

Package ‘DiceDesign’

January 2, 2012

Type Package

Title Designs of Computer Experiments

Version 1.1

Date 2011-08-29

Author Jessica Franco, Delphine Dupuy and Olivier Roustant.

Maintainer D. Dupuy <dupuy@emse.fr>

Description Space-Filling Designs and Uniformity Criteria.

License GPL-3

Suggests rgl, randtoolbox, lattice

Encoding latin1

URL <http://emse.dice.fr/>

Repository CRAN

Date/Publication 2011-09-04 05:18:03

R topics documented:

DiceDesign-package	2
coverage	4
discrepancyCriteria	5
dmaxDesign	6
factDesign	8
greenwood.table	9
meshRatio	9
mindist	10
OA131	12
OA131_scrambled	13
rss2d	13

rss3d	16
runif.faure	18
straussDesign	19
unif.test.quantile	21
unif.test.statistic	22

Index	24
--------------	-----------

DiceDesign-package *Designs of Computer Experiments*

Description

Space-Filling Designs (SFD) and criteria of uniformity.

Details

Package: DiceDesignGamma
 Type: Package
 Version: 1.1
 Date: 2011-08-29
 License: GPL-3

This package provides tools to create some specific Space-Filling Design (SFD) and to test their quality:

- Strauss SFD and Maximum entropy SFD,
- Discrepancies criteria, distance measures,
- Radial scanning statistic

Note

This work was conducted within the frame of the DICE (Deep Inside Computer Experiments) Consortium between ARMINES, Renault, EDF, IRSN, ONERA and TOTAL S.A. (<http://emse.dice.fr/>).

In this package only Faure's sequence is implemented. Note that the **randtoolbox** package provides the following quasi random sequences: the Sobol sequence, the Halton (hence Van Der Corput) sequence and the Torus sequence (also known as Kronecker sequence).

We refer to the **lhs** package to construct Latin Hypercube Designs.

Author(s)

J. Franco, D. Dupuy and O. Roustant. Thanks to A. Jourdan for discussions about OA131.

(maintainer: D. Dupuy <dupuy@emse.fr>)

References

- Fang K.-T., Li R. and Sudjianto A. (2006) Design and Modeling for Computer Experiments, *Chapman & Hall*.
- Santner T.J., Williams B.J. and Notz W.I. (2003) The Design and Analysis of Computer Experiments, *Springer*, 121-161.
- Roustant O., Franco J., Carraro L., Jourdan A. (2010), A radial scanning statistic for selecting space-filling designs in computer experiments, MODA-9 proceedings, <http://www.emse.fr/~roustant>

Examples

```
# *****
# Designs of experiments
# *****

# A maximum entropy design with 20 points in [0,1]^2
p <- dmaxDesign(20,2,0.9,200)
plot(p$design,xlim=c(0,1),ylim=c(0,1))

# *****
# Criteria: L2-discrepancy
# *****
dp <- discrepancyCriteria(p$design,type=c('L2','C2'))
# Coverage measure
covp <- coverage(p$design)

# *****
# Radial scanning statistic: Detection of defects of Sobol designs
# *****

# requires randtoolbox package
library(randtoolbox)

# in 2D
rss <- rss2d(design=sobol(n=20, dim=2), lower=c(0,0), upper=c(1,1),
type="1", col="red")

# in 8D. All pairs of dimensions are tried to detect the worst defect
# (according to the specified goodness-of-fit statistic).
d <- 8
n <- 10*d
rss <- rss2d(design=sobol(n=n, dim=d), lower=rep(0,d), upper=rep(1,d),
type="1", col="red")

# avoid this defect with scrambling ?
# 1. Faure-Tezuka scrambling (type "?sobol" for more details and options)
rss <- rss2d(design=sobol(n=n, dim=d, scrambling=2), lower=rep(0,d),
upper=rep(1,d), type="1", col="red")
# 2. Owen scrambling
rss <- rss2d(design=sobol(n=n, dim=d, scrambling=1), lower=rep(0,d),
upper=rep(1,d), type="1", col="red")
```

 coverage

Coverage

Description

Compute the coverage measure

Usage

coverage(design)

Arguments

design a matrix (or a data.frame) representing the design of experiments representing the design of experiments in the unit cube $[0,1]^d$. If this last condition is not fulfilled, a transformation into $[0,1]^d$ is applied before the computation of the criteria.

