Package 'RcmdrPlugin.DoE'

October 7, 2023

Version 0.12-5

Date 2023-10-07

Title R Commander Plugin for (Industrial) Design of Experiments

Maintainer Ulrike Groemping <ulrike.groemping@bht-berlin.de>

Depends R (>= 2.10.0), utils, DoE.base (>= 0.22-8), FrF2 (>= 1.2-10), DoE.wrapper (>= 0.8-6), tcltk

Imports Rcmdr, RcmdrMisc

Suggests FrF2.catlg128

RcmdrModels rsm

Description Provides a platform-independent GUI for design of experiments.
The package is implemented as a plugin to the R-Commander, which is a more general graphical user interface for statistics in R based on tcl/tk.
DoE functionality can be accessed through the menu Design that is added to the R-Commander menus.

License GPL (>= 2)

URL https://prof.bht-berlin.de/groemping/DoE/

Author Ulrike Groemping [aut, cre], Fox John [ctb]

NeedsCompilation no

Repository CRAN

Date/Publication 2023-10-07 09:10:02 UTC

R topics documented:

`RcmdrPlugin.DoE-package`

R-Commander plugin package that implements design of experiments facilities from packages DoE.base, FrF2 and DoE.wrapper into the R-Commander

---

Description

The plugin adds a Design menu to the R-Commander. From this, various types of experimental plans can be generated, imported, augmented with responses or further runs, and - in a limited way - analysed. Of course, other R-Commander functionality can also be used on the designs.

The plugin is currently a beta version. Reports on bugs or inconveniences are very welcome.

Prior Remark for Windows users

The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

Suggests field is not meant as a suggestion

For ensuring valid checks of package **RcmdrPlugin.DoE**, it was necessary to include package **FrF2.catlg128** under suggested packages. Note that most users will not need that package and should not install it.

Beta Version!

RcmdrPlugin.DoE is currently in beta shape. The calculation functions have been reasonably tested within the underlying calculation packages. Usage of the menus has of course also been tested, but there are very likely untested scenarios where something does not work properly. Bug reports are therefore invited!

The logged commands are currently aiming to be correct but not to be didactic, since they have been generated with the purpose to obtain a valid result from the inputs with as little programming effort as possible. They often contain many superfluous inputs, since default inputs could also be omitted. For example, while `FrF2(8,4)` would produce a regular fractional factorial design with 8 runs, 4 factors and default factor names and levels, the respective logged command is three rows long.

The help files will have to be improved upon. Suggestions are welcome (where did you get stuck?).

Current users of this package have to accept acting as beta testers. It is particularly important to never work with the one and only data original, but have the valuable experimental data themselves stored in a safe place.

Covered methodology and planned extensions

Full factorial designs, orthogonal main effects designs, regular and non-regular 2-level fractional factorial designs, central composite and Box-Behnken designs, latin hypercube samples, and simple D-optimal designs can currently be generated from the GUI.

Extensions to cover further latin hypercube designs as well as more advanced D-optimal designs (with blocking) are planned for the future.

RcmdrPlugin.DoE implements functionality from packages **DoE.base**, **FrF2**, **DoE.wrapper** (which in turn implements functionality from several other packages), and also from package **rsm**. All package allow more flexibility through command line programming than through using the GUI. Nevertheless, usage of the GUI should be sufficient for reasonable creation and analysis of results at least in standard situations.

There is a tutorial (see references) with example data; it is, however, far from being comprehensive. For experienced users of other DoE Software, the menus may be reasonably self-explanatory. Please feel free to contact me with issues regarding the documentation; it is very informative and will help improve documentation, if I understand where users get stuck. However, I am of course NOT a help desk for experimentation or issues with using R or the R-Commander.

Author(s)

Ulrike Groemping

References

Groemping, U. (2011). Tutorial for designing experiments using the R package **RcmdrPlugin.DoE**. Report 4/2011, Reports in Mathematics, Physics and Chemistry, Department II, Beuth University of Applied Sciences Berlin.
The report itself: http://www1.bht-berlin.de/FB_II/reports/Report-2011-004.pdf.
Example data: https://prof.bht-berlin.de/fileadmin/prof/groemp/downloads/Tutorial_DoEGUI_exampleData.zip.

See Also

See also DoE.base-package, FrF2-package, DoE.wrapper-package for the packages behind this R-Commander plugin.

DoEGlossary *Glossary for DoE terminology as used in RcmdrPlugin.DoE*

Description

This glossary is intended for users who have heard about the concepts of experimental design, orthogonality, and so forth but would nevertheless like to look up some of the terminology.

