Package 'qualityTools'

February 24, 2016

Type Package

Title Statistical Methods for Quality Science

Version 1.55

Date 2016-02-23

Author Thomas Roth

Maintainer Thomas Roth <thomas.roth@alumni.tu-berlin.de>

Description Contains methods associated with the Define, Measure, Analyze, Improve and Control (i.e. DMAIC) cycle of the Six Sigma Quality Management methodology.It covers distribution fitting, normal and non-normal process capability indices, techniques for Measurement Systems Analysis especially gage capability indices and Gage Repeatability (i.e Gage RR) and Reproducibility studies, factorial and fractional factorial designs as well as response surface methods including the use of desirability functions. Improvement via Six Sigma is project based strategy that covers 5 phases: Define - Pareto Chart; Measure - Probability and Quantile-Quantile Plots, Process Capability Indices for various distributions and Gage RR Analyze i.e. Pareto Chart, Multi-Vari Chart, Dot Plot; Improve - Full and fractional factorial, response surface and mixture designs as well as the desirability approach for simultaneous optimization of more than one response variable. Normal, Pareto and Lenth Plot of effects as well as Interaction Plots; Control - Quality Control Charts can be found in the 'qcc' package. The focus is on teaching the statistical methodology used in the Quality Sciences.

License GPL-2

URL http://www.r-qualitytools.org

LazyLoad yes

Depends R(>= 2.15.0), graphics, methods, Rsolnp, MASS

Imports

Suggests

Encoding latin1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-02-24 17:40:35

R topics documented:

`qualityTools-package` *Statistical Methods for Quality Sciences*

Description

This Package contains methods associated with the (**D**)**efine** (**M**)**easure** (**A**)**nalyze** (**I**)**mprove** and (**C**) **ontrol** (i.e. **DMAIC**) cycle of the Six Sigma Quality Management methodology.

1. **Define:** Pareto Chart
2. **Measure:** Probability and Quantile-Quantile Plots, Process Capability Ratios for various distributions and Gage R&R
3. **Analyze:** Pareto Chart, Multi-Vari Chart, Dot Plot
4. **Improve:** Full and fractional factorial, response surface and mixture designs as well as the desirability approach for simultaneous optimization of more than one response variable. Normal, Pareto and Lenth Plot of effects as well as Interaction Plots etc.
5. **Control:** Quality Control Charts can be found in the qcc package

Details

Package:	qualityTools
Type:	Package
Version:	0.96.2
Date:	2012-07-01
URL:	http://r-qualitytools.org
License:	GPL-2
LazyLoad:	yes

Note

This package is primarily used for teaching! The package vignette is available under http://www.r-qualitytools.org.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Maintainer: Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
example(paretoChart)
example(mvPlot)
example(dotPlot)
example(qqPlot)
example(ppPlot)
example(pcr)
example(gageRR)
example(facDesign)

example(fracDesign)
example(effectPlot)
example(taguchiDesign)
example(rsmDesign)
example(paretoPlot)
example(wirePlot)
example(contourPlot)
example(mixDesign)
example(desirability)
example(optimum)
```

---

adSim *Bootstrap-based Anderson-Darling Test for Univariate Distributions*

Description

Anderson-Darling test for univariate distributions with bootstrap-based p-value determination. It also enables the p-value determination from tabled critical values.

Usage

```
adSim(x, distribution = "normal", b = 10000)
```

Arguments

x	A numeric vector.
distribution	A character string. Distributions "cauchy", "exponential", "gumbel", "gamma", "log-normal", "lognormal", "logistic", "normal" and "weibull" are recognized.
b	A numeric value giving the frequency of bootstraps. Any value in [1000,1000000] is allowed. If b is set to NA, the Anderson-Darling test will be applied without simulation. b should be chosen carefully. High values such as 1 mio can easily make your computer run up to 3 hours (depending on the distribution, on the sample size and on your computer system).

Details

First parameter estimation for the tested distribution is performed. In the majority of cases Maximum-Likelihood-Estimation is used as fitting method and is mainly provided by fitdistr() from package MASS. Parameters of normal and log-normal distribution are fitted by mean and standard deviation. Cauchy parameters are fitted by the sums of the weighted order statistic, when p-value determination should be done from tabled critical values. The Anderson-Darling statistic is calculated based on the estimated parameters. Parametric bootstrapping provides the distribution of the Anderson-Darling test which is used in order to determine a p-value. Simulation-based Anderson-Darling distribution and critical values for selected quantiles are printable. When simulation is not desired, a p-value is obtained from tabled critical values (no exact expression exists except for log-normal, normal and exponential distribution). Tabled critical values for selected quantiles are printable as well.

Value

An object of class "adSim", a list with six components,

distribution	the distribution the Anderson-Darling test was applied for
parameter_estimation	the estimated parameters
Anderson_Darling	the value of the Anderson-Darling test
p_value	the corresponding p-value (simulated or tabled value)
crititical_values	the corresponding critical values (simulated or tabled 0.75, 0.90, 0.95, 0.975 and 0.99 quantiles)
simAD	bootstrap-based Anderson-Darling distribution

Author(s)

Marco Wannicke
Thomas Roth <thomas.roth@tu-berlin.de>

References

Stute, W./ Manteiga, W./ Quindimil, M.(1993): Bootstrap based goodness-of-fit-tests; Metrika, Vol. 40, p.243-256; Physica Verlag
D'Agostino, R.B./ Stephens, M.A. (1986): Goodness-Of-Fit Techniques; illustrated edition; p.97-193; Marcel Dekker Inc.

See Also

http://www.r-qualitytools.org

Examples

```
x <- rnorm(25,32,2)
adSim(x)
adSim(x,"logistic",2000)
adSim(x,"cauchy",NA)
```

`aliasTable` *Display an alias table*

Description

Function to do display an alias table for the aliased effects of a fractional factorial design.

Usage

```
aliasTable(fdo, degree, show = TRUE)
```

Arguments

fdo	needs to be an object of class facDesign.
degree	numeric giving the degree of interaction i.e. degree=3 means up to threeway interactions.
show	a logigal value specifying whether the alias table should be shown or not. By default show is set to 'TRUE'.

Value

The function `aliasTable()` returns a matrix indicating the aliased effects.

Note

For a more detailed example which shows the effect of applying `aliasTable()` to an object of class facDesign, please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

fracDesign,
fracChoose,
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#create a fractional factorial design
fracFac = fracDesign(k = 3, gen = "C = AB")
#display the alias table
aliasTable(fracFac)
```

---

as.data.frame-methods *Methods for Function as.data.frame in Package base*

Description

Methods for function as.data.frame in Package base

Methods

signature(x = "MSALinearity") object of class MSALinearity

signature(x = "ANY") ANY

signature(x = "desOpt") object of class desOpt

signature(x = "facDesign") object of class facDesign

signature(x = "mixDesign") object of class mixDesign

signature(x = "taguchiDesign") object of class taguchiDesign

signature(x = "pbDesign") object of class taguchiDesign

signature(x = "steepAscent") object of class steepAscent

signature(x = "gageRR") object of class gageRR

See Also

http://www.r-qualitytools.org

as.data.frame.desOpt *Coerce to a data.frame*

Description

S3 generic for desOpt class.

Usage

```
## S3 method for class 'desOpt'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class desOpt.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.desOpt returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

as.data.frame.facDesign *Coerce to a data.frame*

Description

S3 generic for facDesign class.

Usage

```
## S3 method for class 'facDesign'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class facDesign.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.facDesign returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

---

as.data.frame.gageRR *Coerce to a data.frame*

Description

S3 generic for gageRR class.

Usage

```
## S3 method for class 'gageRR'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class gageRR.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.gageRR returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

---

as.data.frame.mixDesign

Coerce the object of class mixDesign to a data.frame

Description

S3 generic for mixDesign class.

Usage

```
## S3 method for class 'mixDesign'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class mixDesign.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.mixDesign returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

as.data.frame.MSALinearity

Coerce to a data.frame

Description

S3 generic for MSALinearity class.

Usage

```
## S3 method for class 'MSALinearity'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class MSALinearity.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.MSALinearity returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

as.data.frame.pbDesign

Coerce to a data.frame

Description

S3 generic for pbDesign class.

Usage

```
## S3 method for class 'pbDesign'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class pbDesign.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.pbDesign returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdar@mailbox.tu-berlin.de>

See Also

http://www.r-qualitytools.org

---

as.data.frame.steepAscent

Coerce to a data.frame

Description

S3 generic for steepAscent class.

Usage

```
## S3 method for class 'steepAscent'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class steepAscent.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.steepAscentDesign returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

---

as.data.frame.taguchiDesign

Coerce to a data.frame

Description

S3 generic for taguchiDesign class.

Usage

```
## S3 method for class 'taguchiDesign'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	needs to be an object of class taguchiDesign.
row.names	vector containing the row names.
optional	logical value. If 'TRUE', setting row names and converting column names (to syntactic names: see make.names) is optional. By default optional is set to 'TRUE'.
...	additional arguments to be passed to or from methods.

Value

The function as.data.frame.taguchiDesign returns a data frame.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

`averagePlot` *Function to create average Plots*

Description

`averagePlot` creates all x-y plots of averages by size out of an object of class `gageRR`.
Therfore the averages of the multiple readings by each operator on each part are plotted with the reference
value or overall part averages as the index.

Usage

```
averagePlot(x, main, xlab, ylab, col, ask=TRUE, single=FALSE, ...)
```

Arguments

`x`	needs to be an object of class `gageRR`.
`main`	a main title for the plot
`xlab`	a label for the x axis
`ylab`	a label for the y axis
`col`	plotting color
`ask`	a logical value. If 'TRUE' (default) the user is asked for input, before a new figure is drawn.
`single`	a logical value.If 'TRUE' a new graphic device will be opened for each plot. By default `single` is set to 'FALSE'. For further information see details.
`...`	arguments to be passed to methods, such as graphical parameters (see `par`).

Details

`averagePlot` will split the screen into maximal 3x2 subscreens in which plots will be plotted. If the six subscreens are not enough the function will (by default) ask the user before plotting the next few plots. If `ask` is set to 'FALSE' the function will open as many graphic devices as necessary to show all plots.
There are two possible ways to avoid the described internal routine of splitting the screen:

- If `single` is set to 'TRUE' all plots will be plotted in seperate single devices.
- If `par(mfrow)` is unequal to 'c(1,1)' the function will adapt the setting given by `par`.

Note

Please do read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org`.

Author(s)

Thomas Roth: <thomas.roth@tu-berlin.de>
Etienne Stockhausen: <stocdarf@mailbox.tu-berlin.de>

References

The idea of the plot and the example given by `example(averagePlot)` are out of:

- CHRYSLER Group LLC; FORD Motor Company; GENERAL MOTORS Corporation: Measurement System Analysis (MSA),
p.114, 4rd ed. Southfield: AIAG, 2010.

See Also

gageRR
par
http://www.r-qualitytools.org

Examples

```
# create gageRR-object
gdo = gageRRDesign(Operators = 3, Parts = 10, Measurements = 3,
                   randomize = FALSE)
# vector of responses
y = c(0.29,0.08, 0.04,-0.56,-0.47,-1.38,1.34,1.19,0.88,0.47,0.01,0.14,-0.80,
      -0.56,-1.46, 0.02,-0.20,-0.29,0.59,0.47,0.02,-0.31,-0.63,-0.46,2.26,
      1.80,1.77,-1.36,-1.68,-1.49,0.41,0.25,-0.11,-0.68,-1.22,-1.13,1.17,0.94,
      1.09,0.50,1.03,0.20,-0.92,-1.20,-1.07,-0.11, 0.22,-0.67,0.75,0.55,0.01,
      -0.20, 0.08,-0.56,1.99,2.12,1.45,-1.25,-1.62,-1.77,0.64,0.07,-0.15,-0.58,
      -0.68,-0.96,1.27,1.34,0.67,0.64,0.20,0.11,-0.84,-1.28,-1.45,-0.21,0.06,
      -0.49,0.66,0.83,0.21,-0.17,-0.34,-0.49,2.01,2.19,1.87,-1.31,-1.50,-2.16)
# appropriate responses
response(gdo)=y
# perform and gageRR
gdo=gageRR(gdo)
averagePlot(gdo,pch=19)
```

`averagePlot-methods` *Methods for Function averagePlot*

Description

Methods for function `averagePlot`

Methods

`signature(object = "gageRR")` function to create an average plot for an object of class gageRR.

See Also

http://www.r-qualitytools.org

`block-methods` *Get and set methods*

Description

Get and set the Blocking structure for a 'facDesign' object

Usage

```
## S4 method for signature 'facDesign'
block(object)
## S4 replacement method for signature 'facDesign'
block(object) <- value
```

Arguments

`object`	a `facDesign` object
`value`	data.frame or vector

Methods

`signature(x = "facDesign")` Get and set the Blocking structure for a `facDesign` object.

Note

`block` is to be rewritten

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

response
facDesign
http://www.r-qualitytools.org

Examples

```
#NA in response column
fdo = facDesign(k = 3)
block(fdo)
```

`blockGen-methods` *Get and set methods*

Description

Get and set the generators for blocking

Usage

```
## S4 method for signature 'facDesign'
blockGen(object)
## S4 replacement method for signature 'facDesign'
blockGen(object) <- value
```

Arguments

`object`	a `facDesign` object
`value`	data.frame or vector

Methods

`signature(x = "facDesign")` Get and set the generators for blocking.

Note

`blockGen` is to be rewritten

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

`response`
`facDesign`
http://www.r-qualitytools.org/html/Improve.html

`blocking` *Blocking*

Description

Blocks a given factorial or response surface design.

Usage

```
blocking(fdo, blocks, BoR = FALSE, random.seed, useTable = "rsm", gen)
```

Arguments

`fdo`	needs to be an object of class `facDesign`.
`blocks`	numerical value giving the number of blocks.
`BoR`	logical value indicating whether the replicates should be blocked or not. By default `BoR` is set to 'FALSE'.
`random.seed`	numerical variable for `set.seed` to generate repeatable results for randimization within blocks.
`useTable`	character indicating which table to use. The follwing options will be accepted: • "rms" - table from reference • "calc" - table calculated by qualityTools
`gen`	giving the generator that will be used.

Value

The function `blocking()` returns a an object of class facDesign with blocking structure.

Note

`blocking` is still buggy at this time and Blocking on Replicates is not yet implemented and Blocking requires k >= 3 factors. This will change soon.
For an example in context which shows the effect of applying `blocking()` to an object of class `facDesign`, please read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org/html/Improve.html`.

Author(s)

Thomas Roth `<thomas.roth@tu-berlin.de>`

References

MYERS, Raymond H.; MONTGOMERY, Douglas C.; ANDERSON-COOK, Christine M.: Response Surface Methodology. New York: WILEY ,2009

See Also

facDesign
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#create a 2^3 full factorial design
fdo = facDesign(k = 3)

#make it a design with 2 blocks
blocking(fdo, 2)

#create a response surface design for 3 factors
fdo = rsmDesign(k = 3)

#make it a design with 3 blocks (i.e. 1 block for star part and 2 blocks
#for the cube part)
blocking(fdo, 3)
```

---

centerCube-methods *Get and set methods*

Description

Set or get the Center Points of the cube portion for an object of class facDesign

Usage

```
## S4 method for signature 'facDesign'
centerCube(x)
## S4 replacement method for signature 'facDesign'
centerCube(x) <- value
```

Arguments

x	a facDesign object
value	data.frame or vector

Methods

signature(objectc = "facDesign") Get and set the centerCube for the factors in an object of class facDesign. So far used internally.

See Also

centerStar
cube
star
http://www.r-qualitytools.org

Examples

```
#create a response surface design for k = 3 factors
rsdo = rsmDesign(k = 3)

#split design into two blocks
rsdo = blocking(rsdo, 2)

#set two Center Points per block
centerCube(rsdo) = data.frame(A = c(0,0), B = c(0,0), C = c(0,0))

#get the centerPoints of the cube portion
centerCube(rsdo)
```

---

centerStar-methods *Get and set methods*

Description

Get and set the Center Points of the star portion for an object of class facDesign

Usage

```
## S4 method for signature 'facDesign'
centerStar(x)
## S4 replacement method for signature 'facDesign'
centerStar(x) <- value
```

Arguments

x	a facDesign object
value	data.frame or vector

Methods

`signature(objectc = "facDesign")` Get and set the Center Points of the star portion for an object of class facDesign. So far used internally.

See Also

centerCube
cube
star
http://www.r-qualitytools.org

Examples

```
#create a response surface design for k = 3 factors
rsdo = rsmDesign(k = 3)
centerStar(rsdo)
```

cg *Function to calculate and visualize the gage capability.*

Description

Function visualize the given values of measurement in a run chart and in a histogram. Furthermore the "centralized Gage potential index" Cg and the "non-centralized Gage Capability index" Cgk are calculated and displayed.

Usage

```
cg             (x, target, tolerance, ref.interval, facCg, facCgk, n = 0.2,
                type, col, pch, xlim, ylim, conf.level = 0.95, cex.val = 1.5)

cgToleranceView(x, target, tolerance, ref.interval, facCg, facCgk, n = 0.2,
               type, col, pch, xlim, ylim, main, conf.level = 0.95, cex.val = 1,
                cgOut = TRUE)

cgHist         (x, target, tolerance, ref.interval, facCg, facCgk, n = 0.2, col,
                xlim, ylim, main, conf.level = 0.95, cex.val = 1, cgOut = TRUE)

cgRunChart     (x, target, tolerance, ref.interval, facCg, facCgk, n = 0.2,
                type, col, pch, xlim, ylim,main, conf.level = 0.95, cex.val = 1,
                cgOut = TRUE)
```

Arguments

x	a vector containing the measured values.
target	a numeric value giving the expected target value for the x-values.
tolerance	vector of length 2 giving the lower and upper specification limits.
ref.interval	numeric value giving the confidence intervall on which the calculation is based. By default it is based on 6 sigma methodology. Regarding the normal distribution this relates to `pnorm(3) - pnorm(-3)` which is exactly 99.73002 percent If the calculation is based on an other sigma value `ref.interval` needs to be adjusted. To give an example: If the sigma-level is given by 5.15 the `ref.interval` relates to `pnorm(5.15/2)-pnorm(-5.15/2)` which is exactly 0.989976 percent.
facCg	numeric value as a factor for the calculation of the gage potential index. The default Value for facCg is '0.2'.
facCgk	numeric value as a factor for the calulation of the gage capability index. The default value for facCgk is '0.1'.
n	numeric value between '0' and '1' giving the percentage of the tolerance field (values between the upper and lower specification limits given by tolerance) where the values of x should be positioned. Limit lines will be drawn. Default value is '0.2'.

type	what type of plot should be drawn in the run chart. Possible types see plot.
col	color of the curve in the run chart.
pch	variable specifies the symbols of the run chart. Details see par.
xlim	vector of length 2 giving the limits for the x axis of the run chart.
ylim	vector of length 2 giving the limits for the y axis of the run chart.
main	an overall title for the plot: see title.
conf.level	confidence level for internal t.test checking the significance of the bias between target and mean of x. The default value is '0.95'. The result of the t.test is shown in the histogram on the left side.
cex.val	numeric value giving the size of the text in the legend. See also par.
cgOut	logical value deciding wether the Cg and Cgk values should be plotted in a legend. Only available for the function cgHist, cgToleranceView and cgRunChart. The default value for cgOut is 'TRUE'.

Details

The calculation of the potential and actual gage capability are based on the following formulae:

- Cg = (facCg * tolerance[2]-tolerance[1])/ref.interval
- Cgk = (facCgk * abs(target-mean(x))/(ref.interval/2)

If the usage of the historical process variation is preferred the values for the tolerance tolerance must be adjusted manually. That means in case of the 6 sigma methodolgy for example, that tolerance = 6 * sigma[process].

Value

Function returns a list of numeric values. The first element contains the calculated centralized gage potential index Cg and the second contains the non-centralized gage capability index Cgk.

Note

Support for other distributions than normal might be included later in an update.
For a more detailed example which shows the usage cg() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Measure.html.

Author(s)

Thomas Roth: <thomas.roth@tu-berlin.de>
Etienne Stockhausen: <stocdarf@mailbox.tu-berlin.de>

References

- DIETRICH, Edgar; SCHULZE, Alfred: Pruefprozesseignung, 3rd ed. Munich: Carl Hanser, 2007.
- DIETRICH, Edgar et al: Eignungsnachweis von Messsystemen, 3rd ed. Munich: Carl Hanser, 2008.

See Also

gageLin
gageRR
plot
par
http://www.r-qualitytools.org/html/Measure.html

Examples

