Package ‘DiceView’

February 19, 2015

Title Plot methods for computer experiments design and surrogate.
Version 1.3-1
Date 2012-03-29
Author Yann Richet, Yves Deville, Clement Chevalier
Maintainer Yann Richet <yann.richet@irsn.fr>
Description View 2D/3D sections or contours of computer experiments designs, surrogates or test functions.
Depends methods, DiceKriging, DiceEval, rgl
License GPL-3
URL http://promethee.irsn.org/
Repository CRAN
NeedsCompilation no
Date/Publication 2013-11-15 15:04:01

R topics documented:

contourview ................................................................. 2
contourview.fun ......................................................... 3
contourview.km ............................................................ 4
contourview.list ......................................................... 6
sectionview ................................................................. 8
sectionview.fun .......................................................... 9
sectionview.km ............................................................ 10
sectionview.list .......................................................... 12
sectionview3d ............................................................... 14
sectionview3d.fun ......................................................... 15
sectionview3d.km .......................................................... 16
sectionview3d.list ......................................................... 18
view .............................................................................. 19

Index 21
contourview

Plot a contour view of a kriging or modelPredict model including design points, or a function.

Description

Plot a contour view of a kriging or modelPredict model. It is useful for a better understanding of a model behaviour.

Usage

contourview(model, ...)

Arguments

model       an object of class "km", a list that can be used in a "modelPredict" call, or a function.

...         other arguments of the contourview.km, contourview.list or contourview.fun function

Author(s)

Yann Richet, IRSN

See Also

sectionview3d

Examples

## A 2D example - Branin-Hoo function
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design факт <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design факт <- data.frame(design факт); names(design факт) <- c("x1", "x2")
y <- branin(design факт)

## kriging model 1 : matern5_2 covariance structure, no trend, no nugget effect
m1 <- km(design = design факт, response = y)

contourview(m1)

contourview(branin, dim = 2, add=TRUE)
**Description**

Plot a contour view of a function.

**Usage**

```r
contourview.fun(fun,
    dim = ifelse(is.null(center), 2, length(center)),
    center = NULL, axis = NULL, npoints = 20, nlevels = 10,
    col = "blue", filled = FALSE, mfrow = NULL,
    Xname = NULL, yname = NULL, Xscale = 1, yscale = 1,
    xlim = c(0, 1), ylim = NULL, title = NULL, add = FALSE,
    ...)
```

**Arguments**

- `fun`: an object of class "function".
- `dim`: the dimension of fun arguments.
- `center`: optional coordinates (as a list or data frame) of the center of the section view if the model's dimension is > 2.
- `axis`: optional matrix of 2-axis combinations to plot, one by row. The value `NULL` leads to all possible combinations i.e. `choose(D, 2)`.
- `npoints`: an optional number of points to discretize plot of response surface and uncertainties.
- `col`: color for the surface.
- `filled`: use `filled.contour`.
- `nlevels`: number of contour levels to display.
- `mfrow`: an optional list to force `par(mfrow = ...) call. The default value `NULL` is automatically set for compact view.
- `xlim`: a list to give x range for all plots.
- `ylim`: an optional list to force y range for all plots.
- `Xname`: an optional list of string to overload names for X.
- `yname`: an optional string to overload name for y.
- `Xscale`: an optional factor to scale X.
- `yscale`: an optional factor to scale y.
- `title`: an optional overload of main title.
- `add`: to print graphics on an existing window.
- `...`: further arguments passed to the first call of `plot3d`.
Details

Experimental points are plotted with fading colors. Points that fall in the specified section (if any) have the color specified `col_points` while points far away from the center have shaded versions of the same color. The amount of fading is determined using the Euclidean distance between the plotted point and center. The variables chosen with their number are to be found in the `X` slot of the model. Thus they are 'spatial dimensions' but not 'trend variables'.

Author(s)

Yann Richet, IRSN

See Also

See `sectionview3d.fun`.

Examples

```r
## A 2D example - Branin-Hoo function.
contourview.fun(branin, dim = 2)
```

Description

Plot a contour view of a kriging model: mean response surface, fitted points and confidence surfaces. Provide a better understanding of the kriging model behaviour.

