

# Package ‘ECmexico2012’

May 24, 2012

**Type** Package

**Title** Ecole-Chercheur Mexico 2012

**Version** 1.0-0

**Date** 2012-05-24

**Author** EC-Mexico-2012

**Maintainer** Herve Monod <herve.monod@jouy.inra.fr>

**Depends** R (>= 1.8.0), sensitivity, akima, car, evd, lhs, mgcv, rgl, triangle

**Suggests** orthopolynom

**Description** ECmexico2012

**License** GPL

**LazyLoad** yes

**Collate** 'ECmexico2012-package.R' 'TPLibModels.R' 'TPLibUtile.R' 'TPLibData.R'

**Encoding** latin1

## R topics documented:

ECmexico2012-package	2
convertfrom.basep	3
convertinto.basep	3
convertU2N	4
copyright	4
crossing	5
fungus.factors	5
fungus.model	6
fungus.simule	6
inverses.basep	7
ishigami.factors	8
ishigami.model	8
ishigami.simule	9
lhs.plan	9
lhs2intervalle	10

loiGeneriqueTronquee . . . . .	11
loiGumbelTronquee . . . . .	12
loiLogNormaleTronquee . . . . .	12
loiNormaleTronquee . . . . .	13
morris.mexico . . . . .	14
perspPlus . . . . .	14
planor.kernelcheck.basep . . . . .	15
regular.fraction . . . . .	16
representative.basep . . . . .	17
samplingLHS . . . . .	17
samplingOptLHS . . . . .	19
samplingSimple . . . . .	20
TP1.ICmorris . . . . .	21
TP1corr . . . . .	21
TP1histo . . . . .	22
TP1indices.aov . . . . .	22
TP1pavage . . . . .	23
TP1tirage . . . . .	23
weed.decision . . . . .	24
weed.factors . . . . .	24
weed.fun . . . . .	24
weed.model . . . . .	25
weed.simule . . . . .	26
wwdm.climates . . . . .	27
wwdm.factors . . . . .	27
wwdm.model . . . . .	27
wwdm.simule . . . . .	28

## Index 30

---

ECmexico2012-package

*Librairie R de l'ECOLE-CHERCHEURS MEXICO, ECULLY, 4-8 juin 2012*

---

## Description

Librairie R de l'ECOLE-CHERCHEURS MEXICO, ECULLY, 4-8 juin 2012

## Details

Package: ECmexico2012  
 Type: Package  
 Version: 0.0-1  
 Date: 2012-05-24  
 Depends: R (>= 1.8.0), sensitivity, akima, car, evd, lhs, mgcv, rgl, triangle  
 License: GPL  
 LazyLoad: yes  
 Collate: 'ECmexico2012.R' 'TPLibModels.R' 'TPLibUtile.R' 'TPLibData.R'  
 Encoding: latin1

**Author(s)**

ec-mexico, maintenir: Hervé Monod

---

convertfrom.basep    *Utilitaire plans fractionnaires: conversion base p vers base 10...*

---

**Description**

Utilitaire plans fractionnaires: conversion base p vers base 10

**Usage**

```
convertfrom.basep(x, p)
```

**Arguments**

x	matrice dont les lignes forment des nombres en base p
p	un nombre entier premier

**Value**

vecteur des valeurs en base 10

**Note**

les coefficients sont supposés ordonnés par puissances croissantes de p

**Examples**

```
conv.into2 <- convertinto.basep( x=c(0:16), p=2 )  
convertfrom.basep(x=conv.into2, p=2)
```

---

convertinto.basep    *Utilitaire plans fractionnaires: conversion base 10 vers base p...*

---

**Description**

Utilitaire plans fractionnaires: conversion base 10 vers base p

**Usage**

```
convertinto.basep(x, p)
```

**Arguments**

x	vecteur des valeurs en base 10
p	un nombre entier premier

**Value**

matrice dont les lignes forment des nombres en base p

**Note**

les coefficients sont ordonnés par puissances croissantes de p

**Examples**

```
convertinto.basep( x=c(0:16), p=2 )
convertinto.basep( x=c(0:16), p=3 )
```

---

convertU2N	<i>Transformation d'un échantillon d'une loi uniforme vers une loi normale...</i>
------------	---

---

**Description**

Transformation d'un échantillon d'une loi uniforme vers une loi normale de même moyenne et d'écart.type assurant que la loi uniforme couvre une probabilité égale à l'argument couverture de la loi normale

**Usage**

```
convertU2N(x, param, couverture=0.95)
```

**Arguments**

x	échantillon d'une loi considérée uniforme
param	ligne d'un data.frame de type xxx.factors
couverture	une probabilité entre 0 et 1

**Examples**

```
convertU2N(seq(8,10,length=11), fungus.factors["Topt",])
```

---

copyright	<i>Copyright(c) INRA 2012</i>
-----------	-------------------------------

---

**Description**

Copyright(c) INRA 2012

---

crossing	<i>Génération d'un plan factoriel complet à partir des nombres de modalités de...</i>
----------	---

---

**Description**

Génération d'un plan factoriel complet à partir des nombres de modalités de  $s$  facteurs

**Usage**

```
crossing(n, start=1)
```

**Arguments**

n	vecteur des nombres de modalités
start	valeur du 1er chiffre utilisé pour les modalités des facteurs

**Value**

matrice à 1 ligne par combinaison et 1 colonne par facteur

**Examples**

```
crossing( c(2,3,4), start=0 )
```

---

fungus.factors	<i>Facteurs d'entrée de l'AS du modèle "fungus"</i>
----------------	---

---

**Description**

Facteurs d'entrée du modèle "fungus" (croissance champignon selon l'humidité)

**Value**

data.frame à 5 lignes (facteurs) et 4 colonnes (spécifs)

**Note**

Les valeurs dans fungus.factors correspondent à *Alternaria Brassicae*, ravageur du colza (Magarey et al, 2005)

---

fungus.model

Modèle "fungus" (croissance champignon selon l'humidité)...