Details

The coverage criterion is defined by

$$coverage = \frac{1}{\bar{\gamma}} \left[\frac{1}{n} \sum_{i=1}^n (\gamma_i - \bar{\gamma})^2 \right]^{1/2}$$

where γ_i is the minimal distance between the point x_i and the other points of the design and $\bar{\gamma}$ is the mean of the γ_i .

Note that for a regular mesh, cov=0. Then, a small value of cov means that the design is close to a regular grid.

Value

A real number equal to the value of the coverage criterion for the design.

Author(s)

J. Franco

References

Gunzburger M., Burkdart J. (2004) *Uniformity measures for point samples in hypercubes* <http://people.sc.fsu.edu/~burkardt/pdf/ptmeas.pdf>.

See Also

other distance criteria like [meshRatio](#) and [mindist](#).
discrepancy measures provided by [discrepancyCriteria](#).

Examples

```

dimension <- 2
n <- 40
X <- matrix(runif(n*dimension),n,dimension)
coverage(X)

```

discrepancyCriteria *Discrepancy measure*

Description

Compute discrepancy criteria.

Usage

```
discrepancyCriteria(design,type='all')
```

Arguments

design a matrix (or a data.frame) corresponding to the design of experiments. The discrepancy criteria are computed for a design in the unit cube $[0,1]^d$. If this condition is not satisfied the design is automatically rescaled.

type type of discrepancies (single value or vector) to be computed:

'all'	all type of discrepancies (default)
'L2'	star L2-discrepancy
'C2'	centered L2-discrepancy
'M2'	modified L2-discrepancy
'S2'	symmetric L2-discrepancy
'W2'	wrap-around L2-discrepancy

Details

The discrepancy measures how far a given distribution of points deviates from a perfectly uniform one. Different L2 discrepancies are available in DiceDesign. For example, if we denote by $Vol(J)$ the volume of a subset J of $[0; 1]^d$ and $A(X; J)$ the number of points of X falling in J , the L2-star discrepancy is:

$$D_{L2}(X) = \left[\int_{[0,1]^{2d}} \left(\frac{A(X, J_{a,b})}{n} - Vol(J_{a,b}) \right)^2 da db \right]^{1/2}$$

where $a = (a_1; \dots; a_d)'$, $b = (b_1; \dots; b_d)'$ and $J_{a,b} = [a_1; b_1] \times \dots \times [a_d; b_d]$. The other L2-discrepancies are defined according to the same principle with different form from the subset J . Among all the possibilities, discrepancyCriteria implements only the L2 discrepancies because it can be expressed analytically even for high dimension.

Centered L2-discrepancy is computed using the analytical expression done by Hickernell (1998). The user will refer to Fleming and Manteufel (2005) to have more details about the wrap around discrepancy.

Value

A real number equal to the centered L2-discrepancy of the design.

Author(s)

J. Franco & D. Dupuy

References

Fang K.T, Li R. and Sudjianto A. (2006) Design and Modeling for Computer Experiments, *Chapman & Hall*.

Franco J. (2008) Planification d'expériences numérique en phase exploratoire pour la simulation des phénomènes complexes, *PhD thesis, Ecole Nationale Supérieure des Mines de Saint Etienne*.

Hickernell F.J. (1998) A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, **67**, 299-322.

Fleming J.B. and Manteufel R.D. (2005) *Replicated Latin Hypercube Sampling*, 46th Structures, Structural Dynamics & Materials Conference, 16-21 April 2005, Austin (Texas) – AIAA 2005-1819.

See Also

distance criteria ([coverage](#), [meshRatio](#) and [mindist](#))

Examples

```
dimension <- 2
n <- 40
X <- matrix(runif(n*dimension),n,dimension)
discrepancyCriteria(X)
```

dmaxDesign

Maximum Entropy Designs

Description

Space-Filling Designs with n experiments based on covariance matrix in $[0,1]^d$.

Usage

```
dmaxDesign(n, dimension, range, niter_max=1000)
```

Arguments

n	number of experiments
dimension	number of variables
range	range of variogram
niter_max	number of iterations

Details

Maximum entropy design is a kind of optimal design based on Shannon's definition of entropy as the amount of information. Originally, maximum entropy sampling was proposed by Shewry and Wynn (1987). The goal of the design is to maximize the entropy defined as the determinant of the correlation matrix using a Fedorov-Mitchell exchange algorithm.

