Package ‘xtal’

December 29, 2015

Type Package

Title Crystallization Toolset

Version 1.15

Date 2015-12-28

Author Qingan Sun, Xiaojun Li

Maintainer Qingan Sun <quinsun@gmail.com>

Description This is the tool set for crystallographer to design and analyze crystallization experiments, especially for ribosome from Mycobacterium tuberculosis.

License GPL-2 | GPL-3

Depends methods,graphics, grDevices, stats, utils

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2015-12-29 22:31:03

R topics documented:

`xtal-package` *Crystallization Tool*

Description

This is the tool set for crystallographer to design and analyze crystallization experiments, especially for ribosome from Mycobacterium tuberculosis.

Details

Package:	xtal
Type:	Package
Version:	1.0
Date:	2015-12-28
License:	GPL-2|GPL-3
Depends:	methods,graphics, grDevices, stats, utils

Author(s)

Qingan Sun, Xiaojun Li

Maintainer: Qingan Sun <quinsun@gmail.com>

`Design-class` *Class* `"Design"`

Description

Virtual Class of the experiment design

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`volume`: Object of class `"numeric"` volume of each well in matrix block

`stock`: Object of class `"data.frame"` stock composition

`portion`: Object of class `"array"` portion of each stock in each well

Methods

design2Screen `signature(object = "Design")`: ...

Author(s)

Qingan Sun, Xiaojun Li

See Also

`Design8Vertex`

Examples

```
showClass("Design")
```

`design2Screen` *Constructor of Screen from Design object*

Description

extract matrix info from Design object and put into new Screen object

Usage

```
design2Screen(object)
```

Arguments

`object`	Design class

Value

new Screen object

Author(s)

Qingan Sun, Xiaojun Li

Examples

```
# set up a Design object
# please read the 'design8Vertex' for detail
stock=matrix(nrow=8,ncol=3)
colnames(stock)=c("PEG","pH","salt")
stock[,1]=rep(c(6,16),4)
stock[,2]=rep(c(8,8,9.5,9.5),2)
stock[,3]=rep(c(0,300),each=4)
stock=data.frame(stock)
dim=list(5:0/5,3:0/3,3:0/3)
```

```
test8Vertex=design8Vertex(900,stock,dim)
# construct a new Screen object
testScreen<-design2Screen(test8Vertex)
```

`design2Screen-methods` *~~ Methods for Function* `design2Screen` ~~

Description

~~ Methods for function `design2Screen` ~~

Methods

`signature(object = "Design")` constructor of Screen from Design object

`design8Vertex` *constructor of Class Design8Vertex*

Description

Caculate the portion matrix of each stock from the 'dim', and call the new(Design8Vertex)

Usage

```
design8Vertex(volume, stock, dim)
```

Arguments

`volume`	numeric, for volume of each well in matrix block
`stock`	dataframe, the composition of each stock (8 stock in the 8-vertex design)
`dim`	list of three vectors, the dilution of stock 1 in 3 dimensions: 6X4X4

Value

new object of Design8Vertex class

Author(s)

Qingan Sun, Xiaojun Li

Examples

```
# set the stock with 3 variables: PEG concentration, pH, and salt concentration
stock<-matrix(nrow=8,ncol=3)
colnames(stock)<-c("PEG","pH","salt")
stock[,1]<-rep(c(6,16),4)
stock[,2]<-rep(c(8,8,9.5,9.5),2)
stock[,3]<-rep(c(0,300),each=4)
stock<-data.frame(stock)
dim<-list(5:0/5,3:0/3,3:0/3) # the dilution serial of stock1
#call the function and return a new object
test8Vertex<-design8Vertex(900,stock,dim)
```

`Design8Vertex-class` *Class* `"Design8Vertex"`

Description

Design Class contains the info for 8-Vertex setting of crystallization matrix

Objects from the Class

Objects can be created by calls of the form `new("Design8Vertex", ...)`.