---

### Description

Modèle "fungus" (croissance champignon selon l'humidité)

### Usage

```
fungus.model (param=fungus.factors$nominal, temperature=10)
```

### Arguments

param            vecteur de longueur 5 comprenant un jeu de valeurs de Tmin, Topt, Tmax, Wmin, Wmax

temperature    scalaire ou vecteur de temperatures

### Value

scalaire ou vecteur de longueur egale au nombre de temperatures

### Note

Des valeurs min, max et nominales des paramètres sont donnees dans fungus.factors. Elles correspondent à *Alternaria Brassicae*, ravageur du colza (Magarey et al, 2005).

### References

Magarey RD, Sutton TB, Thayer CL (2005). A simple generic infection model for foliar fungal plant pathogens. *Phytopathology* 95, 92-100.

### Examples

```
fungus.model( fungus.factors$nominal, temperature=c(10,15,18,21,30) )
```

---

fungus.simule

Simulation du modèle "fungus"...

---

### Description

Simulation du modèle "fungus"

### Usage

```
fungus.simule(X, temperature=10, tout=FALSE)
```

**Arguments**

<code>x</code>	vecteur de longueur 5 ou matrice N x 5 comprenant un ou plusieurs jeux de valeurs de Tmin, Topt, Tmax, Wmin, Wmax
<code>temperature</code>	scalaire ou vecteur de temperatures
<code>tout</code>	TRUE si l'on veut les entrées ET les sorties dans le tableau de sortie

**Value**

Data.frame a N lignes et p colonnes, ou p est la longueur de 'temperature'

**Note**

Des valeurs min, max et nominales des paramètres sont donnees dans fungus.factors. Elles correspondent à *Alternaria Brassicae*, ravageur du colza (Magarey et al, 2005).

**References**

Magarey RD, Sutton TB, Thayer CL (2005). A simple generic infection model for foliar fungal plant pathogens. *Phytopathology* 95, 92-100.

**Examples**

```
scenarios <- rbind(fungus.factors$binf, fungus.factors$nominal, fungus.factors$bsup)
fungus.simule( scenarios, temperature=c(10,15,18,21,30) )
```

---

<code>inverses.basep</code>	<i>Calcul basique des inverses modulo p...</i>
-----------------------------	--

---

**Description**

Calcul basique des inverses modulo  $p$

**Usage**

```
inverses.basep(p)
```

**Arguments**

<code>p</code>	un nombre entier premier
----------------	--------------------------

**Value**

vecteur des inverses

**Examples**

```
inverses.basep(5)
(inverses.basep(17) * (1:16)) %%17
```

---

<code>ishigami.factors</code>	<i>Facteurs d'entrée du modèle Ishigami</i>
-------------------------------	---

---

### Description

Facteurs d'entrée du modèle Ishigami, décrit dans *Saltelli et al., 2000*

### Value

`data.frame` à 3 lignes (facteurs) et 4 colonnes (spécifs)

---

<code>ishigami.model</code>	<i>Modèle d'Ishigami, décrit dans Saltelli et al., 2000</i>
-----------------------------	---

---

### Description

Modèle d'Ishigami

### Usage

```
ishigami.model(param=ishigami.factors$nominal)
```

### Arguments

<code>param</code>	vecteur de longueur 3 ou matrice N x 3 des paramètres chacun des paramètres doit varier entre -pi et +pi
--------------------	--

### Value

scalaire, ou vecteur de longueur N

### Note

Appelle la fonction `ishigami.fun` de la librairie `sensitivity`

### Examples

```
ishigami.model( c(-1,0,-1) )
ishigami.model( rbind( c(1,1,1), c(-1,0,-1) ) )
```



---

ishigami.simule	<i>Simulation du modèle d'Ishigami, décrit dans Saltelli et al., 2000</i>
-----------------	---

---

**Description**

Simulation du modèle d'Ishigami

**Usage**

```
ishigami.simule(X, tout=FALSE)
```

**Arguments**

X	matrice ou dataframe N x 3 des valeurs d'entrée, comprises entre -pi et +pi
tout	TRUE si l'on veut les entrées ET les sorties dans le tableau de sortie

**Value**

matrice ou dataframe si 'tout==TRUE', un vecteur sinon

**Note**

Appelle la fonction ishigami.fun de la librairie sensitivity

**Examples**

```
ishigami.simule( c(-1,0,-1) )
ishigami.simule( rbind( c(1,1,1),c(-1,0,-1) ) )
```

---

lhs.plan	<i>Tire selon le plan hyper-cube latin un échantillon de valeurs de paramètres...</i>
----------	---

---

**Description**

Tire selon le plan hyper-cube latin un échantillon de valeurs de paramètres

**Usage**

```
lhs.plan(taille, plage, repet, tout=FALSE)
```

**Arguments**

taille	de l'échantillon
plage	objet de type .factors
repet	NULL ou une ligne d'un objet de type .factors
tout	TRUE si l'on veut conserver l'échantillon de base dans $[0, 1]^p$

**Value**

la matrice  $N \times p$  de l'échantillon si `tout=FALSE` ou une liste à 2 composantes, `plan` et `lhs.tirage`, si `tout=TRUE`

**Note**

L'objet `.factors` donné dans l'argument `plage` sert à spécifier le nom des facteurs et les bornes de leurs intervalles d'incertitude. L'argument `repet` permet d'inclure un facteur supplémentaire qualitatif, dont les modalités sont obtenues par des tirages aléatoires indépendants avec remise.