The spatial correlation matrix is defined by $C = (\rho_{ij})$:

$$\rho_{ij} = \begin{cases} 1 - \gamma(h_{ij}) & \text{if } h_{ij} \leq a, \\ 0 & \text{if } h_{ij} > a, \end{cases}$$

where h_{ij} is the distance between x_i and x_j , a denotes the range of the variogram and γ is a spherical variogram:

$$\gamma(h) = 1.5 \frac{h}{a} - 0.5 \left(\frac{h}{a} \right)^3 \quad \text{for } h \leq a$$

Value

A list with components:

n	the number of points
design	the design of experiments
dimension	the number of variables
range	the range of the variogram
niter_mx	the number of iterations
design_init	the initial distribution
det_init	the value of the determinant for the initial distribution
det_end	the value of the determinant at the end of the procedure

Author(s)

J. Franco

References

Currin C., Mitchell T., Morris M. and Ylvisaker D. (1991) *Bayesian Prediction of Deterministic Functions With Applications to the Design and Analysis of Computer Experiments*, American Statistical Association, **86**, 416, 953-963.

Shewry, M. C. and Wynn and H. P. (1987) *Maximum entropy sampling*, Journal of Applied Statistics 14, 165-170.

Examples

```
n <- 20
dimension <- 2
range <- 0.9
niter_max <- 200
out <- dmaxDesign(n,dimension,range,niter_max)
```

factDesign

Full Factorial Designs

Description

Create a factorial design with $n = \text{pow}(\text{levels}, \text{dimension})$ experiments in $[0,1]^d$.

Usage

```
factDesign(dimension, levels)
```

Arguments

dimension	an integer given the number of input variables
levels	an integer given the number of levels

Details

It is possible to take a different number of levels for any factor. In this case, the argument levels should be a vector.

Value

factDesign returns a list containing all the input arguments detailed before, plus the following components:

n	the number of experiments
design	the design of experiments

Author(s)

G. Pujol and J. Franco

Examples

```
## First example
g <- factDesign(2,7)
plot(g$design,xlim=c(0,1),ylim=c(0,1))
## Second example
g <- factDesign(2,c(2,7))
plot(g$design,xlim=c(0,1),ylim=c(0,1))
```

greenwood.table	<i>Upper percentage points for Greenwood statistic</i>
-----------------	--

Description

Upper percentage points for Greenwood statistic.

Usage

```
data(greenwood.table)
```

Format

A data frame of 23 observations on the following 5 variables:

n sample size

q10 quantile at significance level 0.1

q5 quantile at significance level 0.05

q2.5 quantile at significance level 0.025

q1 quantile at significance level 0.01

Source

D Agostino R.B., Stephens M.A. (1986), Goodness-of-fit techniques, CRC Press, New York, page 340, table 8.3.

meshRatio	<i>MeshRatio measure</i>
-----------	--------------------------

Description

The meshRatio criterion is the ratio between the maximum and the minimum distance between two points of the experimental design.

Usage

```
meshRatio(design)
```

Arguments

design a matrix (or a data.frame) representing the design of experiments in the unit cube $[0,1]^d$. If this last condition is not fulfilled, a transformation into $[0,1]^d$ is applied before the computation of the criteria.

Details

The meshRatio criterion is defined by

$$\text{meshRatio} = \frac{\max_{1 \leq i \leq n} \gamma_i}{\min_{1 \leq i \leq n} \gamma_i}$$

where γ_i denotes the minimal distance between the point x_i and the other points of the design.

Note that for a regular mesh, meshRatio=1.

Value

A real number equal to the value of the meshRatio criterion for the design.

Author(s)

J. Franco

References

Gunzburger M. and Burkardt J. (2004) *Uniformity measures for point samples in hypercubes* <http://people.sc.fsu.edu/~burkardt/pdf/ptmeas.pdf>.

See Also

other distance criteria like [meshRatio](#) and [mindist](#).

discrepancy measures provided by [discrepancyCriteria](#).

Examples

```
dimension <- 2
n <- 40
X <- matrix(runif(n*dimension),n,dimension)
meshRatio(X)
```

mindist

Mindist measure

Description

Compute the mindist criterion (also called maximin)

Usage

```
mindist(design)
```

Arguments

design a matrix (or a data.frame) representing the design of experiments in the unit cube $[0,1]^d$. If this last condition is not fulfilled, a transformation into $[0,1]^d$ is applied before the computation of the criteria.

