Package 'multisensi'

October 13, 2022

Type Package

Title Multivariate Sensitivity Analysis

Version 2.1-1

Date 2018-04-04

Author Caroline Bidot <caroline.bidot@inra.fr>, Matieyendou Lamboni <matieyendou.lamboni@gmail.com>, Hervé Monod <herve.monod@inra.fr>

Maintainer Hervé Monod <herve.monod@inra.fr>

Description Functions to perform sensitivity analysis on a model with multivariate output.

License CeCILL-2

Repository CRAN

LazyLoad yes

Depends R (>= 2.8.0)

Suggests MASS

Imports stats, graphics, utils, grDevices, sensitivity, knitr

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Date/Publication 2018-04-10 10:27:07 UTC

R topics documented:

`multisensi-package` *Multivariate sensitivity Analysis*

Description

Sensitivity Analysis (SA) for models with multivariate output

Details

This package generalises sensitivity analysis to simulation models with multivariate output. It makes it easy to run a series of independent sensitivity analyses on a set of output variables and to plot the results. Alternatively, it allows to apply sensitivity analyses to the variables resulting from the application of a multivariate method (such as PCA or splines or polynomial regression) to the output data (Lamboni et al., 2009).

The function `multisensi` integrates all the different possible methods implemented in the package. Besides, the user may consider the functions which have existed since the first version of the package:

i) `gsi` function for the Generalised Sensitivity Analysis (Lamboni et al., 2011, Xiao and Li, 2016) based on inertia decomposition. This method synthesizes the information that is spread between the time outputs or between the principal components and produces a unique sensitivity index for each factor.

ii) gsi function for the componentwise sensitivity analysis obtained by computing sensitivity indices on principal components (Campbell et al., 2006)

iii) dynsi function for the dynamic sensitivity analysis obtained by computing sensitivity indices on each output variable.

In the first version of **multisensi**, sensitivity indices were based on using a factorial design and a classical ANOVA decomposition. It is now possible to use other methods for the design and for the sensitivity analysis.

Simulation model management

The **multisensi** package works on simulation models coded either in R or using an external language (typically as an executable file). Models coded in R must be either functions or objects that have a predict method, such as lm objects. Models defined as functions will be called once with an expression of the form `y <- f(X)` where `X` is a vector containing a combination of levels of the input factors, and `y` is the output vector of length q, where q is the number of output variables. If the model is external to R, for instance a computational code, it must be analyzed with the decoupled approach: the methods require an input data frame (`X`) containing all the combinations of the input levels and the outputs data frame (`Y`) containing the response of the model corresponding to these combinations. The size of `X` is $n * p$ and the size of `Y` is $n * q$ where p is the number of the input factor, q is the number of the model outputs and n is the number of all the combinations of the input levels. This approach can also be used on R models that do not fit the required specifications.

References

Lamboni, M., Makowski, D., Monod, H., 2009. Multivariate global sensitivity analysis for dynamic crop models. Field Crops Research, volume 113, pp. 312-320.

Lamboni, M., Makowski, D., Monod, H., 2011. Multivariate sensitivity analysis to measure global contribution of input factors in dynamic models. Reliability Engineering & System Safety, volume 96, pp. 450-459.

Xiao, H., Li, L., 2016. Discussion of paper by M. Lamboni, H. Monod, D. Makowski Multivariate sensitivity analysis to measure global contribution of input factors in dynamic models, Reliab. Eng. Syst. Saf. 96 (2011) 450-459. Reliability Engineering & System Safety, volume 147, pp. 194-195.

Saltelli, A., Chan, K., Scott, E.M. eds, 2000. *Sensitivity Analysis* Wiley, New York.

`analysis.anoasg` *Runs a series of analyses of variance*

Description

The `analysis.anoasg` function runs a series of analyses of variance on the columns of a data.frame, by using the aov function.

Usage

```
analysis.anoasg(Y, plan, nbcomp = 2, sigma.car = NULL,
                 analysis.args = list(formula = 2,
                                      keep.outputs = FALSE))
```

Arguments

Y	a data.frame of output variables or principal components.
plan	a data.frame containing the design.
nbcomp	the number of Y variables to analyse (the first nbcomp variables of Y will be analysed).
sigma.car	NULL or sum of squares of Y. If not NULL, compute the Generalised Sensitivity Indices (saved in the last column of the data.frame mSI/tSI/iSI outputs.
analysis.args	a list of arguments. The formula component is for ANOVA formula like "A+B+c+A:B" OR an integer giving the maximum interaction order (1 for main effects). If it contains keep.outputs=TRUE, the outputs associated with the analysis of each variable are returned (see section Value).

Value

A list containing:

SI	data.frame of sensitivity indices
mSI	data.frame of first-order sensitivity indices
tSI	data.frame of total sensitivity indices
iSI	data.frame of interaction sensitivity indices
inertia	vector of Inertia explained by the variables
indic.fact	0-1 matrix to indicate the factors associated with each factorial effect
Hpredict	prediction of outputs
outputkept	if analysis.args$keep.outputs=TRUE, list of the outputs returned by the sensitivity analysis performed on each variable
call.info	list with first element analysis="anova"

See Also

aov

Examples

```
# Test case : the Winter Wheat Dynamic Models (WWDM)
#  input factors design
 data(biomasseX)
# output variables (precalculated to speed up the example)
 data(biomasseY)

 res <- analysis.anoasg(biomasseY, biomasseX,
```

```
                    nbcomp = 2, sigma.car = NULL,
                    analysis.args = list(formula = 2,
                                         keep.outputs = FALSE))
```

`analysis.sensitivity` *Runs a series of sensitivity analyses by a function from the* **sensitivity** *package*

Description

The `analysis.sensitivity` function runs a series of sensitivity analyses on the columns of a data.frame, using a method implemented in the **sensitivity** package.