```
#simple example with default values
cg(rnorm(125,mean = 10.01 ,sd = 0.1), target = 10, tolerance = c(8,12))
#example with larger n and adjusted ref. interval
cg(rnorm(25,mean = 1.01 ,sd = 0.5), ref.interval=pnorm(5.5/2)-pnorm(-5.5/2), n=0.3)
#example with changed factors for Cg and Cgk
cg(rnorm(75, sd = 0.1), facCg = 0.15, facCgk = 0.075, tolerance = c(-10,10)/6)
```

`code2real` *Coding*

Description

Function to calculate the real value of a coded value.

Usage

```
code2real(low, high, codedValue)
```

Arguments

`low`	numeric value giving the lower boundary.
`high`	numeric value giving the higher boundary.
`codedValue`	numeric avlue giving the coded value that will be calculated.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

Examples

```
code2real(160, 200, 0)
```

---

compPlot *Function to create comparison Plots*

Description

compPlot creates comparison x-y plots of an object of class gageRR.
The averages of the multiple readings by each operator on each part are plotted against each other with the operators as indices. This plot compares the values obtained by one operator to those of another.

Usage

```
compPlot(x, main, xlab, ylab, col, cex.lab, fun = NULL, ...)
```

Arguments

x	needs to be an object of class gageRR.
main	a main title for the plot
xlab	a label for the x axis
ylab	a label for the y axis
col	plotting color
cex.lab	the magnification to be used for x and y labels relative to the current setting of cex
fun	(optional) function that will be applied to the multiple readings of each part. fun should be an object of class function like mean,median, sum, etc. By default, fun is set to 'NULL' and all readings will be plotted.
...	arguments to be passed to methods, such as graphical parameters (see par).

Note

Please do read the vignette for the package qualityTools at http://www.r-qualitytools.org.

Author(s)

Thomas Roth: <thomas.roth@tu-berlin.de>
Etienne Stockhausen: <stocdarf@mailbox.tu-berlin.de>

References

The idea of the plot and the example given by `example(compPlot)` are out of:

- CHRYSLER Group LLC; FORD Motor Company; GENERAL MOTORS Corporation: Measurement System Analysis (MSA),
  p.115, 4rd ed. Southfield: AIAG, 2010.

See Also

gageRR
par
http://www.r-qualitytools.org

Examples

```
#create gageRR-object
gdo = gageRRDesign(Operators = 3, Parts = 10, Measurements = 3,
      randomize = FALSE)
#vector of responses
y = c(0.29,0.08, 0.04,-0.56,-0.47,-1.38,1.34,1.19,0.88,0.47,0.01,0.14,-0.80,
      -0.56,-1.46, 0.02,-0.20,-0.29,0.59,0.47,0.02,-0.31,-0.63,-0.46,2.26,
      1.80,1.77,-1.36,-1.68,-1.49,0.41,0.25,-0.11,-0.68,-1.22,-1.13,1.17,0.94,
      1.09,0.50,1.03,0.20,-0.92,-1.20,-1.07,-0.11, 0.22,-0.67,0.75,0.55,0.01,
      -0.20, 0.08,-0.56,1.99,2.12,1.45,-1.25,-1.62,-1.77,0.64,0.07,-0.15,-0.58,
      -0.68,-0.96,1.27,1.34,0.67,0.64,0.20,0.11,-0.84,-1.28,-1.45,-0.21,0.06,
      -0.49,0.66,0.83,0.21,-0.17,-0.34,-0.49,2.01,2.19,1.87,-1.31,-1.50,-2.16)
#appropriate responses
response(gdo)=y
#perform and gageRR
gdo=gageRR(gdo)
compPlot(gdo,pch=19)
```

`compPlot-methods` *Methods for Function compPlot*

Description

Methods for function `compPlot`

Methods

`signature(object = "gageRR")` function to create an comparison plot for an object of class gageRR.

See Also

http://www.r-qualitytools.org

confounds *Confounded Effects*

Description

Function to display confounded effects of a fractional factorial design in a human readable way.

Usage

```
confounds(x, depth = 2)
```

Arguments

x	needs to be an object of class facDesign.
depth	numeric value - up to depth-way confounded interactions are printed

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

contourPlot *Contour Plot*

Description

Creates a contour diagramm for an object of class facDesign.

Usage

```
contourPlot(x, y, z, data = NULL, xlim, ylim, main, xlab, ylab, form = "fit",
            col = 1, steps, factors, fun)
```

Arguments

x	name providing the Factor A for the plot.
y	name providing the Factor B for the plot.
z	name giving the Response variable.
data	needs to be an object of class facDesign and contains the names of x,y,z.
xlim	vector giving the range of the x-axis.
ylim	vector giving the range of the y-axis.

main	an overall title for the plot: see title.
xlab	a title for the x axis: title.
ylab	a title for the y axis: title.
form	a character string or a formula with the syntax "y~ x+y + x*y". If form is a character it has to be one out of the following: • "quadratic" • "full" • "interaction" • "linear" • "fit" "fit" takes the formula from the fit in the facDesign object fdo. Quadratic or higher orders should be given as I(Variable^2). By default form is set as "fit".
col	a predefined (1, 2, 3 or 4) or self defined colorRampPalette or color to be used (i.e. "red").
steps	number of grid points per factor. By default steps = 25.
factors	list of 4th 5th factor with value i.e. factors = list(D = 1.2, E = -1), if nothing is specified values will be the mean of the low and the high value of the factors.
fun	function to be applied to z "desirability".

Details

This function can be used to display the desirability of each response by specifying fun = "desirability" or the fun = "overall" (i.e. composed) desirability of all responses. The required desirabilities can be set using desires.

Value

The function contourPlot() returns an invisible list with the following entries:

- x - locations of grid lines for x at which the values in z are measured
- y - locations of grid lines for y at which the values in z are measured
- z - a matrix containing the values of z to be plotted

Note

For an example in context which shows the usage of the function contourPlot() to an object of class facDesign, please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

wirePlot
filled.contour
persp
colorRampPalette
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#create a response surface design and assign random data to response y
fdo = rsmDesign(k = 3, blocks = 2)
response(fdo) = data.frame(y = rnorm(nrow(fdo)))
par(mfrow = c(2,3))

#I - display linear fit
contourPlot(A,B,y, data = fdo, form = "linear")
#II - display full fit (i.e. effect, interactions and quadratic effects
contourPlot(A,B,y, data = fdo, form = "full")
#III - display a fit specified before
fits(fdo) = lm(y ~ B + I(A^2) , data = fdo)
contourPlot(A,B,y, data = fdo, form = "fit")
#IV - display a fit given directly
contourPlot(A,B,y, data = fdo, form = "y ~ A*B + I(A^2)")
#V - display a fit using a different colorRamp
contourPlot(A,B,y, data = fdo, form = "full", col = 2)
#VI - display a fit using a self defined colorRamp
myColour = colorRampPalette(c("green", "gray","blue"))
contourPlot(A,B,y, data = fdo, form = "full", col = myColour)
```

---

contourPlot3 *function to create a ternary plot (contour plot) for an object of class mixDesign*

Description

This function creates a ternary plot (contour plot) for mixture designs (i.e. object of class mixDesign)

Usage

```
contourPlot3(x, y, z, response, data = NULL, main, xlab, ylab, zlab, border,
             form = "linear", col = 1, col.text, cex.axis, axes = TRUE,
             steps, factors)
```

Arguments

x	factor 1 of the mixDesign object.
y	factor 2 of the mixDesign object.

z	factor 3 of the mixDesign object.
response	the response of the mixDesign object.
data	the mixDesign object from which x,y,z and the response are taken.
main	giving an overall title for the plot: see title.
xlab	giving a title for the x axis : see title.
ylab	giving a title for the y axis : see title.
zlab	giving a title for the z axis : see title.
border	numeric or character (for example "red") value specifying the color of the surroundimg the ternary plot. By default border is set to "white"
form	a character string or a formula with the syntax "y ~ A + B +C". If form is a character string it has to be one out of the following: • "linear" • "quadratic" • "fullCubic" • "specialCubic" How the form influences the output is described in the reference listed below. By default form is set as "linear".
col	a predefined (1, 2, 3 or 4) or self defined colorRampPalette.
col.text	a numerical value or a character string (like "red") giving the color oft the inscription of the axes. The default value for col.text is '1'.
cex.axis	numerical value giving the size of the inscription of the axes. The default value cex.axis is '1'.
axes	logical value specifying whether the axes should be plotted or not. By default axes is set to 'TRUE'.
steps	resolution of the plot (number of rows for the square matrix), also number of grid points per factor. By default steps = 25.
factors	list of factors for categorizing with setting in case there are more than 3 factors (not yet implemented).

Value

contourPlot3 returns an invisible matrix containing the response values as NA's and numerics.

Note

This method is still under construction!
For an example in context which shows the usage of the function contourPlot3() to an object of class mixDesign, please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

MYERS, Raymond H.; MONTGOMERY, Douglas C.; ANDERSON-COOK, Christine M.: Response Surface Methodology. New York: WILEY ,2009

See Also

wirePlot3
mixDesign
colorRampPalette
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#yarn example p.564 Response Surface Methodology
mdo = mixDesign(3,2, center = FALSE, axial = FALSE, randomize = FALSE,
                replicates  = c(1,1,2,3))
names(mdo) = c("polyethylene", "polystyrene", "polypropylene")
units(mdo) = "percent"
elongation = c(11.0, 12.4, 15.0, 14.8, 16.1, 17.7, 16.4, 16.6, 8.8, 10.0, 10.0,
               9.7, 11.8, 16.8, 16.0)
response(mdo) = elongation

dev.new(14,14); par(mfrow = c(2,2))
contourPlot3(A, B, C, elongation, data = mdo, form = "linear")
contourPlot3(A, B, C, elongation, data = mdo, form = "quadratic", col = 2)
contourPlot3(A, B, C, elongation, data = mdo,
             form = "elongation ~ I(A^2) - B:A + I(C^2)", col = 3)
contourPlot3(A, B, C, elongation, data = mdo, form = "quadratic",
             col = colorRampPalette(c("yellow", "white", "red")))
```

---

cp *Process Capability Indices*

Description

Calculates the process capability cp, cpk, cpkL (onesided) and cpkU (onesided) for a given dataset and distribution.
A histogramm with a density curve is displayed along with the specification limits and a Quantile-Quantile Plot for the specified distribution.
Lower-, upper and total fraction of nonconforming entities are calculated. Box Cox Transformations are supported as well as the calculation of Anderson Darling Test Statistics.

Usage

```
cp(x, distribution = "normal", lsl, usl, target, boxcox = FALSE,
    lambda = c(-5,5), main, xlim, ylim, grouping = NULL, std.dev = NULL,
    conf.level = 0.9973002, start, lineWidth = 1, lineCol = "red",
    lineType = "solid", specCol = "red3", specWidth = 1, cex.text = 2,
    cex.val = 1.5, cex.col = "darkgray", plot = TRUE, bounds.lty = 3,
    bounds.col = "red", ...)
```

Arguments

x	numeric vector containing the values for which the process capability should be calculated.
distribution	character string specifying the distribution of x. The function cp will accept the following caracter strings for distribution: • "normal" • "log-normal" • "exponential" • "logistic" • "gamma" • "weibull" • "cauchy" • "beta" • "f" • "t" • "geometric" • "poisson" • "negative-binomial" By default distribution is set to "normal".
lsl	numeric value for the lower specification limit.
usl	numeric value for the upper specification limit.
target	(optional) numeric value giving the target value.
boxcox	logical value specifying whether a Box-Cox transformation should be performed or not. By default boxcox is set to 'FALSE'.
lambda	(optional) lambda for the transformation, default is to have the function estimate lambda.
main	an overall title for the plot: see title.
xlim	vector giving the range of the x-axis.
ylim	vector giving the range of the y-axis.
grouping	(optional) If grouping is given the standard deviation is calculated as mean standard deviation of the specified subgroups corrected by the factor c4 and expected fraction of nonconforming is calculated using this standard deviation.

std.dev	(optional) historical standard devation (only provided for normal distribution!).
conf.level	numeric value between '0' and '1' giving the confidence interval. By default conf.level is 0.9973 (99.73%) which is the reference interval bounded by the 99.865% and 0.135% quantile.
start	a named list giving the parameters to be fitted with initial values. Must be supplied for some distribution (see fitdistr of the R-package MASS).
lineWidth	a numeric value specifying the width of the line for the density curve.
lineCol	numerical value or character string (like "red") specifying the color of the line for the density curve.
lineType	character string specifying the line type e.g. "dashed", "solid", etc.
specCol	numerical value or character string specifying the color for the specification limits.
specWidth	numerical value specifying the line width for the specification limits.
cex.text	numerical value specifying the cex for lsl, usl and target.
cex.val	numerical value specifying the cex for the process capability ratios.
cex.col	numerical value or character string specifying the color for lsl, usl and target.
bounds.col	graphical parameter. For further details see ppPlot or qqPlot.
bounds.lty	graphical parameter. For further details see ppPlot or qqPlot.
plot	logical value. If set to 'FALSE' the graphical output will be omitted.\ By default plot is set to 'TRUE'.
...	some other graphical parameters.

Details

Distribution fitting is deligated to function fitdistr of the R-package MASS as well as the calculation of lambda for the Box Cox Transformation. p-values for Anderson Darling Test are reported for the most important distributions.
cpk is always min(cpK, cpL).

- pt stands for total fraction nonconforming
- pu stands for upper fraction nonconforming
- pl stands for lower fraction nonconforming
- cp stands for process capability index
- cpkL stands for lower process capability index
- cpkU stands for upper process capability index
- cpk stands for minimum process capability index

Note

At this point there's no distinction made between process performance P_pk and process capability. The latter implies a process that is in statistical contro whereas process performance is estimated for a process that might not have been demonstrated to be in a state of statistical control.
For a detailed example which shows the usage of the function `cp()` please read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org/html/Analyze.html`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

- ISO (2007). Statistical methods - Process performance and capability statistics for measured quality characteristics (ISO 21747:2006).
- MITTAG, H.-J.; RINNE, H.: Prozessfaehigkeitsmessung fuer die industrielle Praxis. Muinch: Hanser, 1999.
- KOTZ, Samuel; LOVELACE, Cynthia R.: Process capability indices in theory and practice. London,New York: Arnold, 1998

See Also

`qqPlot`
`ppPlot`
`http://www.r-qualitytools.org/html/Analyze.html`

Examples

```
x = rweibull(30, 2, 8) +100
#process capability for a weibull distribution
cp(x, "weibull", lsl = 100, usl = 117)

#box cox transformation and one sided
cp(x, boxcox = TRUE, lsl = 1)

#process capability assuming a normal distribution
cp(x, "normal", lsl = 0, usl = 17)

#process capability for a normal distribution and data in subgroups
#some artificial data with shifted means in subgroups
x = c(rnorm(5, mean = 1), rnorm(5, mean = 2), rnorm(5, mean = 0))

#grouping vector
group = c(rep(1,5), rep(2,5), rep(3,5))

#calculate process capability
cp(x, grouping = group) #compare to sd(x)
```

cube-methods *Get and set methods*

Description

Get and set the cube portion for an object of class facDesign

Usage

```
## S4 method for signature 'facDesign'
cube(x)
## S4 replacement method for signature 'facDesign'
cube(x) <- value
```

Arguments

x	a facDesign object
value	data.frame or vector

Methods

signature(objectc = "facDesign") Get and set the cube for the factors in an object of class facDesign. So far used internally.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

facDesign
fracDesign
rsmDesign
star
http://www.r-qualitytools.org

Examples

```
rsdo = rsmDesign(k = 3, alpha = 2)
cube(rsdo)
```

`desirability` *Desirability Function*

Description

Creates desirability functions for use in the optimization of multiple responses.

Usage

```
desirability(response, low, high, target = "max", scale = c(1, 1),
             importance = 1, constraints)
```

Arguments

`response`	name of the response.
`low`	lowest acceptable value for the response.
`high`	highest acceptable value for the response.
`target`	desired target value of the response. `target` can be "max","min" or any specefic numeric value.
`scale`	numeric value giving the scaling factors for one and two sided transformations.
`importance`	a value ranging from 0.1 to 10, used to calculate a weighted importance i.e. with importances 1,2 and 4 D=[(d1)^1,(d2)^2,(d3)^4]^(1/7).
`constraints`	not yet implemented.

Details

For a product to be developed different values of responses are desired leading to multiple response optimization. Minimization, Maximization as well as a specific target value are defined using desirability functions. A desirability function tranforms the values of a response into [0,1] where 0 stands for a non acceptable value of the response and 1 for values where higher/lower (depending on the direction of the optimization) values of the response have little merit.
A first desirability function was specified by Harrington (1965), Derringer and Suich (1980) came up with a modified appproach to transform several responses into a desirability function which was extended with the possibility of specifying weights Derringer (1994). Castillo, Montgomery and McCarville came up with a another modification. The first and the latter are not implemented!

Value

This function returns a desirability function.

Note

For an example in context which shows the usage of the function `desirability()` please read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org/html/Improve.html`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

- HARRINGTON, E.C.:'The Desirability Function', Journal of the American Society for Quality Control, pp. 494-498, 1965.
- DERRINGER, G.; SUICH, R. 'Simulaneous Optimization of Several Response Variables', Journal of Quality Technology, vol. 12, no. 4. 214-219, 1980.
- DERRINGER, G.: 'A Balancing Act: Optimizing a Product's Properties', Quality Progress, vol. 27, no. 6, pp. 51-58, 1994.
- CASTILLO; MONTGOMERY; MCCARVILLE: 'Modified Desirability Functions for Multiple Response Optimization', Journal of Quality Technology, vol. 28, no. 3, pp. 337-345, 1996.

See Also

desires
optimum
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#Maximization of a response
#define a desirability for response y where higher values of y are better
#as long as the response is smaller than high
d = desirability(y, low = 6, high = 18, target = "max")

#show and plot the desirability function
d; plot(d)

#Minimization of a response including a scaling factor
#define a desirability for response y where lower values of y are better
#as long as the response is higher than low
d = desirability(y, low = 6, high = 18, scale = c(2),target = "min")

#show and plot the desirability function
d; plot(d)

#Specific target of a response is best including a scaling factor
#define a desirability for response y where desired value is at 8
#and values lower than 6 as well as values higher than 18 are not acceptable
d = desirability(y, low = 6, high = 18, scale = c(0.5,2),target = 12)

#show and plot the desirability function
d; plot(d)
```

desirability-class *Class "desirability"*

Description

desirability Class

Objects from the Class

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

Slots

`response`: Object of class `"character"` ~~

`low`: Object of class `"numeric"` ~~

`high`: Object of class `"numeric"` ~~

`target`: Object of class `"ANY"` ~~

`scale`: Object of class `"numeric"` ~~

`importance`: Object of class `"numeric"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("desirability")
```

desires-methods *Get and set methods*

Description

Set or get the desirability for each response of a factorial design. The desirability is stored in the facDesign object. Setting desirabilities is required for optimization of multiple responses.

Usage

```
## S4 method for signature 'facDesign'
desires(x)
## S4 replacement method for signature 'facDesign'
desires(x) <- value
```

Arguments

x	an object of class facDesign
value	an object of class desirability; see example

Methods

signature(objectc = "facDesign") Get and set the desires for an object of class facDesign.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

DERRINGER, G.; SUICH, R. 'Simulaneous Optimization of Several Response Variables', Journal of Quality Technology, vol. 12, no. 4. 214-219, 1980.

See Also

desirability
fits
optimum
http://www.r-qualitytools.org

Examples

```
#create a response surface design
fdo = rsmDesign(k = 2, blocks = 2, alpha = "both")

#set two responses for the response surface designs
response(fdo) = data.frame(y= rnorm(14, 12, sd =  2),
                           y2 =  -2*fdo[,4]^2 - fdo[,5]^2 + rnorm(14, 12))

#set a fit for each response
fits(fdo) = lm(y ~ A*B , data = fdo)
fits(fdo) = lm(y2 ~ A*B + I(A^2) + I(B^2), data = fdo)

#define a desirability for response y
d = desirability(y, 6, 18, scale = c(0.5, 2), target = 12)

#plot the desirability function
plot(d)

#set the desirability for y and y2 in the factorial design fdo
desires(fdo) = d
desires(fdo) = desirability(y2, 6, 18, scale = c(1, 1), target = "min")
desires(fdo)
```

desOpt-class *Class "desOpt"*

Description

desOpt Class

Objects from the Class

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

Slots

`facCoded`: Object of class `"list"` ~~

`facReal`: Object of class `"list"` ~~

`responses`: Object of class `"list"` ~~

`desirabilities`: Object of class `"list"` ~~

`overall`: Object of class `"numeric"` ~~

`all`: Object of class `"data.frame"` ~~

`fdo`: Object of class `"facDesign"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("desOpt")
```

distr-class *Class "distr"*

Description

distr Class

Objects from the Class

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

Slots

x: Object of class "vector" ~~

name: Object of class "character" ~~

parameters: Object of class "numeric" ~~

sd: Object of class "numeric" ~~

n: Object of class "numeric" ~~

loglik: Object of class "numeric" ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("distr")
```

---

distrCollection-class *Class "distrCollection"*

Description

distrCollection Class

Objects from the Class

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

Slots

distr: Object of class "list" ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("distrCollection")
```

`distribution` *Distribution*

Description

calculates the most likely parameters for a given distribution.