Details

The mindist criterion is defined by

$$\text{mindist} = \min_{x_i \in X} (\gamma_i)$$

where γ_i is the minimal distance between the point x_i and the other points x_k of the design.

A higher value corresponds to a more regular scattering of design points.

Value

A real number equal to the value of the mindist criterion for the design.

Author(s)

J. Franco

References

Gunzburger M., Burkardt J. (2004) *Uniformity measures for point samples in hypercubes* <http://people.sc.fsu.edu/~burkardt/pdf/ptmeas.pdf>.

Jonshon M.E., Moore L.M. and Ylvisaker D. (1990) *Minmax and maximin distance designs*, J. of Statis. Planning and Inference, 26, 131-148.

Chen V.C.P., Tsui K.L., Barton R.R. and Allen J.K. (2003) *A review of design and modeling in computer experiments*, Handbook of Statistics, 22, 231-261.

See Also

other distance criteria like [meshRatio](#) and [mindist](#).

discrepancy measures provided by [discrepancyCriteria](#).

Examples

```
dimension <- 2
n <- 40
X <- matrix(runif(n*dimension),n,dimension)
mindist(X)
```

OA131

*A 3D orthogonal array of strength 2***Description**

A 3-dimensional linear orthogonal array (OA) of strength 2 with 49 points. The design points are equally spaced into 2 dimensional coordinate planes. However by construction, such OAs satisfy a linear relation, here: $x_1 + 3x_2 + x_3 = 0 \pmod{7}$. As a consequence, the design points are contained in parallel planes orthogonal to (1,3,1). Actually, they are also contained in parallel planes orthogonal to other directions, as (2,-1,2) or (3,2,3), since the congruence relation leads to $2x_1 - x_2 + 2x_3 = 0 \pmod{7}$ or $3x_1 + 2x_2 + 3x_3 = 0 \pmod{7}$. For instance, they are contained in 4 parallel planes orthogonal to (2,-1,2).

Usage

```
data(OA131)
```

Format

A data frame with 49 observations on the following 3 variables.

x1 first coordinate

x2 second coordinate

x3 third coordinate

Source

Roustant O., Franco J., Carraro L., Jourdan A. (2010), A radial scanning statistic for selecting space-filling designs in computer experiments, MODA-9 proceedings, www.emse.fr/~roustant

Examples

```
data(OA131)

# centering and reducing to [0,1]^3
OA <- (OA131 + 0.5)/7

pairs(OA, xlim=c(0,1), ylim=c(0,1))
## Not run: library(lattice)
cloud(x3~x1+x2, data=OA, xlim=c(0,1), ylim=c(0,1), zlim=c(0,1),
      screen = list(z = 50, x = -70, y = 0))
## End(Not run)
```

OA131_scrambled	<i>A scrambled 3D orthogonal array of strength 2</i>
-----------------	--

Description

This design is obtained by adding a uniform noise to each coordinate of the orthogonal array OA131.

Usage

```
data(OA131_scrambled)
```

Format

A data frame with 49 observations on the following 3 variables.

x1 first coordinate

x2 second coordinate

x3 third coordinate

Source

Roustant O., Franco J., Carraro L., Jourdan A. (2010), A radial scanning statistic for selecting space-filling designs in computer experiments, MODA-9 proceedings, www.emse.fr/~roustant

Examples

```
data(OA131)
data(OA131_scrambled)

pairs(OA131, xlim=c(0,1), ylim=c(0,1))
pairs(OA131_scrambled, xlim=c(0,1), ylim=c(0,1))
```

rss2d	<i>2D graphical tool for defect detection of Space-Filling Designs.</i>
-------	---

Description

For a 2-dimensional design, the 2D radial scanning statistic (RSS) scans angularly the domain. In each direction, it compares the distribution of projected points to their theoretical distribution under the assumption that all design points are drawn from uniform distribution. For a d-dimensional design, all pairs of dimensions are scanned. The RSS detects the defects of low discrepancy sequences or orthogonal arrays, and can be used for selecting space-filling designs.