Usage

```
analysis.sensitivity(Y, plan, nbcomp = 2, sigma.car = NULL,
                     analysis.args = list(keep.outputs = FALSE))
```

Arguments

`Y`	a data.frame of output variables or principal components.
`plan`	an object containing the design. It must be created by a function from the **sensitivity** package with argument `model=NULL`.
`nbcomp`	the number of `Y` variables to analyse (the first `nbcomp` variables of `Y` will be analysed).
`sigma.car`	NULL or sum of squares of Y. If not NULL, compute the Generalised Sensitivity Indices (saved in the last column of the data.frame mSI/tSI/iSI outputs.
`analysis.args`	a list of arguments. If it contains `keep.outputs=TRUE`, the outputs associated with the analysis of each variable are returned (see section `Value`).

Details

The argument `plan` must be an object created by a method implemented in the **sensitivity** package. Thus it belongs to a class such as `morris` or `fast99`. The name of the class is stored in the element `call.info$fct` of the output returned by `analysis.sensitivity`.

Value

A list containing:

`SI`	data.frame of sensitivity indices or other importance measures returned by the function from the **sensitivity** package used. Sometimes empty but kept for compatibility reasons.
`mSI`	data.frame of first-order sensitivity indices
`tSI`	data.frame of total sensitivity indices

iSI	data.frame of interaction sensitivity indices
inertia	empty (kept for compatibility reasons)
indic.fact	0-1 matrix to indicate the factors associated with each factorial effect
Hpredict	empty (kept for compatibility reasons)
outputkept	if analysis.args$keep.outputs=TRUE, list of the outputs returned by the sensitivity analysis performed on each variable
call.info	list with first element analysis="sensitivity" and second element fct storing the class name of the argument plan

Examples

```
# Test case : the Winter Wheat Dynamic Models (WWDM)
library(sensitivity) # to use fast99
# input factors design
data(biomasseX)
# input climate variable
data(Climat)

# example of the sensitivity:fast99 function
# design
newplan <- fast99(model = NULL, factors = names(biomasseX), n = 100,
            q = "qunif", q.arg = list(list(min = 0.9, max = 2.8),
                         list(min = 0.9, max = 0.99),
                         list(min = 0.6, max = 0.8),
                         list(min = 3, max = 12),
                         list(min = 0.0035, max = 0.01),
                         list(min = 0.0011, max = 0.0025),
                         list(min = 700, max = 1100)))
# simulations
wwdm.Y <- simulmodel(model=biomasse, plan=newplan$X, climdata=Climat)
# analysis
res <- analysis.sensitivity(data.frame(wwdm.Y), plan=newplan, nbcomp=4)
```

---

basis.ACP *A function to decompose multivariate data by principal components analysis (PCA)*

Description

The basis.ACP function decomposes a multivariate data set according to principal components analysis.

Usage

```
basis.ACP(simuls, basis.args = list())
```

Arguments

`simuls`	a data.frame of size `N` x `T`, typically a set of `N` simulation outputs of length `T`.
`basis.args`	an empty list of arguments for the PCA decomposition.

Details

This function uses `prcomp`.

Value

`H`	a data.frame of size `N` x `T`, containing the coefficients of the PCA decomposition. It is equal to the `x` output of function `prcomp`.
`L`	a matrix of size `T` x `T`. It contains the eigenvectors of the PCA decomposition.
`call.info`	list with the element `reduction="pca"`

See Also

`prcomp`

Examples

```
data(biomasseY)
res <- basis.ACP(biomasseY)
```

`basis.bsplines` *A function to decompose multivariate data on a B-spline basis*

Description

The `basis.bsplines` function decomposes a multivariate data set on a B-spline basis defined by its knots and mdegree parameters.

Usage

```
basis.bsplines(simuls, basis.args = list(knots = 5, mdegree = 3))
```

Arguments

`simuls`	a data.frame of size `N` x `T`, typically a set of `N` simulation outputs of length `T`.
`basis.args`	a list of arguments for the B-spline decomposition. The `knots` argument is the number of knots or the vector of knot positions. The `mdegree` argument is the polynomial degree. For the optional `x.coord` argument, see the Details section.

Details

The optional `x.coord` element of the list in `basis.args` can be used to specify the support of the B-spline decomposition, if different from `1:T`. It must be a vector of length `T`.

Value

H	a data.frame of size N x d, where d is the dimension of the B-spline decomposition. It contains the coefficients of the decomposition for each row of the simuls data.frame.
L	a matrix of size T x d. It contains the vectors of the B-spline basis.
call.info	list with the element reduction="b-splines"

See Also

bspline, sesBsplinesNORM

Examples

```
data(biomasseY)

res <- basis.bsplines(biomasseY,basis.args=list(knots=7,mdegree=3))
```

---

basis.mine *A function to decompose multivariate data on a user-defined basis*

Description

The basis.mine function decomposes a multivariate data set on a user-defined basis.