Usage

```r
contourview.km(model, type = "UK", center = NULL,
axis = NULL, npoints = 20, nlevels = 10,
col_points = "red", col_surf = "blue", filled = FALSE,
bg_blend = 1, mfrow = NULL, Xname = NULL, yname = NULL,
Xscale = 1, yscale = 1, xlim = NULL, ylim = NULL,
title = NULL, add = FALSE, ...)
```

Arguments

- `model`: an object of class "km".
- `type`: the kriging type to use for model prediction.
- `center`: optional coordinates (as a list or data frame) of the center of the section view if the model’s dimension is > 2.
- `axis`: optional matrix of 2-axis combinations to plot, one by row. The value NULL leads to all possible combinations i.e. choose(D, 2).
contourview.km

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>npoints</td>
<td>an optional number of points to discretize plot of response surface and uncertainties.</td>
</tr>
<tr>
<td>col_points</td>
<td>color of points.</td>
</tr>
<tr>
<td>col_surf</td>
<td>color for the surface.</td>
</tr>
<tr>
<td>filled</td>
<td>use filled.contour</td>
</tr>
<tr>
<td>nlevels</td>
<td>number of contour levels to display.</td>
</tr>
<tr>
<td>bg_blend</td>
<td>an optional factor of alpha (color channel) blending used to plot design points outside from this section.</td>
</tr>
<tr>
<td>mfrow</td>
<td>an optional list to force \texttt{par(mfrow = \ldots)} call. The default value \texttt{NULL} is automatically set for compact view.</td>
</tr>
<tr>
<td>xlim</td>
<td>an optional list to force x range for all plots. The default value \texttt{NULL} is automatically set to include all design points.</td>
</tr>
<tr>
<td>ylim</td>
<td>an optional list to force y range for all plots. The default value \texttt{NULL} is automatically set to include all design points (and their 1-99 percentiles).</td>
</tr>
<tr>
<td>Xname</td>
<td>an optional list of string to overload names for X.</td>
</tr>
<tr>
<td>yname</td>
<td>an optional string to overload name for y.</td>
</tr>
<tr>
<td>Xscale</td>
<td>an optional factor to scale X.</td>
</tr>
<tr>
<td>yscale</td>
<td>an optional factor to scale y.</td>
</tr>
<tr>
<td>title</td>
<td>an optional overload of main title.</td>
</tr>
<tr>
<td>add</td>
<td>to print graphics on an existing window.</td>
</tr>
<tr>
<td>...</td>
<td>further arguments passed to the first call of \texttt{plot3d}.</td>
</tr>
</tbody>
</table>

Details

Experimental points are plotted with fading colors. Points that fall in the specified section (if any) have the color specified \texttt{col_points} while points far away from the center have shaded versions of the same color. The amount of fading is determined using the Euclidean distance between the plotted point and \texttt{center}. The variables chosen with their number are to be found in the \texttt{X} slot of the model. Thus they are 'spatial dimensions' but not 'trend variables'.

Note

The confidence bands are computed using normal quantiles and the standard error given by \texttt{predict.km}.

Author(s)

Yann Richet, IRSN

See Also

See \texttt{sectionview3d.km} and the \texttt{km} function in the \texttt{DiceKriging} package.
Examples

## a Rd example - Branin-Hoo function. See DiceKriging package manual
## a 16-points factorial design, and the corresponding response

d <- 2; n <- 16

design_fact <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design_fact <- data.frame(design_fact); names(design_fact)<-c("x1", "x2")
y <- branin(design_fact)

## kriging model 1: matern5_2 covariance structure, no trend, no nugget effect

m1 <- km(design = design_fact, response = y)

## the same as contourview.km
contourview(m1)

## change colors
contourview(m1, col_points = "firebrick", col_surf = "SpringGreen2")

## change colors, use finer grid and add needles
contourview(m1, npoints = c(50, 30), col_points = "orange",
col_surf = "SpringGreen2")

## Display reference function
contourview(branin,dim=2,add=TRUE,col='red')

---

**contourview.list**  
*Plot a contour view of a model, including design points*

---

### Description

Plot a contour view of a model, thus providing a better understanding of its behaviour.

