a matrix. Defaults to FALSE, which returns a matrix

Value

When x contains a poset, either a matrix or a list (when `as.list` is `TRUE`) with the indices that make up the poset. If there's no poset, the function returns a missing value.

The returned matrix hax 2 columns, each named after the respective poset function. In case a list is requested, the function returns a named list with 2 elements, each element containing the indices related to the poset function of the same name (either `L` or `R`).

Examples

```
data(DysData)
DysPimEnv <- new.pim.env(DysData)
poset(DysPimEnv)
```

`print` *Print methods for the different object types*

Description

Printing `pim`, `pim.environment`, `pim.formula` and `pim.poset` objects.

Usage

```
print(x, ...)

## S4 method for signature 'pim'
print(x, digits = max(3L, getOption("digits") - 3L), show.vcov = FALSE, ...)

## S4 method for signature 'pim.environment'
print(x, digits = max(3L, getOption("digits") - 3L), n = 6L, ...)

## S4 method for signature 'pim.poset'
print(x, digits = max(3L, getOption("digits") - 3L), n = 6L, ...)

## S4 method for signature 'pim.formula'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	the object
`...`	arguments passed to other methods. Currently ignored
`digits`	an integer that defines the number of digits printed
`show.vcov`	a logical value indicating whether the variance- covariance matrix should be shown or not. Defaults to `FALSE`
`n`	number of observations shown by `print`

Value

invisible NULL

Examples

```
data(FEVData)
Model <- pim(FEV~ Smoke*Sex , data=FEVData)
print(Model)
print(penv(Model))
# You get the drift
```

`response` *Extract response from a pim.formula or a pim object*

Description

This function extracts the response from a `pim.formula` for use in `pim.fit`.

Usage

```
response(object)

## S4 method for signature 'pim.formula'
response(object)

## S4 method for signature 'pim'
response(object)
```

Arguments

`object`	an object of class pim or pim.formula.

Value

The response variable with pseudo-observations for a pim.

See Also

`pim-class` and `pim.formula-class` for more information on the classes, and `pim`, `pim.fit` and `pim.formula` for more information on related functions.

Examples

```
data('FEVData')
Model <- pim(FEV~ Smoke*Sex , data=FEVData)
response(Model)

# In pieces
FEVenv <- new.pim.env(FEVData, compare="unique")
FEVform <- new.pim.formula(
  Age ~ I(L(Height) - R(Height))  ,
  FEVenv
)
response(FEVform)
```

`sandwich.estimator` *Pseudo-observation variance sandwich estimator*

Description

The functions described here all implement an estimator for the variance of the coefficients. This function is not exported.

Usage

```
sandwich.estimator(
  U,
  U.diff,
  g1,
  g2,
  shared.factor = 1,
  switched.factor = 1,
  self.factor = 1
)
```

Arguments

`U`	See the formula for sandwich estimator: holds `U_{ij}`
`U.diff`	See the formula for sandwich estimator: holds the partial derivatives of `U`.
`g1, g2`	Index in the original observations of the "left" and "right" part of the pseudo-observations.
`shared.factor`	Factor by which all `UijUik` or `UijUlj` will be multiplied
`switched.factor`	Factor by which all `UijUki` or `UijUjl` will be multiplied
`self.factor`	Factor by which all `UijUij` or `-UijUji` will be multiplied

Value

The matrix of the sandwich estimator

SUData *This is the Surgical unit data*

Description

This is the Surgical unit data

Details

These are the columns and their meanings

- `EnT` Enzyme function test score
- `Gender` Gender of the patient (0: male, 1: female)
- `Alcohol` History of alcohol use (0: none, 1: moderate, 2: severe)
- `SurvivalTime` Survival time of each patient (the outcome of interest)

summary.pim *The summary function for the pim class*

Description

The function `summary` is a generic function. We provide a method for objects of the `pim-class`.

Usage

```
summary(object, ...)

## S4 method for signature 'pim'
summary(object, h0 = 0, ...)
```

Arguments

`object`	an object of the class pim.
`...`	arguments passed to other methods. Currently ignored.
`h0`	a numeric value or a vector as long as the number of coefficients with the value that defines the null hypothesis to test against

Value

a `pim.summary` object

Examples

```
data(FEVData)
Model <- pim(FEV~ Age + Smoke*Sex , data=FEVData)
summary(Model)
```

vcov *Methods for vcov*

Description

This package defines an S4 generic for vcov and methods for list and pim classes.

Usage

```
vcov(object, ...)

## S4 method for signature 'pim'
vcov(object, ...)

## S4 method for signature 'list'
vcov(object, ...)
```

Arguments

object	any object.
...	arguments passed to other methods. Currently ignored

Value

the variance-covariance matrix

See Also

vcov in the stats package.

Examples

```
data(FEVData)
Model <- pim(FEV~ Age + Smoke*Sex , data=FEVData)
vcov(Model)
```

vcov.estimators *vcov estimators for pim*

Description

sandwich.vcov and score.vcov are two similar estimators for the variance-covariance matrix of a probabilistic index model. These functions are meant to be used within a call to pim as a value for the argument vcov.

Usage

```
sandwich.vcov(fitted, X, Y, W, link, poset, ...)

score.vcov(fitted, X, Y, W, link, poset, ...)
```

Arguments

fitted	The fitted values (calculated as X %*% coef with X the design matrix and coef the coefficients)
X	the design matrix
Y	a numeric vector with pseudoresponses
W	a numeric vector with weights. If weights are not applicable, set to NULL (the default)
link	a character vector with the link function
poset	a list with the left and right indices. See poset for more information.
...	arguments passed to downstream methods.

Details

You can create your own estimating functions for the variance-covariance matrix. To do so, you have to make sure that your function allows for the exact same arguments. As the function pim.fit calculates the fitted values already, there's no need to incorporate the calculation of these inside the function.

Value

the variance-covariance matrix

Note

You should only use score.vcov in combination with an identity link

See Also

sandwich.estimator for more information on the actual fitting process. pim for a few examples in how these are used

vcov.internal *Internal functions for vcov estimation*

Description

These functions serve as preparation functions to calculute the variance- covariance matrix of a pim using any of the vcov.estimators provided in this package. The result of these preparation functions is used by the sandwich.estimator and score.estimator functions respectively.

Usage

```
U.sandwich(Zbeta, Z, Y, link, W = NULL)

U.score(Zbeta, Z, Y, link, W = NULL)
```

Arguments

Zbeta	fitted values
Z	design matrix
Y	pseudo responses
link	character vector with link function
W	vector with weights

Note

These functions should NOT be called by the user

Index