Usage

```
distribution(x, distribution = "weibull", start, ...)
```

Arguments

`x`	vector of distributed values from which the parameter should be determined.
`distribution`	character string specifying the distribution of `x`. The function `distribution` will accept the following character strings for `distribution`: • "normal" • "chi-squared" • "exponential" • "logistic" • "gamma" • "weibull" • "cauchy" • "beta" • "f" • "t" • "geometric" • "poisson" • "negative binomial" • "log-normal" By default `distribution` is set to "weibull".
`start`	start values if needed.
`...`	...

Value

`distribution()` returns an object of class `distrCollection`.

Note

This is just a wrapper for the fitdistr function of the MASS library!

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

MASS Package.

`doeFactor-class` *Class "doeFactor"*

Description

doeFactor Class

Objects from the Class

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

Slots

`low`: Object of class `"ANY"` ~~

`high`: Object of class `"ANY"` ~~

`name`: Object of class `"character"` ~~

`unit`: Object of class `"character"` ~~

`type`: Object of class `"character"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("doeFactor")
```

`dotPlot` *Function to create a dot plot*

Description

dotPlot creates a dot plot. For data in groups the dotPlot is displayed stacked in one or not-stacked in different plot regions.

Usage

```
dotPlot(x, group, xlim, ylim, col, xlab, ylab, pch, cex, breaks,
        stacked = TRUE, ...)
```

Arguments

x	vector with numeric values which should be plotted.
group	(optional) vector for grouping the values (see examples).
xlim	vector giving the lower and upper limit of the x-axis.
ylim	vector giving the lower and upper limit of the y-axis.
col	vector containing numeric values or strings for different colors for the groups in the dot plot.
xlab	a title for the x axis: title.
ylab	a title for the y axis: title.
pch	a vector of integers specifying symbols or a single character to be used for plotting points for the different groups in the dot plot.
cex	the amount by which points and symbols should be magnified relative to the default.
breaks	a vector giving the breakpoints for the binning of the values in x.
stacked	a logical vector specifying whether the single groups should be plotted in a stacked dot plot or in a single one. By default stacked is set to 'TRUE'.
...	further graphical parameters (see par).

Details

values in x are assigned to the bins defined by breaks. The actual binning is done using hist.

Value

dotPlot a invisible matrix containing NA's and numeric values representing values in a bin. The number of bins is given by the number of columns of the matrix.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

hist
boxplot
http://www.r-qualitytools.org/html/Analyze.html

Examples

```
#create some data and grouping
x = rnorm(28)
g = rep(1:2, 14)

#dot plot with groups and no stacking
dotPlot(x, group = g, stacked = FALSE, pch = c(19, 20),
        main = "Non stacked dot plot")
```

```
#dot plot with groups and stacking
x = rnorm(28)
dotPlot(x, group = g, stacked = TRUE, pch = c(19, 20),
        main = "Stacked dot plot")
```

`effectPlot` *Main Effect Plots*

Description

A main effect plot is created for designs of type taguchiDesign and facDesign.

Usage

```
effectPlot(object, factors, fun = mean, response = NULL, single = FALSE,
           points = FALSE, classic = FALSE, axes = TRUE, lty, xlab, ylab, main,
           ylim, ...)
```

Arguments

`object`	needs to be an object of class `facDesign` or `taguchiDesign`.
`factors`	for which factor is the effectPlot to be created.
`fun`	a function for the construction of the effectPlot such as `mean`, `median`,etc. By default `fun` is set to `mean`.
`response`	response variable. If the response data frame of fdo consists of more then one responses, this variable can be used to choose just one column of the response data frame. `response` needs to be an object of class character with length of '1'. It needs to be the same character as the name of the response in the response data frame that should be plotted. By default `response` is set to 'NULL'.
`single`	logical value. If 'TRUE' device region can be set up using for instance `par(mfrow = c(2,2))`. By default `single` is set to 'FALSE'.
`points`	logical value. If 'TRUE' points are shown in addition to values out of `fun`. By default `points` is set to 'FALSE'.
`axes`	logical value indicating wheter the axes should be drawn or not. 'TRUE' by default.
`classic`	logical value. 'TRUE' creates an effectPlot as depicted in most textbooks. By default `classic` is set to 'FALSE'.
`lty`	numerical value which specifies the line type used.
`xlab`	a title for the x axis: `title`.
`ylab`	a title for the y axis: `title`.

main	an overall title for the plot: see title.
ylim	vector giving the range of the y-axis.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

effectPlot uses an altered version of the base function interaction.plot to draw each effectPlot.

Note

For an example in context which shows the usage of the function effectPlot() to an object of class facDesign or an object of class taguchiDesign, please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

interactionPlot
paretoPlot
snPlot
facDesign
response
normalPlot
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#effectPlot for a 2^k factorial design
fdo = facDesign(k = 3)
#set response with generic response function
response(fdo) = rnorm(8)
effectPlot(fdo)

#effectPlot for a taguchiDesign
tdo = taguchiDesign("L9_3")
response(tdo) = rnorm(9)
effectPlot(tdo, points = TRUE, col = 2, pch = 16, lty = 3)
```

effectPlot-methods *Methods for Function effectPlot*

Description

Methods for function effectPlot

Methods

signature(object = "facDesign") function to create an effect plot for an object of class facDesign.

signature(object = "taguchiDesign") function to create an effect plot for an object of class taguchiDesign.

See Also

http://www.r-qualitytools.org

errorPlot *Function to create error Charts*

Description

The data from an object of class gageRR can be analyzed by running "Error Charts"
of the individual deviations from the accepted rference values. These "Error Charts" are provided by
the function errorPlot.

Usage

```
errorPlot(x, main, xlab, ylab, col, pch, type, ylim, legend=TRUE, ...)
```

Arguments

x	needs to be an object of class gageRR.
main	a main title for the plot.
xlab	a label for the x axis.
ylab	a label for the y axis.
col	plotting color.
pch	an integer specifying a symbol or a single character to be used as the default in plotting points.
type	graphical parameter (see plot).
ylim	the y limits of the plot

legend	a logical value specifying whether a legend is plotted automatically. By default legend is set to 'TRUE'. If the argument legend is set to 'FALSE' an individual legend can be added by using the function legend afterwards.
...	arguments to be passed to methods, such as graphical parameters (see par).

Details

The plotted values are can be calculated in two ways:

- Error = Observed Value - Reference Value (not yet implemented)
- Error = Observed Value - Average Measurement of Part

The first way is not yet implemented, because it is not yet possible to give a refrence value to the object in x.
This will be implemented later! Therefore errorPlot uses the second way above to calculate the plotted error.

Graphical parameters such as col or pch can be given as single characters or as vectors containing characters or number for the parameters of the individual operators.

Note

Please do read the vignette for the package qualityTools at http://www.r-qualitytools.org.

Author(s)

Thomas Roth: <thomas.roth@tu-berlin.de>
Etienne Stockhausen: <stocdarf@mailbox.tu-berlin.de>

References

The idea of the plot and the example given by example(errorPlot) are out of:

- CHRYSLER Group LLC; FORD Motor Company; GENERAL MOTORS Corporation: Measurement System Analysis (MSA), p.112, 4rd ed. Southfield: AIAG, 2010.

See Also

gageRR
par
http://www.r-qualitytools.org

Examples

```
#create gageRR-object
gdo = gageRRDesign(Operators = 3, Parts = 10, Measurements = 3,
                   randomize = FALSE)
#vector of responses
y = c(0.29,0.08, 0.04,-0.56,-0.47,-1.38,1.34,1.19,0.88,0.47,0.01,0.14,-0.80,
      -0.56,-1.46, 0.02,-0.20,-0.29,0.59,0.47,0.02,-0.31,-0.63,-0.46,2.26,
      1.80,1.77,-1.36,-1.68,-1.49,0.41,0.25,-0.11,-0.68,-1.22,-1.13,1.17,0.94,
      1.09,0.50,1.03,0.20,-0.92,-1.20,-1.07,-0.11, 0.22,-0.67,0.75,0.55,0.01,
      -0.20, 0.08,-0.56,1.99,2.12,1.45,-1.25,-1.62,-1.77,0.64,0.07,-0.15,-0.58,
      -0.68,-0.96,1.27,1.34,0.67,0.64,0.20,0.11,-0.84,-1.28,-1.45,-0.21,0.06,
      -0.49,0.66,0.83,0.21,-0.17,-0.34,-0.49,2.01,2.19,1.87,-1.31,-1.50,-2.16)
#appropriate responses
response(gdo)=y
# perform and gageRR
gdo=gageRR(gdo)
errorPlot(gdo)
```

---

errorPlot-methods *Methods for Function errorPlot*

Description

Methods for function `errorPlot`

Methods

`signature(object = "gageRR")` function to create an error plot for an object of class gageRR.

See Also

http://www.r-qualitytools.org

---

facDesign *facDesign*

Description

Generates a 2^k full factorial design.

Usage

```
facDesign(k = 3, p = 0, replicates = 1, blocks = 1, centerCube = 0)
```

Arguments

k	numeric value giving the number of factors. By default k is set to '3'.
p	numeric integer between '0' and '7'. p is giving the number of additional factors in the response surface design by aliasing effects. For further information see fracDesign and fracChoose. By default p is set to '0'.
replicates	numeric value giving the number of replicates per factor combination. By default replicates is set to '1'.
blocks	numeric value giving the number of blocks. By default blocks is set to '1'. Blocking is only performed for k greater 2.
centerCube	numeric value giving the number of centerpoints within the 2^k design. By default centerCube is set to '0'.

Details

facDesign generates 2^k full factorial designs.

Value

The function facDesign returns an object of class facDesign.

Note

For an example in context which shows the usage of the function facDesign please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

fracDesign
fracChoose
rsmDesign
pbDesign
taguchiDesign
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#returns a 2^3 full factorial design
vp.full = facDesign(k = 3)
#generate some random response
response(vp.full) = rnorm(2^3)
#summary of the full factorial design (especially no defining relation)
summary(vp.full)

#------------
```

```
#returns a full factorial design with 3 replications per factor combination
#and 4 center points
vp.rep = facDesign(k = 2, replicates = 3, centerCube = 4)
#set names
names(vp.rep) = c("Name 1", "Name 2")
#set units
units(vp.rep) = c("min", "F")
#set low and high factor values
lows(vp.rep) = c(20, 40, 60)
highs(vp.rep) = c(40, 60, 80)
#summary of the replicated full factorial Design
summary(vp.rep)
```

---

facDesign-class *Class "facDesign"*

Description

facDesign Class

Objects from the Class

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

Slots

name: Object of class `"character"` ~~

factors: Object of class `"list"` ~~

cube: Object of class `"data.frame"` ~~

star: Object of class `"data.frame"` ~~

centerCube: Object of class `"data.frame"` ~~

centerStar: Object of class `"data.frame"` ~~

generator: Object of class `"ANY"` ~~

response: Object of class `"data.frame"` ~~

block: Object of class `"data.frame"` ~~

blockGen: Object of class `"data.frame"` ~~

runOrder: Object of class `"data.frame"` ~~

standardOrder: Object of class `"data.frame"` ~~

desireVal: Object of class `"list"` ~~

desirability: Object of class `"list"` ~~

fits: Object of class `"list"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("facDesign")
```

---

factors-methods *Get and set methods*

Description

Get and set the `factors` for the factors in an object of class `facDesign`, `mixDesign`, `taguchiDesign` or `pbDesign`

Usage

```
## S4 method for signature 'facDesign'
factors(x)
## S4 replacement method for signature 'facDesign'
factors(x) <- value
## S4 method for signature 'mixDesign'
factors(x)
## S4 replacement method for signature 'mixDesign'
factors(x) <- value
## S4 method for signature 'taguchiDesign'
factors(x)
## S4 replacement method for signature 'taguchiDesign'
factors(x) <- value
## S4 method for signature 'pbDesign'
factors(x)
## S4 replacement method for signature 'pbDesign'
factors(x) <- value
```

Arguments

x	a facDesign, taguchiDesign, pbDesign or mixDesign object.
value	new factors (used internally).

Methods

`signature(object = "facDesign")` Get and set the `factors` in an object of class facDesign.

`signature(object = "mixDesign")` Get and set the `factors` in an object of class mixDesign.

`signature(object = "taguchiDesign")` Get and set the `factors` in an object of class taguchiDesign.

`signature(object = "pbDesign")` Get and set the `factors` in an object of class taguchiDesign.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

See Also

factors
factors
factors
types
response
facDesign
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#NA in response column
fdo = facDesign(k = 3)
factors(fdo)
```

---

fits-methods *Get and set methods*

Description

Set or get the fits (i.e. linear model) for each response of a factorial design. The fit is stored in the facDesign object. Setting fits is required for optimization of multiple responses.

Usage

```
## S4 method for signature 'facDesign'
fits(x)
## S4 replacement method for signature 'facDesign'
fits(x) <- value
```

Arguments

x	an object of class facDesign.
value	an object of class lm.

Methods

signature(objectc = "facDesign") Get and set the fits for an object of class facDesign.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

desires
desirability
rsmDesign
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#create response surface design
fdo = rsmDesign(k = 2, blocks = 2, alpha = "both")

#set two responses for the response surface designs
response(fdo) = data.frame(y= rnorm(14, 12, sd =  2),
                           y2 =  -2*fdo[,4]^2 - fdo[,5]^2 + rnorm(14, 12))

#set a fit for each response
fits(fdo) = lm(y ~ A*B , data = fdo)
fits(fdo) = lm(y2 ~ A*B + I(A^2) + I(B^2), data = fdo)

#get the fitted response for y2
fits(fdo)[["y2"]]
```

---

fracChoose *Choosing a fractional or full factorial design from a table.*

Description

Designs displayed are the classic minimum abberation designs. Choosing a design is done by clicking with the mouse into the appropriate field.

Usage

```
fracChoose()
```

Value

`fracChoose()` returns an object of class facDesign.

Note

For an example in context which shows the usage of the function `fracChoose()` please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

- BOX, George. E. P. and HUNTER, J.S.: "Multifactor Experimental Designs for Exploring Response Surfaces",
The Annals of Mathematical Statistics 28, p.195- 241, 1957

- BOX, George E. P. and HUNTER, J. Stuart and HUNTER, William G.
Statistics for Experimenters (2005) 28, p.272, table 6.22

See Also

facDesign
fracDesign
rsmChoose
rsmDesign
http://www.r-qualitytools.org/html/Improve.html

`fracDesign` *fracDesign*

Description

Generates a 2^k full- or fractional factorial design.

Usage

```
fracDesign(k = 3, p = 0, gen = NULL, replicates = 1, blocks = 1, centerCube = 0,
           random.seed = 1234)
```

Arguments

k	numeric value giving the number of factors. By default k is set to '3'.
p	numeric integer between '0' and '7'. p is giving the number of additional factors in the response surface design by aliasing effects. A 2^k-p factorial design will be generated and the generators of the standard designs available in `fracChoose()` will be used. By default p is set to '0'. Any other value will cause the function to omit the argument gen given by the user and replace it by the one out of the table of standard designs (see: `fracChoose()`). Replicates and blocks can be set anyway!
gen	one or more defining relations for a fractional factorial design. By default gen is set to 'NULL'.
replicates	numeric value giving the number of replicates per factor combination. By default `replicates` is set to '1'.
blocks	numeric value giving the number of blocks. By default `blocks` is set to '1'.

centerCube	numeric value giving the number of centerpoints within the 2^k design. By default centerCube is set to '0'.
random.seed	seed for randomization of the design

Details

fracDesign generates 2^k full- or fractional factorial designs.

Value

fracDesign returns an object of class facDesign.

Note

For an example in context which shows the usage of the function fracDesign() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

facDesign
fracChoose
pbDesign
rsmDesign
taguchiDesign
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#returns a 2^3 full factorial design
vp.full = facDesign(k = 3)
#design in 2 blocks
vp.full = blocking(vp.full, 2)
#generate some random response
response(vp.full) = rnorm(2^3)
#summary of the full factorial design (especially no defining relation)
summary(vp.full)

#returns a 2^4-1 fractional factorial design. Factor D will be aliased with
vp.frac = fracDesign(k = 4, gen = "D=ABC")
#the three-way-interaction ABC (i.e. I = ABCD)
response(vp.frac) = rnorm(2^(4-1))
#summary of the fractional factorial design
summary(vp.frac)

#returns a full factorial design with 3 replications per factor combination
#and 4 center points
```

```
vp.rep = fracDesign(k = 3, replicates = 3, centerCube = 4)
#summary of the replicated fractional factorial Design
summary(vp.rep)
```

---

gageLin *Function to visualize and calucalte the linearity of a gage.*

Description

Function visualize the linearity of a gage by plotting the single and mean bias in one plot and intercalate them with a straight line.
Furthermore the function deliver some characteristic values of linearity studies according to MSA (Measurement System Analysis).

Usage

```
gageLin(object, conf.level = 0.95, ylim, col, pch, lty = c(1, 2),
        stats = TRUE, plot = TRUE)
```

Arguments

object	an object of class MSALinearity. To create such an object see gageLinDesign.
conf.level	an numeric value between '0' and '1', giving the confidence intervall for the analysis. Default value: '0.95'
ylim	a vector with two entries, giving the minimum and the maximum of the y-axis.
col	a vector with four numeric entries. The first gives the color of the single points, the second gives the color of the points for the mean bias, the third gives the color fo the straight interpolation line and the fourth gives the color for the lines representing the confidence interval. If one of the values is missing or negative the points or lines are not plotted. col is by default 'c(1,2,1,4)'
pch	a vector with two numeric or single character entries giving the symbols for the single points (1st entry) and the mean bias (2nd entry). The default vector is "c(20,18)"
lty	a vector with two entries giving the line-style for the interpolating line and the confidence interval lines. For detailed information to the entries please see par. The default value for lty is 'c(1,2)'.
stats	a logigal value. If 'TRUE' (default) the function returns all calculated information.
plot	a logigal value. If 'TRUE' (default) the function deliver a plot.

Value

The function returns an object of class MSALinearity which can be used with e.g. plot or summary.

Author(s)

Thomas Roth: thomas.roth@tu-berlin.de
Etienne Stockhausen: stocdarf@mailbox.tu-berlin.de

References

- Cgrysler Group LCC & Ford Motor Company & General Motors Corporation, Measurement System Analysis - MSA (2010), 4th ed. Southfield,Michigan: Automotive Industry Action Group.

See Also

cg
gageRR
gageLinDesign
response
edit
http://www.r-qualitytools.org

Examples

```
# Results of single runs
A=c(2.7,2.5,2.4,2.5,2.7,2.3,2.5,2.5,2.4,2.4,2.6,2.4)
B=c(5.1,3.9,4.2,5,3.8,3.9,3.9,3.9,3.9,4,4.1,3.8)
C=c(5.8,5.7,5.9,5.9,6,6.1,6,6.1,6.4,6.3,6,6.1)
D=c(7.6,7.7,7.8,7.7,7.8,7.8,7.8,7.7,7.8,7.5,7.6,7.7)
E=c(9.1,9.3,9.5,9.3,9.4,9.5,9.5,9.5,9.6,9.2,9.3,9.4)

# creates Desing
test=gageLinDesign(ref=c(2,4,6,8,10),n=12)
 # creates data.frame with results
Messungen=data.frame(rbind(A,B,C,D,E))
# enter results in Desing
response(test)=Messungen

# no plot and no return
MSALin=gageLin(test,stats=FALSE,plot=FALSE)
# plot only
plot(MSALin)
# summary
summary(MSALin)
```

---

gageLinDesign *Function to create a object of class MSALinearity.*

Description

Function generates an object that can be used with the function gageLin.

Usage

```
gageLinDesign(ref, n = 5)
```

Arguments

ref	is a vector and contains the reference values for each group.
n	is a single value and gives the amount of runs.

Value

The Function returns an object of the class MSALinearity.

Note

The results for the single should be entered with the help of response!

Author(s)

Thomas Roth: thomas.roth@tu-berlin.de
Etienne Stockhausen: stocdarf@mailbox.tu-berlin.de

See Also

gageLin
response
edit
http://www.r-qualitytools.org

Examples

