Package 'sivipm'

April 26, 2019

Version 1.1-4.2

Date 2016-01-13

Title Sensitivity Indices with Dependent Inputs

Author A. Bouvier [aut], J.-P. Gauchi [aut, cre], E. Volatier [ctb]

Maintainer ORPHANED

Depends R(>= 3.6.0), methods

Imports stats, graphics, utils, seqinr

Description Sensitivity indices with dependent correlated inputs, using a method based on PLS regression.

License GPL (>= 2)

Encoding UTF-8

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-26 13:21:19 UTC

X-CRAN-Original-Maintainer Annie Bouvier <annie.bouvier@jouy.inra.fr>

X-CRAN-Comment Orphaned on 2018-06-09 as maintainer has retired and her email address now bounces.

R topics documented:

`sivipm-package` *Sensitivity Indices with Dependent Inputs*

Description

Compute total and individual sensitivity indices, significant components, and confidence intervals for the total sensitivity indices.

Details

The total and individual sensitivity indices are calculated using a method based on the VIP of the PLS regression, proposed by J.P. Gauchi. The significant components are calculated by the SIMCA software rule and by the Lazraq & Cléroux test. The confidence intervals for the total sensitivity indices are determined by the bootstrap method. A fast algorithm is implemented which allows big datasets analysis.

Author(s)

A. Bouvier, J.-P. Gauchi, E. Volatier

Maintainer: Annie Bouvier <Annie.Bouvier@jouy.inra.fr>

References

- Gauchi, J.-P. and Lehuta, S. and Mahévas, S. 2010. Optimal sensitivity analysis under constraints: Application to fisheries. In Procedia - Social and Behavioral Sciences, vol. 2. Elsevier, pp. 7658-7659. Sixth International Conference on Sensitivity Analysis of Model Output (Milan, Italy), 19-22 July; co-organized by the ELEUSI research center of Bocconi University and by the Joint Research Center of the European Commission.
- Gauchi, J.-P. 2012. Global Sensitivity Analysis: The SIVIP method (SAS/IML language). Rapport technique 2012-3. INRA, UR1404, F-78350 Jouy-en-Josas, France.
- Gauchi, J.-P. 2015. A practical method of global sensitivity analysis under constraints. Rapport technique 2015-1. INRA, UR1404, F-78350 Jouy-en-Josas, France.
- Lazraq, A. and Cléroux, R. 2001. The PLS multivariate regression model: testing the significance of successive PLS components. Journal of Chemometrics. Vol. 15(6), pp 523-536.
- Lazraq, A. and Cléroux, R. and Gauchi, J.-P. 2003. Selecting both latent and explanatory variables in the PLS1 regression model. Chemometrics and Intelligent Laboratory Systems, Vol. 66(2), pp 117-126. Elsevier. doi:10.1016/S0169-7439(03)00027-3.
- SIMCA Software. http://www.umetrics.com/products/simca.

bind.polyX *Bind Several polyX Objects*

Description

Bind several objects of class polyX.

Usage

```
## S4 method for signature 'polyX'
bind(x, ...)
```

Arguments

x	Object of class polyX.
...	Objects of class polyX to be binded to x. As many as required. It is assumed that they have been built on the same dataset of inputs.

Value

An object of class polyX, coding for a polynomial including the monomials of all the polynomials in the argument list. Its degree is the maximal degree of these polynomials. Duplicated monomials are removed.

See Also

takeoff.polyX

Examples

```
# Create first polynomial
P <- vect2polyX(cornell0[,1:3],c("1","2","3", "3*3*3", "3*3"))
# Create second polynomial
P2 <- vect2polyX(cornell0[,1:3], c("1","2","3", "2*3"))
# Bind them: PP=X1 + X2 + X3 + X3*X3*X3 + X3*X3 + X2*X3
PP <- bind.polyX(P, P2)
```

cornell0 *Motor Octane Number Measurements*

Description

Motor octane number measurements in different mixtures.

Usage

```
data("cornell0")
```

Format

A data frame with 12 observations on the following 8 variables. All are numeric.

[,1]	Distillation	Direct distillation
[,2]	Reformat	Reformat
[,3]	NaphthaT	Naphtha thermal cracked
[,4]	NaphthaC	Naphtha catalytic cracked
[,5]	Polymer	Polymer
[,6]	Alkylat	Alkylat
[, 7]	Gasoline	Natural gasoline
[, 8]	Y	Octane number

Source

Kettaneh-Wold N. (1992) Analysis of mixture data with partial least squares. Chemometrics and Intellignet Laboratory Systems, vol. 14, pp. 57-69.