Slots

`volume`**:** Object of class `"numeric"` the volume of each well in matrix block

`stock`**:** Object of class `"data.frame"` stock composition

`portion`**:** Object of class `"array"` portion of each stock in each well

Extends

Class "`Design`", directly.

Methods

writeTecan `signature(object = "Design8Vertex", fileName = "ANY", source = "ANY", destination = "ANY",`
...

writeTecan `signature(object = "Design8Vertex", fileName = "ANY", source = "ANY", destination = "ANY",`
...

Note

preferred constructor design8Vertex

Author(s)

Qingan Sun, Xiaojun Li

See Also

`design8Vertex` `Design`

Examples

```
showClass("Design8Vertex")
```

`Exp-class` *Class* `"Exp"`

Description

Store of experiment info of screen matrix and crystal score

Objects from the Class

Objects can be created by calls of the form `new("Exp", ...)`.

Slots

`screen`**:** Object of class `"Screen"` the screen condition

`score`**:** Object of class `"numeric"` score of crystal quality in each condition

Methods

getOptimal `signature(zga = "Exp")`: ...

Author(s)

Qingan Sun, Xiaojun Li

See Also

`getOptimal`

Examples

```
showClass("Exp")
```

getCondition — *Getter of the condition matrix in Screen object*

Description

Getter of the condition matrix in Screen object

Usage

```
getCondition(object)
```

Arguments

`object`

Value

matrix of screen condition

Author(s)

Qingan Sun, Xiaojun Li

Examples

```
# set up a Design object
# please read the 'design8Vertex' for detail
stock=matrix(nrow=8,ncol=3)
colnames(stock)=c("PEG","pH","salt")
stock[,1]=rep(c(6,16),4)
stock[,2]=rep(c(8,8,9.5,9.5),2)
stock[,3]=rep(c(0,300),each=4)
stock=data.frame(stock)
dim=list(5:0/5,3:0/3,3:0/3)
test8Vertex=design8Vertex(900,stock,dim)
# construct a new Screen object
testScreen<-design2Screen(test8Vertex)
condition<-getCondition(testScreen)
```

getCondition-methods — ~~ *Methods for Function* `getCondition` ~~

Description

~~ Methods for function `getCondition` ~~

Methods

```
signature(object = "Screen")
```

`getOptimal` *get Optimal condition in the crystallization plate*

Description

local regression for the crystal score across the screen matrix, and return the condition with the highest score

Usage

```
getOptimal(zga)
```

Arguments

`zga`	Exp object of experiment containing screen setting and score results

Author(s)

Qingan Sun, Xiaojun Li

Examples

```
# set up a Design object
# please read the 'design8Vertex' for detail
stock=matrix(nrow=8,ncol=3)
colnames(stock)=c("PEG","pH","salt")
stock[,1]=rep(c(6,16),4)
stock[,2]=rep(c(8,8,9.5,9.5),2)
stock[,3]=rep(c(0,300),each=4)
stock=data.frame(stock)
dim=list(5:0/5,3:0/3,3:0/3)
test8Vertex=design8Vertex(900,stock,dim)
# set up a Screen object
# please read the 'design2Screen' for detail
testScreen<-design2Screen(test8Vertex)
# the score from evaluation of crystal quality in screen plate
# in this example, the crystals are scored according to
# the scale proposed by Carter & Carter
score=c(2,2,1,1,1,1,1,2,3,3,2,2,1,1,4,4,4,4,1,4,5,3,3,3,2,2,2,2,2,3,1,2,4,3,3,2,1,1,
1,1,1,1,1,1,3,4,3,4,2,2,4,3,3,3,1,4,3,3,4,2,1,2,3,3,4,3,1,3,3,3,5,4,2,2,2,3,
4,3,2,2,5,4,5,5,2,2,5,5,5,5,2,2,5,5,5,4)
testExp=new(Class='Exp',screen=testScreen, score=score)
testOpt<-getOptimal(testExp)
```

`getOptimal-methods` ~~ *Methods for Function* `getOptimal` ~~

Description

~~ Methods for function `getOptimal` ~~

Methods

`signature(zga = "Exp")`

`screen` *Screen constructor*

Description

constructor of Screen object

Usage

```
screen(fac, condition, position)
```

Arguments

`fac`	
`condition`	
`position`	

Author(s)

Qingan Sun, Xiaojun Li

`Screen-class` *Class* `"Screen"`

Description

Class for screen matrix

Objects from the Class

Objects can be created by calls of the form `new("Screen", ...)`.