It can of course not replace a good training or a good book on experimental design.

Glossary

star A $\star$ indicates a paragraph or section, that requires more statistical understanding than necessary for successful application of experimentation and can be skipped by readers who do not want to go in too deeply. Do not confuse with resolution III*.

2fi short for 2-factor interaction: The effect of a factor depends on the level of a second factor, e.g. in a sled test example reported in Grove and Davis, 1992 (p.25 f.), speed of airbag inflation shows almost no effect on chest acceleration for the smaller airbag in the experiment, while a higher speed strongly decreases chest acceleration for the larger air bag. One can also look at this example the other way round: For high speed, increasing the airbag size reduces chest acceleration, while for low speed, increasing the airbag size increases chest acceleration.

2-level fractional factorial A 2-level array is a row-by-column table, each column of which contains two distinct values only. A "2-level orthogonal array" is a 2-level array with particular balance properties: If for each column, one value is denoted by -1 and the other by +1, the element wise product of any pair of two columns always sums to 0. This provides an easy means of checking for orthogonality of any 2-level array. Balance of the example plan can be expressed verbally as follows: Each possible level combination of two factors in the plan occurs equally often, e.g. short float rod/small tank, long float rod/small tank, short float rod/large tank, long float rod/large tank each occur twice. A "regular 2-level fractional factorial" is a special 2-level orthogonal array, for which all effects are either fully aliased or not aliased at all (no partial aliasing). A "non-regular 2-level fractional factorial" is a 2-level orthogonal array, for which partial aliasing occurs.

aberration $\star$Roughly, aberration is a criterion for comparing two designs of the same resolution: the design is better if aberration is smaller. Aberration measures the extent of the most severe aliasing: The larger the aberration, the more aliasing of the most severe type (resolution III: main effect with 2fi, resolution IV: 2fi with 2fi (and main effect with 3-factor interaction, not listed in tables of CATALOGUE.XLS)) is present in the design. Technically, aberration is the number of shortest words.

active effect An effect is said to be "active", if it does have an influence on the response, and inactive otherwise.

aliasing Two effects are said to be completely aliased, if they cannot be seperated from each other by the design. An extreme example would be running an experiment on tires such that large tires of material 1 are compared to small tires of material 2. As there are neither large tires of material 2 nor small tires of material 1, the main effects of tire size and material are completely aliased. (Designs with aliasing between main effects would be resolution II and are not implemented in package **FrF2**. If one really intends to alias two main effects with each other, one can always combine them into one, e.g. in the above example the combined factor could be called "tire".) Likewise, aliasing between other effects like between two 2fis or between a main effect and a 2fi means that the effects cannot be separated from each other by the design. More technically, in the present context, aliasing refers to structurally determined identity between several effects that arises in fractional factorials.

Design "Design" is the shortest term for an experimental plan or experimental layout.

Degrees of freedom (df) The data contain a certain number n of independent pieces of information, e.g. in case of a designed experiment we have the number of experimental runs (or a multiple of this number in case of proper replication). In any case, one needs to estimate the overall average from the responses of the experiment so that there are n - 1 independent pieces of information left. These are the "degrees of freedom" to start with. We then introduce factors

the effects of which we want to estimate. Each effect has degrees of freedom (df) associated with it, e.g. the df for the main effect of a 2-level factor are 1 (=2-1), while the df for the main effect of a 4-level factor would be 3 (=4-1). In total, the sum of the df of all effects that we estimate cannot exceed the df that we have gained from collecting data. If we subtract the df for each effect that is estimated from the overall df, the remaining number are the so-called error degrees of freedom or residual df. In DoE, we often "saturate" models, i.e. we assign all available degrees of freedom to effects so that residual df are 0. In this case, the only way of assessing whether or not an effect is active is by halfnormal plots, full-normal plots, or effects Paretos; "proper" statistical analysis like Analysis of Variance does require some error degrees of freedom.

DoE Design of experiment, also experimental design: The statistical discipline that takes care of planning / designing experiments. Some people also use this expression for the design itself, or for the whole process of running a designed experiment.