Usage

```
basis.mine(simuls, basis.args = list(
        baseL=1*outer(sort(0:(ncol(simuls)-1)%%5),0:4,"==") ) )
```

Arguments

simuls	a data.frame of size N x T, typically a set of N simulation outputs of length T.
basis.args	a list of arguments for the polynomial decomposition. The baseL argument is a matrix of size T x d containing the coordinates of the d basis vectors.

Details

The default basis.args argument generates a projection on a moving-average basis. But if in the multisensi function this basis.args argument is not given for reduction=basis.mine, the execution will be stopped.

Value

H	a data.frame of size N x d, where d is the number of basis vectors. It contains the coefficients of the decomposition for each row of the simuls data.frame.
L	a matrix of size T x d. It contains the vectors of the user-defined basis.
call.info	list with the element reduction="matrix"

Examples

```
data(biomasseY)
M <- 1*outer(sort(0:(ncol(biomasseY)-1)%%5),0:4,"==")
norm.M <- sqrt(colSums(M^2))
for (i in 1:ncol(M)){
    M[,i]=M[,i]/norm.M[i]
}

res <- basis.mine(biomasseY, basis.args=list(baseL=M))
```

---

basis.osplines — *A function to decompose multivariate data on an orthogonal B-spline basis (O-spline)*

Description

The `basis.osplines` function decomposes a multivariate data set on an orthogonalised B-spline (or O-spline) basis defined by its knots and mdegree parameters.

Usage

```
basis.osplines(simuls, basis.args = list(knots = 5, mdegree = 3))
```

Arguments

`simuls`	a data.frame of size `N` x `T`, typically a set of `N` simulation outputs of length `T`.
`basis.args`	a list of arguments for the O-spline decomposition. The `knots` argument is the number of knots or the vector of knot positions. The `mdegree` argument is the polynomial degree. For the optional `x.coord` argument, see the Details section.

Details

The optional `x.coord` element of the list in `basis.args` can be used to specify the support of the O-spline decomposition, if different from `1:T`. It must be a vector of length `T`.

Value

`H`	a data.frame of size `N` x `d`, where `d` is the dimension of the O-spline decomposition. It contains the coefficients of the decomposition for each row of the `simuls` data.frame.
`L`	a matrix of size `T` x `d`. It contains the vectors of the O-spline basis.
`call.info`	list with the element `reduction="o-splines"`

See Also

`bspline`, `sesBsplinesORTHONORM`

Examples

```
data(biomasseY)

res <- basis.osplines(biomasseY,basis.args=list(knots=7,mdegree=3))
```

---

basis.poly   *A function to decompose multivariate data on a polynomial basis*

Description

The basis.poly function decomposes a multivariate data set on a polynomial basis.

Usage

```
basis.poly(simuls, basis.args = list(degree = 3))
```

Arguments

simuls	a data.frame of size N x T, typically a set of N simulation outputs of length T.
basis.args	a list of arguments for the polynomial decomposition. The degree argument is the maximum degree of the polynomial basis. For the optional x.coord argument, see the Details section.

Details

This function uses poly. The optional x.coord element of the list in basis.args can be used to specify the support of the polynomial decomposition, if different from 1:T. It must be a vector of length T.

Value

H	a data.frame of size N x (d+1), where d is the degree of the polynomial decomposition. It contains the coefficients of the decomposition for each row of the simuls data.frame.
L	a matrix of size T x (d+1). It contains the vectors of the polynomial basis.
call.info	list with the element reduction="polynomial"

See Also

poly

Examples

```
data(biomasseY)

res <- basis.poly(biomasseY,basis.args=list(degree=3))
```

`biomasse` *The Winter Wheat Dynamic Model*

Description

The Winter Wheat Dynamic Model, a toy model to illustrate the main multisensi methods

Usage

```
biomasse(input, climdata, annee = 3)
```

Arguments

`input`	vector of input values.
`annee`	year.
`climdata`	a meteorological data.frame specific to biomasse.

Details

The Winter Wheat Dry Matter model (WWDM) is a dynamic crop model running at a daily time step (Makowski et al, 2004). It has two state variables, the above-ground winter wheat dry matter $U(t)$, in g/m^2 and the leaf area index LAI(t) with t the day number from sowing ($t = 1$) to harvest ($t = 223$). In the **multisensi** package implementation, the `biomasse` function simulates the output for only one parameter set (the first row of `input` if it is a matrix or a data.frame).

Value

a vector of daily dry matter increase of the Winter Wheat biomass, over 223 days

References

Makowski, D., Jeuffroy, M.-H., Gu\'erif, M., 2004 Bayesian methods for updating crop model predictions, applications for predicting biomass and grain protein content. In: Bayesian Statistics and Quality Modelling in the Agro-Food Production Chain (van Boeakel et al. eds), pp. 57-68. Kluwer, Dordrecht

Monod, H., Naud, C., Makowski, D., 2006 Uncertainty and sensitivity analysis for crop models. In: Working with Dynamic Crop Models (Wallach D., Makowski D. and Jones J. eds), pp. 55-100. Elsevier, Amsterdam

biomasseX *A factorial input design for the main example*

Description

Factorial design (resolution V) data for the 7 WWDM model input factors

Usage

```
data(biomasseX)
```

Format

A data frame with 2187 observations on the following 7 variables.