**Examples**

```
lhs.plan( taille = 10, plage = fungus.factors, repet = NULL, tout = FALSE)
lhs.plan( taille = 10, plage = wwdm.factors[1:7,],
          repet = wwdm.factors[8,], tout = FALSE)
```

---

lhs2intervalle	<i>Projection de valeurs tirées entre 0 et 1 sur une autre plage de variation...</i>
----------------	--

---

**Description**

Projection de valeurs tirées entre 0 et 1 sur une autre plage de variation

**Usage**

```
lhs2intervalle(matrice, minAmax)
```

**Arguments**

<code>matrice</code>	matrice ou <code>data.frame</code> 0-1 à $N$ lignes et $p$ colonnes
<code>minAmax</code>	vecteur des bornes de l'intervalle cible

**Value**

`matrice`

**Note**

Utilisé dans `lhs.plan`.

**Examples**

```
## Not run: TODO
```

---

`loiGeneriqueTronquee`*Fonctions génériques pour loi tronquée*

---

## Description

Fonctions génériques pour loi tronquée

## Usage

```
d.trunc.distr(x, distr, trunc.int, ...)
p.trunc.distr(q, distr, trunc.int, ...)
q.trunc.distr(p, distr, trunc.int, ...)
r.trunc.distr(n, distr, trunc.int, ...)
```

## Arguments

<code>x</code>	vecteur de quantiles
<code>q</code>	vecteur de quantiles
<code>p</code>	vecteur de probabilités
<code>n</code>	taille de l'échantillon aléatoire à générer
<code>distr</code>	intitulé de la loi à tronquer
<code>trunc.int</code>	bornes de la troncature
<code>...</code>	paramètres de la loi à tronquer

## Value

densités, probabilités, quantiles, échantillon aléatoire

## Examples

```
d.trunc.distr(x=c(-2.5,-1.96,-1,0,1,1.96,2.5), distr = 'norm',
              trunc.int = c(-2, 2), mean = 0, sd = 1)
d.trunc.distr(x=c(-2.5,-1.96,-1,0,1,1.96,2.5),
              distr = c('dnorm', 'pnorm', 'qnorm', 'rnorm'),
              trunc.int = c(-2, 2), mean = 0, sd = 1)
```

---

loiGumbelTronquee    *The Truncated Gumbel distribution*

---

### Description

Fonctions associées à la loi de Gumbel tronquée

### Usage

```
dtgumbel(x, loc = 0, scale = 1, min = -1e6, max = 1e6)
```

```
ptgumbel(q, loc = 0, scale = 1, min = -1e6, max = 1e6)
```

```
qtgumbel(p, loc = 0, scale = 1, min = -1e6, max = 1e6)
```

```
rtgumbel(n, loc = 0, scale = 1, min = -1e6, max = 1e6)
```

### Arguments

x	vecteur de quantiles
q	vecteur de quantiles
p	vecteur de probabilités
n	taille de l'échantillon aléatoire à générer
loc	paramètre de position de la loi à tronquer
scale	paramètre d'échelle de la loi à tronquer
min	borne inférieure de la troncature
max	borne supérieure de la troncature

### Value

densités, probabilités, quantiles, échantillon aléatoire

---

loiLogNormaleTronquee  
*The Truncated LogNormal distribution*

---

### Description

Fonctions associées à la loi LogNormale tronquée

### Usage

```
dtlnorm(x, meanlog = 0, sdlog = 1, min = -1e6, max = 1e6)
```

```
ptlnorm(q, meanlog = 0, sdlog = 1, min = -1e6, max = 1e6)
```

```
qtlnorm(p, meanlog = 0, sdlog = 1, min = -1e6, max = 1e6)
```

```
rtlnorm(n, meanlog = 0, sdlog = 1, min = -1e6, max = 1e6)
```

**Arguments**

x	vecteur de quantiles
q	vecteur de quantiles
p	vecteur de probabilités
n	taille de l'échantillon aléatoire à générer
meanlog	moyenne de la loi à tronquer
sdlog	écart-type de la loi à tronquer
min	borne inférieure de la troncature
max	borne supérieure de la troncature

**Value**

densités, probabilités, quantiles, échantillon aléatoire

---

loiNormaleTronquee *The Truncated Normal distribution*

---

**Description**

Fonctions associées à la loi Normale tronquée

**Usage**

```
dtnorm(x, mean = 0, sd = 1, min = -1e6, max = 1e6)
ptnorm(q, mean = 0, sd = 1, min = -1e6, max = 1e6)
qtnorm(p, mean = 0, sd = 1, min = -1e6, max = 1e6)
rtnorm(n, mean = 0, sd = 1, min = -1e6, max = 1e6)
```

**Arguments**

x	vecteur de quantiles
q	vecteur de quantiles
p	vecteur de probabilités
n	taille de l'échantillon aléatoire à générer
mean	moyenne de la loi à tronquer
sd	écart-type de la loi à tronquer
min	borne inférieure de la troncature
max	borne supérieure de la troncature