D-optimal A D-optimal design minimizes the volume of the confidence ellipsoid for the model coefficients (by maximizing the determinant of the model information matrix); requires that a model is specified in advance

effect The term "effect" stands for all kinds of impact that a factor or a combination of factors can have on the response. Effect summarizes main effects, 2fis, and even higher-order interactions in one expression. It is also customary to split main effects of multilevel factors into several contributions, e.g. the main effect of a 3-level factor may be split into a linear and a quadratic effect. Strictly speaking, the effects are the theoretically "true" impacts on the response, that cannot be observed in an experiment, as there always is experimental error (in CAE experiments, there is often no random error, but there is model error instead). Sometimes, the term effect is also used for the estimated effects, but one should always bear in mind that estimated effects are subject to experimental error and / or model error and / or possible aliasing, so that they are indicative of true effects, but need to be verified (e.g. by confirmation runs, cf. Grove and Davis, 1992, p.83).

estimate When running an experiment (setting aside CAE), the response is subject to measurement error and many other sources of random or systematic deviation from `the truth`. Therefore, effects will not be known after the experiment, but can only be `estimated`. For estimating a main effect of a 2-level factor, for example, one has to calculate the difference between the average response for the two levels. Using the term "estimating" instead of "calculating" emphasizes the fact that there is some variation involved in the actually calculated numbers, as using a different part, or measuring on a different machine or under different environmental conditions (or ...) would have returned a different number. With CAE, things are somewhat different: In a CAE experiment, there is often no measurement error (or very little from computational inaccuracies), but there almost certainly is model error, i.e. the CAE model does not perfectly reflect reality. Hence, any CAE experiment yields insights into model reality, but not necessarily into real world reality.

experiment "By an experiment we mean any test, evaluation or trial designed to evaluate a change, which requires the collection of measurements (data) so that a judgment (prediction) on the performance of the siystem can be formed, and a decision can be made as to whether further re-design (with further experimentation and prediction) is necessary." (cited from Grove and Davis, 1992, p.1) In this glossary, the term is used in a more restricted sense for a statistically designed experiment, in which experimentation is done following an experimental plan in a number of factors, each of which has a predetermined set of levels.

factor A factor is a characteristic of the system under study that is to be systematically varied over a predetermined set of levels in an experiment.

fractional factorial A regular fractional factorial is a design obtained from a full factorial in fewer factors by deliberately assigning additional factors to particular interactions (i.e. by choosing generating contrasts), thereby introducing perfect aliasing between some effects. One can also start from the full factorial in all factors, and state that a regular fractional factorial is a design obtained by taking a regular (!) fraction of the full factorial only. This notion is closer to the expression, of course.

A non-regular fractional factorial analogously takes a non-regular fraction of the experiment one would have to run in case of a full factorial.

full factorial A full factorial is a design that consists of every possible combination of levels of all the factors. A full factorial of `k` 2-level factors has `2^k` runs (if unreplicated).

generating contrast A generating contrast or "generator" is needed for generating regular fractional factorials. It is obtained by assigning an additional factor to an interaction column from a full factorial, e.g. ABCDE=I when assigning the additional factor E to the 4-factor interaction ABCD in a 16-run design: As E equals ABCD in this case, the product of ABCD with E yields a constant column of + (=+1), which is denoted by I.

base factors The `k` 2-level factors making up a `2^k` run full factorial design are called the base factors. They can be used for generating a regular 2-level fractional factorial.

Hadamard matrix *A (transposed) Hadamard matrix is a matrix of -1 and +1 such that all columns of the matrix are orthogonal. All orthogonal 2-level arrays can be understood as square Hadamard matrices (when adding a first column of solely +1s). Likewise, any square Hadamard matrix with a first column of solely +1s can be used for creating an orthogonal array for 2-level factors.

half-normal plot The half-normal plot is also called Daniel plot, according to Daniel (1959). The idea is as follows: if all factors have no impact on the response, the estimated effects are random numbers with mean 0 and some variance and should roughly follow a normal distribution. The absolute values of these estimated effects thus follow the so-called half-normal distribution. Now, we can depict the sorted standardized absolute effects against the theoretical quantiles of the standard halfnormal distribution (plotting positions). The plotting positions are chosen such that the effects lie nicely on a line, if they behave like normally distributed random numbers with mean 0. In the context of DoE, one usually hopes that most effects lie on a line through the origin, while a few effects are distinctly larger than the continuation of the line. Those effects that stick out well above (or to the right of) the line are considered "active", as they are larger than what one would expect from the size of random variation within the experiment.

If the eye-balled line does not point towards the origin, this often indicates presence of an outlier.

Caution: Occasionally, although some effects stick out dramatically, the largest effect almost lies on the eye-balled line. It is unreasonable to argue that the largest effect is inactive but smaller effects are active. Whenever a particular effect is considered active, all larger (in absolute value) effects have to be considered active as well.