- `Eb` First WWDM input factor name
- `Eimax` Second WWDM input factor name
- `K` Thirth WWDM input factor name
- `Lmax` Fourth WWDM input factor name
- `A` Fifth WWDM input factor name
- `B` Sixth WWDM input factor name
- `TI` Seventh WWDM input factor name

See Also

`biomasse`, `biomasseY`

Examples

```
data(biomasseX)
## maybe str(biomasseX) ; plot(biomasseX) ...
```

biomasseY *Output of the biomasse model for the plan provided in the package*

Description

Simplified output of the biomasse model (one column per decade), especially generated for examples in the package help files

Usage

```
data(biomasseY)
```

Format

A data frame with 2187 rows and 22 output variables (one per decade).

See Also

biomasse, biomasseX

Examples

```
data(biomasseY)
dim(biomasseY)
```

---

bspline *function to evaluate B-spline basis functions*

Description

The `bspline` function evaluates ith B-spline basis function of order m at the values in x, given knot locations in k

Usage

```
bspline(x = seq(0, 1, len = 101), k = knots, i = 1, m = 2)
```

Arguments

`x`	vector or scalar, coordinate where to calculate the B-spline functions
`k`	vector of knot locations
`i`	integer; from 0 to length(knots)+1-m
`m`	integer, degree of the B-Splines

Details

B-splines are defined by recursion : $b_{i,0}(x) = 1$ if $k_j \le x < k_{j+1}$; 0 else.

$$b_{i,m}(x) = \frac{x - k_i}{k_{i+m} - k_i} b_{i,m-1}(x) + \frac{k_{i+m+1} - x}{k_{i+m+1} - k_{i+1}} b_{i+1,m-1}(x)$$

Value

values in x of the ith B-spline basis function of order m

Note

This is essentially an internal function for the **multisensi** package

References

Wood Simon, 2006. *Generalized Additive Models: An Introduction with R* Chapman and Hall/CRC.

Climat *Climate data*

Description

Climate data for the WWDM model (needed by the `biomasse` function)

Usage

```
data(Climat)
```

Format

A data frame with 3126 observations on the following 4 variables.

`ANNEE` a factor with levels 1 to 14, indicating 14 different years

`RG` daily radiation variable

`Tmin` daily maximum temperature

`Tmax` daily minimum temperature

Source

Makowski, D., Jeuffroy, M.-H., Gu\'erif, M., 2004 Bayesian methods for updating crop model predictions, applications for predicting biomass and grain protein content. In: Bayesian Statistics and Quality Modelling in the Agro-Food Production Chain (van Boeakel et al. eds), pp. 57-68. Kluwer, Dordrecht.

Monod, H., Naud, C., Makowski, D., 2006 Uncertainty and sensitivity analysis for crop models. In: Working with Dynamic Crop Models (Wallach D., Makowski D. and Jones J. eds), pp. 55-100. Elsevier, Amsterdam

dynsi *Dynamic Sensitivity Indices: DSI*

Description

dynsi implements the Dynamic Sensitivity Indices. This method allows to compute classical Sensitivity Indices on each output variable of a dynamic or multivariate model by using the ANOVA decomposition

Usage

```
dynsi(formula, model, factors, cumul = FALSE, simulonly=FALSE,
      nb.outp = NULL, Name.File=NULL, ...)
```

Arguments

`formula`	ANOVA formula like `"A+B+c+A:B"` OR an integer equal to the maximum interaction order in the sensitivity model.
`model`	output data.frame OR the name of the R-function which calculates the model output. The only argument of this function must be a vector containing the input factors values.
`factors`	input data.frame (the design) if model is a data.frame OR a list of factors levels such as `factor.example <- list(A=c(0,1),B=c(0,1,4))`.
`cumul`	logical value. If TRUE the sensitivity analysis will be done on the cumalative outputs.
`simulonly`	logical value. If TRUE the program stops after calculating the design and the model outputs.
`nb.outp`	The first nb.outp number of model outputs to be considered. If NULL all the outputs are considered.
`Name.File`	optional name of a R script file containing the R-function that calculates the simulation model. e.g `"exc.ssc"`.
`...`	possible fixed parameters of the model function.

Details

If `factors` is a list of factors, the dynsi function generates a complete factorial design. If it is a data.frame, dynsi expects that each column is associated with an input factor.

Value

dynsi returns a list of class "dynsi" containing the following components:

`X`	a data.frame containing the experimental design (input samples)
`Y`	a data.frame containing the output (response)
`SI`	a data.frame containing the Sensitivity Indices (SI) on each output variable of the model and the Generalised SI (GSI)
`mSI`	a data.frame of first order SI on each output variable and first order GSI
`tSI`	a data.frame containing the total SI on each output variable and the total GSI
`iSI`	a data.frame of interaction SI on each output variable and interaction GSI
`Att`	0-1 matrix of association between input factors and factorial terms in the anovas
`call.info`	a list containing informations on the process (reduction=NULL, analysis, fct, call)
`inputdesign`	either the input data.frame or the sensitivity object used
`outputs`	a list of results on each output variable

...

Note

This function can now be replaced by a call to the `multisensi` function. It is kept for compatibility with Version 1 of the **multisensi** package.

References

M. Lamboni, D. Makowski and H. Monod, 2009. Multivariate global sensitivity analysis for dynamic crop models. Field Crops Research, 113, 312-320.