**Value**

densités, probabilités, quantiles, échantillon aléatoire

---

`morris.mexico`

*Méthode de Morris á la sauce mexicaine*

---

### Description

Adaptation de la méthode de Morris pour l'EC Mexico

### Usage

```
morris.mexico(model, factors, r, design, binf=0, bsup=1, scale=TRUE, ...)
```

### Arguments

<code>model</code>	voir la méthode morris de la librairie sensitivity
<code>factors</code>	voir la méthode morris de la librairie sensitivity
<code>r</code>	voir la méthode morris de la librairie sensitivity
<code>design</code>	voir la méthode morris de la librairie sensitivity
<code>binf</code>	voir la méthode morris de la librairie sensitivity
<code>bsup</code>	voir la méthode morris de la librairie sensitivity
<code>scale</code>	voir la méthode morris de la librairie sensitivity
<code>...</code>	voir la méthode morris de la librairie sensitivity

### Value

voir la méthode morris de la librairie sensitivity

### Note

Corrige un problème détecté dans la version 1.0 de sensitivity en normalisant les facteurs d'entrée avant les calculs principaux

---

`perspPlus`

*Interface conviviale pour des graphiques pour 3 variables...*

---

### Description

Interface conviviale pour des graphiques pour 3 variables

### Usage

```
perspPlus(x, y, z, pcol=c("blue", "green"), pphi=30, ptheta=-30,
  nomx=deparse(substitute(x)), nomy=deparse(substitute(y)),
  nomz=deparse(substitute(z)), type=1)
```

**Arguments**

x	voir la doc de persp
y	voir la doc de persp
z	voir la doc de persp
pcol	code pour le dégradé de couleurs
pphi	angle de vue (colatitude, argument phi de persp)
ptheta	angle de vue (direction azimutale, argument theta de persp)
nomx	chaîne de caractère
nomy	chaîne de caractère
nomz	chaîne de caractère
type	un chiffre. 1: perspective; 2: image; 3: contour; 4: perspective 3D

**Value**

invisible()

**Examples**

```
#perspPlus(x=Tmax,y=Tmin,z = Y10, pcol=c("blue", "green"), pphi=30, ptheta=-30,
#      nomx=deparse(substitute(x)), nomy=deparse(substitute(y)),
#      nomz=deparse(substitute(z)), type=1)
```

---

planor.kernelcheck.basep

*Vérification des confusions d'effets en cours de construction...*


---

**Description**

Vérification des confusions d'effets en cours de construction d'un plan factoriel fractionnaire

**Usage**

```
planor.kernelcheck.basep(PhiStar, admissible, IneligibleSet, p)
```

**Arguments**

PhiStar	matrice clé en cours
admissible	matrice codant les caractères a priori admissibles
IneligibleSet	ensemble des caractères non éligibles
p	un nombre entier premier

**Value**

vecteur logique

---

regular.fraction	<i>Construction de plans factoriels fractionnaires symétriques entre les facteurs.</i>
------------------	--

---

## Description

Construction de plans factoriels fractionnaires symétriques entre les facteurs. Cette fonction permet de générer une fraction de résolution donnée pour  $s$  facteurs à  $p$  modalités en  $p^r$  unités

## Usage

```
regular.fraction(s, p, r, resolution)
```

## Arguments

s	le nombre de facteurs
p	un nombre entier premier égal au nombre de modalités par facteur
r	un nombre entier définissant la taille du plan, égale à $p^r$
resolution	la résolution de la fraction

## Value

liste à deux composantes, `plan` (le plan en base  $p$ ) et `matrice.cle` (la matrice clé contenant les relations de définition), ou NULL si aucune solution n'été trouvée. Le plan est sous la forme d'une matrice composée d'entiers modulo  $p$

## Note

This is a simplified version of a more general library in preparation. In this version, all factors must have the same prime number of levels and only fractions with a given resolution can be constructed. The first  $q$  factors are used as basic factors. The first solution is kept although it may not be the most interesting one (no control of aberration). This function is programmed entirely in R and so it is not efficient with respect to computer time. There is no explicit check on the arguments and so it is up to the user to restrict  $p$  to a prime number such as 2, 3, 5 or 7.

## Examples

```
regular.fraction(s=8, p=2, r=4, resolution=4)
regular.fraction(s=9, p=2, r=4, resolution=4)
```



---

representative.basep

*Fonction générant l'ensemble minimal de représentants des colonnes...*


---

### Description

Fonction générant l'ensemble minimal de représentants des colonnes d'une matrice, en base  $p$

### Usage

```
representative.basep(mat, p)
```

### Arguments

mat	une matrice d'entiers modulo $p$
p	un nombre entier premier

### Value

une matrice d'entiers modulo  $p$

---

samplingLHS

*Tirage aléatoire LHS de N jeux de paramètres...*


---

### Description

Tirage aléatoire LHS de N jeux de paramètres (possibilité d'imposer une matrice de corrélations sur les rangs)

### Usage

```
samplingLHS(dim_x, nom=c(NA), N=1, lois=rep(0, dim_x),
  paramlois=array(0, dim=c(4, dim_x)),
  correl=0, tronq=rep(FALSE, dim_x),
  paramtronq=array(0, dim=c(2, dim_x)))
```

### Arguments

dim_x	nombre de paramètres d'entrée dans le modèle
nom	vecteur des noms des paramètres simulés (défini comme : c("V1", "V2", "V3", ...))
N	nombre de jeux de simulations
lois	vecteur contenant les types de distribution de proba pour chaque entrée: 0=uniforme ; 1=normale ; 2=lognormale ; 3=weibull 4=exponentielle ; 5=beta ; 6=triangulaire ; 7=trapezoidale 10=gumbel. Par défaut, on prend la loi uniforme

paramlois	tableau avec les parametres de chaque loi (max=4) pour chaque entree (range par colonne) : (min,max,0,0) pour uniforme (par défaut : min=0, max=1) (moy,ecart-type,0,0) pour normale, (moy du log, ecart-type du log,0,0) pour lognormale, (forme,echelle,0,0) pour Weibull, (lambda,0,0,0) pour exponentielle, (shape1, shape2,0,0) pour beta, (min,mode,max,0) pour triangulaire, (min,mode1,mode2,max) pour trapezoidale, (mode,echelle,0,0) pour Gumbel
correl	0 → pas de correlation entre parametres; 1 → pas de correlation, on supprime les correlations indesirables par la methode des permutations circulaires 2 → introduction d'une matrice de correlations sur les rangs des parametres via le fichier 'matcorrelrank.dat'
trong	vecteur pour sélectionner ou non une loi tronquée: TRUE pour loi tronquée, FALSE sinon
paramtrong	tableau avec les paramètres de troncature de chaque loi: (min,max) range par colonne