Interpretation of half-normal plots, in case of non-ideal appearance, requires careful thinking, looking for outliers, ... Half-normal plots should be used with great care or not at all in experiments with few runs only (e.g. 8 run experiments). If replications are present, it can be better to use analysis of variance (provided the replications of one and the same run

represent all kinds of random variation that may be encountered between runs) than to use half-normal plots. Whenever half-normal plots are used, one should include all effects - also those associated with empty columns - in the plot. Therefore, screening designs created by RcmdrPlugin.DoE include those empty columns per default.

hereditary principle The idea that an interaction is less likely to be important if one or both of the involved main effects are inactive, is called the "hereditary principle". In case of aliasing between two 2fis, if the interaction column that hosts the two aliased 2fis, is estimated to have a considerable effect, the hereditary principle can sometimes be used to resolve the issue which of the two 2fis is responsible for this finding: If one of the two 2fis involves a main effect that is not estimated to be important, while for the other both the involved main effects are estimated to be important, the hereditary principle suggests that the 2fi with two "parents" is the likely responsible. Nevertheless, it is possible that a 2fi is truly important, although only one (or even none) of the two involved main effects is active. Therefore, in case of ambiguity it is always a good idea to do confirmation runs or follow-up experiments.

I I denotes a constant column of + (or +1, depending on notation) of appropriate length. Therefore, the letter `I` is not used for factor names (neither is `i`).

interaction It is possible that the main effect of a factor `B` given the first level of a factor `A` is substantially different from the main effect of factor `B` given the second level of factor `A` . In this case, it is said that there is a 2-factor interaction between factors `A` and `B`. One can extend the concept of interactions to multi-factor interactions (also called higher-order interactions). In practice, however, interactions between more than 2 factors are very rarely considered important and are very difficult to interpret.

latin hypercube Computer experiments usually do not benefit from repeating runs, because the results will be virtually the same. Therefore, one tries to use designs that do not produce replications when projected into lower-dimensional space (e.g. for modelling with a few factors that turn out to be relevant only). Latin hypercube designs or latin hypercube samples have as many levels for each factor as there are runs. Usually, one tries to choose level combinations such that the design is "space-filling", i.e. fills the experimental space with points so that the experiment provides information everywhere in the experimental region.

level A level is one of the predetermined settings over which a factor is varied (usually very few, in package **FrF2** mostly two, sometimes also four (not yet implemented). For example, the factor temperature could have the two levels 0 and 20 degree Celsius. Determining the levels of a factor is a very important step in planning an experiment, as the results might be very different, when looking at e.g. -15 and 35 degree Celsius instead of 0 and 20.

linear effect, quadratic effect For a quantitative factor, the response can be modeled as a linear function of the size of the factor, i.e. denoting the response by `y` and the factors value by `x`, we can write `y=a+bx`. If the factor has 2 levels only in the experiment, we can estimate `b` from the data, but it is impossible to check, whether the factors effect on the response is well-approximated by a linear function. If the factor has more than 2 levels, we can find out, whether a linear function is appropriate, e.g. by additionally fitting a quadratic effect, `y=a+bx+cx^2`. In 2-level experiments with quantitative factors, it is also customary to incorporate a few so-called center points, which are in the middle between extremes for all factors. By comparing their average to the average results from the other points, linearity can be checked.

main effect The main effect for a 2-level factor is defined as the (true) difference between the (true) responses for the two levels of the factor. It is estimated by the difference between the averages of measured responses for the two levels of the factor. Often, the term "main effect" is used

for the true unobservable difference as well as for the estimated effect from an experiment. It is, however, useful to be aware that there is a difference, that is due to measurement error, aliasing, ...

The main effect for factors with more than two levels is made up of several components or degrees of freedom (df), where df = (levels - 1). For a quantitative 3-level factor (2 df), one often talks about the linear and the quadratic effect, which can in case of equi-distant levels be estimated by the difference between the average responses of the high and the low level (linear effect) and by the difference of the average response of the middle level from the average response of the two outer levels (quadratic effect). Likewise, one can discuss linear, quadratic, and cubic effects for 4-level factors and so on. For qualitative factors - e.g. type of material, shape of gasket etc. - one possibility is to define a reference category, e.g. the current level, and to formulate the main effect in terms of the difference of the response on each level to the response for the reference category.

