Package ‘polychaosbasics’

March 1, 2017

Version 1.1-1
Date 2017-03-01
Title Sensitivity Indexes Calculated from Polynomial Chaos Expansions
Author A. Bouvier [aut], J.-P. Gauchi [cre], A. Bensadoun [aut]
Maintainer Annie Bouvier <annie.bouvier@inra.fr>
Depends methods
Imports utils, lhs, MASS
Suggests graphics, stats
Description Computation of sensitivity indexes by using a method based on a truncated Polynomial Chaos Expansions of the response.
      The necessary condition of the method is: the inputs must be uniformly and independently sampled. Since the inputs are uniformly distributed, the truncated Polynomial Chaos Expansion is built from the multivariate Legendre orthogonal polynomials.
License GPL (>= 2)
URL https://cran.r-project.org/package=polychaosbasics,
      http://genome.jouy.inra.fr/logiciels/polychaosbasics
      PCEfit-class.R PCEpoly-class.R PCESI.R polyLeg.R
NeedsCompilation no
Repository CRAN
Date/Publication 2017-03-01 12:29:25

R topics documented:

  polychaosbasics-package .................................................. 2
  analyticsPolyLeg .......................................................... 3
  getNames ................................................................. 5
  PCEdesign-class ........................................................ 5
  PCEfit-class ............................................................. 6
  PCEpoly-class ............................................................ 7
Description

Computation of sensitivity indexes by using a method based on a truncated Polynomial Chaos Expansions of the response. The necessary condition of the method is: the inputs must be uniformly and independently sampled. Since the inputs are uniformly distributed, the truncated Polynomial Chaos Expansion is built from the multivariate Legendre orthogonal polynomials.

Details

Legendre chaos polynomials are calculated on a provided dataset by function polyLeg or on a simulated LHS by function analyticsPolyLeg.

Then, from the object returned by these functions, the PCESI function calculates sensitivity indexes, metamodel coefficients and some other results.

Author(s)

A. Bouvier [aut], J.-P. Gauchi [cre], A. Bensadoun [aut]

Maintainer: Annie Bouvier <annie.bouvier@inra.fr>

References


Examples

```r
# First example:
# the dataset is simulated by using the Ishigami function
nlhs <- 200 # number of rows
degree <- 6 # polynomial degree
set.seed(42)# fix the seed for reproducible results
pce <- analyticsPolyLeg(nlhs, degree, 'ishigami') # build Legendre polynomial
ret <- PCESI(pce) # compute the PCE sensitivity indexes
print(ret)
```
# Illustrate the result by a plot:
# plot the computer model output against the metamodel output
y.hat <- ret@y.hat  # metamodel output
y.obs <- pce[, "Y"]  # computer model output
## Not run:
x11()
plot(y.hat, y.obs,
     xlab="metamodel output", ylab="computer model output",
     main="Ishigami test", sub="Scatter plot and regression line")
# Add the regression line
reg <- lm(y.hat ~ y.obs)  # linear regression
lines(reg$fitted.values, y.obs)

## End(Not run)

# Second example:
# the dataset is a user dataset
load(system.file("extdata", "FLORSYS1extract.Rda",
                package="polychaosbasics"))
degree <- 4  # polynomial degree
lhs <- FLORSYS1extract[, -ncol(FLORSYS1extract)]  # inputs
Y <- FLORSYS1extract[, ncol(FLORSYS1extract)]  # output
pce <- polylen(lhs, Y, degree)  # build Legendre polynomial
ret <- PCESSI(pce)  # compute the PCE sensitivity indexes
print(ret, all=TRUE)

---

**analyticsPolyLeg**  
*Calculate Legendre Polynomials on a Simulated Dataset*

**Description**

This function calculates Legendre polynomials on a simulated LHS.

The dataset is generated by using the function `randomLHS` (from package `lhs`). The output is then calculated by using the Ishigami [Saltelli, 2000, Chap. 2] or Sobol function [Sobol’, 2003]. Finally, Legendre polynomials are computed after calibration within the bounds [-1, +1].

**Usage**

`analyticsPolyLeg(nlhs, degree, model.fun)`

**Arguments**

- `nlhs` integer equal to the number of rows of the dataset.
- `degree` integer equal to the degree of the polynomial. Should be greater than 1.
- `model.fun` string equal to the required model. Valid values are 'ishigami' and 'sobol'.
Details

• The Ishigami function has three inputs that are linked to the output $Y$ according to:

\[ Y = \sin(X_1) + 7 \times (\sin(X_2))^2 + 0.1 \times (X_3)^4 \times \sin(X_1) \]

Each $X_j$ is a uniform random variable on the interval $[-\pi, +\pi]$.

• The Sobol function has height inputs. The four first ones only are generated by using the function `randomLHS`. The four last are set to 0.5 (see Gauchi, 2017). The output $Y$ is then the product of:

\[ (4 \times X_j - 2 + A_j)/(1 + A_j) \]

for $j$ in 1 to 8, and $A = (1, 2, 5, 10, 20, 50, 100, 500)$

Value

An objet of class `PCEpoly`.