### Usage

```r
contourview.list(model, center = NULL, axis = NULL, 
npoints = 20, nlevels = 10, col_points = "red",
col_surf = "blue", filled = FALSE, bg_blend = 1, 
mfrow = NULL, Xname = NULL, yname = NULL, Xscale = 1, 
yscale = 1, xlim = NULL, ylim = NULL, title = NULL, 
add = FALSE, ...)
```

### Arguments

- **model**: a list that can be used in the modelPredict function of the **DiceEval** package.
- **center**: optional coordinates (as a list or data frame) of the center of the section view if the model’s dimension is > 2.
- **axis**: optional matrix of 2-axis combinations to plot, one by row. The value NULL leads to all possible combinations i.e. choose(D, 2).
contourview.list

- **npoints**: an optional number of points to discretize plot of response surface and uncertainties.
- **col_points**: color of points.
- **col_surf**: color for the surface.
- **filled**: use filled.contour
- **nlevels**: number of contour levels to display.
- **mfrow**: an optional list to force \texttt{par(mfrow = \ldots)} call. The default value \texttt{NULL} is automatically set for compact view.
- **bg_blend**: an optional factor of alpha (color channel) blending used to plot design points outside from this section.
- **xlim**: an optional list to force x range for all plots. The default value \texttt{NULL} is automatically set to include all design points.
- **ylim**: an optional list to force y range for all plots. The default value \texttt{NULL} is automatically set to include all design points.
- **xname**: an optional list of string to overload names for X.
- **yname**: an optional string to overload name for y.
- **xscale**: an optional factor to scale X.
- **yscale**: an optional factor to scale y.
- **title**: an optional overload of main title.
- **add**: to print graphics on an existing window.
- **...**: optional arguments passed to the first call of \texttt{plot3d}.

**Details**

Experimental points are plotted with fading colors. Points that fall in the specified section (if any) have the color specified \texttt{col_points} while points far away from the center have shaded versions of the same color. The amount of fading is determined using the Euclidean distance between the plotted point and center. The variables chosen with their number are to be found in the dataDx element of the model. Thus they are original data variables but not trend variables that may have been created using the model's formula.

**Author(s)**

Yann Richet, IRSN

**See Also**

- \texttt{sectionview.list} for a 2D plot, and the \texttt{modelPredict} function in the \texttt{DiceEval} package. The \texttt{sectionview3d.km} produces a similar plot for km objects.
Examples

```r
## A 2D example - Branin-Hoo function
## a 16-points factorial design, and the corresponding response
n <- 16
design факт <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design факт <- data.frame(design факт); names(design факт) <- c("x1", "x2")
y <- branin(design факт)

## linear model
m1 <- modelFit(design факт, y$x1, type = "Linear", formula = "y-.")

## the same as sectionview3d.list
contourview(m1)
```

---

sectionview

Plot a section view of a kriging or modelPredict model including design points, or a function.

Description

Plot one section view per dimension of a kriging, modelPredict model or function. It is useful for a better understanding of a model behaviour (including uncertainty).

Usage

```r
sectionview(model, ...)
```

Arguments

- `model` an object of class "km", a list that can be used in a "modelPredict" call, or a function.
- `...` other arguments of the `contourview.km`, `contourview.list` or `contourview.fun` function

Author(s)

Yann Richet, IRSN

See Also

See the documentation of `sectionview.km`, `sectionview.list`, or `sectionview.fun` for the arguments.

The `sectionview3d` method provides a 3D version.
Examples

```r
## A 2D example - Branin-Hoo function
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design факт <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design факт <- data.frame(design факт); names(design факт) <- c("x1", "x2")
y <- branin(design факт)

## Kriging model 1: matern5.2 covariance structure, no trend, no nugget effect
m1 <- km(design = design факт, response = y)

sectionview(m1, center = c(.333, .333))
sectionview(branin, dim = 2, center = c(.333, .333), add = TRUE)
```

---

**sectionview.fun**

---

Plot one section view per dimension of a function thus providing a better understanding of the model behaviour.

**Usage**

```r
sectionview.fun(fun, 
                 dim = ifelse(is.null(center), 1, length(center)),
                 center = NULL, axis = NULL, npoints = 100,
                 col_surf = "blue", mfrow = NULL, Xname = NULL, 
                 yname = NULL, Xscale = 1, yscale = 1, xlim = c(0, 1),
                 ylim = NULL, title = NULL, add = FALSE, ...)
```

**Arguments**

- **fun**: an object of class "function".
- **dim**: the dimension of fun arguments.
- **center**: optional coordinates (as a list or data frame) of the center of the section view if the model's dimension is > 1.
- **axis**: optional matrix of 1-axis combinations to plot, one by row. The value NULL leads to all possible combinations i.e. 1:D.
- **npoints**: an optional number of points to discretize plot of response surface and uncertainties.
- **col_surf**: color for the section.
- **mfrow**: an optional list to force `par(mfrow = ...) call. The default value NULL is automatically set for compact view.
xlim a list to give x range for all plots.
ylim an optional list to force y range for all plots.
Xname an optional list of string to overload names for X.
yname an optional string to overload name for y.
Xscale an optional factor to scale X.
yscale an optional factor to scale y.
title an optional overload of main title.
add to print graphics on an existing window.
... further arguments passed to the first call of plot.

