Package ‘safi’

February 20, 2015

Type Package
Title Sensitivity Analysis for Functional Input
Version 1.0
Date 2014-10-28
Author Jana Fruth, Malte Jastrow
Maintainer Jana Fruth <fruth@statistik.tu-dortmund.de>
Description Design and sensitivity analysis for computer experiments with scalar-valued output and functional input, e.g. over time or space. The aim is to explore the behavior of the sensitivity over the functional domain.
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2014-10-28 17:35:33

R topics documented:

safi-package .......................................................... 2
accessSafiDesign .................................................. 2
createSafiDesign .................................................. 3
demo3steps .......................................................... 4
plot.safidesign .................................................. 5
plot.safimodel .................................................. 5
safiModel .......................................................... 6
splitSafiDesign .................................................. 8

Index 9
Description

Sensitivity analysis for functional inputs.

Author(s)

Jana Fruth, Malte Jastrow

References


accessSafiDesign  Print functional input design as convenient matrix

Description

The functional input design is printed at a chosen number of time points to allow for data transfer to the output model.

Usage

accessSafiDesign(s.d, n.timepoints)

Arguments

s.d  safidesign-object
n.timepoints  number of desired time points at which to evaluate the functions, either one value (the same number for all inputs) or a vector of length d.f (different numbers for the inputs)

Value

List of matrices of input design functions (one for each input)
createSafidesign

Examples

# 1 functional input
s.d <- createSafidesign(d.f = 1)
s.d2 <- splitSafidesign(s.d = s.d, new.split.points = list(c(0.25, 0.75)))
accessSafidesign(s.d2, n.timepoints = 20)

# 3 functional inputs
s.d <- createSafidesign(d.f = 3)
s.d2 <- splitSafidesign(s.d = s.d, new.split.points = list(c(0.5), c(0.25, 0.75),
                                                   c(0.25, 0.5, 0.75)))
accessSafidesign(s.d = s.d2, n.timepoints = c(2, 4, 10))

---

createSafidesign  Creating an initial DoE for a given number of (functional) inputs

Description

An initial design object is created.

Usage

createSafidesign(method = "SB", mirrored.runs.included = FALSE, d.f = 1, variable.names)

Arguments

method  a character string specifying the evaluation method to be chose from "SB" and
        "other". Currently only "SB", sequential bifurcation, is implemented. For
        "other" the DoE has to be added to the object manually.
mirrored.runs.included
        boolean. If TRUE mirror runs will be included in the design.
d.f  number of (functional) inputs
variable.names  optional variable names (if NULL names are set to x1,x2,...)

Value

safidesign object containing DoE, split.points and d.f,mirrored.runs.included and method

References

inputs in computer experiments, Reliability Engineering & System Safety, doi: 10.1016/j.ress.2014.07.018,
preprint on HAL: http://hal.archives-ouvertes.fr/hal-00943509.

Bettonvil, B. (1995) Factor screening by sequential bifurcation, Communications in Statistics-
Simulation and Computation, 24, 165-185.
Examples

# one input
s.d <- createSafiDesign()
s.d

# two inputs
s.d <- createSafiDesign(d.f = 2)
s.d

# including mirrored runs
s.d <- createSafiDesign(d.f = 2, mirrored.runs.included = TRUE)
s.d

demo3steps Demo of safi

Description

Demo of a complete sequential sensitivity analysis with 2 functional and 1 scalar input in three sequential steps.

Details

The procedure is performed on an artificial underlying function with two functional inputs \( g_1 \) and \( g_2 \) and one scalar input \( x \).

\[
f(g_1, g_2, x) = \int_0^{1/3} (3 - 9t) g_1(t) dt - \int_{3/10}^{1} [1/30g_2(t) + 3] dt + 8/10 \sin(x)
\]

The linear influences to detect are:

for \( g_1 \): decreasing positive influence in the beginning, then no influence

for \( g_2 \): no influence in the beginning, then constant negative influence

for \( x \): positive influence

The demo shows how the three inputs are analysed in three sequential steps in which the space of the two functional inputs is divided more and more. After the third step, it reveals the linear influences listed above.
plot.safidesign  

Plotting SAFI designs

Description

Plotting the functional inputs against the functional domain

Usage

## S3 method for class 'safidesign'
plot(x, runs = NULL, ...)

Arguments

- x  
safidesign-object
- runs  
vector of runs that shall be plotted, if not provided all runs are plotted
- ...  
optional graphical parameters

Value

One plot for each input, containing all design functions given in runs.