Median For obtaining the median of a group of values, first sort the values by size. For an odd number of values, the median is the middle value in the sorted series. For an even number of values, the median is the average of the two middle values of the sorted series. Thus, the median is a value that splits the data into a bottom and a top half.

mixed level orthogonal array An orthogonal array that contains factors with both 2 level and 3 levels is called a "mixed level orthogonal array" or "mixed level array". Such arrays are implemented under menu "General Factorial".

naming convention for factors It is a convention that factors in experimental plans are abbreviated by capitalized letters from the alphabet, where usually the letter `I` is skipped as it has a special meaning in the literature on experimental plans. Once one runs out of capital letters as there are more than 25 factors, conventions are less unique; some people use numbers, others small letters (e.g. default in this software or Minitab), ... If the small letters are also exhausted, this software uses F1, F2, ...

Interactions in 2-level fractional factorials are denoted by the ordered sequence of factor names of the factors involved, e.g. AB denotes the 2fi between factors A and B, BEFH denotes the four factor interaction between the four included factors. These sequences of letters can be used simultaneously for naming the appropriate column in the design and for naming the whole interaction effect.

Plackett-Burman designs Plackett and Burman (1946) introduced 2-level orthogonal arrays that are not obtained as regular fractional factorials. If the number of runs is not a power of 1, these designs have any particular interaction partially aliased with several main effects, so that on the one hand we don't find unconfounded main effects, but on the other hand, no main effect is completely distorted by a single interaction effect. Note that these designs cannot be used for estimating interactions or for constructing 4-level factors from a pair of 2-level factors. However, a few subsequent analyses have been proposed to use these designs for modelling after using them for screening (requires expert knowledge, not included in this software).

Projection properties, projectivity The term "projection properties" refers to the properties of a design, when attention is restricted to a few (seemingly) active factors only, for example in the case of screening designs, where some factors are ruled out as unimportant. A design is said to be of projectivity 3, if it contains at least one full factorial (plus possibly some repeat runs) in any triple of three factors. Analogously, a design is said to be of projectivity 4, if it contains at least one full factorial (plus possibly some repeat runs) in any set of four factors etc. For Plackett-Burman designs, projection properties have been studied in detail by Lin and Draper (1992), Draper and Lin (1995) and Box and Tyssedal (1996). The work on 16 run Hadamard

matrices by Box and Tyssedal (2001) is also based on projection properties. It forms the basis of the 16 run screening designs used per default in this software.

randomize Randomization is a means of trying to avoid effects of unknown factors that may for example change over time. In the DoE context, it is advisable to randomize the run order, as time might introduce unexpected effects that can lead to misleading results. Randomizing is the default for all experiments, except if hard-to-change factors are declared.

Since experimentation without randomization runs great risks particularly in case of very systematic run orders like the one used for hard-to-change factors, one should normally strive to run an experiment in randomized order. However, if the experiment becomes *much* more difficult this way, it may be acceptable to deviate from this rule.

replication/repetition Both replication and repetition stand for repeating a measurement with exactly the same configuration of factor levels. The difference between these two concepts is as follows (cf. Grove and Davis, 1992, p.155/p.198): Replication should ideally reflect all possibilities of experimental error; for example, if installing a part in the system may cause variation, replication of a run should not just install a part once and measure the response twice, but should at least re-install the part before re-measuring. Better even, a different part with the same configuration should be installed for also capturing part-to-part variation.

In many cases, it may be more useful to run a genuine 16-run design instead of twice replicating an 8-run design. Cost may be a driver of using replication instead of a truly larger design, and the desire for a good estimate of experimental variability may be another driver for replication.

Repetition mainly refers to production and measurement processes. It should not introduce experimental error but rather reflect just the errors that one can expect to occur in normal production or measurement. For example, if some factors refer to machine setup for producing some parts, repetition means that the machine is setup once, and several parts are produced under the same setup.

Per default, this software considers proper replications. These are randomized in blocks, such that the first replicate of each run in randomized order is conducted before the second replication of any run starts. Under option "repeat.only", repeated runs occur all together in a row, assuming that they are repeated measurements.

resolution Resolution is the smallest possible number of factors that appears in a pair of aliased effects, e.g. if there is an alias between a main effect and a 2fi, the resolution is III (resolution is usually denoted in Roman numbers). More technically speaking, resolution is the length of the shortest word.