```
# results of run A-E
A=c(2.7,2.5,2.4,2.5,2.7,2.3,2.5,2.5,2.4,2.4,2.6,2.4)
B=c(5.1,3.9,4.2,5,3.8,3.9,3.9,3.9,3.9,4,4.1,3.8)
C=c(5.8,5.7,5.9,5.9,6,6.1,6,6.1,6.4,6.3,6,6.1)
D=c(7.6,7.7,7.8,7.7,7.8,7.8,7.8,7.7,7.8,7.5,7.6,7.7)
E=c(9.1,9.3,9.5,9.3,9.4,9.5,9.5,9.5,9.6,9.2,9.3,9.4)

# create Design
temp=gageLinDesign(ref=c(2,4,6,8,10),n=12)
# create data.frame for results
results=data.frame(rbind(A,B,C,D,E))
# enter results in Design
response(temp)=results
temp
```

gageRR *Gage R&R - Gage Repeatability and Reproducibility*

Description

Performs a Gage R&R analysis for an object of class gageRR.

Usage

```
gageRR(gdo, method = "crossed", sigma = 6, alpha = 0.25, DM = NULL,
       HM = NULL, tolerance = NULL, dig = 3, ...)
```

Arguments

gdo	needs to be an object of class gageRR.
method	"crossed" which is the typical design for performing a Measurement Systems Analysis using Gage Repeatability and Reproducibility or "nested" which is used for destructive testing (i.e. the same part cannot be measured twice). Operators measure each a different sample of parts under the premise that the parts of each batch are alike. By default method is set to "crossed".
sigma	numeric value giving the number of sigmas. For sigma=6 this relates to 99.73 percent representing the full spread of a normal distribution function (i.e. pnorm(3) - pnorm(-3)). Another popular setting sigma=5.15 relates to 99 percent (i.e. pnorm(2.575) - pnorm(-2.575)). By default sigma is set to '6'.
alpha	alpha value for discarding the interaction Operator:Part and fitting a non-interaction model. By default alpha is set to '0.25'.
DM	By default DM is set to 'NULL'.
HM	By default HM is set to 'NULL'.
tolerance	numeric value giving the tolerance for the measured parts. This is required to calculate the Process to Tolerance Ratio. By default tolerance is set to 'NULL'.
dig	numeric value giving the number of significant digits for format. By default dig is set to '3'.
...	further graphical parameters(see par

Value

gageRR() returns an object of class gageRR and shows typical Gage Repeatability and Reproducibility Output including Process to Tolerance Ratios and the number of distinctive categories (i.e. ndc) the measurement system is able to discriminate with the tested setting.

Note

For an example in context which shows the usage of the function gageRR() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Measure.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de >

See Also

gageLin
cg
gageRRDesign
response
cg
http://www.r-qualitytools.org/html/Measure.html

Examples

```
#create a crossed Gage R&R Design
gdo = gageRRDesign(3,10, 2, randomize = FALSE)

#set the response i.e. Measurements
y = c(23,22,22,22,22,25,23,22,23,22,20,22,22,22,24,25,27,28,23,24,23,24,24,22,
      22,22,24,23,22,24,20,20,25,24,22,24,21,20,21,22,21,22,21,21,24,27,25,27,
      23,22,25,23,23,22,22,23,25,21,24,23)
response(gdo) = y

#perform a Gage R&R
gdo = gageRR(gdo, tolerance = 5)

#summary
summary(gdo)

#standard graphics for Gage R&R
plot(gdo)


##create a crossed Gage R&R Design -
##Vardeman, VanValkenburg 1999 - Two-Way Random-Effects Analyses and Gauge
#gdo = gageRRDesign(Operators = 5, Parts = 2, Measurements = 3, randomize = FALSE)
#
##Measurements
#weight = c(3.481, 3.448, 3.485, 3.475, 3.472,
#           3.258, 3.254, 3.256, 3.249, 3.241,
#           3.477, 3.472, 3.464, 3.472, 3.470,
#           3.254, 3.247, 3.257, 3.238, 3.250,
#           3.470, 3.470, 3.477, 3.473, 3.474,
#           3.258, 3.239, 3.245, 3.240, 3.254)
#
##set the response i.e. Measurements
```

```
#response(gdo) = weight
#
##perform a Gage R&R
#gdo = gageRR(gdo)
#
##summary
#summary(gdo)
#
##standard graphics for Gage R&R
#plot(gdo)
#
```

---

gageRR-class *Class "gageRR"*

Description

gageRR Class

Objects from the Class

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

Slots

X: Object of class `"data.frame"` ~~

ANOVA: Object of class `"aov"` ~~

RedANOVA: Object of class `"aov"` ~~

method: Object of class `"character"` ~~

Estimates: Object of class `"list"` ~~

Varcomp: Object of class `"list"` ~~

Sigma: Object of class `"numeric"` ~~

GageName: Object of class `"character"` ~~

GageTolerance: Object of class `"character"` ~~

DateOfStudy: Object of class `"character"` ~~

PersonResponsible: Object of class `"character"` ~~

Comments: Object of class `"character"` ~~

b: Object of class `"factor"` ~~

a: Object of class `"factor"` ~~

numM: Object of class `"numeric"` ~~

numO: Object of class `"numeric"` ~~

numP: Object of class `"numeric"` ~~

y: Object of class `"numeric"` ~~

facNames: Object of class `"character"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("gageRR")
```

---

gageRRDesign *Gage R&R - Gage Repeatability and Reproducibility*

Description

Creates a Gage R&R design.

Usage

```
gageRRDesign(Operators = 3, Parts = 10, Measurements = 3, method = "crossed",
             sigma = 6, randomize = TRUE)
```

Arguments

`Operators`	numeric value giving a number or a character vector defining the Operators. By default `Operators` is set to '3'.
`Parts`	a number or character vector defining the Parts. By default `parts` is set to '10'.
`Measurements`	a number defining the measurements per part. By default `Measurments` is set to '3'.
`method`	"crossed" which is the typical design for performing a Measurement Systems Analysis using Gage Repeatability and Reproducibility or "nested" which is used for destructive testing (i.e. the same part cannot be measured twice). Operators measure each a different sample of parts under the premise that the parts of each batch are alike. By default `method` is set to "crossed".
`sigma`	For sigma=6 this relates to 99.73 percent representing the full spread of a normal distribution function (i.e. pnorm(3) - pnorm(-3)). Another popular setting sigma=5.15 relates to 99 percent (i.e. `pnorm(2.575) - pnorm(-2.575)`). By default `sigma` is set to '6'.
`randomize`	logicalvalue. 'TRUE' (default) randomizes the gageRR design.

Value

`gagRRDeign` returns an object of class `gageRR`.

Note

For an example in context which shows the usage of the function gageRRDesign() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Measure.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

gageRR
http://www.r-qualitytools.org/html/Measure.html

Examples

```
#create a Gage R&R Design
temp = gageRRDesign(3,10, 2, randomize = FALSE)

#set the response i.e. Measurements
y = c(23,22,22,22,22,25,23,22,23,22,20,22,22,22,24,25,27,28,23,24,23,24,24,22,
      22,22,24,23,22,24,20,20,25,24,22,24,21,20,21,22,21,22,21,21,24,27,25,27,
      23,22,25,23,23,22,22,23,25,21,24,23)
response(temp) = y

#perform a Gage R&R
gdo = gageRR(temp)

#summary
summary(gdo)

#standard graphics for Gage R&R
plot(gdo)
```

---

gamma3 *The gamma Distribution (3 Parameter)*

Description

Density function, distribution function and quantile function for the gamma distribution.

Usage

```
dgamma3(x, shape, scale, threshold)
pgamma3(q, shape, scale, threshold)
qgamma3(p, shape, scale, threshold, ...)
```

Arguments

x, q	vector of quantiles
p	vector of probabilities
shape	shape parameter by default 1
scale	scale parameter by default 1
threshold	threshold parameter by default 0
...	Arguments that can be passed into uniroot.

Details

The gamma distribution with 'scale' parameter alpha, 'shape' parameter c and 'threshold' parameter zeta has density given by

f(x) = (c/alpha) (((x-zeta)/alpha)^(c-1)) exp(-((x-zeta)/alpha)^c)

The cumulative distribution function is given by

F(x) = 1 - exp(-((x-zeta)/alpha)^c)

Value

dgamma3 gives the density, pgamma3 gives the distribution function and qgamma3 gives the quantile function.

Note

qgamma3 calls uniroot for each value of the argument 'p'. The solution is consequently not exact; the ... can be used to obtain a more accurate solution if necessary.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

References

Johnson, L., Kotz, S., Balakrishnan, N. (1995) Continuous Univariate Distributions-Volume 1, 2nd ed. New York: John Wiley & Sons.

See Also

uniroot

Examples

```
##Simple Example
dgamma3(x=1,scale=1,shape=5,threshold=0)
temp=pgamma3(q=1,scale=1,shape=5,threshold=0)
temp
qgamma3(p=temp,scale=1,shape=5,threshold=0)
```

```
#
##Visualized Example
##prepare screen
#dev.new()
#split.screen(matrix(c(0,0.5,0,1, 0.5,1,0,1),byrow=TRUE,ncol=4))
##generate values
#x=seq(0,3,length=1000)
##plot different density functions
#screen(1)
#plot(x,y=dgamma3(x,threshold=0,shape=0.5,scale=1),col="green",
#     xlim=c(0,2.5),ylim=c(0,2.5),type="l",lwd=2,xlab="x",
#     ylab="f(x)",main="Density Function of gamma-Distribution")
#lines(x,y=dgamma3(x,threshold=0,shape=1,scale=1),lwd=2,col="red")
#lines(x,y=dgamma3(x,threshold=0,shape=1.5,scale=2),lwd=2,col="blue")
#lines(x,y=dgamma3(x,threshold=0,shape=5,scale=1),lwd=2,col="orange")
##add legend
#legend("topright",legend=c(expression(paste(alpha, " = 1 ")*
#       paste(c, " = 0.5 ")*paste(zeta," = 0")),
#       expression(paste(alpha, " = 1 ")*paste(c, " = 1 ")*
#       paste(zeta," = 0")),expression(paste(alpha, " = 2 ")*
#       paste(c, " = 1.5 ")*paste(zeta," = 0")),
#       expression(paste(alpha, " = 1 ")*paste(c, " = 5 ")*
#       paste(zeta," = 0"))),col=c("green","red","blue","orange"),
#       text.col="black",lwd=2,bty="0",inset=0.04)
#abline(v=0,lty=2,col="grey")
#abline(h=0,lty=2,col="grey")
##plot different distribution functions
#screen(2)
#plot(x,y=pgamma3(x,threshold=0,shape=0.5,scale=1),col="green",
#     xlim=c(0,2.5),ylim=c(0,1),type="l",lwd=2,xlab="x",ylab="F(x)",
#     main="Cumulative Distribution Function of gamma-Distribution")
#lines(x,y=pgamma3(x,threshold=0,shape=1,scale=1),lwd=2,col="red")
#lines(x,y=pgamma3(x,threshold=0,shape=1.5,scale=2),lwd=2,col="blue")
#lines(x,y=pgamma3(x,threshold=0,shape=5,scale=1),lwd=2,col="orange")
##add legend
#legend("bottomright",legend=c(expression(paste(alpha, " = 1 ")*
#       paste(c, " = 0.5 ")*paste(zeta," = 0")),
#       expression(paste(alpha, " = 1 ")*paste(c, " = 1 ")*
#       paste(zeta," = 0")),expression(paste(alpha, " = 2 ")*
#       paste(c, " = 1.5 ")*paste(zeta," = 0")),
#       expression(paste(alpha, " = 1 ")*paste(c, " = 5 ")*
#       paste(zeta," = 0"))),col=c("green","red","blue","orange"),
#       text.col="black",lwd=2,bty="0",inset=0.04)
#abline(v=0,lty=2,col="grey")
#abline(h=0,lty=2,col="grey")
#close.screen(all=TRUE)
```

---

highs-methods *Get and set methods*

Description

Get and set the highs for the factors in an object of class facDesign and mixDesign.

Usage

```
## S4 method for signature 'facDesign'
highs(object)
## S4 replacement method for signature 'facDesign'
highs(object) <- value
## S4 method for signature 'mixDesign'
highs(object)
## S4 replacement method for signature 'mixDesign'
highs(object) <- value
```

Arguments

object	a facDesign or mixDesign object
value	data.frame or vector

Methods

signature(objectc = "facDesign") Get and set the highs for the factors in an object of class facDesign.

signature(object = "mixDesign") Get and set the highs for the factors in an object of class mixDesign.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

factors
lows
highs
types
http://www.r-qualitytools.org

Examples

```
fdo = facDesign(k=3)
lows(fdo) = c(10, 160, 1)
highs(fdo) = c(20, 200, 2)
summary(fdo)
```

identity-methods *Get method*

Description

Calculates the alias table for a fractional factorial design and prints an easy to read summary of the defining relations such as 'I = ABCD' for a standard 2^(4-1) factorial design.

Usage

```
## S4 method for signature 'facDesign'
identity(x)
```

Arguments

x	a facDesign or taguchiDesign object

Value

identity in character representation
columns

Methods

`signature(x = "facDesign")` Calculates the alias table for a fractional factorial design and prints an easy to read summary of the defining relations such as 'I = ABCD' for a standard 2^(4-1) factorial design.

`signature(x = "taguchiDesign")` Calculates the alias table for a fractional taguchi design and prints an easy to read summary of the defining relations such as 'I = ABC' for a standard "L4_2" taguchi design.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

aliasTable
fracDesign
fracChoose
http://www.r-qualitytools.org

Examples

```
#generate a 2^(4-1) factorial design by assigning the interaction ABC
#to the factor D
vp = fracDesign(k = 4, gen = "D = ABC")

#the defining relation is (D = ABC)*D = I = ABCD. I is the identity.
identity(vp)

#the identity can be seen in the according alias table
aliasTable(vp)
```

`interactionPlot` *interactionPlot*

Description

Display effects of an object of class `facDesign` in a line plot.

Usage

```
interactionPlot(fdo, y = NULL, response = NULL, fun = mean, main, col = 1:2,
                ...)
```

Arguments

`fdo`	needs to be an object of class `facDesign`.
`fun`	function to use for the calculation of the interactions, as `mean`,`median`,etc.
`col`	vector of colors for the plot. Single colors can be given as character strings or numeric values.
`y`	if fdo is a vector, `interactionPlot()` defaults to `interaction.plot` from package `stats` if y is a vector too.
`response`	response variable. If the response data frame of fdo consists of more then one responses, this variable can be used to choose just one column of the response data frame. `response` needs to be an object of class character with length of '1'. It needs to be the same character as the name of the response in the response data frame that should be plotted.
`main`	an overall title for the plot: see `title`.
`...`	further graphical parameters: see `par`.

Details

`interactionPlot()` displays interactions for an object of class `facDesign` (i.e. 2^k full or 2^k-p fractional factorial design).
Parts of the original interactionPlot were integrated.

Value

none

Note

For an example in context which shows the usage of the function `interactionPlot()` to an object of class `facDesign`, please read the vignette for the package `qualityTools` at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

`factors`
`fracDesign`
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#NA in response column and 2 replicates per factor combination
vp = fracDesign(k = 3, replicates = 2)
#generate some data
y = 4*vp[,1] -7*vp[,2] + 2*vp[,2]*vp[,1] + 0.2*vp[,3] + rnorm(16)
response(vp) = y
#show effects and interactions (nothing significant expected)
interactionPlot(vp)
```

---

lnorm3 *The Lognormal Distribution (3 Parameter)*

Description

Density function, distribution function and quantile function for the Lognormal distribution.

Usage

```
dlnorm3(x, meanlog, sdlog, threshold)
plnorm3(q, meanlog, sdlog, threshold)
qlnorm3(p, meanlog, sdlog, threshold, ...)
```

Arguments

x, q	vector of quantiles
p	vector of probabilities
meanlog, sdlog	mean and standard deviation of the distribution on the log scale with default values of '0' and '1' respectively.
threshold	threshold parameter by default 0
...	Arguments that can be passed into uniroot.

Details

The Lognorm distribution with 'meanlog' parameter zeta, 'meansd' parameter sigma and 'threshold' parameter theta has density given by

f(x) = (1/(sqrt(2*pi)*sigma*(x-theta)))
exp(-(((log((x-theta))-zeta)^2)/(2(sigma)^2)))

The cumulative distribution function is given by

F(x) = pnorm((log((x-theta))-zeta)/sigma)

Value

dlnorm3 gives the density, plnorm3 gives the distribution function and qlnorm3 gives the quantile function.

Note

qlnorm3 calls uniroot for each value of the argument 'p'. The solution is consequently not exact; the ... can be used to obtain a more accurate solution if necessary.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

References

Johnson, L., Kotz, S., Balakrishnan, N. (1995) Continuous Univariate Distributions-Volume 1, 2nd ed. New York: John Wiley & Sons.

See Also

uniroot

Examples

```
#Simple Example
dlnorm3(x=2,meanlog=0,sdlog=1/8,threshold=1)
temp=plnorm3(q=2,meanlog=0,sdlog=1/8,threshold=1)
temp
qlnorm3(p=temp,meanlog=0,sdlog=1/8,threshold=1)
#
##Visualized Example
##prepare screen
#dev.new()
#split.screen(matrix(c(0,0.5,0,1, 0.5,1,0,1),byrow=TRUE,ncol=4))
##generate values
#x=seq(0,4,length=1000)
##plot different density functions
#screen(1)
#plot(x,y=dlnorm3(x,0,1/8,1),col="green",xlim=c(0,3),type="l",lwd=2,xlab="x",
#     ylab="f(x)",main="Density Function of Log-Normal-Distribution")
#lines(x,y=dlnorm3(x,0,0.5,0),lwd=2,col="red")
#lines(x,y=dlnorm3(x,0,1,0),lwd=2,col="blue")
#lines(x,y=dlnorm3(x,1,1/8,0),lwd=2,col="orange")
##add legend
#legend("topleft",legend=c(expression(paste(zeta," = 0 ")*
#       paste(sigma, " = 1/8 ")*paste(theta," = 1")),
#       expression(paste(zeta," = 0 ")*paste(sigma, " = 0.5 ")*
#       paste(theta," = 0")),expression(paste(zeta," = 0 ")*
#       paste(sigma, " = 1 ")*paste(theta," = 0")),
#       expression(paste(zeta," = 1 ")*paste(sigma, " = 1/8 ")*
#       paste(theta," = 0"))),col=c("green","red","blue","orange"),
#       text.col="black",lwd=2,bty="o",inset=0.04)
#abline(v=0,lty=2,col="grey")
#abline(h=0,lty=2,col="grey")
##plot different distribution functions
#screen(2)
#plot(x,y=plnorm3(x,0,1/8,1),col="green",xlim=c(0,3),type="l",lwd=2,xlab="x",
#     ylab="F(x)",
#     main="Cumulative Distribution Function of Log-Normal-Distribution")
#lines(x,y=plnorm3(x,0,0.5,0),lwd=2,col="red")
#lines(x,y=plnorm3(x,0,1,0),lwd=2,col="blue")
#lines(x,y=plnorm3(x,1,1/8,0),lwd=2,col="orange")
##add legend
#legend("topleft",legend=c(expression(paste(zeta," = 0 ")*
#       paste(sigma, " = 1/8 ")*paste(theta," = 1")),
#       expression(paste(zeta," = 0 ")*paste(sigma, " = 0.5 ")*
#       paste(theta," = 0")),expression(paste(zeta," = 0 ")*
#       paste(sigma, " = 1 ")*paste(theta," = 0")),
#       expression(paste(zeta," = 1 ")*paste(sigma, " = 1/8 ")*
#       paste(theta," = 0"))),col=c("green","red","blue","orange"),
#       text.col="black",lwd=2,bty="o",inset=0.04)
#abline(v=0,lty=2,col="grey")
#abline(h=0,lty=2,col="grey")
#close.screen(all=TRUE)
```

lows-methods *Get and set methods*

Description

Get and set the lows for the factors in an object of class facDesign.

Usage

```
## S4 method for signature 'facDesign'
lows(object)
## S4 replacement method for signature 'facDesign'
lows(object) <- value
## S4 method for signature 'mixDesign'
lows(object)
## S4 replacement method for signature 'mixDesign'
lows(object) <- value
```

Arguments

object	a facDesign or mixDesign object.
value	data.frame or vector.

Methods

signature(object = "facDesign") Get and set the lows for the factors in an object of class facDesign.

signature(object = "mixDesign") Get and set the lows for the factors in an object of class mixDesign.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

factors
lows
lows
types
http://www.r-qualitytools.org

Examples

```
fdo = facDesign(k=3)
lows(fdo) = c(10, 160, 1)
lows(fdo) = c(20, 200, 2)
summary(fdo)
```

`mixDesign` *Mixture Designs*

Description

function to generate simplex lattice and simplex centroid mixture designs with optional center points and axial points.

Usage