References

- Tenenhaus M., Gauchi J.-P. and Ménardo C. (1995) Régression PLS et applications. Revue de Statistique Appliquée, vol. 53(1), pp. 7-63.
- Tenenhaus M. (1998) La régression PLS. Théorie et pratique. Ed. TECHNIP. Paris.

---

cornell1 *Motor Octane Number Measurements with Missing Values*

Description

This dataset is the `cornell0` dataset where missing values have been introduced in the first 7 variables.

Usage

```
data("cornell1")
```

Source

Kettaneh-Wold N. (1992) Analysis of mixture data with partial least squares. Chemometrics and Intellignet Laboratory Systems, vol. 14, pp. 57-69.

References

- Tenenhaus M., Gauchi J.-P. and Ménardo C. (1995) Régression PLS et applications. Revue de Statistique Appliquée, vol. 53(1), pp. 7-63.
- Tenenhaus M. (1998) La régression PLS. Théorie et pratique. Ed. TECHNIP. Paris.

crpolyX *Create polyX Object from Standard Polynomial*

Description

Create an object of class polyX from raw dataset of X-inputs and standard polynomial.

Usage

```
crpolyX(dataX, d, type = "full")
```

Arguments

dataX	Raw X-inputs. A data.frame with as many rows as observations, and as many columns as X-input variables.
d	Polynomial degree.
type	Polynomial type. A character string among: • full: complete polynomial with all monomials of degree less or equal to d, • power: power terms of degree less or equal to d, • interact: interactions of degree less or equal to d.

Value

An object of class polyX.

Note

The first monomials are always the X-input variables.

See Also

vect2polyX, crpolyXT

Examples

```
X <- cornell0[,1:3]
# Creation of the polynomial of degree 2 including the power terms:
# P= X1 + X2 + X3 + X1*X1 + X2*X2 + X3*X3
P <- crpolyX(X, 2, type="power")
```

crpolyXT *Create polyX Object from Standard Polynomial, Transformed Inputs*

Description

Create an object of class polyX from transformed X-inputs and standard polynomial.

Usage

```
crpolyXT(varnames, dataXT, d, type = "full")
```

Arguments

varnames	X-input names. A character vector of length equal to the number of X-inputs.
dataXT	Transformed X-inputs. A data.frame with as many rows as observations, and which the number of columns is equal or greater than the number of monomials.
d	Polynomial degree.
type	Polynomial type: see crpolyX.

Value

An object of class polyX.

Note

The first monomials are always the X-input variables.

See Also

vect2polyXT, crpolyX

factorsplit *Transform Categorical Variables into Numeric Variables*

Description

Transform each factor column of a dataframe into numeric variables.

- If the factor has more than 2 levels, it is split into as many 0-1 columns as levels. The value 1 is set on the lines where the factor is equal to the level, and 0 elsewhere.
- If the factor has one or two levels, the first level is coded -1 and the second one into +1.

The numeric columns are left unchanged.

Usage

```
factorsplit(data)
```

Arguments

data	Data.frame.

Value

The data.frame data, where the factor columns are replaced by 0/1 or +1/-1 columns. The labels of the added columns are the concatenation of the name of the factor and the name of the level, with the underscore separator.

Note

When using the other functions of the package on a result of factorsplit, be careful when describing the polynomial: the number and names of the variables in the returned data.frame may be different than the genuine ones.

See Also

crpolyX, vect2polyX

Examples

```
a <- data.frame(V=1:3, color=c("red","green","black"),
                       temp=c("hot", "cold", "hot"))
b <- factorsplit(a)
# The columns in b are: V, color_red, color_green,  color_red, temp
```

---

poly-class *Class* "poly"

Description

An S4 class container for a polynomial description.

Slots

P List of length equal to the number of monomials. Each component is a vector of length equal to the polynomial degree. It contains the numbers of the variables in the monomial.

indic Matrix with as many rows as monomials and as many columns as X-inputs. The element (i,j) is 1 when the variable j is in the monomial, 0 otherwise.

Objects from the Class

Objects from this class are usually not intended to be accessed directly by the end-user.

Methods

summary, print, or show: display functions. print has an optional argument, all=FALSE. When it is set to TRUE, all the monomes are displayed. Otherwise, only the polynomial degree, the number of monomials and the number of variables.