Usage

```
rss2d(design, lower, upper, gof.test.type="greenwood",
      gof.test.stat=NULL, transform=NULL, n.angle=360, graphics=1,
      trace=TRUE, lines.lwd = 1, lines.lty = "dotted", ...)
```

Arguments

design	a matrix or data.frame containing the d-dimensional design of experiments. The row no. i contains the values of the d input variables corresponding to simulation no. i
lower	the domain lower boundaries.
upper	the domain upper boundaries.
gof.test.type	an optional character indicating the kind of statistical test to be used to test the goodness-of-fit of the design projections to their theoretical distribution. Several tests are available, see unif.test.statistic . Default is "greenwood".
gof.test.stat	an optional number equal to the goodness-of-fit statistic at level 5%. Default is the modified test statistic for fully specified distribution (see details below).
transform	an optional character indicating what type of transformation should be applied before testing uniformity. Only one choice available "spacings", that lead to over-detection. Default - and recommended - is NULL.
n.angle	an optional number indicating the number of angles used. Default is 360 corresponding to a 0.5-degree discretization step. Note that the RSS curve is continuous.
graphics	an optional integer indicating whether a graph should be produced. If negative, no graph is produced. If superior to 2, the RSS curve only is plotted in the worst 2D coordinate subspace (corr. to the worst value of statistic). If 1 (default), the design is also added, with its projections onto the worst (oblique) axis.
trace	an optional boolean. Turn it to FALSE if you want no verbosity.
lines.lwd	optional number specifying the width of the straight lines involved in the graphical outputs (axis and projections)
lines.lty	optional character string specifying the type of the straight lines involved in the graphical outputs (axis and projections)
...	optional graphical parameters of plot function, to draw the RSS curve.

Value

a list with components:

global.stat	a matrix containing the values of the global statistic (equal to the maximum of statistic values over the RSS curve) for all pairs of dimensions.
worst.case	the worst pair of dimensions, that is the one that gives the worst value of global.stat.
worst.dir	the worst direction, that is the one that gives the worst value of the global statistic in the coordinate plane defined by worst.case.
stat	a vector of length n.angle containing the statistic values for each angle, in the coordinate plane defined by worst.case.

angle	a vector of length n.angle containing the corresponding angles used.
curve	a (2*n.angle)x2 matrix containing the discretized RSS curve.
gof.test.stat	the threshold at significance level 0.05 for the specified goodness-of-fit statistic. It is equal to the radius of the circle superimposed on the RSS figure.

Author(s)

O. Roustant

References

Roustant O., Franco J., Carraro L., Jourdan A. (2010), A radial scanning statistic for selecting space-filling designs in computer experiments, MODA-9 proceedings, www.emse.fr/~roustant

D Agostino R.B., Stephens M.A. (1986), Goodness-of-fit techniques, CRC Press, New York.

See Also

[unif.test.statistic](#), [unif.test.quantile](#), [rss3d](#)

Examples

```
# Detection of defects of Sobol designs

# requires randtoolbox package
library(randtoolbox)

# in 2D
rss <- rss2d(design=sobol(n=20, dim=2), lower=c(0,0), upper=c(1,1), type="l",
  col="red")

# in 8D.
# All pairs of dimensions are tried to detect the worst defect
# (according to the specified goodness-of-fit statistic).
d <- 8
n <- 10*d
rss <- rss2d(design=sobol(n=n, dim=d), lower=rep(0,d), upper=rep(1,d), type="l",
  col="red")

# avoid this defect with scrambling ?
# 1. Faure-Tezuka scrambling (type "?sobol" for more details and options)
rss <- rss2d(design=sobol(n=n, dim=d, scrambling=2), lower=rep(0,d),
  upper=rep(1,d), type="l", col="red")
# 2. Owen scrambling
rss <- rss2d(design=sobol(n=n, dim=d, scrambling=1), lower=rep(0,d),
  upper=rep(1,d), type="l", col="red")
```

 rss3d

3D graphical tool for defect detection of Space-Filling Designs.

Description

For a 3-dimensional design, the 3D radial scanning statistic (RSS) scans angularly the domain. In each direction, it compares the distribution of projected points to their theoretical distribution under the assumption that all design points are drawn from uniform distribution. For a d-dimensional design, all triplets of dimensions are scanned. The RSS detects the defects of low discrepancy sequences or orthogonal arrays, and can be used for selecting space-filling designs.