Details
A multiple rows/columns plot is produced.

Author(s)
Yann Richet, IRSN

See Also
The function `sectionview3d.fun` produces a 3D version.

Examples
```r
## A 2D example - Branin-Hoo function.
sectionview.fun(branin, center=c(.5,.5))
```

---

**sectionview.km**

*Plot section views of a kriging model, including design points*

Description
Plot one section view per dimension of a kriging model thus providing a better understanding of the model behaviour including uncertainty.

Usage
```r
sectionview.km(model, type = "UK", center = NULL,
    axis = NULL, npoints = 100, col_points = "red",
    col_surf = "blue",
    conf_lev = c(0.5, 0.8, 0.9, 0.95, 0.99),
    conf_blend = NULL, bg_blend = 5, mfrow = NULL,
    Xname = NULL, yname = NULL, Xscale = 1, yscale = 1,
    xlim = NULL, ylim = NULL, title = NULL, add = FALSE,
    ...)
```
Arguments

- **model**: an object of class "km".
- **type**: the kriging type to use for model prediction.
- **center**: optional coordinates (as a list or data frame) of the center of the section view if the model's dimension is > 1.
- **axis**: optional matrix of 1-axis combinations to plot, one by row. The value NULL leads to all possible combinations i.e. 1:D.
- **npoints**: an optional number of points to discretize plot of response surface and uncertainties.
- **col_points**: color of points.
- **col_surf**: color for the section.
- **conf_lev**: an optional list of confidence interval values to display.
- **conf_blend**: an optional factor of alpha (color channel) blending used to plot confidence intervals.
- **bg_blend**: an optional factor of alpha (color channel) blending used to plot design points outside from this section.
- **mfrow**: an optional list to force `par(mfrow = ...)` call. The default value NULL is automatically set for compact view.
- **xlim**: an optional list to force x range for all plots. The default value NULL is automatically set to include all design points.
- **ylim**: an optional list to force y range for all plots. The default value NULL is automatically set to include all design points (and their 1-99 percentiles).
- **xname**: an optional list of string to overload names for X.
- **yname**: an optional string to overload name for y.
- **xscale**: an optional factor to scale X.
- **yscale**: an optional factor to scale y.
- **title**: an optional overload of main title.
- **add**: to print graphics on an existing window.
- **...**: further arguments passed to the first call of `plot`.

Details

A multiple rows/columns plot is produced. Experimental points are plotted with fading colors. Points that fall in the specified section (if any) have the color specified `col_points` while points far away from the center have shaded versions of the same color. The amount of fading is determined using the Euclidean distance between the plotted point and center.

Author(s)

Yann Richet, IRSN
See Also

The function `sectionview3d.km` produces a 3D version. For more information on the km class, see the km function in the DiceKriging package.

Examples

```r
## A 2D example - Branin-Hoo function
## a 16-points factorial design, and the corresponding response
n <- 2; d <- 16
design.fact <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design.fact <- data.frame(design.fact); names(design.fact)$<-(c("x1", "x2"))
y <- branin(design.fact)

## kriging model 1 : matern5_2 covariance structure, no trend, no nugget effect
m1 <- km(design = design.fact, response = y)

## display reference function
sectionview(m1, center = c(.333, .333))

## Display reference function
sectionview(branin,dim=2,center=c(.333, .333),add=TRUE,col='red')
```

sectionview.list

Plot a section view of a model, including design points

Description

Plot one section view per dimension of a surrogate model. It is useful for a better understanding of a model behaviour.