Examples

# 1 functional input
s.d <- createSafiDesign(d.f = 1)
plot(s.d)

# 3 functional inputs
s.d <- createSafiDesign(d.f = 3)
s.d2 <- splitSafiDesign(s.d = s.d, new.split.points = list(c(0.5), c(0.25, 0.75), 
       c(0.25, 0.5, 0.75)))
plot(s.d2)
plot(s.d2, runs = 1:2)  # only first two runs

plot.safimodel  

Plotting SAFI model coefficients

Description

Barplots of the functional sensitivities from a safimodel object.

Usage

## S3 method for class 'safimodel'
plot(x, ylim = NULL, ...)

Arguments

- `x` safimodel object
- `ylim` limits for the y-axis. If NULL, limits are computed internally.
- `...` optional graphical parameters

Value

One plot for each functional input.

References


Examples

### simple example

```r
s.d <- createSafiDesign(d.f = 1)
s.d2 <- splitSafiDesign(s.d = s.d, new.split.points = list(c(0.25, 0.75)))

# artificial model output (rising influence)
x <- accessSafiDesign(s.d = s.d2, n.timepoints = 4)
y <- x$x1 * c(0, 1, 2, 3)
s.m <- safimodel(s.d = s.d2, y = y)
plot(s.m)
```

### d.f = 3, mirrored

```r
s.d <- createSafiDesign(d.f = 3, mirrored.runs.included = TRUE)
s.d2 <- splitSafiDesign(s.d, list(c(0.5), c(0.25, 0.75), c(0.25, 0.5, 0.75)))

# artificial model output (x1 without influence, x2 rising, x3 falling)
x <- accessSafiDesign(s.d = s.d2, n.timepoints = 4)
y <- x$x2 * c(0, 1, 2, 3) + x$x3 * c(0, -1, -2, -3)
s.m <- safimodel(s.d2, y = y)
plot(s.m)
```

---

**safiModel**

*Functional sensitivity analysis*

**Description**

computes normalized regression indices for the sensitivity analysis of functional inputs
Usage

```r
safiModel(s.d, y)
```

Arguments

- `s.d` safidesign-object
- `y` model response

Details

If the design was created with method "SB" the coefficients are computed via sequential bifurcation, for method "other" via least squares estimation.

Value

`safiModel` object containing the design and the computed coefficients

References


Examples

```r
### simple example
s.d <- createsafidesign(d.f = 1)
s.d2 <- splitsafidesign(s.d = s.d, new.split.points = list(c(0, 0.75)))

# artificial model output (rising influence)
x <- accesssafidesign(s.d = s.d2, n.timepoints = 4)
y <- x$x1 * c(0, 1, 2, 3)
s.m <- safimodel(s.d = s.d2, y = y)
plot(s.m)

### d.f = 3, mirrored
s.d <- createsafidesign(d.f = 3, mirrored.runs.included = TRUE)
s.d2 <- splitsafidesign(s.d, list(c(0.5, 0.75), c(0.25, 0.5), c(0.25, 0.5, 0.75)))

# artificial model output (x1 without influence, x2 rising, x3 falling)
x <- accesssafidesign(s.d = s.d2, n.timepoints = 4)
y <- x$2 * c(0, 1, 2, 3) + x$3 * c(0, -1, -2, -3)
s.m <- safimodel(s.d2, y = y)
plot(s.m)
```
splitSafiDesign  

### Description

The functional domains of the input functions are split up at given points and a corresponding DoE is generated.

### Usage

```r
splitSafiDesign(s.d, new.split.points)
```

### Arguments

- `s.d`  
  safidesign object  
  - `new.split.points`  
    list of new split points

### Details

For method "other" no runs are added.

### Value

safidesign-object

### References


### Examples

```r
# two inputs
s.d <- createSafiDesign(d.f = 2)
s.d
s.d2 <- splitSafiDesign(s.d = s.d, new.split.points = list(c(0.5), c(0.25, 0.75)))
s.d2

# including mirrored runs
s.d <- createSafiDesign(d.f = 2, mirrored.runs.included = TRUE)
s.d
s.d2 <- splitSafiDesign(s.d, list(c(0.5), c(0.25, 0.75)))
s.d2
```
Index

*Topic `textasciitildekwd1`
createSafiDesign, 3
splitSafiDesign, 8

*Topic `textasciitildekwd2`
createSafiDesign, 3
splitSafiDesign, 8

*Topic `datasets`
demo3steps, 4

*Topic `package`
safi-package, 2

accessSafiDesign, 2
createSafiDesign, 3
demo3steps, 4
plot.safidesign, 5
plot.safimodel, 5

safi (safi-package), 2
safi-package, 2
safiModel, 6
splitSafiDesign, 8