### Value

la matrice des N simulations des `dim_x` parametres

### Note

!!! WARNING !!!! : le LHS et la la troncature ne s'appliquent pas a la loi trapezoidale

\*\*\*\*\*

FONCTION UTILISEE :

truncated.R (fonctions pour lois tronquees)

\*\*\*\*\*

LIBRAIRIES REQUISES (A INSTALLER)

library(triangle)

library(evd) # Gumbel

### Author(s)

B. Iooss

### References

Stein, M. 1987. Technometrics 29:143-151

Iman and Conover. 1982. Commun. Stat. Simul. Comput. 11(3):311-334

McKay, Conover and Beckman. 1979. Technometrics 21: 239-245

samplingOptLHS

*Tirage aleatoire d'un plan LHS optimal de N jeux de parametres...***Description**

Tirage aleatoire d'un plan LHS optimal de N jeux de parametres Le plan peut etre maximin, distance-optimal ou S-optimal

**Usage**

```
samplingOptLHS(dim_x, nom=c(NA), N=1, lois=rep(0, dim_x),
  paramlois=array(0, dim=c(4, dim_x)), tronq=rep(FALSE, dim_x),
  paramtronq=array(0, dim=c(2, dim_x)),
  optimal="maximin", dup=1, pop=100, gen=4, pMut=0.1)
```

**Arguments**

dim_x	nombre de parametres d'entree dans le modele
nom	vecteur des noms des parametres simules (defini comme : <code>c("V1", "V2", "V3", ...)</code> )
N	nombre de jeux de simulations
lois	vecteur contenant les types de distribution de proba pour chaque entree 0=uniforme ; 1=normale ; 2=lognormale ; 3=weibull 4=exponentielle ; 5=beta ; 6=triangulaire ; 7=trapezoidale 10=gumbel Par default, on prend la loi uniforme
paramlois	tableau avec les parametres de chaque loi (max=4) pour chaque entree (range par colonne) : (min,max,0,0) pour uniforme (par default : min=0, max=1) (moy,ecart-type,0,0) pour normale, (moy du log, ecart-type du log,0,0) pour lognormale, (forme,echelle,0,0) pour Weibull, (lambda,0,0,0) pour exponentielle, (shape1, shape2,0,0) pour beta, (min,mode,max,0) pour triangulaire, (min,mode1,mode2,max) pour trapezoidale, (mode,echelle,0,0) pour Gumbel
tronq	TODO
paramtronq	TODO
optimal	type d'optimalite pour le LHS (par default 'maximin') 'distance' pour distance-optimal, 'S' pour S-optimal
dup	facteur pour le nb de points candidats dans les fcts maximinLHS (plan maximin) et improvedLHS (plan dist-optimal)
pop	option de la fct geneticLHS (plan S-optimal) Taper help(geneticLHS) pour en savoir plus
gen	option de la fct geneticLHS (plan S-optimal) Taper help(geneticLHS) pour en savoir plus
pMut	option de la fct geneticLHS (plan S-optimal) Taper help(geneticLHS) pour en savoir plus

**Value**

la matrice des N simulations des dim\_x parametres

**Note**

!!! WARNING !!!! : le LHS et la la troncature ne s'appliquent pas a la loi trapezoidale

\*\*\*\*\*

FONCTION UTILISEE :

truncated.R (fonctions pour lois tronquees)

\*\*\*\*\*

LIBRAIRIES REQUISES (A INSTALLER)

library(triangle)

library(lhs)

library(evd) # Gumbel

\*\*\*\*\*

**Author(s)**

B. Iooss

---

samplingSimple

*Tirage aleatoire simple de N jeux de parametres...*

---

**Description**

Tirage aleatoire simple de N jeux de parametres

**Usage**

```
samplingSimple(dim_x, nom=c(NA), N=1, lois=rep(0, dim_x),
  paramlois=array(0, dim = c(4, dim_x)), tronq=rep(FALSE, dim_x),
  paramtronq=array(0, dim = c(2, dim_x)))
```

**Arguments**

dim_x	nombre de parametres d'entree dans le modele
nom	: vecteur des noms des parametres simules (defini comme : c("V1","V2","V3",...))
N	nombre de jeux de simulations
lois	vecteur contenant les types de distribution de proba pour chaque entree 0=uniforme ; 1=normale ; 2=lognormale ; 3=weibull 4=exponentielle ; 5=beta ; 6=triangulaire ; 7=trapezoidale 10=gumbel Par default, on prend la loi uniforme
paramlois	tableau avec les parametres de chaque loi (max=4) pour chaque entree (range par colonne) : (min,max,0,0) pour uniforme (par default : min=0, max=1) (moy,ecart-type,0,0) pour normale, (moy du log, ecart-type du log,0,0) pour lognormale, (forme,echelle,0,0) pour Weibull, (lambda,0,0,0) pour exponentielle, (shape1, shape2,0,0) pour beta, (min,mode,max,0) pour triangulaire, (min,mode1,mode2,max) pour trapezoidale, (mode,echelle,0,0) pour Gumbel
tronq	vecteur pour selectionner ou non une loi tronquee TRUE pour loi tronquee, FALSE sinon (par default)
paramtronq	tableau avec les parametres de troncature de chaque loi : (min,max) range par colonne