```
mixDesign(p, n = 3, type = "lattice", center = TRUE, axial = FALSE, delta,
          replicates = 1, lower, total = 1, randomize, seed)
```

Arguments

`p`	numerical value giving the amount of factors.
`n`	numerical value specifying the degree (ignored if type = "centroid").
`type`	character string giving the type of design. `type` can be "lattice" or "centroid" (referencing to the first source under the section references]. By default `type` is set to "lattice".
`center`	logical value specifying whether (optional) center points will be added. By default `center` is set to 'TRUE'.
`axial`	logical value specifying whether (optional) axial points will be added. By default `axial` is set to 'FALSE'.
`delta`	numerical value giving the delta (see references) for axial runs. No default setting.
`replicates`	vector with the number of replicates for the different design points i.e. c(center = 1, axial = 1, pureBlend = 1, BinaryBlend = 1, p-3 blend, p-2 blend, p-1 blend). By default `replicates` is set to '1'.
`lower`	vector of lower-bound constraints on the component proportions (i.e. must be given in percent).
`total`	vector with 1 the percentage of the mixture made up by the q - components (e.g. q = 3 and x1 + x2 + x3 = 0.8 –> total = 0.8 with 0.2 for the other factors being held constant) 2 overall total in corresponding units (e.g. 200ml for the overall mixture)
`randomize`	logical value. If 'TRUE' the RunOrder of the mixture design will be randomized (default).
`seed`	numerical value giving the input for `set.seed`.

Value

`mixDesig()` returns an object of class `mixDesign`.

Note

In this version the creation of (augmented) lattice, centroid mixture designs is fully supported. Getters and Setter methods for the mixDesign object exist just as for objects of class `facDesign` (i.e. factorial designs).
The creation of constrained component proportions is partially supported but don't rely on it. Visualization (i.e. ternary plots) for some of these designs can be done with the help of the `wirePlot3` and `contourPlot3` function.
Visualization is however BETA and the analysis of mixture designs is not as convenient as for objects of class `facDesign` (i.e. needs to be done manually, at this point). Feedback is welcome...

For an example in context which shows the usage of the function `mixDesign()` to an object of class `facDesign`, please read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org/html/Improve.html`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

- CORNELL: Experiments with Mixtures - 3rd Ed. New Jersey: Wiley, 2011.
- CHASALOW, S. ; BRAND, R.: Generation of Simplex Lattice Points. Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 44, No. 4 (1995), pp. 534-545.

See Also

`facDesign`
`fracDesign`
`rsmDesign`
`wirePlot3`
`contourPlot3`
`http://www.r-qualitytools.org/html/Improve.html`

Examples

```
#simplex lattice design with center (default response added with NA's)
mixDesign(p = 3, n = 2, center = FALSE)

#simplex lattice design with a center (default response added with NA's)
mixDesign(p = 3, n = 2, center = TRUE)

#simplex lattice design with augmented points (default response added with NA's)
mixDesign(p = 3, n = 2, center = FALSE, axial = TRUE)

#simplex lattice design with augmented points, center and 2 replications
```

```
#(default response added with NA's)
mixDesign(p = 3, n = 2, center = TRUE, axial = TRUE, replicates = 2)

#simplex centroid design giving 2^(p-1) distinct design points
#(default response added with NA's)
mixDesign(p = 3, n = 2, type = "centroid")

#simplex centroid design with augmented points (default response added with NA's)
mixDesign(p = 3, n = 2, type = "centroid", axial = TRUE)



#yarn elongation example from Cornell (2002)
mdo = mixDesign(3,2, center = FALSE, axial = FALSE, randomize = FALSE,
                replicates  = c(1,1,2,3))
#set names (optional)
names(mdo) = c("polyethylene", "polystyrene", "polypropylene")
#units(mdo) = "%" #set units (optional)
#set response (i.e. yarn elongation)
elongation = c(11.0, 12.4, 15.0, 14.8, 16.1, 17.7, 16.4, 16.6, 8.8, 10.0, 10.0,
               9.7, 11.8, 16.8, 16.0)
response(mdo) = elongation

#print a summary of the design
summary(mdo)

#show contourPlot and wirePlot
dev.new(14, 14);par(mfrow = c(2,2))
contourPlot3(A, B, C, elongation, data = mdo, form = "quadratic")
wirePlot3(A, B, C, elongation, data = mdo, form = "quadratic", theta = 190)
wirePlot3(A, B, C, elongation, data = mdo, form = "quadratic", phi = 390,
          theta = 0)
wirePlot3(A, B, C, elongation, data = mdo, form = "quadratic", phi = 90)
```

`mixDesign-class` *Class "mixDesign"*

Description

mixDesign class for simplex lattice and simplex centroid mixture designs with optional center points and augmented points.

Objects from the Class

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

Slots

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

`factors`**:** Object of class `"list"` ~~

`total`: Object of class "numeric" ~~

`lower`: Object of class "numeric" ~~

`design`: Object of class "data.frame" ~~

`designType`: Object of class "character" ~~

`pseudo`: Object of class "data.frame" ~~

`response`: Object of class "data.frame" ~~

`Type`: Object of class "data.frame" ~~

`block`: Object of class "data.frame" ~~

`runOrder`: Object of class "data.frame" ~~

`standardOrder`: Object of class "data.frame" ~~

`desireVal`: Object of class "list" ~~

`desirability`: Object of class "list" ~~

`fits`: Object of class "data.frame" ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

facDesign

Examples

```
showClass("mixDesign")
```

MSALinearity-class *Class* "MSALinearity"

Description

MSALinearity Class

Objects from the Class

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

Slots

X: Object of class "data.frame" ~~

Y: Object of class "data.frame" ~~

model: Object of class "lm" ~~

conf.level: Object of class "numeric" ~~

Linearity: Object of class "numeric" ~~

GageName: Object of class "character" ~~

GageTolerance: Object of class "numeric" ~~

DateOfStudy: Object of class "character" ~~

PersonResponsible: Object of class "character" ~~

Comments: Object of class "character" ~~

facNames: Object of class "character" ~~

Methods

as.data.frame signature(x = "MSALinearity"): ...

plot signature(x = "MSALinearity"): ...

response signature(object = "MSALinearity"): ...

response<- signature(object = "MSALinearity"): ...

show signature(object = "MSALinearity"): ...

summary signature(object = "MSALinearity"): ...

Author(s)

Thomas Roth: thomas.roth@tu-berlin.de
Etienne Stockhausen: stocdarf@mailbox.tu-berlin.de

Examples

```
showClass("MSALinearity")
```

---

mvPlot *Multi-Vari-Charts*

Description

Draws a Multi-Vari Chart for 2, 3 or 4 factors.

Usage

```
mvPlot(response, fac1, fac2, fac3 = NA, fac4 = NA, sort = TRUE, col, pch,
       cex.txt = 1, las = 1, labels = FALSE, quantile = TRUE, FUN = NA)
```

Arguments

response	the values of the response in a vector.response must be declared.
fac1	vector providing factor 1 as shown in the example.fac1 must be declared.
fac2	vector providing factor 2 as shown in the example.fac2 must be declared.
fac3	(optional) vector providing factor 3 as shown in the example. By default fac3 is set to 'NA'.
fac4	(optional) vector providing factor 4 as shown in the example. By default fac3 is set to 'NA'.
sort	logical value indicating whether the sequence of the factors given by fac1 - fac4 should be reordered to minimize the space needed to visualize the Multi-Vari-Chart. By default sort is set to 'TRUE'.
col	graphical parameter. Vector containing numerical values or character strings giving the colors for the different factors. By default col starts with the value '3' and is continued as needed.
pch	graphical parameter. Vector containing numerical values or single characters giving plotting points for the different factors. See points for possible values and their interpretation. Note that only integers and single-character strings can be set as a graphics parameter (and not NA nor NULL). By default pch starts with the value '1' and is continued as needed.
cex.txt	a numerical value giving the amount by which plotting labels at the single points should be magnified relative to the default. By default cex.txt is set to '1'.
las	graphical parameter for the style of x-axis labels. See par for further information.
labels	logical value indicating whether the single points should be labeld with the row-number of the data.frame invisibly returned by the function mvPlot. By default labels is set to 'FALSE'.
quantile	logical value indicating whether the quanitiles (0.00135, 0.5 & 0.99865) should be visualized for the single groups. By default quantile is set to 'TRUE'.
FUN	function to be used for calculation of response for unique settings of the factors e.g. the mean. By default FUN is set to 'NA' and therfore omitted.

Value

mvPlot returns invisibly a data.framein which all plotted points are listed.
The option labels can be used to plot the row-numbers at the single points and to ease the identification.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

See Also

http://www.r-qualitytools.org

Examples

```
#Example I
examp1 = expand.grid(c("Engine1","Engine2","Engine3"),c(10,20,30,40))
examp1 = as.data.frame(rbind(examp1, examp1, examp1))
examp1 = cbind(examp1, rnorm(36, 1, 0.02))
names(examp1) = c("factor1", "factor2", "response")
test1=mvPlot(response = examp1[,3], fac1 = examp1[,2],
             fac2 = examp1[,1],sort=FALSE,las=2,FUN=mean)

#Example II
examp2=expand.grid(c("Op I","Op II","Op III"),c(1,2,3,4),
                   c("20.11.1987","21.11.1987"))
examp2=as.data.frame(rbind(examp2, examp2, examp2))
examp2=cbind(examp2, rnorm(72, 22, 2))
names(examp2) = c("factor1", "factor2", "factor3", "response")
test2=mvPlot(response = examp2[,4], fac1 = examp2[,1],
            fac2 = examp2[,2], fac3 = examp2[,3], sort=TRUE, FUN=mean,
            labels=TRUE)

#Example III
examp3 = expand.grid(c("A","B","C"),c("I","II","III","IV"),c("H","I"),
                     c(1,2,3,4,5))
examp3 = as.data.frame(rbind(examp3, examp3, examp3))
examp3 = cbind(examp3, rnorm(360, 0, 2))
names(examp3) = c("factor1", "factor2", "factor3", "factor4", "response")
test3=mvPlot(response = examp3[,5], fac1 = examp3[,1],
             fac2 = examp3[,2], fac3 = examp3[,3], fac4 = examp3[,4], sort=TRUE,
             quantile=TRUE, FUN=mean)
```

---

names-methods *Methods for function names in Package* base

Description

Methods for function names in Package base.

Usage

```
## S4 method for signature 'facDesign'
names(x)
## S4 replacement method for signature 'facDesign'
names(x) <- value
## S4 method for signature 'mixDesign'
names(x)
## S4 replacement method for signature 'mixDesign'
names(x) <- value
## S4 method for signature 'doeFactor'
names(x)
## S4 replacement method for signature 'doeFactor'
names(x) <- value
## S4 method for signature 'taguchiDesign'
names(x)
## S4 replacement method for signature 'taguchiDesign'
names(x) <- value
## S4 method for signature 'taguchiFactor'
names(x)
## S4 replacement method for signature 'taguchiFactor'
names(x) <- value
## S4 method for signature 'pbDesign'
names(x)
## S4 replacement method for signature 'pbDesign'
names(x) <- value
## S4 method for signature 'pbFactor'
names(x)
## S4 replacement method for signature 'pbFactor'
names(x) <- value

## S4 method for signature 'gageRR'
names(x)
```

Arguments

x	object of class facDesign, mixDesign, taguchiDesign, taguchiFactor, pbDesign, pbFactor, doeFactor or gageRR.
value	character vector.

Methods

signature(x = "ANY") ANY object.

signature(x = "doeFactor") a doeFactor object.

signature(x = "facDesign") a facDesign object.

signature(x = "mixDesign") a mixDesign object.

signature(x = "taguchiDesign") a taguchiDesign object.

signature(x = "taguchiFactor") a taguchiFactor object.

`signature(x = "pbDesign")` a `pbDesign` object.

`signature(x = "pbFactor")` a `pbFactor` object.

`signature(x = "gageRR")` a `gageRR` object.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

See Also

http://www.r-qualitytools.org

---

ncol-methods *Get method*

Description

Get the number of columns for a `facDesign` object

Usage

```
## S4 method for signature 'facDesign'
ncol(x)
```

Arguments

x	a `facDesign` object

Value

numeric - number of columns

Methods

`signature(x = "facDesign")` Get the number of columns for a `facDesign` object.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

`nrow`
http://www.r-qualitytools.org

normalPlot *Normal plot*

Description

function to generate a normal plot of the factor effects for an object of class facDesign.

Usage

```
normalPlot(fdo, threeWay = FALSE, na.last = NA, alpha = 0.05, response = NULL,
           sig.col = c("red1", "red2", "red3"), sig.pch = c(1, 2, 3), main,
           ylim, xlim, xlab, ylab, pch, col, border = "red", ...)
```

Arguments

fdo	object of class facDesign.
threeWay	'TRUE'/'FALSE' plot three-way or higher interactions. By default threeWay is set to 'FALSE'.
na.last	By default set to 'NA'.
alpha	alpha for marking interactions.
response	response variable. If the response data frame of fdo consists of more then one responses, this variable can be used to choose just one column of the response data frame. response needs to be an object of class character with length of '1'. It needs to be the same character as the name of the response in the response data frame that should be plotted. By default response is set to 'NULL'.
sig.col	vector - colors for marking significant interactions. By default sig.col is set to 'c("red1", "red2", "red3")'.
sig.pch	vector - point characters for marking significant interactions. By default sig.pch is set to 'c(1, 2, 3)'.
main	graphical parameter. A main title for the plot, see also title.
ylim	graphical parameter. The y limits of the plot.
xlim	graphical parameter. The x limits (x1, x2) of the plot. Note that x1 > x2 is allowed and leads to a 'reversed axis'.
xlab	graphical parameter. A label for the x axis, defaults to a description of x.
ylab	graphical parameter. A label for the y axis, defaults to a description of y.
pch	graphical parameter. Vector containing numerical values or single characters giving plotting points for the different factors. See points for possible values and their interpretation. Note that only integers and single-character strings can be set as a graphics parameter (and not NA nor NULL).

col	graphical parameter. Single numerical value or character string giving the color for the points.
border	graphical parameter. Single numerical value or character string giving the color of the border line.
...	further graphical parameters see par.

Details

If the given facDesign object fdo contains replicates this function will deliver a normal plot
i.e.: effects divided by the standard deviation (t-value) will be plotted against an appropriate probability
scaling (see: ppoints).
If the given facDesign object fdo contains no replications the standard error can not be calculated.
In that case the function will deliver an effect plot.
i.e.: the effects will be plotted against an appropriate probability scaling. (see: ppoints).

Value

NULL

Note

For a more detailed example which shows the usage of normalPlot() in context please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

facDesign
paretoPlot
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#factorial design
fdo = facDesign(k=3, replicates = 2)
#seed for random numbers
set.seed(123)
#random numbers
y = rnorm(nrow(fdo))
#set the response
response(fdo) = y
#create a normal plot
normalPlot(fdo)
```

nrow-methods *Get method*

Description

Get the number of rows for a facDesign object.

Usage

```
## S4 method for signature 'facDesign'
nrow(x)
```

Arguments

x	a facDesign object

Value

numeric - number of rows

Methods

signature(x = "facDesign") Get the number of rows for a facDesign object.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

nrow
http://www.r-qualitytools.org

oaChoose *Taguchi Designs*

Description

Shows a matrix of possible taguchi designs.

Usage

```
oaChoose(factors1, factors2, level1, level2, ia)
```

Arguments

factors1	number of factors on level1.
factors2	number of factors on level2.
level1	number of levels for factors1.
level2	number of levels for factors2.
ia	number of interactions.

Details

oaChoose returns possible taguchi designs. Specifying the number of factor1 factors with level1 levels (factors1 = 2, level1 = 3 means 2 factors with 3 factor levels) and factor2 factors with level2 levels and desired interactions one or more taguchi designs are suggested.
If all parameters are set to '0', a matrix of possible taguchi designs is shown.

Value

oaChoose returns an object of class taguchiDesign

Note

TODO:

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

TODO:

See Also

facDesign for 2^k factorial designs,
rsmDesign for response surface designs,
fracDesign for fractional factorial design,
gageRRDesign for gage designs
http://www.r-qualitytools.org

Examples

```
oaChoose()
```

optimum *Optimal factor settings*

Description

Calculates the best factors settings with regard to defined desirabilities and constraints.
Two approaches are currently supported, (I) evaluating (all) possible factor settings and (II) using the function optim or gosolnp of the Rsolnp package.
Using optim optim initial values for the factors to be optimized over can be set via start.
The optimality of the solution depends critically on the starting parameters which is why it is recommended to use 'type="gosolnp"' although calculation takes a while.

Usage

```
optimum(fdo, constraints, steps = 25, type = "grid", start, ...)
```

Arguments

fdo	an object of class facDesign with fits and desires set.
constraints	constraints for the factors such as list(A = c(-2,1), B = c(0, 0.8)).
steps	number of grid points per factor if type = "grid".
type	type of search. "grid","optim" or "gosolnp" are supported (see DESCRIPTION).
start	numerical vector giving the initial values for the factors to be optimized over.
...	further aguments.

Details

It is recommened to use 'type="gosolnp"'. Derringer and Suich (1994) desirabilities do not have continuous first derivatives, more precisely they have points where their derivatives do not exist, start should be defined in cases where 'type = "optim"' fails to calculate the best factor setting.

Value

optimum returns an object of class desOpt.

Note

For a example which shows the usage of optimum() in context please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

optim
desirability
desires
gosolnp
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#BEWARE BIG EXAMPLE --Simultaneous Optimization of Several Response Variables--
#Source: DERRINGER, George; SUICH, Ronald: Simultaneous Optimization of Several
#        Response Variables. Journal of Quality Technology Vol. 12, No. 4,
#        p. 214-219

#Define the response suface design as given in the paper and sort via
#Standard Order
fdo = rsmDesign(k = 3, alpha = 1.633, cc = 0, cs = 6)
fdo = randomize(fdo, so = TRUE)

#Attaching the 4 responses
y1 = c(102,120,117,198,103,132,132,139,102,154,96,163,116,153,133,133,140,142,
       145,142)
y2 = c(900,860,800,2294,490,1289,1270,1090,770,1690,700,1540,2184,1784,1300,
       1300,1145,1090,1260,1344)
y3 = c(470,410,570,240,640,270,410,380,590,260,520,380,520,290,380,380,430,
       430,390,390)
y4 = c(67.5,65,77.5,74.5,62.5,67,78,70,76 ,70,63 ,75,65,71 ,70,68.5,68,68,69,
       70)
response(fdo) = data.frame(y1, y2, y3, y4)[c(5,2,3,8,1,6,7,4,9:20),]

#setting names and real values of the factors
names(fdo) = c("silica", "silan", "sulfur")
highs(fdo) = c(1.7, 60, 2.8)
lows(fdo) = c(0.7, 40, 1.8)

#summary of the response surface design
summary(fdo)

#setting the desires
desires(fdo) = desirability(y1, 120, 170, scale = c(1,1), target = "max")
desires(fdo) = desirability(y2, 1000, 1300, target = "max")
desires(fdo) = desirability(y3, 400, 600, target = 500)
desires(fdo) = desirability(y4, 60, 75, target = 67.5)
desires(fdo)

#Have a look at some contourPlots
par(mfrow = c(2,2))
contourPlot(A, B, y1, data = fdo)
contourPlot(A, B, y2, data = fdo)
wirePlot(A, B, y1, data = fdo)
wirePlot(A, B, y2, data = fdo)
```

```
#setting the fits as in the paper
fits(fdo) = lm(y1 ~ A + B + C + A:B + A:C + B:C + I(A^2) + I(B^2) + I(C^2),
               data = fdo)
fits(fdo) = lm(y2 ~ A + B + C + A:B + A:C + B:C + I(A^2) + I(B^2) + I(C^2),
               data = fdo)
fits(fdo) = lm(y3 ~ A + B + C + A:B + A:C + B:C + I(A^2) + I(B^2) + I(C^2),
               data = fdo)
fits(fdo) = lm(y4 ~ A + B + C + A:B + A:C + B:C + I(A^2) + I(B^2) + I(C^2),
               data = fdo)
#fits(fdo)

#calculate the same best factor settings as in the paper using type = "optim"
optimum(fdo, type = "optim")

#calculate (nearly) the same best factor settings as in the paper using type = "grid"
optimum(fdo, type = "grid")
```

`overall` *Overall Desirability*

Description

This is a function to calculate the desirability for each response as well as the overall desirability. The resulting `data.frame` can be used to plot the overall as well as the desirabilities for each response.
This function serves for a visualization of the desirability approach for multiple response optimization.

Usage

```
overall(fdo, steps = 20, constraints, ...)
```

Arguments

`fdo`	needs to be an object of class `facDesign` containing `fits` and `desires`
`steps`	numeric value - points per factor to be evaluated –> specifies also the grid size.
`constraints`	list - constraints for the factors in coded values such as list(A > 0.5, B < 0.2).
`...`	further arguments.

Value

`overall` returns a `data.frame` with a column for each factor, desirability for each response and a column for the overall desirability.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

see desirability.

See Also

facDesign
rsmDesign
desirability
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#arbitrary example with random data!!!
rsdo = rsmDesign(k = 2, blocks = 2, alpha = "both")
set.seed(123)
response(rsdo) = data.frame(y = rnorm(nrow(rsdo)), y2 = rnorm(nrow(rsdo)))
fits(rsdo) = lm(y ~ A*B + I(A^2) + I(B^2), data = rsdo)
fits(rsdo) = lm(y2 ~ A*B + I(A^2) + I(B^2), data = rsdo)
desires(rsdo) = desirability(y, -1, 2, scale = c(1, 1), target = "max")
desires(rsdo) = desirability(y2, -1, 0, scale = c(1, 1), target = "min")
dVals = overall(rsdo, steps = 10, constraints = list(A = c(-0.5,1),
                B = c(0, 1)))

##Uncomment for visualization of desirabilities
#require(lattice)
#contourplot(y ~ A*B,  data = dVals) #desirability of y
#contourplot(y2 ~ A*B,  data = dVals) #desirability of y2
#wireframe(overall ~ A*B, shade = TRUE, data = dVals)
```

---

paretoChart *Pareto Chart*

Description

function to create a pareto chart.