---

polyX-class *Class* "polyX"

Description

An S4 class container for a transformed dataset of inputs and its polynomial description.

Slots

dataX.exp Transformed X-inputs. A data.frame. The variables are the result of the calculation of the monomials of a polynomial on raw X-inputs.

Pindic Object of class poly.

Objects from the Class

Objects from this class are created by "new" or by the functions vect2polyX, vect2polyXT, crpolyX, crpolyXT

Methods

- summary, print, show: display functions. print has an optional argument, all=FALSE. When it is set to TRUE, all the monomes are displayed. Otherwise, only the polynomial degree, the number of monomials, the number of variables and the number of observations.
- bind: bind several polyX objects. See bind.polyX

---

sivip-class *Class* "sivip"

Description

An S4 class container for the result of the function sivipm.

Slots

`fo.isivip` vector of length equal to the number of X-inputs. Individual sensitivity indices.

`fo.isivip.percent` sorted percentages of `fo.isivip`.

`tsivip` vector of length equal to the number of X-inputs. Total sensitivity indices.

`tsivip.percent` sorted percentages of `tsivip`.

`monosignif` a vector of length equal to the number of monomials.

`correlalea` a matrix. The number of rows and columns is equal to the number of Y-inputs.

`simca.signifcomponents` a matrix. The number of rows is equal to the number of components, and the number of columns is equal to the number of Y-inputs plus one.

`lazraq.signifcomponents` a vector of length equal to the number of components.

`output` a list which contains the additional results.

See description in the function `sivipm`.

Objects from the Class

Objects from this class are created by the function `sivipm`.

Methods

`print` display function. `print` has an optional argument, `all=FALSE`. When it is set to TRUE, all the components are displayed. Otherwise, the slots `fo.isivip`, `tsivip` and `output` are hidden.

`show` equivalent to `print(all=TRUE)`

`summary` equivalent to `print(all=FALSE)`

`getNames` display the names of the non-null components

`sivipboot` *Confidence Intervals for the Total Sensitivity Indices by Bootstrap*

Description

Confidence intervals for the total sensitivity indices by a bootstrap method.

Usage

```
sivipboot(Y, XIndic, B, nc=2, graph=FALSE, alea=FALSE,
                  fast=FALSE, alpha=0.05)
```

Arguments

Y	Outputs. A data.frame with as many rows as observations and as many columns as response variables.
XIndic	Object of class polyX which contains the polynomial description and the dataset of inputs.
B	Number of bootstrap replicates.
nc	Number of components.
graph	If TRUE, boxplot display.
alea	If TRUE, an uniform random variable is included in the analysis (see sivipm).
fast	If TRUE, auxiliary results are calculated from the Miller's formulae more adapted to big datasets.
alpha	Level of the bootstrap confidence intervals.

Value

A matrix with as many rows as input variables and two columns: the lower and upper bounds of the total sensitivity indices percentile bootstrap confidence intervals.

See Also

sivipm

Examples

```
X <- cornell0[,1:3] # X-inputs
Y <- as.data.frame( cornell0[,8]) # response variable
# Creation of the polynomial:
P <- vect2polyX(X, c("1", "2", "3", "3*3*3"))
set.seed(15) #alea seed
nloops <- 3 # number of loops, example for fast running
nc <- 2 # number of components
sivipboot(Y, P, nloops, nc, fast=TRUE)
```

---

sivipm *Sensitivity Indices*

Description

Compute total and individual sensitivity indices, significant components and auxiliary results.

Usage

```
sivipm(Y,  XIndic,
        nc=2,  options= c("fo.isivip", "tsivip", "simca", "lazraq"),
        graph = FALSE, alea = FALSE, fast = FALSE,
        output=NULL)
```

Arguments

Y	Outputs. A data.frame with as many rows as observations and as many columns as response variables.
XIndic	Object of class polyX which contains the polynomial description and the dataset of inputs.
nc	Required number of components.
options	Options to select what is calculated. A string vector. Valid values are: • fo.isivip first order individual sensitivity indices, • tsivip total sensitivity indices, • simca significant components calculated by the SIMCA software rule. See Details. • lazraq significant components calculated by the Lazraq and Cléroux test. See Details.
graph	If TRUE, a graph is drawn when options includes tsivip.
alea	If TRUE, an uniform random variable is included in the analysis when options includes tsivip. Then, the non significant monomials are excluded from the total sensitivity indices calculation.
fast	If TRUE, auxiliary results are calculated from the Miller's formulae more adapted to big datasets.
output	If non NULL, additional results are returned in a component named output. Character vector, which valid values are: • isivip to return isivip • betaNat to return betaNat and betaNat0 • VIP to return VIP and VIPind • Q2 to return Q2 and Q2cum • PLS to return PLS results: mweights, weights, x.scores, x.loadings, y.scores, y.loadings, cor.tx, cor.ty, expvar, X.hat, Y.hat See "Value". It is advised to first determine the number of significant components, by setting the options simca or lazraq, before asking for additional results.