**Value**

la matrice des N simulations des `dim_x` parametres

**Note**

!!! WARNING !!!! : la troncature ne s'applique pas a la loi trapezoidale

\*\*\*\*\*

FONCTION UTILISEE :

truncated.R (fonctions pour lois tronquees)

\*\*\*\*\*

LIBRAIRIES REQUISES (A INSTALLER)

library(triangle)

library(evd) # Gumbel

\*\*\*\*\*

**Author(s)**

B. Iooss

---

TP1.ICmorris

*Calcul d'intervalles de confiance pour la méthode Morris...*

---

**Description**

Calcul d'intervalles de confiance pour la méthode Morris

**Usage**

TP1.ICmorris(etude.morris)

**Arguments**

`etude.morris` structure issue de la fonction `morris` du package `sensitivity`

---

TP1corr

*Graphiques de corrélation entre facteurs X et sortie y de la FC...*

---

**Description**

Graphiques de corrélation entre facteurs X et sortie y de la FC issus d'une anova

**Usage**

TP1corr(etude.morris, transfo=TRUE, binf, bsup)

**Arguments**

<code>etude.morris</code>	structure issue de la fonction morris du package sensitivity
<code>transfo</code>	si TRUE, recodage de la matrice X codée dans [0,1]
<code>binf</code>	vecteur des bornes inférieures des gammes des facteurs
<code>bsup</code>	vecteur des bornes supérieures des gammes des facteurs

---

<code>TP1histo</code>	<i>Diagramme des fréquences des valeurs des facteurs échantillonnées...</i>
-----------------------	---

---

**Description**

Diagramme des fréquences des valeurs des facteurs échantillonnées issus d'une anova

**Usage**

```
TP1histo(etude.morris)
```

**Arguments**

<code>etude.morris</code>	structure issue de la fonction morris du package sensitivity
---------------------------	--

---

<code>TP1indices.aov</code>	<i>Calcul et représentation graphique des indices principaux et totaux...</i>
-----------------------------	---

---

**Description**

Calcul et représentation graphique des indices principaux et totaux issus d'une anova

**Usage**

```
TP1indices.aov(table.aov, noms, modeleAOV, titre="")
```

**Arguments**

<code>table.aov</code>	table d'anova issue de la fonction aov()
<code>noms</code>	vecteur des labels des facteurs
<code>modeleAOV</code>	modèle d'anova créé avec formula()
<code>titre</code>	titre du graphique

TP1pavage

*Construction du plan avec tirage dans les pavés défini par un plan P...***Description**

Construction du plan avec tirage dans les pavés défini par un plan P

**Usage**

```
TP1pavage(P, nrep=3, Nbclass=2, binf, bsup)
```

**Arguments**

P	matrice à Nbfac colonnes codée par des entiers de 1 à Nbclass
nrep	nbre de tirages par pavé [entier]
Nbclass	niveau de discrétisation des gammes des facteurs
binf	vecteur des bornes inf des gammes des facteurs
bsup	= vecteur des bornes sup des gammes des facteurs

**Value**

liste contenant les matrices des coordonnées entières (Plan.rep) et réelles (xx) des points tirés au hasard

**Exemples**

```
TP1pavage( rbind(1:3,c(2,2,2)),binf=c(-10,0,100),bsup=c(10,5,600),Nbclass=5)
```

TP1tirage

*Tirage uniforme dans un pavé de  $R^K$ ...***Description**Tirage uniforme dans un pavé de  $R^K$ **Usage**

```
TP1tirage(PAV, binf, bsup, Nbclass)
```

**Arguments**

PAV	vecteur des coordonnées entières d'un pavé codées de 1 à Nbclass
binf	vecteur des bornes inf des gammes des facteurs
bsup	vecteur des bornes sup des gammes des facteurs
Nbclass	niveau de discrétisation des gammes des facteurs

**Value**vecteur à  $K$  éléments

## Examples

```
TP1tirage( PAV=c(1,2,3),binf=c(-10,0,100),bsup=c(10,5,600),Nbclass=5)
```

---

<code>weed.decision</code>	<i>Décisions par défaut pour le modèle "Weed"</i>
----------------------------	---

---

## Description

Dataframe des décisions par défaut pour le modèle "Weed"

## Value

data.frame à 8 lignes (années) et 3 colonnes (facteurs)

---

<code>weed.factors</code>	<i>Facteurs d'entrée du modèle "Weed"</i>
---------------------------	---

---

## Description

Facteurs d'entrée du modèle "Weed" (ou "mauvaises herbes")

## Value

data.frame à 20 lignes (facteurs) et 4 colonnes (spécifs)

---

<code>weed.fun</code>	<i>Fonction de base du modèle "Weed"</i>
-----------------------	--

---

## Description

Fonction de base du modèle "Weed": calcul sur 1 année du modèle "Weed".