Usage

`sectionview.list(model, center = NULL, axis = NULL, npoints = 100, col_points = "red", col_surf = "blue", bg_blend = 5, mfrow = NULL, Xname = NULL, yname = NULL, xscale = 1, yscale = 1, xlim = NULL, ylim = NULL, title = NULL, add = FALSE, ...)`

Arguments

- `model`: a list that can be used as model with the `modelPredict` function of the DiceEval package.
- `center`: optional coordinates (as a list or data frame) of the center of the section view if the model’s dimension is > 1.
- `axis`: optional matrix of 1-axis combinations to plot, one by row. The value NULL leads to all possible combinations i.e. 1:D.
- `npoints`: an optional number of points to discretize plot of response surface and uncertainties.
sectionview.list

- **col_points**: color of points.
- **col_surf**: color for the section.
- **bg_blend**: an optional factor of alpha (color channel) blending used to plot design points outside from this section.
- **mfrow**: an optional list to force `par(mfrow = ...)` call. Default (NULL value) is automatically set for compact view.
- **xlim**: an optional list to force x range for all plots. The default value NULL is automatically set to include all design points.
- **ylim**: an optional list to force y range for all plots. The default value NULL is automatically set to include all design points.
- **xname**: an optional list of string to overload names for X.
- **yname**: an optional string to overload name for y.
- **xscale**: an optional factor to scale X.
- **yscale**: an optional factor to scale y.
- **title**: an optional overload of main title.
- **add**: to print graphics on an existing window.
- **...**: optional arguments passed to the first call of plot().

### Details
A multiple rows/columns plot is produced. Experimental points are plotted with fading colors. Points that fall in the specified section (if any) have the color specified `col_points` while points far away from the center have shaded versions of the same color. The amount of fading is determined using the Euclidean distance between the plotted point and center.

### Author(s)
Yann Richet, IRSN

### See Also
See `sectionview3d.list` for a 3d version, and the `modelPredict` function in the `DiceEval` package.

### Examples
```r
## A 2D example: Branin-Hoo function. See the DiceKriging package manual
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design.fact <- data.frame(design.fact); names(design.fact) <- c("x1", "x2")
y <- branin(design.fact)

## linear model
m1 <- modelFit(design.fact, y$x1, type = "Linear", formula = "Y~."

sectionview.list(m1, center = c(.333,.333))
```
sectionview3d

Plot a 3-D (using RGL) view of a kriging or modelPredict model, including design points

Description

Plot a 3-D view of a kriging or modelPredict model. It is useful for a better understanding of a model behaviour.

Usage

sectionview3d(model, ...)

Arguments

model an object of class "km", a list that can be used in a "modelPredict" call, or a function.

... other arguments of the sectionview3d.km, sectionview3d.list or sectionview3d.fun function

Author(s)

Yann Richet, IRSN

See Also

sectionview

Examples

## a 2D example - Branin-Hoo function. See DiceKriging package manual
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design.fact <- data.frame(design.fact); names(design.fact)<-c("x1", "x2")
y <- branin(design.fact)

## kriging model 1: maternS_2 covariance structure, no trend, no nugget effect
m1 <- km(design = design.fact, response = y)

## the same as sectionview3d.km
sectionview3d(m1)

sectionview3d(branin, dim = 2, add = TRUE)
sectionview3d.fun

Plot a 3-D (using RGL) view of a function

Description

Plot a 3-D view of a function. Provide a better understanding of the model behaviour.

Usage

sectionview3d.fun(fun, 
  dim = ifelse(is.null(center), 2, length(center)), 
  center = NULL, axis = NULL, npoints = 20, col = "blue", 
  Xname = NULL, yname = NULL, Xscale = 1, yscale = 1, 
  xlim = c(0, 1), ylim = NULL, title = NULL, add = FALSE, 
  ...)

Arguments

fun an object of class "function".
dim the dimension of fun arguments.
center optional coordinates (as a list or data frame) of the center of the section view if 
the model’s dimension is > 2.
axis optional matrix of 2-axis combinations to plot, one by row. The value NULL leads 
to all possible combinations i.e. choose(0, 2).
npoints an optional number of points to discretize plot of response surface and uncer-
tainties.
col color for the surface.
xlim a list to give x range for all plots.
ylim an optional list to force y range for all plots.
Xname an optional list of string to overload names for X.
yname an optional string to overload name for y.
Xscale an optional factor to scale X.
yscale an optional factor to scale y.
title an optional overload of main title.
add to print graphics on an existing window.
... further arguments passed to the first call of plot3d.