Usage

```
paretoChart(x, weight, showTable = TRUE, las = 0, main, col, border, xlab,
            ylab = "Frequency", percentVec, ...)
```

Arguments

x	vector of qualitative values.
weight	vector with weights for x (not yet implemented!)
showTable	logical value specifing whether a frequeny table will be added above the graph. By default showTable is set to 'TRUE'.

las	graphical parameter. Numeric in 0,1,2,3 giving the style of axis labels (see: par).
main	an overall title for the plot: see title.
col	numerical value or character string defining the fill-color of the bars.
border	numerical value or character string defining the border-color of the bars.
xlab	a title for the x axis: title.
ylab	a title for the y axis: title.
percentVec	numerical vector giving the position and values of tick marks for percentage axis.
...	further graphical parameters.

Value

paretoChart returns a data.frame of percentages.

Note

For an example which shows the usage of the function paretoChart() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Define.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
#artifical defects dataset
defects = LETTERS[runif(80, 1, 10)]
#paretoChart of the defects
paretoChart(defects)
```

---

paretoPlot *paretoPlot*

Description

Display standardized effects and interactions of a 'facDesign' object in a pareto plot.

Usage

```
paretoPlot(fdo, threeWay = FALSE, abs = TRUE, decreasing = TRUE, na.last = NA,
           alpha = 0.05, response = NULL, xlim, ylim, xlab, ylab, main,
           single = TRUE, ...)
```

Arguments

fdo	an object of class facDesign
threeWay	logical. If TRUE, threeway-interactions are displayed as well.
abs	logical. If TRUE, absolute effects and interactions are displayed.
alpha	the significance level used to calculate the critical value
response	response variable. If the response data frame of fdo consists of more then one responses, this variable can be used to choose just one column of the response data frame. response needs to be an object of class character with length of '1'. It needs to be the same character as the name of the response in the response data frame that should be plotted. By default response is set to 'NULL'.
decreasing	logical. If TRUE, effects and interactions are sorted decreasing.
na.last	na.last
xlab	graphical parameter
ylab	graphical parameter
xlim	graphical parameter
ylim	graphical parameter
main	graphical parameter
single	a logical value.If 'TRUE' a new graphic device will be opened for each column of the respond dataframe of fdo (response(fdo)). If set to 'FALSE' par(mfrow) will be set internally. By default single is set to 'TRUE'.
...	graphical parameters

Details

paretoPlot displays a pareto plot of effects and interactions for an object of class facDesign (i.e. 2^k full or 2^k-p fractional factorial design). For a given significance level alpha, a critical value is calculated and added to the plot. Standardization is achieved by dividing estimates with their standard error. For unreplicated fractional factorial designs a Lenth Plot is generated.

Value

a list of effects for each response in the 'facDesign' object

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

Design and Analysis of experiments - Volume2 - Advanced Experimental Design - Hinkelmann/Kempthorne

See Also

factors, fracDesign, facDesign

Examples

```
#factorial design with replications
#NA in response column and 2 replicates per factor combination
vp = fracDesign(k = 3, replicates = 2)
#generate some data
y1 = 4*vp[,1] -7*vp[,2] + 2*vp[,2]*vp[,1] + 0.2*vp[,3] + rnorm(16)
y2 = 9*vp[,1] -2*vp[,2] + 5*vp[,2]*vp[,1] + 0.5*vp[,3] + rnorm(16)
response(vp) = data.frame(y1,y2)
#show effects and interactions (nothing significant expected)
paretoPlot(vp)

#fractional factorial design --> Lenth Plot
vp = fracDesign(k = 4, gen = "D = ABC")
#generate some data
y = rnorm(8)
response(vp) = y
#show effects and interactions (nothing significant expected)
paretoPlot(vp)
```

---

pbDesign *Plackett-Burman Designs*

Description

Creates a Plackett-Burman design.

Usage

```
pbDesign(n, k, randomize = TRUE, replicates = 1)
```

Arguments

n	integer value giving the number of trials.
k	integer value giving the number of factors.
randomize	logical value ('TRUE'/'FALSE') - randomizes the RunOrder of the design. By default randomize is set to 'TRUE'.
replicates	Integer giving the number of replicates.

Value

pbDesign returns an object of class pbDesign.

Note

This function creates Placket-Burman Designs up to n=48. Bigger Designs are not implemented because of lack in practicability. For the creation either the number of factors or the number of trials can be denoted. Wrong combinations will lead to an error message. Originally Placket-Burman-Design are applicable for number of trials divisible by 4. If n is not divisble by 4 this function will take the next larger Placket-Burman Design and truncate the last rows and columns.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

References

- Plackett, R.L. and Burman J.P.: The design of optimum multifactorial experiments. Biometrika 33 (1946), 305-332
- Myers, R.H.; Montgomery, D.C. and Anderson-Cook, C.M.: Response Surface Methodology. p. 165. New Jersey: Wiley & Sons, 2009.

See Also

facDesign for 2^k factorial designs
rsmDesign for response surface designs
fracDesign for fractional factorial design
gageRRDesign for gage designs
http://www.r-qualitytools.org/html/Improve.html

Examples

```
pbdo=pbDesign(5)
```

---

pbDesign-class *Class "pbDesign"*

Description

pbDesign Class

Objects from the Class

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

Slots

name: Object of class "character" ~~

factors: Object of class "list" ~~

design: Object of class "data.frame" ~~

designType: Object of class "character" ~~

replic: Object of class "data.frame" ~~

response: Object of class "data.frame" ~~

Type: Object of class "data.frame" ~~

block: Object of class "data.frame" ~~

runOrder: Object of class "data.frame" ~~

standardOrder: Object of class "data.frame" ~~

desireVal: Object of class "list" ~~

desirability: Object of class "list" ~~

fits: Object of class "data.frame" ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

Examples

```
showClass("pbDesign")
```

---

pbFactor-class *Class "pbFactor"*

Description

pbFactor Class

Objects from the Class

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

Slots

values: Object of class "ANY" ~~

name: Object of class "character" ~~

unit: Object of class "character" ~~

type: Object of class "character" ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de> Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

Examples

```
showClass("pbFactor")
```

---

pcr *Process Capability Indices*

Description

Calculates the process capability cp, cpk, cpkL (onesided) and cpkU (onesided) for a given dataset and distribution.
A histogramm with a density curve is displayed along with the specification limits and a Quantile-Quantile Plot for the specified distribution.
Lower-, upper and total fraction of nonconforming entities are calculated. Box-Cox Transformations are supported as well as the calculation of Anderson Darling Test Statistics.

Usage

```
pcr(x, distribution = "normal", lsl, usl, target, boxcox = FALSE,
    lambda = c(-5,5), main, xlim, ylim, grouping = NULL, std.dev = NULL,
    conf.level = 0.9973002, start, lineWidth = 1, lineCol = "red",
    lineType = "solid", specCol = "red3", specWidth = 1, cex.text = 2,
    cex.val = 1.5, cex.col = "darkgray", plot = TRUE, bounds.lty = 3,
    bounds.col = "red", ...)
```

Arguments

x	numeric vector containing the values for which the process capability should be calculated.
distribution	character string specifying the distribution of x. The function cp will accept the following caracter strings for distribution:

- "normal"
- "log-normal"
- "exponential"
- "logistic"
- "gamma"
- "weibull"
- "cauchy"
- "gamma3"
- "weibull3"
- "lognormal3"
- "beta"

- "f"
- "t"
- "geometric"
- "poisson"
- "negative-binomial"

By default `distribution` is set to "normal".

`lsl`	numeric value for the lower specification limit.
`usl`	numeric value for the upper specification limit.
`target`	(optional) numeric value giving the target value.
`boxcox`	logical value specifying whether a Box-Cox transformation should be performed or not. By default `boxcox` is set to 'FALSE'.
`lambda`	(optional) lambda for the transformation, default is to have the function estimate lambda.
`main`	an overall title for the plot: see `title`.
`xlim`	vector giving the range of the x-axis.
`ylim`	vector giving the range of the y-axis.
`grouping`	(optional) If grouping is given the standard deviation is calculated as mean standard deviation of the specified subgroups corrected by the factor c4 and expected fraction of nonconforming is calculated using this standard deviation.
`std.dev`	(optional) historical standard devation (only provided for normal distribution!).
`conf.level`	numeric value between '0' and '1' giving the confidence interval. By default `conf.level` is 0.9973 (99.73%) which is the reference interval bounded by the 99.865% and 0.135% quantile.
`start`	a named list giving the parameters to be fitted with initial values. Must be supplied for some distribution (see `fitdistr` of the R-package `MASS`).
`lineWidth`	a numeric value specifying the width of the line for the density curve.
`lineCol`	numerical value or character string (like "red") specifying the color of the line for the density curve.
`lineType`	character string specifying the line type e.g. "dashed", "solid", etc.
`specCol`	numerical value or character string specifying the color for the specification limits.
`specWidth`	numerical value specifying the line width for the specification limits.
`cex.text`	numerical value specifying the cex for lsl, usl and target.
`cex.val`	numerical value specifying the cex for the process capability ratios.
`cex.col`	numerical value or character string specifying the color for lsl, usl and target.
`plot`	logical value. If set to 'FALSE' the graphical output will be omitted.\ By default `plot` is set to 'TRUE'.
`bounds.col`	graphical parameter. For further details see `ppPlot` or `qqPlot`.
`bounds.lty`	graphical parameter. For further details see `ppPlot` or `qqPlot`.
`...`	some other graphical parameters.

Details

Distribution fitting is deligated to function `fitdistr` of the R-package `MASS` as well as the calculation of lambda for the Box Cox Transformation. p-values for Anderson Darling Test are reported for the most important distributions.
cpk is always min(cpK, cpL).

- pt stands for total fraction nonconforming
- pu stands for upper fraction nonconforming
- pl stands for lower fraction nonconforming
- cp stands for process capability index
- cpkL stands for lower process capability index
- cpkU stands for upper process capability index
- cpk stands for minimum process capability index

For a Box-Cox tranformation a data vector with positive values is needed to estimate an optimal value of lambda for the Box-Cox power transformation of teh values. The Box-Cox power transformation is used to bring the distribution of the data vector to a form close to normal. Estimation of the optimal lambda is deligated to the function `boxcox` of the MASS package. The Box-Cox transformation has the form y(lambda)=(y^lambda-1)/lambda for lambda not equal to zero, and y(lambda)=log(y) for lambda equal to zero. The function boxcox computes the profile log-likelihoods for a range of values of parameter lambda. Function boxcox.lamba returns the value of lambda with the maximum profile log-likelihood.

In case no specification limits are given, 'lsl' and 'usl' are calculated to support a process capability index of 1.

Value

The function `pcr` returns a list with `lambda`, `cp`, `cpl`, `cpu`, `ppt`, `ppl`, `ppu`, `A`, `usl`, `lsl`, `target`.

Note

At this point there's no distinction made between process performance P_pk and process capability. The latter implies a process that is in statistical contro whereas process performance is estimated for a process that might not have been demonstrated to be in a state of statistical control.
Currently work is being done to support the usage of three parameter weibull, loglogistic and gamma distribution.
For a detailed example which shows the usage of the function `cp()` please read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org/html/Analyze.html`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

- ISO(2006). Statistical methods - Process performance and capability statistics for measured quality characteristics (ISO 21747).
- ISO/TR(2007).Statistical methods in process management - capability and performance - part 4: process capability estimates and performance (ISO/TR 22514-4).
- MITTAG, H.-J.; RINNE, H.: Prozessfaehigkeitsmessung fuer die industrielle Praxis. Muinch: Hanser, 1999.
- KOTZ, Samuel; LOVELACE, Cynthia R.: Process capability indices in theory and practice. London,New York: Arnold, 1998.
- Process Capability Statistics for Non-Normal Distributions in R https://web.warwick.ac.uk/statsdept/user2011/TalkSlides/Contributed/18Aug_0950_FocusVI_5-ProcessOptimization_2-Roth.pdf

See Also

qqPlot
ppPlot
http://www.r-qualitytools.org/html/Analyze.html

Examples

```
x = rweibull(30, 2, 8) +100
#process capability for a weibull distribution
pcr(x, "weibull", lsl = 100, usl = 117)

#box cox transformation and one sided
pcr(x, boxcox = TRUE, lsl = 1)

#boxcox with lambda=2
pcr(x, boxcox = 2, lsl = 1)

#process capability assuming a normal distribution
pcr(x, "normal", lsl = 0, usl = 17)

#process capability for a normal distribution and data in subgroups
#some artificial data with shifted means in subgroups
x = c(rnorm(5, mean = 1), rnorm(5, mean = 2), rnorm(5, mean = 0))

#grouping vector
group = c(rep(1,5), rep(2,5), rep(3,5))

#calculate process capability
pcr(x, grouping = group) #compare to sd(x)
```

plot-methods *Methods for Function plot in Package* graphics

Description

Methods for function plot in Package graphics

Methods

signature(x = "MSALinearity") object of class MSALinearity.

signature(x = "ANY") ANY

signature(x = "desirability") object of class desirability.

signature(x = "distr") object of class distr.

signature(x = "distrCollection") object of class distrCollection.

signature(x = "steepAscent") object of class distrCollection.

signature(x = "gageRR") object of class gageRR.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

ppPlot *Probability Plots for various distributions*

Description

ppPlot creates a Probability plot of the values in x including a line.

Usage

```
ppPlot(x, distribution,  confbounds = TRUE, alpha, probs, main, xlab, ylab,
       xlim, ylim, border = "red", bounds.col = "black", bounds.lty = 1,
       grid = TRUE, box = TRUE, stats = TRUE, start, ...)
```

Arguments

x	vector containing the sample for ppPlot.
distribution	character string specifying the distribution of x. The function ppPlot will support the following character strings for distribution: • "beta" • "cauchy" • "chi-squared" • "exponential" • "f" • "gamma" • "geometric" • "log-normal" • "lognormal" • "logistic" • "negative binomial" • "normal" • "Poisson" • "t" • "weibull" By default distribution is set to "normal".
confbounds	boolean value: 'TRUE' if confidence bounds should be drawn (default value).
alpha	significance level for the confidence bounds, set on '0.05' by default.
probs	vector containing the percentages for the y axis. All the values need to be between '0' and '1' If probs is missing it will be calculated internally.
main	an overall title for the plot: see title.
xlab	a title for the x axis: title.
ylab	a title for the y axis: title.
xlim	vector giving the range of the x-axis.
ylim	vector giving the range of the y-axis.
border	numerical value or single character string giving the color of interpolation line. By default border is set to "red".
bounds.col	numerical value or single character string giving the color of confidence bounds lines. By default bounds is set to "black".
bounds.lty	numerical value giving the color of confidence bounds lines. By default bounds is set to '1'.
grid	logical value, deciding whether a grid will be added to the current plot. By default grid is set to 'TRUE'.
box	logical value, deciding whether a box is drawn to around the current plot. By default box is set to 'TRUE'.

stats	logical value deciding whether ppPlot returns the values listed under value. By default stats is set to 'TRUE'.
start	A named list giving the parameters to be fitted with initial values. Must be supplied for some distribution. (see Details)
...	further graphical parameters (see par).

Details

Distribution fitting is deligated to function fitdistr of the R-package MASS.
For computation of the confidence bounds the variance of the quantiles is estimated using the delta method,
which implies estimation of observed Fisher Information matrix as well as the gradient of the CDF of the fitted distribution.
Where possible, those values are replaced by their normal approximation.

Value

ppPlot returns a list containing the following:
x - x coordinates
y - y coordinates
int - Intercept
slope - slope

Note

For an example in context which shows the usage of the function ppPlot() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Analyze.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

qqPlot fitdistr in R-package MASS
http://www.r-qualitytools.org/html/Analyze.html

Examples

```
#set up the plotting window for 6 plots
par(mfrow = c(3,2))

#generate random data from weibull distribution
x = rweibull(20, 8, 2)

#Probability Plot for different distributions
ppPlot(x, "log-normal")
ppPlot(x, "normal")
ppPlot(x, "exponential", DB = TRUE)
ppPlot(x, "cauchy")
```

```
ppPlot(x, "weibull")
ppPlot(x, "logistic")
```

`print.adtest` *Test Statistics*

Description

Generic S3 function for objects of class `adtest`.

Usage

```
## S3 method for class 'adtest'
print(x, digits = 4, quote = TRUE, prefix = "", ...)
```

Arguments

`x`	needs to be an object of class `adtest`.
`digits`	minimal number of significant digits, see `print.default`.
`quote`	logical, indicating whether or not strings should be printed with surrounding quotes. By default `quote` is set to 'TRUE'.
`prefix`	single character or character string that will be printet in front of `x`.
`...`	further arguments passed to or from other methods.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

`print.default`
http://www.r-qualitytools.org

print.invisible *Print Function*

Description

Generic S3 function for objects of class invisible.

Usage

```
## S3 method for class 'invisible'
print(x, ...)
```

Arguments

x	needs to be an object of class invisible.
...	further arguments passed to or from other methods.

Note

print.invisible is used to hide the internals of the method simProc.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

simProc
print.default
http://www.r-qualitytools.org

qqPlot *Quantile-Quantile Plots for various distributions*

Description

qqPlot creates a QQ plot of the values in x including a line which passes through the first and third quartiles.

Usage

```
qqPlot(x, y, confbounds = TRUE, alpha, main, xlab, ylab, xlim, ylim,
       border = "red", bounds.col = "black", bounds.lty = 1, start, ...)
```

Arguments

x	the sample for qqPlot
y	character string specifying the distribution of x. The function qqPlot will support the following character strings for y: • "beta" • "cauchy" • "chi-squared" • "exponential" • "f" • "gamma" • "geometric" • "log-normal" • "lognormal" • "logistic" • "negative binomial" • "normal" • "Poisson" • "t" • "weibull" By default distribution is set to "normal".
confbounds	boolean value: 'TRUE' if confidence bounds should be drawn (default value).
alpha	significance level for the confidence bounds, set on '0.05' by default.
main	an overall title for the plot: see title.
xlab	a title for the x axis: title.
ylab	a title for the y axis: title.
xlim	vector giving the range of the x-axis.
ylim	vector giving the range of the y-axis.
border	numerical value or single character string giving the color of interpolation line. By default border is set to "red".
bounds.col	numerical value or single character string giving the color of confidence bounds lines. By default bounds is set to "black".
bounds.lty	numerical value giving the color of confidence bounds lines. By default bounds is set to '1'.
start	A named list giving the parameters to be fitted with initial values. Must be supplied for some distribution: (see Details).
...	further graphical parameters: (see par).

Details

Distribution fitting is deligated to function fitdistr of the R-package MASS.
For computation of the confidence bounds the variance of the quantiles is estimated using the delta method,
which implies estimation of observed Fisher Information matrix as well as the gradient of the CDF of the fitted distribution.
Where possible, those values are replaced by their normal approximation.

Value

a list containing the x and y quantiles

x	sample quantiles
y	theoretical quantiles

Note

For an example in context which shows the usage of the function qqPlot() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Analyze.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

ppPlot fitdistr in R-package MASS
http://www.r-qualitytools.org/html/Analyze.html

Examples

```
#set up the plotting window for 6 plots
par(mfrow = c(3,2))

#generate random data from weibull distribution
x = rweibull(20, 8, 2)

#Quantile-Quantile Plot for different distributions
qqPlot(x, "log-normal")
qqPlot(x, "normal")
qqPlot(x, "exponential", DB = TRUE)
qqPlot(x, "cauchy")
qqPlot(x, "weibull")
qqPlot(x, "logistic")
```

randomize *Randomization*

Description

function to do randomize the run order of factorial designs.

Usage