## Usage

```
weed.fun(decision, param)
```

## Arguments

<code>decision</code>	data.frame à 1 ligne et 3 colonnes Soil, Crop, Herb
<code>param</code>	vecteur des paramètres: mu, v, phi, beta.1, beta.0, chsi.1, chsi.0, delta.new, delta.old, mh, mc, Smax.1, Smax.0, Ymax, rmax, gamma et des variables d'état initiales d.im1, S.im1, SSBa.im1, DSBa.im1 (d,S,SSBa,DSBa)



**Value**

un vecteur de longueur 5 composé de:

- Sproduction de graines par eqn(m<sup>2</sup>)
- ddensité d'adventices à l'émergence (plantes par eqn(m<sup>2</sup>))
- SSBabanque de graines en surface après travail du sol (graines par eqn(m<sup>2</sup>))
- DSBabanque de graines en profondeur après travail du sol (graines par eqn(m<sup>2</sup>))
- Yieldrendement (t par ha)

---

weed.model	<i>Modèle "Weed" pour un jeu de paramètres et un jeu de décisions</i>
------------	---

---

**Description**

Modèle "Weed" pour un jeu de paramètres et un jeu de décisions sur n années

**Usage**

```
weed.model(param, decision=weed.decision, tout=FALSE)
```

**Arguments**

param	vecteur des paramètres: mu, v, phi, beta.1, beta.0, chsi.1, chsi.0, delta.new, delta.old, mh, mc, Smax.1, Smax.0, Ymax, rmax, gamma et des variables d'état initiales d.im1, S.im1, SSBa.im1, DSBa.im1 (d,S,SSBa,DSBa)
decision	data.frame à 3 colonnes Soil, Crop, Herb de valeurs 0-1 et n lignes, où n est le nombre d'années simulées
tout	TRUE si l'on veut les entrées ET les sorties dans le tableau de sortie

**Value**

une matrice n x 5 composée de:

- Sproduction de graines par eqn(m<sup>2</sup>)
- ddensité d'adventices à l'émergence (plantes par eqn(m<sup>2</sup>))
- SSBabanque de graines en surface après travail du sol (graines par eqn(m<sup>2</sup>))
- DSBabanque de graines en profondeur après travail du sol (graines par eqn(m<sup>2</sup>))
- Yieldrendement (t par ha)

**Note**

Voir weed.factors pour les valeurs min, max et nominal des parametres

**Examples**

```
decision <- data.frame(Soil=c(0,1),Crop=c(0,1),Herb=c(0,1))
weed.model( weed.factors$nominal, decision=decision )
```

weed.simule

*Simulations en série du modèle "Weed"***Description**

Simulations en série du modèle "Weed"

**Usage**

```
weed.simule(X, decision=weed.decision, sortie="rdt.total",
            nom.sortie, tout=FALSE)
```

**Arguments**

X	matrice ou data.frame des jeux de paramètres: mu, v, phi, beta.1, beta.0, chsi.1, chsi.0, delta.new, delta.old, mh, mc, Smax.1, Smax.0, Ymax, rmax, gamma et des variables d'état initiales d.im1, S.im1, SSBa.im1, DSBa.im1 (d,S,SSBa,DSBa)
decision	data.frame à 3 colonnes Soil, Crop, Herb et n lignes, où n est le nombre d'années simulées
sortie	fonction ou mot-clé donnant la nature de la ou des variables en sortie de chaque simulation (voir DETAILS)
nom.sortie	noms de la ou des variables de sortie retenues
tout	TRUE si l'on veut les entrées ET les sorties dans le tableau de sortie

**Value**

un data.frame incluant en colonnes la ou les sorties retenues. Suivant la valeur de 'tout', les entrées sont restituées ou non.

**Note**

Le paramètre 'sortie' peut être:

- soit une fonction calculant la ou les variables de sortie de chaque simulation à partir du tableau n x 5 des sorties de 'weed.model';
- soit un mot-cle pré-défini:
  - annee.finale pour avoir les 5 sorties de la dernière année;
  - rdt.total pour avoir la somme des rendements sur les n années (default);
  - banque.finale pour avoir la banque de graines en dernière année.

Pour rappel, les 5 sorties de 'weed.model' sont:

- Sproduction de graines par eqn(m<sup>2</sup>)
- ddensité d'adventices à l'émergence (plantes par eqn(m<sup>2</sup>))
- SSBabanque de graines en surface après travail du sol (graines par eqn(m<sup>2</sup>))
- DSBabanque de graines en profondeur après travail du sol (graines par eqn(m<sup>2</sup>))
- Yieldrendement (t par ha)

**Examples**

```
jeux.param <- rbind(weed.factors$binf, weed.factors$nominal, weed.factors$bsup)
weed.simule( jeux.param, sortie=function(x){sum(x[,5])}, nom.sortie="rdt.total")
weed.simule( jeux.param, sortie="annee.finale", nom.sortie="rdt.total")
```

---

wwdm.climates	<i>Séries climatiques sur 14 années, utilisées par le modèle wwdm</i>
---------------	---

---

**Description**

Séries climatiques sur 14 années, utilisées par le modèle wwdm

**Value**

data.frame à N lignes (1 par jour) et 4 colonnes (ANNEE, RG, Tmin, Tmax)

---

wwdm.factors	<i>Facteurs d'entrée du modèle "wwdm"</i>
--------------	---

---

**Description**

Facteurs d'entrée du modèle "wwdm"

**Value**

data.frame à 8 lignes (facteurs) et 4 colonnes (spécifs)

---

wwdm.model	<i>Modèle "wwdm" pour un jeu de paramètres</i>
------------	--

---

**Description**

Modèle wwdm (winter wheat dry matter) de croissance du blé, modèle de culture très simple, dynamique à pas de temps journalier

**Usage**

```
wwdm.model(param, year, climate=wwdm.climates)
```

**Arguments**

param	vecteur de paramètres de wwdm de longueur 7 ou 8
year	soit NULL soit un nombre compris entre 1 et 14
climate	nom du data.frame contenant les données climatiques