The literature sometimes uses resolution `III*` for designs of resolution III with specific favourable properties. These will be made use of in the future. They have not been implemented yet.

response In an experiment, the response is a quantity that quantifies the performance of the system under investigation. It can be a measurement itself, a quantity calculated from several measurements, ...

response surface design Experimental plans that are meant to model the response as a smooth function of a set of quantitative factors - often second order, i.e. including quadratics of each factor and linear by linear interactions between each pair of factors - are called response surface designs. Such designs can be created by augmenting appropriate regular 2-level fractional factorial designs with center and star points.

robustness experiment A robustness experiment (or robustness study) is dedicated to finding factors (so-called control factors) that can be used in designing the product (or process) for making the response insensitive (=robust) to so-called noise factors that cannot be influenced by

the engineer (like customer usage conditions, outside temperature, ...). For analysis of data from robustness experiments (setting aside the famous but questionable S/N ratio approach, cf. e.g. Grove and Davis 1992, Chapter 11), 2fis between noise factors and control factors are extremely important.

run Any configuration of factor levels for which a response is recorded is called a run. Some people use the term "run" for distinct configurations only so that an 8-run design with 2 replications would still be called an 8-run design, while other people would call such a design a 16-run design. This software adopts the first view.

run order Run order refers to the temporal order in which the runs in an experiment are arranged. Usually, the run order should be randomized. However, it is nevertheless often helpful for thinking about the experimental structure, if one can look at the experiment in standard order. Therefore, the attribute run.order of all designs allows to switch back and forth between randomized and standard order.

clear "clear" refers to the absence of aliasing. It is customary in the literature to talk of effects of being clear if they are neither aliased with a main effect nor with a 2fi. They may be aliased with higher-order interactions.

WLP *short for "Word length pattern", cf. "word"

word *For regular 2-level fractional factorial designs, any product of factors that resolves to `I` is termed a word. For example, if the product CDEF yields a constant column of "+1", CDEF is a word of length four (as there are four factors involved). The number of (non-trivial) words for a design is `2^g-1` with g denoting the number of generating contrasts.

For irregular designs and mixed-level designs, it is possible to determine the number of generalized words (usually of lengths 3 and 4), by summing up the scalar products of the intercept column with all length 3 or 4 combinations of orthogonal effect columns for three or four different factors (Xu and Wu 2001).

Yates order Frank Yates is a renowned experimenter, who found an ordering of the rows of an experiment that made it easy to calculate effects even in the absence of computers. Originally, "Yates order" is the term for this order of the rows. It is now customary to list the columns in the exactly analogous order as well: This order comes about by appending each new factor as the last column and then appending its products with all the preceding columns in the order of occurrence. This leads to the sequence A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, ... It can therefore be said that the columns are listed in Yates order, although "Yates order" originally referred to the order of the rows.

The list `Yates` in this software indicates for each column position, which factors are involved in the respective effect. The column numbers of the thus-obtained matrix are a customary way to denote generating contrasts in the literature on 2-level fractional factorials.

Author(s)

Ulrike Groemping

See Also

See also `pb` and `FrF2` for the functions that do the calculations, `catlg` for the catalogue of regular 2-level fractional factorial designs, and `Menu.2level` for overall help on 2-level factorials in this DoE software.

`editDataset.design` *function to add a warning to the Rcmdr editDataset function*

Description

invokes a warning message before the data.frame method of editDataset is called

Usage

```
## S3 method for class 'design'
editDataset(data, dsname, ...)
```

Arguments

`data`	an object of class `design`
`dsname`	a character string argument, which is always missing in the usage of the function
`...`	not used

Details

It is discouraged to edit design objects from within R. Users should export the design and re-import it after adding response data.

Nevertheless, it is possible to add response values by adding numeric columns to the design with the editor (never edit factors!), and users will have to attach the responses added to an edited design to their original design afterwards, using e.g. the
Modify design -> Add response variables ...
menu, providing the edited data set as the R object that contains the responses.

Value

a data.frame with `.edited` appended to the original design name and without any class design properties.

If editing involved adding response data, the edited file can be used for adding respones to the design object (see Details section).

Warning

Editing a class design object is seldom useful.

Author(s)

Ulrike Groemping

`Menu.2level` *2-level factorial designs*

Description

There are two types of orthogonal 2-level factorial designs, regular fractional factorial designs and screening designs. This help file is about when to apply these.

Quantitative or Qualitative

Both types of design are suitable for quantitative and qualitative factors alike.

Screening designs or Plackett-Burman designs

Screening designs are particularly useful, if the experiment is intended to pick a few important factors out of a list of many candidate factors. It often can not be ruled out that factors interact with each other, but the interactions are not of interest at the screening stage. If the main effects are stronger than the interactions, which is very often the case, the screening experiment has a good chance of selecting the key factors for further experimentation. With very high expertise and some luck, a screening experiment might even be sufficient to draw further conclusions.