```
randomize(fdo, random.seed, so = FALSE)
```

Arguments

fdo	needs to be an object of class facDesign.
random.seed	seed for randomness. See set.seed.
so	logical value specifying whether the standard order should be used or not. By default so is set to 'FALSE'.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

response-methods *Get and set methods*

Description

Set or get the response for a facDesign, mixDesign, gageRRDesign, taguchiDesign or pbDesign object.

Usage

```
## S4 method for signature 'facDesign'
response(object)
## S4 replacement method for signature 'facDesign'
response(object) <- value
## S4 method for signature 'mixDesign'
response(object)
## S4 replacement method for signature 'mixDesign'
response(object) <- value
## S4 method for signature 'gageRR'
```

```
response(object)
## S4 replacement method for signature 'gageRR'
response(object) <- value
## S4 method for signature 'taguchiDesign'
response(object)
## S4 replacement method for signature 'taguchiDesign'
response(object) <- value
## S4 method for signature 'pbDesign'
response(object)
## S4 replacement method for signature 'pbDesign'
response(object) <- value
## S4 method for signature 'steepAscent'
response(object)
## S4 replacement method for signature 'steepAscent'
response(object) <- value
## S4 method for signature 'MSALinearity'
response(object)
## S4 replacement method for signature 'MSALinearity'
response(object) <- value
```

Arguments

object	a facDesign, mixDesign, gageRRDesign, taguchiDesign or pbDesign object
value	new response vector

Methods

signature(object = "MSALinearity") Function to create the respone for an object of class MSALinearity.

signature(object = "facDesign") response is a generic accessor function, and response<- is a generic replacement function. The default methods get and set the response attribute of a facDesign object.
Value must be of same length as nrow(object). If value is shorter/longer than nrow(object) the setting of the response will fail.

signature(object = "mixDesign") response is a generic accessor function, and response<- is a generic replacement function. The default methods get and set the response attribute of a mixDesign object.
Value must be of same length as nrow(object). If value is shorter/longer than nrow(object) the setting of the response will fail.

signature(object = "gageRR") response is a generic accessor function, and response<- is a generic replacement function. The default methods get and set the response attribute of gageRRDesign object.
Value must be of same length as nrow(object). If value is shorter/longer than nrow(object) the setting of the response will fail.

signature(object = "taguchiDesign") response is a generic accessor function, and response<- is a generic replacement function. The default methods get and set the response attribute of taguchiDesign object. value must be of same length as nrow(object). If value is shorter/longer than nrow(object) the setting of the response will fail.

`signature(object = "pbDesign")` response is a generic accessor function, and `response<-` is a generic replacement function. The default methods get and set the response attribute of pbDesign object. value must be of same length as `nrow(object)`. If value is shorter/longer than `nrow(object)` the setting of the response will fail.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

See Also

factors
fracDesign
taguchiDesign
mixDesign
http://www.r-qualitytools.org

Examples

```
#NA in response column
fdo = fracDesign(k = 3)
fdo

#response
y = rnorm(8)

#2^k numeric values in response column
response(fdo) = y
fdo
```

---

rsmChoose *Choosing a response surface design from a table*

Description

Designs displayed are central composite designs with orthogonal blocking and near rotatability. Choosing a design is done by clicking with the mouse into the appropriate field.

Usage

```
rsmChoose()
```

Details

Useful Central Composite Designs with orthogonal blocking and near rotatability

Value

object of class facDesign.

Note

for an example for usage please visit http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

- BOX, George. E. P. and HUNTER, J.S.: "Multifactor Experimental Designs for Exploring Response Surfaces",
  The Annals of Mathematical Statistics 28, p.195-233, 1957

- MYERS, R.H.; MONTGOMERY, D.C.; ANDERSON-COOK, C.M.:
  Response Surface Methodology (2009) p.329, table 7.5. New Jersey: Wiley.

See Also

fracChoose
rsmDesign
http://www.r-qualitytools.org/html/Improve.html

---

rsmDesign *Generate a response surface design (i.e. central composite design)*

Description

Generates a response surface design containing a cube, centerCube, star and centerStar portion.

Usage

```
rsmDesign(k = 3, p = 0, alpha = "rotatable", blocks = 1, cc = 1, cs = 1,
          fp = 1, sp = 1, faceCentered = FALSE)
```

Arguments

k	integer value giving the number of factors. By default k has the value '3'.
p	integer value giving the number of additional factors in the response surface design by aliasing effects. By default p has the value '0'.
alpha	character string - alpha shoud be "rotatable" (default), "orthogonal" or "both". If "both" values for cc and cs will be DISCARDED.

blocks	integer value giving the number of blocks in the response surface design. By default blocks has the value '1'.
cc	integer value giving the number of centerpoints (per block) in the cube portion (i.e. the factorial 2^k design) of the response surface design. Replications of centerpoints in the cube portion can be set with this.By default cc has the value '1'.
cs	integer value giving the number of centerpoints in the star portion (alpha) of the response surface design. Replications of centerpoints in the star portion can be set with this. By default cs has the value '1'.
fp	integer value giving the number of replications per factorial point (i.e. corner points e.g. (-1,1)). By default fp has the value '1'.
sp	integer value giving the number of replications per star point (i.e. alpha). By default sp has the value '1'.
faceCentered	logical value wheter to use a faceCentered response surface design or not (i.e. alpha = 1). By default faceCentered has the value 'FALSE'.

Details

Generated designs consist of a cube, centerCube, star and a centerStar portion. The replication structure can be set with the parameters cc (centerCube), cs (centerStar), fp (factorialPoints) and sp (starPoints).

Value

rsmDesign returns an object of class facDesign.

Note

For an example in context which shows the usage of the function rsmDesign() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

facDesign
fracDesign
fracChoose
pbDesign
rsmChoose
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#central composite design for 2 factors with 2 blocks, alpha = 1.41,
#5 centerpoints in the cube portion and 3 centerpoints in the star portion:
rsmDesign(k = 2, blocks = 2, alpha = sqrt(2),cc = 5, cs = 3)

#central composite design with both, orthogonality and near rotatability
rsmDesign(k = 2, blocks = 2, alpha = "both")

#central composite design with
#2 centerpoints in the factorial portion of the design i.e 2
#1 centerpoint int the star portion of the design i.e. 1
#2 replications per factorial point i.e. 2^3*2 = 16
#3 replications per star points 3*2*3 = 18
#makes a total of 37 factor combinations
rsdo = rsmDesign(k = 3, blocks = 1, alpha = 2, cc = 2, cs = 1, fp = 2, sp = 3)
nrow(rsdo) #37
```

---

runOrd-methods *Get and set methods*

Description

Get and set the runOrd for a facDesign x.

Usage

```
## S4 method for signature 'facDesign'
runOrd(x)
## S4 replacement method for signature 'facDesign'
runOrd(x) <- value
```

Arguments

x	a facDesign object
value	new runOrd vector

Methods

```
signature(object = "facDesign")
```

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

Examples

```
#NA in response column
fdo = facDesign(k = 3)

runOrd(fdo)
```

---

sigma-methods *Get and set methods*

Description

Get and set the sigma for an object of class gageRR.

Usage

```
## S4 method for signature 'gageRR'
sigma(x)
## S4 replacement method for signature 'gageRR'
sigma(x) <- value
```

Arguments

x	object of class gageRR.
value	number of sigmas for calculation the Number of Distinct Categories (signal-to-noise-ratio)in a Gage R&R analysis.

Methods

signature(objectc = "gageRR") Get and set the sigma for an object of class gageRR.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

gageRRDesign
http://www.r-qualitytools.org

Examples

```
x = gageRRDesign(Operators = 3, Parts = 10, Measurements = 3)
#default 6sigma
sigma(x)
sigma(x) = 5.15
sigma(x)
```

simProc *Simulated Process*

Description

This is a function to simulate a black box process for teaching the use of designed experiments. The optimal factor settings can be found using a sequential assembly strategy i.e. apply a 2^k factorial design first, calculate the path of the steepest ascent, again apply a 2^k factorial design and augment a star portion to find the optimal factor settings. Of course other strategies are possible.

Usage

```
simProc(x1, x2, x3, noise = TRUE)
```

Arguments

x1	numeric vector containing the values for factor 1.
x2	numeric vector containing the values for factor 2.
x3	numeric vector containing the values for factor 3.
noise	logical value deciding whether noise should be added or not. Default setting is 'FALSE'.

Details

factor 1 is best within [40, 250]; factor 2 within [90, 240]

Value

simProc returns a numeric value within the range [0,1].

Note

For an example in context which shows the usage of the function simProc() please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

facDesign for 2^k factorial designs
rsmDesign for response surface designs
fracDesign for fractional factorial design
http://www.r-qualitytools.org/html/Improve.html

Examples

```
simProc(120, 140, 1)
simProc(120, 220, 1)
simProc(160, 140, 1)
```

---

snPlot *Signal-to-Noise-Ratio Plots*

Description

A Signal-to-Noise-Ratio plot is created for designs of type taguchiDesign with at least two replicates.

Usage

```
snPlot(object, type="nominal", factors, fun = mean, response = NULL,
       single = FALSE, points = FALSE, classic = FALSE, axes = TRUE,
       lty, xlab, ylab, main, ylim, ...)
```

Arguments

object	needs to be an object of class taguchiDesign.
type	character variable setting the type of the signal-to-noise-ratio plot. The following three can be chosen: • "nominal": Creates a nominal-the-best signal-to-noise-ratio plot. This type of signal-to-noise-ratio plot is used if the goal is to equalize the observed values to a nominal value. For example, the output impedance of an electric device should be as close to specification as possible to garantee a high level of quality. • "smaller": Creates a smaller-the-better signal-to-noise-ratio plot. This type of signal-to-noise-ratio plot is used if the goal is to minimize the observed values. For example, the usage of an of an automatic peeling device: the remains of skin on potatoes should be as small as possible to ensure a high level of quality. • "larger": Creates a larger-the-better signal-to-noise-ratio plot. This type of signal-to-noise-ratio plot is used if the goal is to maximize the observed values. For example, the amount of fruits on trees in a cultivation should be maximised to maximise the profit. By default fun is set to "nominal"
factors	for which factor is the effectPlot to be created.
fun	a function for the construction of the effectPlot such as mean, median,etc. By default fun is set to mean.

response	response variable. If the response data frame of fdo consists of more then one responses, this variable can be used to choose just one column of the response data frame. response needs to be an object of class character with length of '1'. It needs to be the same character as the name of the response in the response data frame that should be plotted. By default response is set to 'NULL'.
single	logical value. If 'TRUE' device region can be set up using for instance par(mfrow = c(2,2)). By default single is set to 'FALSE'.
points	logical value. If 'TRUE' points are shown in addition to values out of fun. By default points is set to 'FALSE'.
classic	logical value. 'TRUE' creates an effectPlot as depicted in most textbooks. By default classic is set to 'FALSE'.
axes	logical value indicating wheter the axes should be drawn or not. 'TRUE' by default.
lty	numerical value which specifies the line type used.
xlab	a title for the x axis: title.
ylab	a title for the y axis: title.
main	an overall title for the plot: see title.
ylim	vector giving the range of the y-axis.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

snPlot uses effectPlot and creates an effect plot for the signal-to-noise ratios as target values. Depending on the used type the target values for the single replications of the taguchi design will be calculated in the following way:

- "nominal": SN = 10*log(mean(y)/var(y))
- "smaller": SN = -10*log((1/n)*sum(y^2))
- "larger": SN = -10*log((1/n)*sum(1/y^2))

Signal-to-Noise ratio plots are an additional tool to estimate the effects of the single factors. Beside the effect plot, which is used to identify the factor with the most effect to a process or something like that, the signal-to-noise ratio plot can be used to judge the variance an d therefore the validity of the results of an effect plot.

Value

snPlot invisibly returns a data.frame containing all the single Signal-to-Noise ratios.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

References

- http://www.statsoft.com/textbook/experimental-design
- APTE, P.R.: Taguchi Method for Dynamic Problems - 3 Day Taguchi Method Workshop at Unimap. Bombay, 2012.

See Also

interactionPlot
paretoPlot
facDesign
response
normalPlot
http://www.r-qualitytools.org/html/Improve.html

Examples

```
tdo = taguchiDesign("L9_3",replicates=3)
response(tdo) = rnorm(27)
snPlot(tdo, points = TRUE, col = 2, pch = 16, lty = 3)
```

---

standOrd-methods *Get and set method*

Description

Get and set the standard order of a factorial design.

Usage

```
## S4 method for signature 'facDesign'
standOrd(x)
## S4 replacement method for signature 'facDesign'
standOrd(x) <- value
```

Arguments

x	a facDesign object
value	new standOrd vector

Methods

signature(object = "facDesign") So far used internally only.

See Also

http://www.r-qualitytools.org

---

star-methods *Get and set methods*

Description

function to set and get the star portion of a response surface design

Usage

```
## S4 method for signature 'facDesign'
star(x)
## S4 replacement method for signature 'facDesign'
star(x) <- value
```

Arguments

x	a 'facDesign' object
value	data.frame holding the star portion.

Methods

signature(x = "facDesign") Get and set the star for the factors in an object of class facDesign. Could be used to exchange a default star portion with a specific one. So far used internally.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

rsmDesign
rsmChoose
http://www.r-qualitytools.org

Examples

```
#rotatable response surface design with 3 factors
design = rsmDesign(k = 3)
#star portion of the design
star(design)
```

`starDesign` *Axial Design*

Description

starDesign is a function to create the star portion of a response surface design. The `starDesign` function can be used to create a star portion of a response surface design for a sequential assembly strategy.
One can either specify k and p and alpha and cs and cc OR simply simply pass an object of class `facDesign` to the data. In the latter an object of class `facDesign` otherwise a list containing the axial runs and centerpoints is returned.

Usage

```
starDesign(k, p = 0, alpha = c("both", "rotatable", "orthogonal"), cs, cc, data)
```

Arguments

`k`	integer value giving number of factors.
`p`	integer vaue giving the number of factors via aliasing. By default set to '0'.
`alpha`	if no numeric value is given defaults to "both" i.e. "orthogonality" and "rotatibility" which can be set as character strings too.
`cs`	integer value giving the number of centerpoints in the star portion of the design.
`cc`	integer value giving the number of centerpoints in the cube portion of the design.
`data`	optional - an object of class `facDesign`.

Value

`starDesign` returns a `facDesign` object if an object of class `facDesign` is given or a list containing entries for axial runs and center points in the cube and the star portion of a design.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

MYERS, R.H.; MONTGOMERY, D.C.; ANDERSON-COOK, C.M.: Response Surface Methodology. New Jersey: Wiley,2009.

See Also

`facDesign` for 2^k factorial designs
`fracDesign` for 2^k-p fractional factorial designs
`rsmDesign` for response surface designs
`mixDesign` for mixture designs
`http://www.r-qualitytools.org`

Examples

```
#Example 1 - sequential assembly
#factorial design with one center point in the cube portion
fdo = facDesign(k = 3, centerCube = 1)
fdo

#set the response via generic response method
response(fdo) = 1:9

#sequential assembly of a response surface design (rsd)
rsd = starDesign(data = fdo)
rsd

#Example 2 - returning a list
starDesign(k = 3, cc = 2, cs = 2, alpha = "orthogonal")
starDesign(k = 3, cc = 2, cs = 2, alpha = "rotatable")
starDesign(k = 3, cc = 2, cs = 2, alpha = "both")
```

---

steepAscent *Steepest Ascent*

Description

steepAscent is a method to calculate the steepest ascent for a facDesign object.

Usage

```
steepAscent(factors, response, size = 0.2, steps = 5, data)
```

Arguments

factors	list containing vector of factor names (coded) to be included in calculation, first factor is the reference factor.
response	character - response given in data.
size	numeric integer value giving the step size in coded units for the first factor given in factors. By default size is set to '0.2'.
steps	numeric integer value giving the number of steps. By default step is set to '5'.
data	needs to be an object of class facDesign.

Details

A first order model is fitted for the factors given in factors. Based on the step size given the steepest ascent is calculated.

Value

steepAscent returns an object of class steepAscent.

Note

This is the steepest ascent for a single response considering main effects only.
For an example in context which shows the usage of the function steepAscent() to an object of class facDesign, please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

http://www.itl.nist.gov/div898/handbook/pri/section5/pri5311.htm

See Also

desires for multiple response optimization usign desirabilities
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#Example from References
fdo = facDesign(k = 2, centerCube = 5)
lows(fdo) = c(170, 150)
highs(fdo) = c(230, 250)
names(fdo) = c("temperature", "time")
units(fdo) = c("C", "minutes")
yield = c(32.79, 24.07, 48.94, 52.49, 38.89, 48.29, 29.68, 46.5, 44.15)
response(fdo) = yield
summary(fdo)

sao = steepAscent(factors = c("B", "A"), response = "yield", size = 1,
                  data = fdo)
sao
obs.yield = c(NA, 56.2, 71.49, 75.63, 72.31, 72.10)
response(sao) = obs.yield
plot(sao, type = "b", col = 2, main = "Steepest Ascent")
```

---

steepAscent-class *Class "steepAscent"*

Description

steepAscent Class

Objects from the Class

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

Slots

`name`: Object of class `"character"` ~~

`X`: Object of class `"data.frame"` ~~

`response`: Object of class `"data.frame"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("steepAscent")
```

---

summary-methods *Methods for Function summary in Package base*

Description

Methods for function `summary` in Package `base`.

Methods

`signature(object = "ANY")` ANY

`signature(object = "MSALinearity")` object of class `MSALinearity`.

`signature(object = "distrCollection")` object of class `distrCollection`.

`signature(object = "facDesign")` object of class `facDesign`.

`signature(object = "mixDesign")` object of class `mixDesign`.

`signature(object = "taguchiDesign")` object of class `taguchiDesign`.

`signature(object = "pbDesign")` object of class `pbDesign`.

`signature(object = "gageRR")` object of class `gageRR`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

See Also

http://www.r-qualitytools.org

summaryFits *Fit Summary*

Description

Function to give an overview of fitted linear models for objects of class facDesign.

Usage

```
summaryFits(fdo, lmFit = TRUE, curvTest = TRUE, origFit = TRUE)
```

Arguments

fdo	needs to be an object of class facDesign.
lmFit	logical value deciding whether the fits from the object fdo should be taken or not. By default lmFit is set to 'TRUE'.
curvTest	logical value deciding wheter curvature tests will be performed or not. By default curvTest is set to 'TRUE'.
origFit	logical value. If 'TRUE' (default) the original values of the fits will be displayed.

Note

This is BETA!

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

http://www.r-qualitytools.org

taguchiChoose *Taguchi Designs*

Description

Shows a matrix of possible taguchi designs

Usage

```
taguchiChoose(factors1 = 0, factors2 = 0, level1 = 0, level2 = 0, ia = 0)
```

Arguments

factors1	integer number of factors on level1. By default set to '0'.
factors2	integer number of factors on level2. By default set to '0'.
level1	integer number of levels for factors1.
level2	integer number of levels for factors2. By default set to '0'
ia	integer number of interactions. . By default set to '0'.

Details

taguchiChoose returns possible taguchi designs.
Specifying the number of factor1 factors with level1 levels (factors1 = 2, level1 = 3 means 2 factors with 3 factor levels) and factor2 factors with level2 levels and desired interactions one or more taguchi designs are suggested.
If all parameters are set to 0, a matrix of possible taguchi designs is shown.

Value

taguchiChoose returns an object of class taguchiDesign.

Note

It is recommended to consult a textbook on taguchi methods to be fully aware of the benefits and limitations that taguchi designs impose.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

http://www.york.ac.uk/depts/maths/tables/taguchi_table.htm

See Also

facDesign for 2^k factorial designs
rsmDesign for response surface designs
fracDesign for fractional factorial design
gageRRDesign for gage designs
http://www.r-qualitytools.org

Examples

```
taguchiChoose()
```

`taguchiDesign` *Taguchi Designs*

Description

Creates a taguchi design.

Usage

```
taguchiDesign(design, randomize = TRUE, replicates = 1)
```

Arguments

`design` design needs to be one of the following characters, which specifies the orthogonal array of the taguchi design:

- "L4_2" for three two-level factors
- "L8_2" for seven two-level factors
- "L9_3" for four three-level factors
- "L12_2" for 11 two-level factors
- "L16_2" for 16 two-level factors
- "L16_4" for 16 four-level factors
- "L18_2_3" for one two-level and seven three-level factors
- "L25_5" for six five-level factors
- "L27_3" for 13 three-level factors
- "L32_2" for 32 two-level factors
- "L32_2_4" for one two-level factor and nine four-level factors
- "L36_2_3_a" for 11 two-level factors and 12 three-level factors
- "L36_2_3_b" for three two-level factors and 13 three-level factors
- "L50_2_5" for one two-level factor and eleven five-level factors
- "L8_4_2" for one four-level factor and four two-level factors

- "L16_4_2_a" for one four-level factor and 12 two-level factors
- "L16_4_2_b" for two four-level factors and nine two-level factors
- "L16_4_2_c" for three four-level factors and six two-level factors
- "L16_4_2_d" for five four-level factors and two two-level factors
- "L18_6_3" for one six-level factors and six three-level factors

`randomize` logical value ('TRUE'/'FALSE') - randomizes the RunOrder of the design. By default `randomize` is set to 'TRUE'.