A. Saltelli, K. Chan and E. M. Scott eds, 2000. *Sensitivity Analysis*. Wiley, New York.

See Also

gsi, multisensi

Examples

```
# Test case : the Winter Wheat Dynamic Models (WWDM)
# input factors design,
 data(biomasseX)
# input Climate variables
 data(Climat)
# output variables (precalculated to speed up the example)
 data(biomasseY)
#
 DYNSI <- dynsi(2, biomasseY, biomasseX)
 summary(DYNSI)
 print(DYNSI)
 plot(DYNSI, color=heat.colors)
 #graph.bar(DYNSI,col=1, beside=F)  # sensitivity bar plot
                                    # for the first output (col=1)
 #graph.bar(DYNSI,col=2, xmax=1)    #
```

---

graph.bar *Sensitivity index bar plot*

Description

A function that plots sensitivity indices by a bar graph

Usage

```
graph.bar(x, col = 1, nb.plot = 15, xmax = NULL,
          beside = TRUE, xlab = NULL, ...)
```

Arguments

x	an object of class gsi or dynsi
col	the column number of GSI to represent in the bar graph
nb.plot	number of input factors to be considered
xmax	a user-defined maximal x value ($x \leq 1$) in all the bar graphs that show sensitivity indices; or NULL if the user wants to keep default values

beside	if TRUE, the main and total sensitivity indices are represented by two bars; if FALSE, they are represented by the same bar
xlab	a label for the x axis
...	graphical parameters

---

graph.pc *Principal Components graph for gsi objects*

Description

A function that plots the Principal Components (PCs) and the sensitivity indices on each PC

Usage

```
graph.pc(x, nb.plot = 15, nb.comp = NULL, xmax = NULL,
         beside = TRUE, cor.plot=FALSE, xtick=TRUE, type="l", ...)
```

Arguments

x	gsi object.
nb.plot	number of input factors to be considered.
nb.comp	number of PCs.
xmax	a user-defined maximal x value ($x \leq 1$) in all the bar graphs that show sensitivity indices; or NULL if the user wants to keep default values.
beside	if TRUE, the main and total sensitivity indices are represented by two bars; if FALSE, they are represented by the same bar.
cor.plot	if TRUE a correlation graph is made to represent the PCs ; if FALSE (default) a functionnal boxplot of the PCs is plotted.
xtick	if TRUE, put column names of outputs (Y) as ticks for the x axis.
type	what type of plot should be drawn for correlation graph ("l" for lines).
...	graphical parameters.

grpe.gsi	*Group factor GSI, obsolete function*

Description

An obsolete function that computed the GSI of a group factor as one factor

Usage

```
grpe.gsi(GSI, fact.interet)
```

Arguments

`GSI`	a gsi or dynsi object
`fact.interet`	input factor to be grouped

Note

This is essentially an internal function for the **multisensi** package

gsi	*Generalised Sensitivity Indices: GSI*

Description

The gsi function implements the calculation of Generalised Sensitivity Indices. This method allows to compute a synthetic Sensitivity Index for the dynamic or multivariate models by using factorial designs and the MANOVA decomposition of inertia. It computes also the Sensitivity Indices on principal components

Usage

```
gsi(formula, model, factors, inertia = 0.95, normalized = TRUE,
    cumul = FALSE, simulonly = FALSE, Name.File = NULL, ...)
```

Arguments

`formula`	ANOVA formula like `"A+B+C+A:B"` OR an integer equal to the maximum interaction order in the sensitivity model
`model`	output data.frame OR the name of the R-function which calculates the model output. The only argument of this function must be a vector containing the input factors values
`factors`	input data.frame (the design) if model is a data.frame OR a list of factors levels such as : `factor.example <- list(A=c(0,1),B=c(0,1,4))`

`inertia`	cumulated proportion of inertia (a scalar < 1) to be explained by the selected Principal components OR number of PCs to be used (e.g 3)
`normalized`	logical value. TRUE (default) computes a normalized Principal Component analysis.
`cumul`	logical value. If TRUE the PCA will be done on the cumulative outputs
`simulonly`	logical value. If TRUE the program stops after calculating the design and the model outputs
`Name.File`	optional name of a R script file containing the R-function that calculates the simulation model. e.g `"exc.ssc"`
`...`	possible fixed parameters of the model function

Details

If factors is a list of factors, the gsi function generates a complete factorial design. If it is a data.frame, gsi expects that each column is associated with an input factor.

Value

gsi returns a list of class "gsi", containing all the input arguments detailed before, plus the following components:

`X`	a data.frame containing the experimental design (input samples)
`Y`	a data.frame containing the output matrix (response)
`H`	a data.frame containing the principal components
`L`	a data.frame whose columns contain the basis eigenvectors (the variable loadings)
`lambda`	the variances of the principal components
`inertia`	vector of inertia percentages per PCs and global criterion
`cor`	a data.frame of correlation between PCs and outputs
`SI`	a data.frame containing the Sensitivity Indices (SI) on PCs and the Generalised SI (GSI)
`mSI`	a data.frame of first order SI on PCs and first order GSI
`tSI`	a data.frame containing the total SI on PCs and the total GSI
`iSI`	a data.frame of interaction SI on PCs and interaction GSI
`pred`	a data.frame containing the output predicted by the metamodel arising from the PCA and anova decompositions
`residuals`	a data.frame containing the residuals between actual and predicted outputs
`Rsquare`	vector of dynamic coefficient of determination
`Att`	0-1 matrix of association between input factors and factorial terms in the anovas
`scale`	logical value, see the arguments
`normalized`	logical value, see the arguments
`cumul`	logical value, see the arguments

`call.info`	a list containing informations on the process (reduction, analysis, fct, call)
`inputdesign`	either the input data.frame or the sensitivity object used
`outputs`	a list of results on each output variable

...