Regular (fractional) factorial designs

Regular Fractional Factorial designs of resolution III are also sometimes used for screening. If the aliasing is not too severe, if e.g. just three main effects are aliased with one 2-factor interaction each (i.e. one word of length three), this can be reasonable. However, with more severe aliasing, using a resolution III design is more risky than using a screening design, because main effects are completely aliased with 2-factor interactions, which can lead to severe bias and wrong conclusions.

Regular (Fractional) Factorial designs of resolution IV or higher are particularly useful, if not only main effects but also 2-factor interactions are expected to be active. Note that interactions between more than two factors are usually considered negligible. This is usually appropriate, if the experimental region is not too large, since many functions can be approximated well by first- or second-order polynomials over small regions.

Regular (Fractional) Factorial designs can be run in blocks. This is advisable, whenever the experimental units are not homogeneous, but smaller groups of units (the blocks) can be made reasonably homogeneous (e.g., batches of material, etc.). The number of blocks must be a power of 2. If there are some experimental factors, that can only be varied for a complete block, a so-called splitplot design is needed; this can be accomodated by function `FrF2`, but has not been implemented in this GUI so far.

If package **FrF2.catlg128** is installed (this will not be necessary for standard use), there is an Expert menu entry for loading catalogues from this package for usage with regular fractional factorial 2-level designs. On using this menu entry, the package is loaded, and a list of catalogues is shown. Do not unnecessarily load them, because some of them are very large.

Warning

Important: For both types of design, like also for other factorial designs, the experiment must conduct all experimental runs as determined in the design, because the design properties will deteriorate in ways not easily foreseeable, if some combinations are omitted.

It must be carefully considered in the planning phase, whether it is possible to conduct all experimental runs, or whether there might be restrictions that do not permit all combinations to be run (e.g.: three factors, each with levels "small" and "large", where the combination with all three factors at level "large" is not doable because of space restrictions). If such restrictions are encountered, the design should be devised in a different way from the beginning. If possible, reasonable adjustments to levels should ensure that a factorial design becomes feasible again. Alternatively, a non-orthogonal D-optimal design can take the restrictions into account. *Unfortunately, this functionality is not yet implemented in this GUI.*

Author(s)

Ulrike Groemping

References

Box G. E. P, Hunter, W. C. and Hunter, J. S. (2005) *Statistics for Experimenters, 2nd edition*. New York: Wiley.

See Also

See Also `pb` for the function behind the screening designs, `FrF2` for the function behind the regular fractional factorial designs, and `catlg` for a catalogue of regular fractional factorial designs, and `DoEGlossary` for a glossary of terms relevant in connection with orthogonal 2-level factorial designs.

`Menu.addcenter` *Adding center points to a 2-level design*

Description

This help file describes adding center points to a 2-level design from package **FrF2**.

Details

A name for the augmented design must be given.

The number of center points is needed.

Center points are distributed systematically over the design - they are not randomized. The bottom text box requests the number of positions over which the center points are to be distributed.

Author(s)

Ulrike Groemping

See Also

See also function `add.center` for the function that does the work

`Menu.addresponse` *Adding (a) response(s) to an existing design*

Description

This help file describes how to use the menu for adding (a) response(s).

Details

Select a design (default: active design) and decide, whether existing response data are to be replaced (useful e.g. if existing response contains missing values that are to be overwritten).

Decide whether response data are already in R or are available from a `csv` file. In the latter case, changing the working directory is possible (but not necessary).

If the response data are already in R or are to be calculated, fill in the appropriate information in the entry field for the response (for playing with random data, this could e.g. be `rnorm(8)`). Otherwise, select a csv file and choose the correct decimal setting. The chosen decimal setting will become the default also for other menus that require a decimal setting.

Finally, provide a valid file name for the modified design.

NOTE: If the response data are not in a csv file, they can be imported from all kinds of other file formats using from R commanders data management menu. The menu *Design* offers the sub menu *Modify Designs* with the *Add responses* menu item for this purpose.

WARNING: Package **RcmdrPlugin.DoE** has been subjected to some testing, but must still be considered a beta version. Especially for data handling routines like adding responses, importing csv files and the like, it is strongly recommended to **always keep a master copy of your valuable experimental data safely stored without touching it**. (This would even hold for a final version of the any software, but is of course far more important for a beta version.

Author(s)

Ulrike Groemping

See Also

See also function `add.response` for the function that does the work

Menu.Analyze *Analyze experimental designs*

Description

This is a brief explanation on the analysis facilities for experimental designs in the Analyze Design menu.