`replicates` Integer giving the number of replicates.

Details

an overview of possible taguchi designs is possible with `taguchiChoose`.

Value

`taguchiDesign` returns an object of class `taguchiDesign`.

Note

For an example in context which shows the usage of the function `taguchiDesign` please read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org/html/Improve.html`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

Box, Bisgard S.G.E.P.; Fung, C. A.: An explanation and critique of taguchis contributions to quality engineering. In: Quality and Reliability Engineering International (1988)

See Also

`facDesign` for 2^k factorial designs
`rsmDesign` for response surface designs
`fracDesign` for fractional factorial design
`pbDesign` for response surface designs
`gageRRDesign` for gage designs
`http://www.r-qualitytools.org/html/Improve.html`

Examples

```
tdo = taguchiDesign("L9_3")
values(tdo) = list(A = c("material 1","material 2","material 3"),
                   B = c(29, 30, 35))
names(tdo) = c("Factors", "Are", "Documented", "In The Design")
response(tdo) = rnorm(9)
summary(tdo)
effectPlot(tdo)
```

---

taguchiDesign-class *Class "taguchiDesign"*

Description

taguchiDesign Class

Objects from the Class

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

Slots

`name`: Object of class `"character"` ~~

`factors`: Object of class `"list"` ~~

`design`: Object of class `"data.frame"` ~~

`designType`: Object of class `"character"` ~~

`replic`: Object of class `"data.frame"` ~~

`response`: Object of class `"data.frame"` ~~

`Type`: Object of class `"data.frame"` ~~

`block`: Object of class `"data.frame"` ~~

`runOrder`: Object of class `"data.frame"` ~~

`standardOrder`: Object of class `"data.frame"` ~~

`desireVal`: Object of class `"list"` ~~

`desirability`: Object of class `"list"` ~~

`fits`: Object of class `"data.frame"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("taguchiDesign")
```

taguchiFactor-class *Class "taguchiFactor"*

Description

taguchiFactor Class

Objects from the Class

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

Slots

`values`: Object of class `"ANY"` ~~

`name`: Object of class `"character"` ~~

`unit`: Object of class `"character"` ~~

`type`: Object of class `"character"` ~~

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

Examples

```
showClass("taguchiFactor")
```

tolerance-methods *Get and set methods*

Description

Get and set the `tolerance` for an object of class `gageRR`.

Usage

```
## S4 method for signature 'gageRR'
tolerance(x)
## S4 replacement method for signature 'gageRR'
tolerance(x) <- value
```

Arguments

`x`	a 'gageRR' object
`value`	data.frame or vector

Methods

`signature(objectc = "gageRR")` Get and set the `tolerance` for an object of class `gageRR`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

`gageRRDesign`
http://www.r-qualitytools.org

Examples

```
x = gageRRDesign(Operators = 3, Parts = 10, Measurements = 3)
#default 6tolerance
tolerance(x)
#100 units
tolerance(x) = 100
tolerance(x)
```

---

types-methods *Get and set methods*

Description

Get and set the `types` for the factors in an object of class `facDesign`.

Usage

```
## S4 method for signature 'facDesign'
types(x)
## S4 replacement method for signature 'facDesign'
types(x) <- value
```

Arguments

x	a `facDesign` object
value	data.frame or vector

Methods

`signature(x = "facDesign")` Get and set the `types` for the factors in an object of class `facDesign`.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

factors
lows
highs
types
http://www.r-qualitytools.org

Examples

```
#NA in response column
fdo = fracDesign(k = 3)
summary(fdo)
types(fdo) = c("numeric","numeric", "factor" )
names(fdo) = c("Time", "Temperature", "Catalyst")
summary(fdo)
```

---

units-methods *Get and set methods*

Description

Get and set the units for the factors in an object of class facDesign, etc.

Usage

```
## S4 method for signature 'facDesign'
units(x)
## S4 replacement method for signature 'facDesign'
units(x) <- value
```

Arguments

x	a facDesign or mixDesign object
value	data.frame or vector

Methods

signature(x = "facDesign") Get and set the units for the factors in an object of class facDesign.

signature(x = "mixDesign") Get and set the units for the factors in an object of class mixDesign.

signature(x = "taguchiDesign") Get and set the units for the factors in an object of class taguchiDesign.

signature(x = "pbDesign") Get and set the units for the factors in an object of class taguchiDesign.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

See Also

factors
lows
highs
types
http://www.r-qualitytools.org

Examples

```
#NA in response column
fdo = fracDesign(k = 2)
summary(fdo)
units(fdo) = c("min","C")
names(fdo) = c("Time", "Temperature")
summary(fdo)
```

---

values-methods *Get and Set methods*

Description

Methods for function values

Usage

```
## S4 method for signature 'taguchiDesign'
values(object)
## S4 replacement method for signature 'taguchiDesign'
values(object) <- value
## S4 method for signature 'pbDesign'
values(object)
## S4 replacement method for signature 'pbDesign'
values(object) <- value
```

Arguments

object	a taguchiDesign or a pbDesign object
value	a list of values

Methods

signature(object = "taguchiDesign") Get and set the values for the factors in an object of class taguchiDesign.

signature(object = "pbDesign") Get and set the values for the factors in an object of class pbDesign.

See Also

http://www.r-qualitytools.org

`weibull3` *The Weibull Distribution (3 Parameter)*

Description

Density function, distribution function and quantile function for the Weibull distribution.

Usage

```
dweibull3(x, shape, scale, threshold)
pweibull3(q, shape, scale, threshold)
qweibull3(p, shape, scale, threshold, ...)
```

Arguments

`x, q`	vector of quantiles
`p`	vector of probabilities
`shape`	shape parameter by default 1
`scale`	scale parameter by default 1
`threshold`	threshold parameter by default 0
`...`	Arguments that can be passed into `uniroot`.

Details

The Weibull distribution with 'scale' parameter alpha, 'shape' parameter c and 'threshold' parameter zeta has density given by

f(x) = (c/alpha) (((x-zeta)/alpha)^(c-1)) exp(-((x-zeta)/alpha)^c)

The cumulative distribution function is given by

F(x) = 1 - exp(-((x-zeta)/alpha)^c)

Value

`dweibull3` gives the density, `pweibull3` gives the distribution function and `qweibull3` gives the quantile function.

Note

`qweibull3` calls `uniroot` for each value of the argument 'p'. The solution is consequently not exact; the ... can be used to obtain a more accurate solution if necessary.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>
Etienne Stockhausen <stocdarf@mailbox.tu-berlin.de>

References

Johnson, L., Kotz, S., Balakrishnan, N. (1995) Continuous Univariate Distributions-Volume 1, 2nd ed. New York: John Wiley & Sons.

See Also

uniroot

Examples

```
#Simple Example
dweibull3(x=1,scale=1,shape=5,threshold=0)
temp=pweibull3(q=1,scale=1,shape=5,threshold=0)
temp
qweibull3(p=temp,scale=1,shape=5,threshold=0)

#Visualized Example
#prepare screen
#dev.new()
#split.screen(matrix(c(0,0.5,0,1, 0.5,1,0,1),byrow=TRUE,ncol=4))
##generate values
#x=seq(0,3,length=1000)
##plot different density functions
#screen(1)
#plot(x,y=dweibull3(x,threshold=0,shape=0.5,scale=1),col="green",
#     xlim=c(0,2.5),ylim=c(0,2.5),type="l",lwd=2,xlab="x",
#     ylab="f(x)",main="Density Function of Weibull-Distribution")
#lines(x,y=dweibull3(x,threshold=0,shape=1,scale=1),lwd=2,col="red")
#lines(x,y=dweibull3(x,threshold=0,shape=1.5,scale=2),lwd=2,col="blue")
#lines(x,y=dweibull3(x,threshold=0,shape=5,scale=1),lwd=2,col="orange")
##add legend
#legend("topright",legend=c(expression(paste(alpha, " = 1 ")*
#       paste(c, " = 0.5 ")*paste(zeta," = 0")),
#       expression(paste(alpha, " = 1 ")*paste(c, " = 1 ")*
#       paste(zeta," = 0")),expression(paste(alpha, " = 2 ")*
#       paste(c, " = 1.5 ")*paste(zeta," = 0")),
#       expression(paste(alpha, " = 1 ")*paste(c, " = 5 ")*
#       paste(zeta," = 0"))),col=c("green","red","blue","orange"),
#       text.col="black",lwd=2,bty="0",inset=0.04)
#abline(v=0,lty=2,col="grey")
#abline(h=0,lty=2,col="grey")
##plot different distribution functions
#screen(2)
#plot(x,y=pweibull3(x,threshold=0,shape=0.5,scale=1),col="green",
#     xlim=c(0,2.5),ylim=c(0,1),type="l",lwd=2,xlab="x",ylab="F(x)",
#     main="Cumulative Distribution Function of Weibull-Distribution")
#lines(x,y=pweibull3(x,threshold=0,shape=1,scale=1),lwd=2,col="red")
#lines(x,y=pweibull3(x,threshold=0,shape=1.5,scale=2),lwd=2,col="blue")
#lines(x,y=pweibull3(x,threshold=0,shape=5,scale=1),lwd=2,col="orange")
####add legend
#legend("bottomright",legend=c(expression(paste(alpha, " = 1 ")*
#       paste(c, " = 0.5 ")*paste(zeta," = 0")),
```

```
#       expression(paste(alpha, " = 1 ")*paste(c, " = 1 ")*
#       paste(zeta," = 0")),expression(paste(alpha, " = 2 ")*
#       paste(c, " = 1.5 ")*paste(zeta," = 0")),
#       expression(paste(alpha, " = 1 ")*paste(c, " = 5 ")*
#       paste(zeta," = 0"))),col=c("green","red","blue","orange"),
#       text.col="black",lwd=2,bty="0",inset=0.04)
#abline(v=0,lty=2,col="grey")
#abline(h=0,lty=2,col="grey")
#close.screen(all=TRUE)
```

`whiskersPlot` *Function to create Whiskers Charts*

Description

In a Whiskers Chart, the high and low data values and the average (median) by part-by-operator are plotted to provide insight into the consistency between operators, to indicate outliers and to discover part-operator interactions. The Whiskers Chart reminds of boxplots for every part and every operator.

Usage

```
whiskersPlot(x, main, xlab, ylab, col, ylim, legend=TRUE, ...)
```

Arguments

`x`	needs to be an object of class `gageRR`.
`main`	a main title for the plot
`xlab`	a label for the x axis
`ylab`	a label for the y axis
`col`	plotting color
`ylim`	the y limits of the plot
`legend`	a logical value specifying whether a legend is plotted automatically. By default `legend` is set to 'TRUE'. If the argument `legend` is set to 'FALSE' an individual legend can be added by using the function `legend` afterwards.
`...`	arguments to be passed to methods, such as graphical parameters (see `par`).

Details

Graphical parameters such as `col` or `pch` can be given as single characters or as vectors containing characters or number for the parameters of the individual operators.

Note

Please do read the vignette for the package `qualityTools` at `http://www.r-qualitytools.org`.

Author(s)

Thomas Roth: <thomas.roth@tu-berlin.de>
Etienne Stockhausen: <stocdarf@mailbox.tu-berlin.de>

References

The idea of the plot and the example given by `example(whiskersPlot)` are out of:

- CHRYSLER Group LLC; FORD Motor Company; GENERAL MOTORS Corporation: Measurement System Analysis (MSA),
p.111, 4rd ed. Southfield: AIAG, 2010.

See Also

gageRR
par
http://www.r-qualitytools.org

Examples

```
# create gageRR-object
gdo = gageRRDesign(Operators = 3, Parts = 10, Measurements = 3,
                   randomize = FALSE)
# vector of responses
y = c(0.29,0.08, 0.04,-0.56,-0.47,-1.38,1.34,1.19,0.88,0.47,0.01,0.14,-0.80,
      -0.56,-1.46, 0.02,-0.20,-0.29,0.59,0.47,0.02,-0.31,-0.63,-0.46,2.26,
      1.80,1.77,-1.36,-1.68,-1.49,0.41,0.25,-0.11,-0.68,-1.22,-1.13,1.17,0.94,
      1.09,0.50,1.03,0.20,-0.92,-1.20,-1.07,-0.11, 0.22,-0.67,0.75,0.55,0.01,
      -0.20, 0.08,-0.56,1.99,2.12,1.45,-1.25,-1.62,-1.77,0.64,0.07,-0.15,-0.58,
      -0.68,-0.96,1.27,1.34,0.67,0.64,0.20,0.11,-0.84,-1.28,-1.45,-0.21,0.06,
      -0.49,0.66,0.83,0.21,-0.17,-0.34,-0.49,2.01,2.19,1.87,-1.31,-1.50,-2.16)
 # appropriate responses
response(gdo)=y
# perform and gageRR
gdo=gageRR(gdo)

whiskersPlot(gdo)
```

---

whiskersPlot-methods *Methods for Function whiskersPlot*

Description

Methods for function `whiskersPlot`

Methods

`signature(object = "gageRR")` function to create a whiskers plot for an object of class gageRR.

See Also

http://www.r-qualitytools.org

`wirePlot` *3D Plot*

Description

Creates a wireframe diagramm for an object of class `facDesign`.

Usage

```
wirePlot(x, y, z, data = NULL, xlim, ylim, zlim, main, xlab, ylab, border, sub,
        zlab, form = "fit", phi, theta, ticktype, col = 1, steps, factors, fun,
        plot)
```

Arguments

`x`	name providing the Factor A for the plot.
`y`	name providing the Factor B for the plot.
`z`	name giving the Response variable.
`data`	needs to be an object of class `facDesign` and contains the names of x,y,z.
`xlim`	vector giving the range of the x-axis.
`ylim`	vector giving the range of the y-axis.
`zlim`	vector giving the range of the y-axis.
`main`	an overall title for the plot: see `title`.
`xlab`	a title for the x axis: `title`.
`ylab`	a title for the y axis: `title`.
`zlab`	a title for the z axis: `title`.
`border`	numerical value or character string giving the color of the line drawn around the surface facets.
`sub`	a sub title for the plot: `title`.
`form`	a character string or a formula with the syntax "y~ x+y + x*y". If `form` is a character it has to be one out of the following: • "quadratic" • "full" • "interaction" • "linear" • "fit" "fit" takes the formula from the fit in the facDesign object fdo. Quadratic or higher orders should be given as I(Variable^2). By default `form` is set as "fit".

phi	numerical value. Angle (in degree) defining the viewing direction. Phi gives the colatitude. By default phi is set to '30'.
theta	numerical value. Angle (in degree) defining the viewing direction. Theta gives the azimuthal direction. By default theta is set to '-30'.
ticktype	graphical parameter. Character string specifying wheter the ticks on the axes should be "simple" or "detailed" (default).
col	a predefined (1, 2, 3 or 4) or self defined colorRampPalette or color to be used (i.e. "red").
steps	number of grid points per factor. By default steps = 25.
factors	list of 4th 5th factor with value i.e. factors = list(D = 1.2, E = -1), if nothing is specified values will be the mean of the low and the high value of the factors.
fun	function to be applied to z "desirability".
plot	logical value. If 'TRUE' (default) a plot is created (most likely to disappear with time)

Details

This function can be used to display the desirability of each response by specifying fun = "desirability" or the fun = "overall" (i.e. composed) desirability of all responses. The required desirabilities can be set using desires.

Value

The function wirePlot() returns an invisible list with the following entries:

- x - locations of grid lines for x at which the values in z are measured
- y - locations of grid lines for y at which the values in z are measured
- z - a matrix containing the values of z to be plotted

Note

For an example in context which shows the usage of the function wirePlot() to an object of class facDesign, please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

See Also

contourPlot
filled.contour
persp
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#create a response surface design and assign random data to response y
fdo = rsmDesign(k = 3, blocks = 2)
response(fdo) = data.frame(y = rnorm(nrow(fdo)))

par(mfrow = c(3,2))

#I - display linear fit
wirePlot(A,B,y, data = fdo, form = "linear")

#II - display full fit (i.e. effect, interactions and quadratic effects
wirePlot(A,B,y, data = fdo, form = "full")

#III - display a fit specified before
fits(fdo) = lm(y ~ B + I(A^2) , data = fdo)
wirePlot(A,B,y, data = fdo, form = "fit")

#IV - display a fit given directly
wirePlot(A,B,y, data = fdo, form = "y ~ A*B + I(A^2)")

#V - display a fit using a different colorRamp
wirePlot(A,B,y, data = fdo, form = "full", col = 2)

#V - display a fit using a self defined colorRamp
myColour = colorRampPalette(c("green", "gray","blue"))
wirePlot(A,B,y, data = fdo, form = "full", col = myColour)
```

`wirePlot3` *function to create a ternary plot (3D wire plot) for an object of class mixDesign*

Description

This function creates a ternary plot for mixture designs (i.e. object of class `mixDesign`).

Usage

```
wirePlot3(x, y, z, response, data = NULL, main, xlab, ylab, zlab,
          form = "linear", phi, theta, col = 1, steps, factors)
```

Arguments

`x`	factor 1 of the `mixDesign` object.
`y`	factor 2 of the `mixDesign` object.
`z`	factor 3 of the `mixDesign` object.
`response`	the response of the `mixDesign` object.

data	the mixDesign object from which x,y,z and the response are taken.
main	giving an overall title for the plot: see title.
xlab	giving a title for the x axis : see title.
ylab	giving a title for the y axis : see title.
zlab	giving a title for the z axis : see title.
form	a character string or a formula with the syntax "y ~ A + B +C". If form is a character string it has to be one out of the following: • "linear" • "quadratic" • "fullCubic" • "specialCubic" How the form influences the output is described in the reference listed below. By default form is set as "linear".
phi	numerical value specifying the angle (in degree) through which the plot is rotated about an imagined horizontal line. By default phi is set as '30'.
theta	numerical value specifying the angle (in degree) through which the plot is rotated about an imagined vertical line. By default phi is set as '30'.
col	a predefined (1, 2, 3 or 4) or self defined colorRampPalette.
steps	resolution of the plot (number of rows for the square matrix), also number of grid points per factor. By default steps = 25.
factors	list of factors for categorizing with setting in case there are more than 3 factors (not yet implemented).

Value

wirPlot3 returns an invisible matrix containing the response values as NA's and numerics.

Note

This method is still under construction!!
For an example in context which shows the usage of the function wirePlot3() to an object of class mixDesign, please read the vignette for the package qualityTools at http://www.r-qualitytools.org/html/Improve.html.

Author(s)

Thomas Roth <thomas.roth@tu-berlin.de>

References

- CORNELL: Experiments with Mixtures - 3rd Ed. New Jersey: Wiley, 2011.
- MYERS, Raymond H.; MONTGOMERY, Douglas C.; ANDERSON-COOK, Christine M.: Response Surface Methodology. New York: WILEY ,2009.

See Also

contourPlot3
http://www.r-qualitytools.org/html/Improve.html

Examples

```
#yarn elongation example p.564 Response Surface Methodology
mdo = mixDesign(3,2, center = FALSE, axial = FALSE, randomize = FALSE,
                replicates  = c(1,1,2,3))
names(mdo) = c("polyethylene", "polystyrene", "polypropylene")
units(mdo) = "percent"
elongation = c(11.0, 12.4, 15.0, 14.8, 16.1, 17.7, 16.4, 16.6, 8.8, 10.0, 10.0,
               9.7, 11.8, 16.8, 16.0)
response(mdo) = elongation

dev.new(14,14); par(mfrow = c(2,2))
wirePlot3(A, B, C, elongation, data = mdo, form = "linear")
wirePlot3(A, B, C, elongation, data = mdo, form = "quadratic", col = 2)
wirePlot3(A, B, C, elongation, data = mdo,
          form = "elongation ~ I(A^2) - B:A + I(C^2)", col = 3)
wirePlot3(A, B, C, elongation, data = mdo, form = "quadratic",
          col = colorRampPalette(c("yellow", "white", "red")))
```

[-methods *Methods for Function [in Package base*

Description

Methods for function [in Package base.

Methods

signature(x = "ANY") ANY

signature(x = "distrCollection") object of class distrCollection.

signature(x = "facDesign") object of class facDesign.

signature(x = "steepAscent") object of class steepAscent.

signature(x = "gageRR") object fo class gageRR.

signature(x = "nonStructure") nonStructure

See Also

http://www.r-qualitytools.org

Index