Slots

`fac`**:** Object of class `"character"` ~~

`condition`**:** Object of class `"matrix"` ~~

`position`**:** Object of class `"matrix"` ~~

Methods

getCondition `signature(object = "Screen")`: ...

screenCsv `signature(object = "Screen", fileName = "character")`: ...

Author(s)

Qingan Sun, Xiaojun Li

See Also

`screenCsv`

Examples

```
showClass("Screen")
```

`screenCsv` *csv exporter of Screen*

Description

write Screen condition matrix into csv file

Usage

```
screenCsv(object, fileName)
```

Arguments

`object`	Screen class
`fileName`	charactor string of csv file

Author(s)

Qingan Sun, Xiaojun Li

Examples

```
# set up a Design object
# please read the 'design8Vertex' for detail
stock=matrix(nrow=8,ncol=3)
colnames(stock)=c("PEG","pH","salt")
stock[,1]=rep(c(6,16),4)
stock[,2]=rep(c(8,8,9.5,9.5),2)
stock[,3]=rep(c(0,300),each=4)
stock=data.frame(stock)
dim=list(5:0/5,3:0/3,3:0/3)
test8Vertex=design8Vertex(900,stock,dim)
# construct a new Screen object
testScreen<-design2Screen(test8Vertex)
screenCsv(testScreen,fileName="OPT.csv")
```

`screenCsv-methods` *~~ Methods for Function* `screenCsv` *~~*

Description

~~ Methods for function `screenCsv` ~~

Methods

`signature(object = "Screen", fileName = "character")`

`writeTecan` *Tecan worklist writer*

Description

calculate the parameter for the Tecan Robot from Design object, out put a worklist file in csv format

Usage

```
writeTecan(object, fileName, source, destination, liquidType)
```

Arguments

`object`	Design class
`fileName`	character string of output worklist
`source`	character string of source name, default 'Source1' (15ml tube rack)
`destination`	character string of destination name, default 'Destination' (96-well deep block)
`liquidType`	vector of charactor of liquid type

Details

if no liquidType is provided for the stock solution, the default will be set to B

Note

This method has polymorphism. If the liquidType is missing for input, it will take it as 'B' as 'Buffer' for granted

Author(s)

Qingan Sun, Xiaojun Li

Examples

```
# set up a Design object
# please read the 'design8Vertex' for detail
stock=matrix(nrow=8,ncol=3)
colnames(stock)=c("PEG","pH","salt")
stock[,1]=rep(c(6,16),4)
stock[,2]=rep(c(8,8,9.5,9.5),2)
stock[,3]=rep(c(0,300),each=4)
stock=data.frame(stock)
dim=list(5:0/5,3:0/3,3:0/3)
test8Vertex=design8Vertex(900,stock,dim)
writeTecan(test8Vertex,"testTecan_defaultLiquid.csv","Source1",'Destination')
liquidType=rep(c('B','P'),4)
writeTecan(test8Vertex,"testTecan_setLiquid.csv","Source1",'Destination',liquidType=liquidType)
```

`writeTecan-methods` ~~ *Methods for Function* `writeTecan` ~~

Description

~~ Methods for function `writeTecan` ~~

Methods

```
signature(object = "Design8Vertex", fileName = "ANY", source = "ANY", destination = "ANY", liquidTy

signature(object = "Design8Vertex", fileName = "ANY", source = "ANY", destination = "ANY", liquidTy
```

Index