Note

This function can now be replaced by a call to the `multisensi` function. It is kept for compatibility with Version 1 of the **multisensi** package.

References

M. Lamboni, D. Makowski and H. Monod, 2009. Multivariate global sensitivity analysis for dynamic crop models. Field Crops Research, volume 113. pp. 312-320

M. Lamboni, D. Makowski and H. Monod, 2009. Multivariate sensitivity analysis to measure global contribution of input factors in dynamic models. Submitted to Reliability Engineering and System Safety.

See Also

`dynsi`, `multisensi`

Examples

```
# Test case : the Winter Wheat Dynamic Models (WWDM)
#  input factors design
 data(biomasseX)
# input climate variable
 data(Climat)
# output variables (precalculated to speed up the example)
 data(biomasseY)
#
 GSI <- gsi(2, biomasseY, biomasseX, inertia=3, normalized=TRUE, cumul=FALSE,
            climdata=Climat)
 summary(GSI)
 print(GSI)
 plot(x=GSI, beside=FALSE)
 #plot(GSI, nb.plot=4)         # the 'nb.plot' most influent factors
                               # are represented in the plots
 #plot(GSI,nb.comp=2, xmax=1)  # nb.comp = number of principal components
 #plot(GSI,nb.comp=3, graph=1) # graph=1 for first figure; 2 for 2nd one
                               # and 3 for 3rd one; or 1:3 etc.
 #graph.bar(GSI,col=1, beside=F)  # sensitivity bar plot on the first PC
 #graph.bar(GSI,col=2, xmax=1)    #
```

`multisensi` *A function with multiple options to perform multivariate sensitivity analysis*

Description

The `multisensi` function can conduct the different steps of a multivariate sensitivity analysis (design, simulation, dimension reduction, analysis, plots). It includes different options for each of these steps.

Usage

```
multisensi(design = expand.grid, model, reduction = basis.ACP,
           dimension = 0.95, center = TRUE, scale = TRUE,
           analysis = analysis.anoasg, cumul = FALSE,
           simulonly = FALSE, Name.File = NULL,
           design.args = list(), basis.args = list(),
           analysis.args = list(), ...)
```

Arguments

`design`	EITHER a function such as expand.grid to generate the design OR a data.frame of size `N x P` containing `N` combinations of levels of the `P` input factors OR a function from the **sensitivity** package such as fast99 OR an object generated by a function from the **sensitivity** package. The first and third cases require additional information to be given in the `design.args` argument.
`model`	EITHER a function to run the model simulations OR a data.frame of size `N x T` containing `N` realizations of `T` output variables.
`reduction`	EITHER a function to decompose the multivariate output on a basis of smaller dimension OR `NULL`. The first case requires additional information to be given in the `basis.args` argument. In the second case, sensitivity analyses are performed on the raw output variables.
`dimension`	EITHER the number of variables to analyse, specified by an integer or by the minimal proportion of inertia (a scalar < 1) to keep in the output decomposition OR a vector specifying a subset of columns in the output data.frame OR NULL if all variables must be analysed.
`center`	logical value. If TRUE (default value) the output variables are centred.
`scale`	logical value. If TRUE (default value) the output variables are normalized before applying the reduction function.
`analysis`	a function to run the sensitivity analysis. Additional information can be given in the `analysis.args` argument.
`cumul`	logical value. If TRUE the output variables are replaced by their cumulative sums.
`simulonly`	logical value. If TRUE the program stops after the model simulations.

Name.File	Name of file containing the R-function model.
design.args	a list of arguments for the function possibly given in the design argument.
basis.args	a list of arguments for the function given in the reduction argument. See the function help for more precision.
analysis.args	a list of arguments for the function possibly given in the analysis argument. See the function help for more precision.
...	optional parameters of the function possibly given in the model argument.

Value

an object of class dynsi if reduction=NULL, otherwise an object of class gsi. See the functions dynsi and gsi for more information.

See Also

dynsi, gsi

Examples

```
## Test case : the Winter Wheat Dynamic Models (WWDM)
#  input factors design
 data(biomasseX)
# input climate variable
 data(Climat)
# output variables (precalculated to speed up the example)
 data(biomasseY)

# to do dynsi process
# argument reduction=NULL
resD <- multisensi(design=biomasseX, model=biomasseY, reduction=NULL,
                 dimension=NULL, analysis=analysis.anoasg,
                 analysis.args=list(formula=2,keep.outputs = FALSE))
summary(resD)


# to do gsi process
#------------
# with dimension reduction by PCA
# argument reduction=basis.ACP
resG1 <- multisensi(design=biomasseX, model=biomasseY, reduction=basis.ACP,
                   dimension=0.95, analysis=analysis.anoasg,
                   analysis.args=list(formula=2,keep.outputs = FALSE))
summary(resG1)

plot(x=resG1, beside=FALSE)

#------------
# with dimension reduction by o-splines basis
# arguments reduction=basis.osplines
# and basis.args=list(knots= ... , mdegree= ... )
```

```
resG2 <- multisensi(design=biomasseX, model=biomasseY, reduction=basis.osplines,
                   dimension=NULL, center=FALSE, scale=FALSE,
                   basis.args=list(knots=11, mdegree=3), analysis=analysis.anoasg,
                   analysis.args=list(formula=2,keep.outputs = FALSE))
summary(resG2)

#------------
library(sensitivity) # to use fast99

# with dimension reduction by o-splines basis
# and sensitivity analysis with sensitivity:fast99
resG3 <- multisensi(design=fast99, model=biomasse,
                   analysis=analysis.sensitivity,
                   design.args=list(factors = names(biomasseX), n = 100,
                   q = "qunif", q.arg = list(list(min = 0.9, max = 2.8),
                   list(min = 0.9, max = 0.99), list(min = 0.6, max = 0.8),
                   list(min = 3, max = 12), list(min = 0.0035, max = 0.01),
                   list(min = 0.0011, max = 0.0025),
                   list(min = 700, max = 1100))), climdata=Climat,
                   reduction=basis.osplines,
                   basis.args=list(knots=7, mdegree=3),
                   center=FALSE,scale=FALSE,dimension=NULL)
summary(resG3)
```