Note

The returned values are dependent on the random seed.

References


See Also

• Function `polyLeg` calculates Legendre polynomials on a user dataset.

• Function `PCESI` calculates PCE sensitivity indexes from the returned object.

Examples

```r
nlhs <- 200 # number of rows in the dataset
degree <- 6 # polynomial degree
set.seed(42) # fix the seed for reproductible results
pce <- analyticsPolyLeg(nlhs, degree, 'ishigami')
print(pce)
```
getNames

Description
Display the names, class and length of all the slots of a \texttt{PCEpoly} or \texttt{PCEfit} object.

Usage
getNames(object)

Arguments
object object from class \texttt{PCEpoly} or \texttt{PCEfit}.

Details
It is a generic function. Its methods are defined in classes \texttt{PCEpoly} and \texttt{PCEfit}.

Value
Doesn’t return any value.

See Also
Classes \texttt{PCEpoly} and \texttt{PCEfit}.

Examples
# Build Legendre polynomial degree 6 on a dataset
# simulated by using the Ishigami function:
pce <- analyticsPolyLeg(100, 6, 'ishigami')
# Display what contains the returned PCEpoly object:
getNames(pce)

PCEdesign-class

Description
Container of the polynomial description structure, as it is stored in objects from class \texttt{PCEpoly}.

Objects from the Class
Objects from this class are created by calls to functions \texttt{polyLeg} or \texttt{analyticsPolyLeg}. They are stored in the slot \texttt{design} in the object of class \texttt{PCEpoly} returned by these functions.
Slots

.Data: matrix with as many columns as inputs and as many rows as monomials plus one. Element
(i,j) is an integer equal to the degree of the input j in the monomial i-1. The first row is
equal to zero: it is for the constant term.

Methods

print signature(x = "PCEdesign", all=FALSE, ...): method of function print. If option
all is set to TRUE, all the monomials are printed. The additional arguments are passed to the
print.default function.

show signature(object = "PCEdesign"): same as function print, without any arguments.

See Also

• Functions polyLeg and analyticsPolyLeg, creators of objects from this class.
• Class PCEpoly in which objects from this class are stored.

---

PCEfit-class Class "PCEfit"

Description

Container of the results of PCE sensitivity indexes computation.

Objects from the Class

Objects from this class are created by calls to function PCESI.

Slots

indexes: matrix with as many rows as inputs and 3 columns. Values of the PCE sensitivity indexes.
The row labels are the inputs numbers. The column labels are LE, PE, TPE.

• indexes[i, "LE"] is the Linear Effect of the input i.
• indexes[i, "PE"] is the Polynomial Effect (called "SU" in Sudret, 2008). It is the effect
  of the monomials in which only the input i appears.
• indexes[i, "TPE"] is the Total Polynomial Effect (often called "SUT"). It is the effect
  of all the monomials in which the input i appears.

indexes.percent: matrix. Percentages of the PCE sensitivity indexes, i.e values of indexes ex-
pressed as percentages of the sums of their columns.

fit: vector of length 2. The values of R2 and RMSEP (RMSEP: Root Mean Square Error Predic-
tion).

IMSI: vector of length equal to the number of monomials. Individual monomial sensitivity indexes.

coef: vector of length equal to the number of monomials plus one. Regression coefficients. The
first one is the constant term.
y.hat: vector of length equal to the number of rows of the dataset. Metamodel outputs.
design: object of class PCEdesign. Matrix coding the polynomial structure.
call.PCEpoly: expression of class ‘call’. The command which creates the PCEpoly object used
as input in the creator command.

Methods

getNames signature(object = "PCEfit"): display the names, class and length of all the comp-
onents. See the description of the generic function getNames.
print signature(object = "PCEfit", all=FALSE, ...): method of function print. When
option all is set to FALSE (the default), only the components indexes, indexes.percent
and fit are printed. The additional arguments are passed to the print.default function.
show signature(object = "PCEfit"): same as function print, without any arguments.

References

Global sensitivity analysis using polynomial chaos expansions. Bruno Sudret. In Reliability En-

See Also

Function PCESI, creator of objects from this class.

Examples

showClass("PCEfit")

Description

Container of the PCE design. It stores the computed values and the structure description of the
Legendre polynomial.

Objects from the Class

Objects from this class are created by calls to functions polyLeg or analyticsPolyLeg.

Slots

.Data: matrix. The computed values of the Legendre polynomial. The number of rows is the
number of rows of the LHS. The number of columns is the number of monomials plus one.
The first column is equal to one: it is for the constant term.
design: object of class PCEdesign. Matrix coding the polynomial structure.
nvx: integer equal to the number of inputs.
call: expression of class ‘call’. The command which creates the object.
Methods

getNames signature(object = "PCEpoly"): display the names, class and length of all the components. See the description of the generic function getNames.

print signature(object = "PCEpoly", all=FALSE, ...): method of function print. The polynomial expression is printed when option all is set to TRUE. The additional arguments are passed to the print.default function.

show signature(object = "PCEpoly"): same as function print, without any arguments.