**Value**

vecteur des 223 gains journaliers de biomasse calculés par WWDM

### Note

Le modèle a deux variables d'état, l'indice de surface foliaire (LAI) et la biomasse aérienne du blé d'hiver. La sortie de la fonction est le gain de poids journalier de la matière sèche en 'g per m2 per day'. Une simulation correspond à une année climatique. Par défaut, les données climatiques sont lues dans le data.frame `wwdm.climates`, qui contient 14 années climatiques, et l'année doit être spécifiée par un nombre entre 1 et 14. Il y a deux façons de faire cela, soit par l'argument `year`, soit par la 8ème coordonnée de l'argument `param` si `year=NULL`. Lorsque `year=NULL` et qu'il n'y a que 7 coordonnées dans `param`, l'année utilisée est l'année numéro 3

### References

Makowski, D., Jeuffroy, M.-H., Guérif, 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 Boekel 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

### Examples

```
#data()
#wwdm.model()
#sum( wwdm.model() ) #biomasse cumulee
#wwdm.model(param=wwdm.factors$nominal, year=NULL, climate=wwdm.climates)
#wwdm.model(param=wwdm.factors$nominal, year=5)
```

---

wwdm.simule

*Simulations en série du modèle "wwdm"*

---

### Description

Fonction gérant une série de simulations de `wwdm`, modèle de culture dynamique à pas de temps journalier, pour le blé

### Usage

```
wwdm.simule(X, year, tout=FALSE, transfo=FALSE,
            b1=wwdm.factors$binf[1:Nbfac], b2=wwdm.factors$bsup[1:Nbfac])
```

### Arguments

<code>X</code>	dataframe à 7 ou 8 colonnes de valeurs des paramètres de <code>wwdm</code>
<code>year</code>	soit NULL, soit une valeur unique entre 1 et 14
<code>tout</code>	TRUE si l'on veut les entrées ET les sorties dans le tableau de sortie
<code>transfo</code>	TRUE si <code>X</code> contient des valeurs codées entre 0 et 1
<code>b1</code>	vecteur des 7 ou 8 bornes inférieures des paramètres si <code>transfo=TRUE</code>
<code>b2</code>	vecteur des 7 ou 8 bornes supérieures des paramètres si <code>transfo=TRUE</code>

**Value**

Biomasse aérienne accumulée avant la récolte, en g per m2

**Note**

Le modèle a deux variables d'état, l'indice de surface foliaire (LAI) et la biomasse aérienne du blé d'hiver. La fonction `wwdm.simule` ne donne en sortie que la biomasse aérienne accumulée avant la récolte, en 'g per m2'. Une simulation correspond à une année climatique. Il est possible de préciser l'année climatique, soit simulation par simulation en ajoutant une colonne 'year' à 'X', soit globalement en utilisant l'argument 'year'

**References**

Makowski, D., Jeuffroy, M.-H., Guérif, 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 Boekel 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

**Examples**

```
jeux.parametres <- as.data.frame(rbind(wwdm.factors$binf,  
                                     wwdm.factors$nominal, wwdm.factors$bsup))  
names(jeux.parametres) <- wwdm.factors$name  
wwdm.simule(jeux.parametres)
```

# Index

convertfrom.basep, 3  
convertinto.basep, 3  
convertU2N, 4  
copyright, 4  
crossing, 5

d.trunc.distr  
    (*loiGeneriqueTronquee*), 11  
dtgumbel(*loiGumbelTronquee*), 12  
dtlnorm(*loiLogNormaleTronquee*),  
    12  
dtnorm(*loiNormaleTronquee*), 13

ECmexico2012  
    (*ECmexico2012-package*), 2  
ECmexico2012-package, 2

fungus.factors, 5  
fungus.model, 6  
fungus.simule, 6

inverses.basep, 7  
ishigami.factors, 8  
ishigami.model, 8  
ishigami.simule, 9

lhs.plan, 9  
lhs2intervalle, 10  
LoiGeneriqueTronquee  
    (*loiGeneriqueTronquee*), 11  
loiGeneriqueTronquee, 11  
loiGumbelTronquee, 12  
loiLogNormaleTronquee, 12  
loiNormaleTronquee, 13

morris.mexico, 14

p.trunc.distr  
    (*loiGeneriqueTronquee*), 11  
perspPlus, 14  
planor.kernelcheck.basep, 15  
ptgumbel(*loiGumbelTronquee*), 12  
ptlnorm(*loiLogNormaleTronquee*),  
    12  
ptnorm(*loiNormaleTronquee*), 13

q.trunc.distr  
    (*loiGeneriqueTronquee*), 11  
qtgumbel(*loiGumbelTronquee*), 12  
qtlnorm(*loiLogNormaleTronquee*),  
    12  
qtnorm(*loiNormaleTronquee*), 13

r.trunc.distr  
    (*loiGeneriqueTronquee*), 11  
regular.fraction, 16  
representative.basep, 17  
rtgumbel(*loiGumbelTronquee*), 12  
rtlnorm(*loiLogNormaleTronquee*),  
    12  
rtnorm(*loiNormaleTronquee*), 13

samplingLHS, 17  
samplingOptLHS, 19  
samplingSimple, 20

TP1.ICmorris, 21  
TP1corr, 21  
TP1histo, 22  
TP1indices.aov, 22  
TP1pavage, 23  
TP1tirage, 23

weed.decision, 24  
weed.factors, 24  
weed.fun, 24  
weed.model, 25  
weed.simule, 26  
wwdm.climates, 27  
wwdm.factors, 27  
wwdm.model, 27  
wwdm.simule, 28