Usage

```
rss3d(design, lower, upper, gof.test.type = "greenwood",
      gof.test.stat = NULL, transform = NULL, n.angle = 60,
      graphics = 1, trace = TRUE)
```

Arguments

design	a matrix or data.frame containing the d-dimensional design of experiments. The row no. i contains the values of the d input variables corresponding to simulation no. i
lower	the domain lower boundaries.
upper	the domain upper boundaries.
gof.test.type	an optional character indicating the kind of statistical test to be used to test the goodness-of-fit of the design projections to their theoretical distribution. Several tests are available, see unif.test.statistic . Default is "greenwood".
gof.test.stat	an optional number equal to the goodness-of-fit statistic at level 5%. Default is the modified test statistic for fully specified distribution (see details below).
transform	an optional character indicating what type of transformation should be applied before testing uniformity. Only one choice available "spacings", that lead to over-detection. Default - and recommended - is NULL.
n.angle	an optional number indicating the number of angles used. Default is 60 corresponding to a 3-degree discretization step. Note that the RSS surface is continuous.
graphics	an optional integer indicating whether a graph should be produced. If negative, no graph is produced. Otherwise (default), the design is plotted in the worst 3D coordinate subspace (corr. to the worst value of statistic), with its projections onto the worst (oblique) axis.
trace	an optional boolean. Turn it to FALSE if you want no verbosity.

Details

The RSS surface is continuous. However for computational purposes, a discretization is used. The default discretization step is tunable with n.angle.

Value

a list with components:

<code>global.stat</code>	an array containing the values of the global statistic (equal to the maximum of statistic values over the RSS surface) for all triplets of dimensions.
<code>print.out</code>	the same as <code>global.stat</code> , but with a user-friendly printing.
<code>worst.case</code>	the worst triplet of dimensions, that is the one that gives the worst value of <code>global.stat</code> .
<code>worst.dir</code>	the worst direction, that is the one that gives the worst value of the statistic in the coordinate 3D subspace defined by <code>worst.case</code> .
<code>stat</code>	a matrix of size <code>n.angle*n.angle</code> containing the statistic values for each angles (spherical coordinates).
<code>angle</code>	a matrix of size <code>n.angle*n.angle</code> containing the corresponding angles used (spherical coordinates).
<code>gof.test.stat</code>	the threshold at significance level 0.05 for the specified goodness-of-fit statistic.

Author(s)

O. Roustant

References

- Roustant O., Franco J., Carraro L., Jourdan A. (2010), A radial scanning statistic for selecting space-filling designs in computer experiments, MODA-9 proceedings, www.emse.fr/~roustant
- D Agostino R.B., Stephens M.A. (1986), Goodness-of-fit techniques, CRC Press, New York.

See Also

[unif.test.statistic](#), [unif.test.quantile](#), [rss2d](#)

Examples

```
# An orthogonal array in 3D
data(OA131)

# centering the design points of this 7-levels design
OA <- (OA131 + 0.5)/7

# 2D projections onto coordinate axis
pairs(OA, xlim=c(0,1), ylim=c(0,1))
# Now let us look at the 3D properties with the 3D RSS (requires the rgl package)
rss <- rss3d(OA, lower=c(0,0,0), upper=c(1,1,1))
# The worst direction detected is nearly proportional to (2,-1,2)
# (type "?OA131" for explanations about this linear orthogonal array)
print(rss$worst.dir)

# Now, scramble this design
# X <- (OA131 + matrix(runif(49*3, 49, 3)))/7
```

```

# or load the design obtained this way
data(OA131_scrambled)
OA2 <- OA131_scrambled

# no feature is detected by the 2D RSS:
rss <- rss2d(OA2, lower=c(0,0,0), upper=c(1,1,1))
# 4 clusters are detected by the 3D RSS:
rss <- rss3d(OA2, lower=c(0,0,0), upper=c(1,1,1))

# Defect detection of 8D Sobol sequences
# All triplets of dimensions are tried to detect the worst defect
# (according to the specified goodness-of-fit statistic).
# requires randtoolbox library to generate the Sobol sequence
## Not run:
library(randtoolbox)
d <- 8
n <- 10*d
rss <- rss3d(design=sobol(n=n, dim=d), lower=rep(0,d), upper=rep(1,d))
## End(Not run)

```

runif.faure

Low discrepancy sequence : Faure

Description

Generate a Faure sequence with n experiments in $[0,1]^d$.