See Also

Functions polyLeg and analyticsPolyLeg, creators of objects from this class.

Examples

showClass("PCEpoly")

---

PCESI 

Compute Sensitivity Indexes from a PCE Design

Description

Calculation of PCE sensitivity indexes and related results.

Usage

PCESI(poly)

Arguments

poly an object of class PCEpoly. The design to analyze (a Legendre polynomial).

Value

An object of class PCEfit.

Note

• By default, only a part of the returned object is displayed by the functions print() and show(). To see the hidden components, use the function print() with the option all=TRUE or the function getNames (see methods of class PCEfit).

• It is advised to increase gradually the polynomial degree, up to the returned object contains a R2 value near from 1 and a low RMSEP value.

See Also

• Functions polyLeg and analyticsPolyLeg, creators of objects from class PCEpoly.

• Class PCEfit for description of the returned structure.
### Examples

```r
# Dataset simulated by using the Ishigami function
dlhs <- 200 # number of rows
degree <- 6 # polynomial degree
set.seed(42)# fix the seed for reproducible results
pce <- analyticsPolyleg(nlhs, degree, 'ishigami')# build the PCE design
ret <- PCESI(pce) # compute the PCE sensitivity indexes
print(ret, all=TRUE)
```

---

**polyLeg**

*Calculate Legendre Polynomials from a Dataset*

**Description**

This function calculates Legendre polynomials on a user LHS. Legendre polynomials are computed after calibration within the bounds [-1, +1].

**Usage**

`polyLeg(lhs, Y, degree)`

**Arguments**

- **lhs**: matrix with as many columns as inputs. Dataset of inputs. Generally, a space filling design is used for forming this dataset. Typically, this is a simple LHS (see McKay, 1979) or a modified LHS.
- **Y**: vector of length equal to the number of rows in `lhs`. Model outputs.
- **degree**: integer greater than 1. Maximal degree of the polynomial.

**Value**

An objet of class `PCEpoly`.

**References**


**See Also**

- Function `analyticsPolyleg` builds Legendre polynomials from a simulated dataset.
- Function `PCESI` calculates PCE sensivity indexes from the returned object.
Examples

```r
load(system.file("extdata", "FLORSYS1extract.Rda", package="polychaosbasics"))
degree <- 4 # polynomial degree
lhs <- FLORSYS1extract[, -ncol(FLORSYS1extract)] # inputs
Y <- FLORSYS1extract[, ncol(FLORSYS1extract)] # output
pce <- polyLeg(lhs, Y, degree)
print(pce)
```

show

Methods 'show' for Classes of the Package 'polychaosbasics'

Description

Methods for the generic function 'show' are defined for the classes `PCEpoly`, `PCEdesign` and `PCEfit`. They do the same thing as the print methods when those are invoked without any argument.

Arguments

object

object from class `PCEpoly`, `PCEdesign` or `PCEfit`.

Value

Doesn't return any value.

See Also

Classes `PCEpoly`, `PCEdesign` and `PCEfit`.

Examples

```r
# Build Legendre polynomial degree 6 on a dataset
# simulated by using the Ishigami function:
pce <- analyticsPolyLeg(100, 6, 'ishigami')
# Standard display the returned PCEpoly object:
pce # it is equivalent to 'show(pce)'
```
Index

*Topic classes
  PCEdesign-class, 5
  PCEfit-class, 6
  PCEpoly-class, 7
*Topic methods
  show, 10
  show, PCEdesign-method (show), 10
  show, PCEfit-method (show), 10
  show, PCEpoly-method (show), 10
*Topic package
  polychaosbasics-package, 2
*Topic regression
  analyticsPolyLeg, 3
  PCESI, 8
  polychaosbasics-package, 2
  polyLeg, 9
analyticsPolyLeg, 2, 3, 5–9

getNames, 5, 7, 8
getNames, PCEfit-method (getNames), 5
getNames, PCEpoly-method (getNames), 5

lhs, 3

PCEdesign, 7, 10
PCEdesign-class, 5
PCEfit, 5, 8, 10
PCEfit-class, 6
PCEpoly, 4–10
PCEpoly-class, 7
PCESI, 2, 4, 6, 7, 8, 9
polychaosbasics
  (polychaosbasics-package), 2
polychaosbasics-package, 2
polyLeg, 2, 4–8, 9
print, 6–8
print.default, 6–8
print.PCEdesign (PCEdesign-class), 5
print.PCEfit (PCEfit-class), 6
print.PCEpoly (PCEpoly-class), 7

randomLHS, 3, 4