`multivar` *A function to decompose the output data set and reduce its dimension*

Description

The function `multivar` applies a multivariate method to decompose the output variables on a given basis.

Usage

```
multivar(simuls, dimension = NULL, reduction, centered = TRUE,
         scale = TRUE, basis.args = list())
```

Arguments

`simuls`	a data.frame of size `N x T`, typically a set of `N` simulation outputs of length `T`
`dimension`	the number of variables to analyse, specified by an integer (for example 3) or by the minimal proportion of inertia (for example 0.95) to keep in the output decomposition
`reduction`	a function to decompose the multivariate output on a basis of smaller dimension
`centered`	logical value. If TRUE the output variables are centred.
`scale`	logical value. If TRUE the output variables are normalized.
`basis.args`	a list of arguments for the function given in the `reduction` argument. See the function help for more precision.

Value

A list containing:

`H`	a data.frame of size `N x d`, where `d` is the number of basis vectors. It contains the coefficients of the decomposition for each row of the `simuls` data.frame.
`L`	a matrix of size `T x d`. It contains the vectors of the user-defined basis.
`sdev`	standard deviations of the columns of `H`
`nbcomp`	number of components kept from the decomposition
`SStot`	total sums of squares of the simulations (after application of `centered` and `scale`)
`centering`	either 0 or the column averages of `simuls`
`scaling`	either 1 or `sdY`, depending on the `scale` argument
`sdY`	standard deviations of the columns of `simuls`
`cor`	correlation matrix (L*sdev), of size `T x nbcomp`
`scale`	kept in case the option scale has been changed in the function
`importance`	cumulated percentage of SS_H (sdev^2) with respect to SStot
`call.info`	list with the element `reduction` storing the name of the argument `reduction`

See Also

`basis.ACP`, `basis.bsplines`, `basis.poly`, `basis.osplines`

Examples

```
data(biomasseY)

res <- multivar(biomasseY, dimension=0.95, reduction=basis.ACP)
```

`planfact` *Complete factorial design in lexical order*

Description

Function that generates a complete factorial design in lexical order

Usage

```
planfact(nb.niv, make.factor = TRUE)
```

Arguments

`nb.niv`	vector containing the number of each input levels
`make.factor`	logical value. If TRUE the columns of the output are of class factor

Value

`plan`	data frame of the complete factorial design

Note

This is essentially an internal function for the **multisensi** package

---

`planfact.as` *Complete factorial design*

Description

Computation of a complete factorial design for model input factors.

Usage

```
planfact.as(input)
```

Arguments

`input`	list of factor levels

Value

`comp2`	complete factorial design of model input

Note

This is essentially an internal function for the **multisensi** package. It is almost equivalent to the function `expand.grid`.

`plot.dynsi` *Plot method for dynamic sensitivity results*

Description

Plot method for dynamic sensitivity results of class dynsi

Usage

```
## S3 method for class 'dynsi'
plot(x, normalized=FALSE, text.tuning = NULL, shade=FALSE,
                   color=NULL, xtick=TRUE, total.plot=FALSE, gsi.plot=TRUE, ...)
```

Arguments

x	a dynsi object.
normalized	logical value, FALSE => SI plotted within var(Y).
text.tuning	NULL or a small integer to improve the position of input factor labels.
shade	if TRUE, put different shadings to enhance the different factorial effects in the plot (long).
color	a palette of colors to enhance the different factorial effects in the plot (for example `color=heat.colors`).
xtick	if TRUE, put column names of outputs (Y) as ticks for the x axis.
total.plot	logical value, TRUE => a new plot is produced with the total SI.
gsi.plot	logical value, TRUE => a new plot is produced for the Generalised Sensitivity Indice.
...	graphical parameters.

Details

For labels that would be partly positioned outside the plot frame, the argument "text.tuning" may allow to get a better positioning. If it is equal to n, say, these labels are moved by n positions inside the frame, where 1 position corresponds to 1 output variable on the x-axis.

See Also

dynsi, multisensi

---

plot.gsi — *Plot method for generalised sensitivity analysis*

Description

Plot method for generalised sensitivity analysis of class gsi

Usage

```
## S3 method for class 'gsi'
plot(x, nb.plot = 10, nb.comp = 3, graph = 1:3, xmax=NULL,
                   beside=TRUE, cor.plot=FALSE, xtick=TRUE, type="l",...)
```

Arguments

x	a gsi object.
nb.plot	number of input factors to be considered.
nb.comp	number of Principal Components to be plotted.
graph	figures number: 1 or 2 or 3. 1 is for plotting the PCs and their sensitivity indices, 2 is for plotting the Generalised Sensitivity Indice, 3 is for plotting the Rsquare.