Author(s)

Yann Richet, IRSN

See Also

sectionview
Examples
### A 2D example - Branin-Hoo function.
```r
sectionview3d.fun(branin, dim = 2)
```

---

sectionview3d.km

Plot a 3-D (using RGL) view of a kriging model, including design points

Description

Plot a 3-D view of a kriging model: mean response surface, fitted points and confidence surfaces. Provide a better understanding of the kriging model behaviour.

Usage

```r
sectionview3d.km(model, type = "UK", center = NULL,
    axis = NULL, npoints = 20, col_points = "red",
    col_surf = "blue", col_needles = NA,
    conf_lev = c(0.95), conf_blend = NULL, bg_blend = 5,
    Xname = NULL, yname = NULL, xscale = 1, yscale = 1,
    xlim = NULL, ylim = NULL, title = NULL, add = FALSE,
    ...)
```

Arguments

- `model`: an object of class "km".
- `type`: the kriging type to use for model prediction.
- `center`: optional coordinates (as a list or data frame) of the center of the section view if the model’s dimension is > 2.
- `axis`: optional matrix of 2-axis combinations to plot, one by row. The value NULL leads to all possible combinations i.e. choose(2, 2).
- `npoints`: an optional number of points to discretize plot of response surface and uncertainties.
- `col_points`: color of points.
- `col_surf`: color for the surface.
- `col_needles`: color of "needles" for the points. The default NA corresponds to no needle plotted. When a valid color is given, needles are plotted using the same fading mechanism as for points.
- `conf_lev`: an optional list of confidence interval values to display.
- `conf_blend`: an optional factor of alpha (color channel) blending used to plot confidence intervals.
- `bg_blend`: an optional factor of alpha (color channel) blending used to plot design points outside from this section.
sectionview3d.km

xlim an optional list to force x range for all plots. The default value NULL is automatically set to include all design points.

ylim an optional list to force y range for all plots. The default value NULL is automatically set to include all design points (and their 1-99 percentiles).

Xname an optional list of string to overload names for X.

yname an optional string to overload name for y.

Xscale an optional factor to scale X.

yscale an optional factor to scale y.

title an optional overload of main title.

add to print graphics on an existing window.

... further arguments passed to the first call of plot3d.

Details

Experimental points are plotted with fading colors. Points that fall in the specified section (if any) have the color specified col_points while points far away from the center have shaded versions of the same color. The amount of fading is determined using the Euclidean distance between the plotted point and center. The variables chosen with their number are to be found in the X slot of the model. Thus they are 'spatial dimensions' but not 'trend variables'.

Note

The confidence bands are computed using normal quantiles and the standard error given by predict.km.

Author(s)

Yann Richet, IRSN

See Also

See sectionview.km and the km function in the DiceKriging package.

Examples

## A 2D example - Branin-Hoo function. See DiceKriging package manual
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design.fact <- data.frame(design.fact); names(design.fact)<-c("x1", "x2")
y <- branin(design.fact)

## kriging model 1 : matern5_2 covariance structure, no trend, no nugget effect
m1 <- km(design = design.fact, response = y)

## the same as sectionview3d.km
sectionview3d(m1)
## change colors

```r
sectionview3d(m1, col_points = "firebrick", col_surf = "SpringGreen2")
```

## change colors, use finer grid and add needles

```r
sectionview3d(m1, npoints = c(50, 30), col_points = "orange",
              col_surf = "SpringGreen2", col_needles = "firebrick")
```

---

### `sectionview3d.list`

Plot a 3-D (using RGL) view of a model, including design points

---

**Description**

Plot a 3-D view of a model, thus providing a better understanding of its behaviour.

**Usage**

```r
sectionview3d.list(model, center = NULL, axis = NULL,
                   npoints = 20, col_points = "red", col_surf = "blue",
                   col_needles = NA, bg_blend = 5, Xname = NULL,
                   yname = NULL, Xscale = 1, yscale = 1, xlim = NULL,
                   ylim = NULL, title = NULL, add = FALSE, ...)
```