Usage

```
runif.faure(n, dimension)
```

Arguments

n	the number of experiments
dimension	the number of variables (<100)

Details

A quasirandom or low discrepancy sequence, such as the Faure, Halton, Hammersley, Niederreiter or Sobol sequences, is "less random" than a pseudorandom number sequence, but more useful for such tasks as approximation of integrals in higher dimensions, and in global optimization. This is because low discrepancy sequences tend to sample space "more uniformly" than random numbers.

see **randtoolbox** or **fOptions** packages for other low discrepancy sequences.

Value

runif.halton returns a list containing all the input arguments detailed before, plus the following component:

design the design of experiments

Author(s)

J. Franco

References

Faure H. (1982) *Discrèpance de suites associées à un système de numération (en dimension s)*, Acta Arith. 41, 337-351

Examples

```
f <- runif.faure(20,2)
plot(f$design,xlim=c(0,1),ylim=c(0,1))
```

straussDesign *Designs based on Strauss process*

Description

Space-Filling Designs based on Strauss process

Usage

```
straussDesign(n,dimension,RND,alpha=0.5,repulsion=0.001,NMC=1000,
constraints1D=0,repulsion1D=0.0001,init=NULL)
```

Arguments

n	the number of experiments
dimension	the number of input variables
RND	a real number which represents the radius of interaction
alpha	the potential power (default, fixed at 0.5)
repulsion	the repulsion parameter in the unit cube (gamma)
NMC	the number of McMC iterations (this number must be large to converge)
constraints1D	1 to impose 1D projection constraints, 0 otherwise
repulsion1D	the repulsion parameter in 1D
init	initial design (default equals NULL and the initialization corresponds to an uniform distribution)

Details

Strauss designs are Space-Filling designs initially defined from Strauss process:

$$\pi(X) = k\gamma^{s(X)}$$

where $s(X)$ is the number of pairs of points (x^i, x^j) of the design $X = (x^1, \dots, x^n)$ that are separated by a distance no greater than the radius of interaction RND, k is the normalizing constant and γ is the repulsion parameter. This distribution corresponds to the particular case $\alpha=0$.

For the general case, a stochastic simulation is used to construct a Markov chain which converges to a spatial density of points $\pi(X)$ described by the Strauss-Gibbs potential. In practice, the Metropolis-Hastings algorithm is implemented to simulate a distribution of points which converges to the stationary law:

$$\pi(X) \propto \exp(-U(X))$$

with a potential U defined by:

$$U(X) = \beta \sum_{1 \leq i < j \leq n} \varphi(\|x^i - x^j\|)$$

where $\beta = -\ln \gamma$, $\varphi(h) = \left(1 - \frac{h}{\text{RND}}\right)^\alpha$ if $h \leq \text{RND}$ and 0 otherwise.

The input parameters of `straussDesign` function can be interpreted as follows:

- RND is used to compute the number of pairs of points of the design separated by a distance no more than RND. A point is said "in interaction" with another if the spheres of radius RND/2 centered on these points intersect.
- α is the potential power α . The case $\alpha=0$ corresponds to Strauss process (0-1 potential).
- `repulsion` is equal to the γ parameter of the Strauss process. Note that γ belongs to $]0,1[$.
- `constraints1D` allows to specify some constraints into the margin. If `constraints1D==1`, two repulsion parameters are needed: one for the all space (`repulsion`) and the other for the 1D projection (`repulsion1D`). Default values are `repulsion=0.001` and `repulsion1D=0.001`. Note that the value of the radius of interaction in the one-dimensional axis is not an input parameter and is automatically fixed at $0.75/n$.

Value

A list containing:

<code>n</code>	the number of experiments
<code>dimension</code>	the number d of variables
<code>init</code>	the initial distribution of n points $[0, 1]^d$
<code>radius</code>	the radius of interaction
<code>alpha</code>	the potential power α
<code>repulsion</code>	the repulsion parameter γ
<code>NMC</code>	the number of iterations MCMC
<code>constraints1D</code>	an integer indicating if constraints on the factorial axis are imposed. If its value is different from zero, a component <code>repulsion1D</code> containing the value of the repulsion parameter γ in dimension 1 is added at the list.
<code>design</code>	the design of experiments in $[0,1]^d$

Author(s)

J. Franco

References

J. Franco, X. Bay, B. Corre and D. Dupuy (2008) *Planification d'expériences numériques à partir du processus ponctuel de Strauss* <http://hal.archives-ouvertes.fr/hal-00260701/fr/> (March 06, 2008).