`xmax`	a user-defined maximal x value ($x \leq 1$) in all the bar graphs that show sensitivity indices; or NULL if the user wants to keep default values.
`beside`	if TRUE, the main and total sensitivity indices are represented by two bars; if FALSE, they are represented by the same bar.
`cor.plot`	if TRUE a correlation graph is made to represent the PCs ; if FALSE (default) a functionnal boxplot of the PCs is plotted.
`xtick`	if TRUE, put column names of outputs (Y) as ticks for the x axis.
`type`	what type of plot should be drawn for correlation graph ("l" for lines).
`...`	graphical parameters.

See Also

`gsi`, `multisensi`, `graph.bar`, `graph.pc`

`predict.gsi` *A function to predict multivariate output*

Description

The function `predict.gsi` generates predicted multivariate output for user-specified combinations of levels of the input factors.

Usage

```
## S3 method for class 'gsi'
predict(object, newdata, ...)
```

Arguments

`object`	Object of class gsi.
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used. need to be same factors and levels as for obtained the gsi object.
`...`	others parameters

Details

Only available if the `gsi` object was obtained with analysis.anoasg and analysis.args$keep.outputs=TRUE.

Value

a data.frame of predicted values for newdata

See Also

`gsi`, `multisensi`, `analysis.anoasg`

Examples

```
data(biomasseX)
data(biomasseY)
x=multisensi(design=biomasseX,model=biomasseY,basis=basis.ACP,
             analysis=analysis.anoasg,
             analysis.args=list(formula=2, keep.outputs=TRUE))
newdata=as.data.frame(apply(biomasseX,2,unique))
predict(x,newdata)
```

---

print.dynsi *print DYNSI*

Description

A function to print DYNSI results

Usage

```
## S3 method for class 'dynsi'
print(x, ...)
```

Arguments

x	a dynsi object
...	print parameters

See Also

dynsi, multisensi

---

print.gsi *print GSI*

Description

function to print GSI results

Usage

```
## S3 method for class 'gsi'
print(x, ...)
```

Arguments

x	a gsi object
...	print parameters

See Also

gsi, multisensi

`quality` *quality of any approximation*

Description

Function that computes the sensitivity quality after making some assumptions about the number of PCs and the number of interactions

Usage

```
quality(echsimul, echsimul.app)
```

Arguments

`echsimul`	model outputs
`echsimul.app`	Predicted model output

Value

A list with the following components:

moy.biais mean of the residuals

residuals biais

coef.det R-square

Note

This is essentially an internal function for the **multisensi** package

---

`sesBsplinesNORM` *normalized B-splines basis functions*

Description

The `sesBsplinesNORM` evaluates B-Splines basis functions at some points.

Usage

```
sesBsplinesNORM(x = seq(0, 1, len = 101), knots = 5, m = 2)
```

Arguments

`x`	vector, coordinates where to calculate the B-spline functions
`knots`	number of knots or vector of knots locations
`m`	integer, degree of the B-Splines

Value

`x`	as input
`bsplines`	matrix, values in x of all B-spline basis functions of order m
`knots`	vector of knots locations
`projecteur`	inverse matrix of `bsplines`

Note

This is essentially an internal function for the **multisensi** package

See Also

`bspline`, `basis.bsplines`

`sesBsplinesORTHONORM` *orthogonalized B-splines basis functions*

Description

The `sesBsplinesORTHONORM` evaluates O-Splines basis functions at some points.

Usage

```
sesBsplinesORTHONORM(x = seq(0, 1, len = 101), knots = 5, m = 2)
```

Arguments

`x`	vector, coordinates where to calculate the B-spline functions
`knots`	number of knots or vector of knots locations
`m`	integer, degree of the B-Splines

Value

`x`	as input
`osplines`	matrix, values in x of all O-spline basis functions of order m
`knots`	vector of knots locations
`projecteur`	inverse matrix of `osplines`

Note

This is essentially an internal function for the **multisensi** package

See Also

`bspline`, `basis.osplines`

`simulmodel` *Model simulation*

Description

Function that simulates the model outputs

Usage

```
simulmodel(model, plan, nomFic = NULL, verbose = FALSE, ...)
```

Arguments

`model`	name of R-function
`plan`	data frame of input design
`nomFic`	name of file that contains the model function
`verbose`	verbose
`...`	... possible fixed parameters of the R-function

Details

The model function must be a R-function. Models defined as functions will be called once with an expression of the form `y <- f(X)` where `X` is a vector containing a combination of levels of the input factors, and `y` is the output vector of length q, where q is the number of output variables

Value

data frame of model outputs

Note

This is essentially an internal function for the **multisensi** package

summary.dynsi *dynsi summary*

Description

Function to summarize the dynamic sensitivity results

Usage

```
## S3 method for class 'dynsi'
summary(object, ...)
```

Arguments

object	a dynsi object
...	summary parameters

See Also

dynsi, multisensi

summary.gsi *summary of GSI results*

Description

function to summarize the GSI results

Usage

```
## S3 method for class 'gsi'
summary(object, ...)
```

Arguments

object	a GSI object
...	summary parameters

See Also

gsi, multisensi

yapprox *Prediction based on PCA and anovas (NOT ONLY)*

Description

A function that predicts the model output after PCA and aov analyses

Usage

```
yapprox(multivar.obj, nbcomp = 2, aov.obj)
```

Arguments

multivar.obj	output of the multivar function
nbcomp	number of columns
aov.obj	aov object

Value

model output predictions

Note

This is essentially an internal function for the **multisensi** package

Index