Details

- When the option simca or lazraq is set, the significant components are computed by the SIMCA software rule, or, by the Lazraq and Cléroux inferential test, at confidence level 0.95, respectively. The option simca is ignored if there are missing values. The option lazraq is ignored if there are missing values and more than one response variables.
- When the option alea is set, the non significant monomials are those for which the individual sensitivity indices is less or equal than the one of the random variable. These non significant monomials are excluded from the total sensitivity indices calculation.
- To analyze big datasets, the option fast is advised.

Value

An object of class sivip, whose slots are:

`fo.isivip and fo.isivip.percent`
: When `options` includes `fo.isivip`, values and sorted percentages of the first order individual sensitivity indices.

`tsivip and tsivip.percent`
: When `options` includes `tsivip`, values and sorted percentages of the total sensitivity indices for each input variable.

`monosignif`
: When `alea` is TRUE, and `options` includes `tsivip`, logical vector which indicates the significant monomials.

`correlalea`
: When `alea` is TRUE, and `options` includes `tsivip`, the correlation matrix between the random variable and the outputs.

`simca.signifcomponents`
: When `options` includes `simca`, the significant components calculated by the S. Wold's rule (SIMCA software rule). Logical matrix with `nc` rows and as many columns as response variables. Values are TRUE for the components where the test is positive at 95% level, FALSE otherwise.

`lazraq.signifcomponents`
: When `options` includes `simca`, the significant components calculated by the Lazraq and Cléroux inferential test. Logical vector of length `nc` with TRUE for the components where the test is positive at 95% level, FALSE otherwise.

`output`
: When `output` is not NULL, a list with additional results, whose components depend on `output` option.

  - `isivip` Individual sensitivity indices for each monomial. Vector of length equal to the number of monomials.
  - `betaNat` Natural `beta`. Matrix with as many rows as monomials and as many columns as response variables.
  - `betaNat0` Natural `beta0` coefficient. Vector of length equal to the number of response variables.
  - `VIP` Matrix of `nc` columns and as many rows as monomials.
  - `VIPind` Matrix with as many rows as response variables and as many columns as monomials.
  - `Q2` Matrix with as many columns as response variables and `nc` rows.
  - `Q2cum` Matrix with as many columns as response variables + one column and `nc` rows.
  - `PLS` PLS results. The dimension of the components are indicated below in brackets. `nmono` denotes the number of monomials, `ny`, the number of response variables and `nobs` the number of observations.
    - `betaCR` (`beta` centered and reduced. Vector `ny`),
    - `mweights` (`nc` X `nmono`),
    - `x.scores` (`nc` X `nobs`),
    - `x.loadings` (`nc` X `nmono`),
    - `y.scores` (`nc` X `nobs`),
    - `y.loadings` (`nc` X `ny`),

- weights (nc X nmono),
- cor.tx (nc X nmono),
- cor.ty (nc X ny),
- expvar (4 X nc),
- x.hat(nobs X nmono),
- y.hat (nobs X ny).

Note

If the output is multivariate, tsivip are the generalized total sensitivity indices (GTSIVIP) and isivip are the generalized individual sensitivity indices (GISIVIP).

Examples

```
X <- cornell0[,1:3] # X-inputs
Y <- as.data.frame( cornell0[,8]) # response variable
# Creation of the polynomial:
P <- vect2polyX(X, c("1", "2", "3", "3*3*3"))
# Compute total sensitivity indices:
A <- sivipm(Y, P, options=c("tsivip"))
# See the names of the returned components
getNames(A)
# The main results
summary(A)
# All the results
print(A, all=TRUE)
# Calculation by using the fast algorithm:
B <- sivipm(Y, P, fast = TRUE, options=c("tsivip"))
```

---

takeoff.polyX *Remove Monomials from a polyX Object*

Description

Remove monomials from an object of class polyX.