**Arguments**

- `model`: a list that can be used in the `modelPredict` function of the `DiceEval` package.
- `center`: optional coordinates (as a list or data frame) of the center of the section view if the model's dimension is > 2.
- `axis`: optional matrix of 2-axis combinations to plot, one by row. The value `NULL` leads to all possible combinations i.e. `choose(D, 2)`.
- `npoints`: an optional number of points to discretize plot of response surface and uncertainties.
- `col_points`: color of points.
- `col_surf`: color for the surface.
- `col_needles`: color of "needles" for the points. The default `NA` corresponds to no needle plotted. When a valid color is given, needles are plotted using the same fading mechanism as for points.
- `bg_blend`: an optional factor of alpha (color channel) blending used to plot design points outside from this section.
- `xlim`: an optional list to force x range for all plots. The default value `NULL` is automatically set to include all design points.
- `ylim`: an optional list to force y range for all plots. The default value `NULL` is automatically set to include all design points.
- `Xname`: an optional list of string to overload names for X.
- `yname`: an optional string to overload name for y.
view

Xscale  an optional factor to scale X.
yscale  an optional factor to scale y.
title  an optional overload of main title.
add  to print graphics on an existing window.
...  optional arguments passed to the first call of plot3d.

Details

Experimental points are plotted with fading colors. Points that fall in the specified section (if any) have the color specified col_points while points far away from the center have shaded versions of the same color. The amount of fading is determined using the Euclidean distance between the plotted point and center. The variables chosen with their number are to be found in the data$x element of the model. Thus they are original data variables but not trend variables that may have been created using the model's formula.

Author(s)

Yann Richet, IRSN

See Also

sectionview.list for a 2D plot, and the modelPredict function in the DiceEval package. The sectionview3d.km produces a similar plot for km objects.

Examples

```r
## A 2D example - Branin-Hoo function
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design.fact <- data.frame(design.fact); names(design.fact) <-c("x1", "x2")
y <- branin(design.fact)

## linear model
m1 <- modelFit(design.fact, y$x1, type = "Linear", formula = "y~.")

## the same as sectionview3d.list
sectionview3d(m1)
```

Description

Plot a view of a kriging, modelPredict model or function. It is useful for a better understanding of a model behaviour. This function is just a wrapping of all other plotting functions (section, contour, section3d), for all supported types (km, list, function).
Usage

```r
view(type = "auto", model, ...)
```

Arguments

- **type**: a string to describe the type of view to display: "auto", "section", "xy", "section3d", "3d", "contour".
- **model**: an object of class "km", a list that can be used in a "modelPredict" call, or a function.
- **...**: other arguments of the `sectionview`, `sectionview3d` or `contourview` function

Author(s)

Yann Richet, IRSN

See Also

`sectionview`, `sectionview3d`, `contourview`

Examples

```r
## A 2D example - Branin-Hoo function. See DiceKriging package manual
## a 16-points factorial design, and the corresponding response
n <- 2; d <- 16
design.fact <- expand.grid(seq(0, 1, length = 4), seq(0, 1, length = 4))
design.fact <- data.frame(design.fact); names(design.fact)<-c("x1", "x2")
y <- branin(design.fact)

## kriging model 1 : matern5_2 covariance structure, no trend, no nugget effect
m1 <- km(design = design.fact, response = y)

## the same as sectionview3d
view("3d",m1)
view("3d",branin, dim = 2, col='red', add = TRUE)
```
Index

*Topic models
  contourview, 2
  contourview.fun, 3
  contourview.km, 4
  contourview.list, 6
  sectionview.fun, 9
  sectionview.km, 10
  sectionview.list, 12
  sectionview3d, 14
  sectionview3d.km, 16
  sectionview3d.list, 18
  view, 19

contourview, 2, 20
contourview, function-method
  (contourview), 2
contourview, km-method (contourview), 2
contourview, list-method (contourview), 2
contourview.fun, 3
contourview.km, 4
contourview.list, 6

km, 5, 12, 17

modelPredict, 7, 13, 19

sectionview, 8, 14, 15, 20
sectionview, function-method
  (sectionview), 8
sectionview, km-method (sectionview), 8
sectionview, list-method (sectionview), 8
sectionview.fun, 8, 9
sectionview.km, 8, 10, 17
sectionview.list, 7, 8, 12, 19
sectionview3d, 2, 8, 14, 20
sectionview3d, function-method
  (sectionview3d), 14
sectionview3d, km-method
  (sectionview3d), 14
sectionview3d, list-method
  (sectionview3d), 14

sectionview3d.fun, 4, 10, 15
sectionview3d.km, 5, 7, 12, 16, 19
sectionview3d.list, 13, 18

view, 19
view, character, function-method (view), 19
view, character, km-method (view), 19
view, character, list-method (view), 19