Examples

```
# Strauss-Gibbs designs in dimension 2 (n=20 points)
S1 <- straussDesign(n=20,dimension=2,RND=0.2)
plot(S1$design,xlim=c(0,1),ylim=c(0,1))
theta <- seq(0,2*pi,by =2*pi/(100 - 1))
for(i in 1:S1$n){
  lines(S1$design[i,1]+S1$radius/2*cos(theta),
        S1$design[i,2]+S1$radius/2*sin(theta),col='red')
}
# 2D-Strauss design
S2 <- straussDesign(n=20,dimension=2,RND=0.2,NMC=200,
constraints1D=0,alpha=0,repulsion=0.01)

plot(S2$design,xlim=c(0,1),ylim=c(0,1))

# 2D-Strauss designs with constraints on the axis
S3 <- straussDesign(n=20,dimension=2,RND=0.18,NMC=200,
constraints1D=1,alpha=0.5,repulsion=0.1,repulsion1D=0.01)

plot(S3$design,xlim=c(0,1),ylim=c(0,1))
rug(S3$design[,1],side=1)
rug(S3$design[,2],side=2)
```

unif.test.quantile *Quantile of some uniformity tests*

Description

Computes the quantile of a uniformity test at a given significance level (see available tests and levels below).

Usage

```
unif.test.quantile(type, n, alpha)
```

Arguments

type	a character indicating which test is used. The choices are the following: "greenwood", "qm" (for Quesenberry-Miller), "ks" (Kolmogorov-Smirnov), "cvm" (Cramer-Von Mises) and "V" (D+ + D- from Kolmogorov-Smirnov).
n	an integer equal to the sample size.
alpha	a real number equal to significance level. At present stage, only four values are available: 0.1, 0.05, 0.025 and 0.01.

Details

Modified statistics are used. For $\alpha = 0.05$, the quantile is (see D Agostino and Stephens, 1986, section 4.4.): $1.358/(\sqrt{n}) + 0.12 + 0.11/\sqrt{n}$ for Kolmogorov-Smirnov and $0.461/(1+1/n) + 0.4/n - 0.6/n^2$ for Cramer-von Mises. When the design size is < 20 , the corrected value seems to be a good approximation, but the non asymptotical value should be preferred.

Value

A real number equal to the quantile of the specified test at significance level α for n observations.

Author(s)

O. Roustant

References

D Agostino R.B., Stephens M.A. (1986), Goodness-of-fit techniques, CRC Press, New York.

See Also

[unif.test.statistic](#), [rss2d](#), [rss3d](#)

`unif.test.statistic` *Statistic of some uniformity tests*

Description

Computes the statistic of a uniformity test (see available tests below).

Usage

```
unif.test.statistic(x, type, transform=NULL)
```

Arguments

x	a vector containing the sample values.
type	a character indicating which test is used. The choices are the following: "greenwood", "qm" (for Quesenberry-Miller), "ks" (Kolmogorov-Smirnov), "cvm" (Cramer-Von Mises) and "V" (D+ + D- from Kolmogorov-Smirnov).
transform	an optional character indicating what type of transformation should be applied before testing uniformity. Default is NULL.

Value

A real number equal to the statistic of the specified test.

Author(s)

O. Roustant

References

D Agostino R.B., Stephens M.A. (1986), Goodness-of-fit techniques, CRC Press, New York.

See Also

[unif.test.quantile](#), [rss2d](#)

Index

*Topic **datasets**

greenwood.table, 9
OA131, 12
OA131_scrambled, 13

*Topic **design**

coverage, 4
discrepancyCriteria, 5
dmaxDesign, 6
factDesign, 8
meshRatio, 9
mindist, 10
rss2d, 13
rss3d, 16
runif.faure, 18
straussDesign, 19
unif.test.quantile, 21
unif.test.statistic, 22

*Topic **package**

DiceDesign-package, 2

coverage, 4, 6

DiceDesign (DiceDesign-package), 2

DiceDesign-package, 2

discrepancyCriteria, 4, 5, 10, 11

dmaxDesign, 6

factDesign, 8

greenwood.table, 9

meshRatio, 4, 6, 9, 10, 11

mindist, 4, 6, 10, 10, 11

OA131, 12

OA131_scrambled, 13

rss2d, 13, 17, 22, 23

rss3d, 15, 16, 22

runif.faure, 18

straussDesign, 19

unif.test.quantile, 15, 17, 21, 23

unif.test.statistic, 14–17, 22, 22