Usage

```
## S4 method for signature 'polyX'
takeoff(P, monomials)
```

Arguments

P	Object of class polyX.
monomials	Integer or character vector. The monomials to be removed. When integer, the numbers of the monomials. When character, their expression using the variable names and the character "*" to denote interaction.

Value

An object of class polyX.

See Also

bind.polyX

Examples

```
X <- cornell0[,1:2] # X-inputs
# Polynomial creation
monomials <- c("Distillation","Reformat","Distillation*Reformat")
P <- vect2polyX (X, monomials)
# Remove the third  monomial:
P2 <- takeoff(P, c(3))
# Same expressed with using variable names:
P2 <- takeoff(P, c("Distillation*Reformat"))
```

---

vect2polyX *Create polyX Object from Polynomial Description*

Description

Create an object of class polyX from a dataset of X-inputs and a vector that describes the polynomial.

Usage

```
vect2polyX(dataX, monomials)
```

Arguments

dataX	Raw X-inputs. A data.frame with as many rows as observations, and as many columns as variables.
monomials	Polynomial description. A character vector. Each element describes a monomial. The input variables are coded either by their numbers, or by their names. The character "*" denotes interaction between variables.

Value

An object of class polyX.

See Also

crpolyX, vect2polyXT

Examples

```
X <- cornell0[,1:3]
# Monomials expressed by variable numbers:
monomials <- c("1","2","3", "1*2*3")
polyXI <- vect2polyX (X, monomials)
# Monomials expressed by variable names:
monomials <- c("Distillation", "Reformat", "NaphthaT",
               "Distillation*Reformat*NaphthaT")
polyXI <- vect2polyX (X, monomials)
```

`vect2polyXT` *Create polyX Object from Polynomial Description*

Description

Create an object of class `polyX` from a dataset of transformed inputs and a vector that describes the polynomial.

Usage

```
vect2polyXT(varnames, dataXT, monomials)
```

Arguments

`varnames`	X-input names. A character vector of length equal to the number of X-inputs.
`dataXT`	Transformed inputs. A data.frame with as many rows as observations, and as many columns as monomials.
`monomials`	Polynomial description. A character vector. Each element describes a monomial. The input variables are coded either by their numbers, or by their names. The character "*" denotes interaction between variables. See `vect2polyX` example.

Value

An object of class `polyX`.

See Also

`crpolyXT`, `vect2polyX`

X18Y2 *Aircraft InfraRed Signature (IRS)*

Description

Aircraft flight measures. The eight first ones are related to flight conditions and IR optical properties of aircraft surfaces. The eight following ones are related to atmospheric conditions. The last two variables are the response values. Three qualitative measures have been transformed into numeric variables: MODEL and CLOUDS are coded as 0/1 and IHAZE is split into three 0/1 variables.

Usage

```
data("X18Y2")
```

Format

A data frame with 180 observations on the following 20 variables. All are numeric.

ALTI	Altitude
MACH	Mach number
POWERS	Engine power setting
EAI	Emissivity of air intake
CAP	Cap
YAW	Yaw
ROLL	Roll
PITCH	Pitch angles
VIS	Visibility
RH	Relative humidity
TA	Ground air temperature
MODEL	Atmospheric model
IHAZE1	Aerosol model urban
IHAZE2	Aerosol model maritime
IHAZE3	Aerosol mode rural
CLOUDS	Cloud presence
HBASE	Base altitude of the cloud layer
HOUR	Hour to compute solar position
IRStot	Typical infrared sensor spectral ranges
IRMS	Multispectral intensity

Source

Lefebvre, S. ONERA, Palaiseau, France.

References

Gauchi, J.-P. 2015. A practical method of global sensitivity analysis under constraints. Rapport technique 2015-1. INRA, UR1404, F-78350 Jouy-en-Josas, France.

`XY180` *Transformed Aircraft InfraRed Signature*

Description

This dataset is the result of monomial calculation on the dataset `X18Y2` (Aircraft InfraRed Signature). The monomials are described in the vignette of the package.

Usage

```
data("XY180")
```

Format

A data frame with 180 observations and 160 variables, labelled "V1", ... "V160". The 158 first ones are monomial values. The last two variables are response values.

Source

Lefebvre, S. ONERA, Palaiseau, France.

References

Gauchi, J.-P. 2015. A practical method of global sensitivity analysis under constraints. Rapport technique 2015-1. INRA, UR1404, F-78350 Jouy-en-Josas, France.

Index