Available analysis options

Generally applicable analysis options The topmost menu entry Default linear model is of interest for all design types and is usable whenever the active dataset is a design with response. However, the default linear model analysis does not work for long version designs with repeated measurements or for parameter designs in long format, as it usually does not make sense in such situations. Rather, such designs should be brought into wide format by using the Design –> Combine or Modify Designs –> Change from long to wide format menu.

Note that the default linear model analysis is a quick first shot that should often be tuned and sometimes (e.g. in many cases splitplot designs) replaced by a different analysis strategy for serious modelling. Tuning can be done by using the built-in linear model functions from the R-Commander Analyze menu. The R-Commander Models menu also offers interesting options for subsequent model diagnostics and graphics.

Analysis options for general factorial designs
For a general factorial design with a response, main effects and interaction plots can be generated (one at a time) by a menu which was slightly from **Rcmdr**s general menu for graphing arithmetic means.

Analysis options specific to 2-level designs
There are two types of orthogonal 2-level factorial designs, regular fractional factorial designs and screening designs. The latter has more interesting analysis options than the former.

Effects plots and main effects plots are of interest for both types of 2-level designs alike, while interaction plots are usually of interest for the regular designs only.

Note that the interpretation of all effects plots, main effects and interaction plots is useful only in connection with knowledge about the alias structure of a design. For regular designs, this can e.g. be obtained from the *Summarize design* menu item within the *Inspect design* sub menu of the *Design* menu. For screening designs, if it can be assumed that interactions are likely to be much less important than main effects, the main effect plots may be interpretable without such thoughts. Considerations involving the interactions become quite complicated with screening designs because of partial aliasing. Advanced users might also want to try the Bayesian methods offered in package `BsMD-package`, which are currently not implemented in **RcmdrPlugin.DoE**.

Analysis options specific to designs with quantitative factors
are available within R-package **rsm** and are implemented in particular for response surface designs; they can also be used for other designs with quantitative variables, especially for latin hypercube designs. (However, for the latter, there are often reasons to use nonparametric approaches.

The menu item *Response surface model ...* creates a response surface analysis with first order (FO) terms, and potentially also two-factor interactions (TWI) and/or pure quadratic (PQ) terms. The resulting object has class `rsm` and can be postprocessed with the subsequent menu items.

The menu item *Steepest slope ...* supports moving the experimental range towards a more promising area, based on a first order model or on a second order model with a saddle point.

The menu item *Plot response surface ...* supports creation of contour, perspective or image plots of a response surface model, and can also be used for linear models with quantitative variables.

General analysis functionality from R Commander
It is of course also possible to use other analysis facilities within the R Commander, or to use the command-line facilities offered in package `rsm` and elsewhere.

Author(s)

Ulrike Groemping

References

Box G. E. P, Hunter, W. C. and Hunter, J. S. (2005) *Statistics for Experimenters, 2nd edition*. New York: Wiley.

See Also

See also `summary.design` for how designs are summarized, `formula.design` for the function that creates the default linear model formula, or `DanielPlot`, `MEPlot` and `IAPlot` for the functions behind the graphical analysis tools for 2-level factors.

Menu.augmentlhs *Menu for augmenting an existing latin hypercube sample*

Description

With this menu, an existing latin hypercube sample for quantitative factors can be augmented. The menu calls function lhs.augment from R-package DoE.wrapper.

Usage

```
Menu.augmentlhs()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with the attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry "Export design" from the "Design" menu, or the full R workspace can be stored using the appropriate entry in the "File" menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See Also `lhs.augment` for the function behind this menu and `Menu.Quantitative` for the available designs for quantitative factors used in this software or its documentation.

`Menu.bbd` *Menu for generating Box-Behnken designs for quantitative factors*

Description

With this menu, Box-Behnken designs for quantitative factors can be generated. The menu calls function bbd.design from R-package DoE.wrapper.

Usage

```
Menu.bbd()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with the attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry "Export design" from the "Design" menu, or the full R workspace can be stored using the appropriate entry in the "File" menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See Also `bbd.design` for the function behind this menu and also function `bbd` from package **rsm** for the function that underlies function `bbd.design`. `Menu.Quantitative` describes the designs for quantitative factors that are available in this software.

`Menu.bbdTab1` *Basic information for Box-Behnken designs*

Description

Basic information for Box-Behnken designs

Brief statistical background

Box-Behnken designs have three levels in all factors and are useful for fitting second order polynomials to experimental data based on several quantitative response variables. For more information, look at the help file for function `bbd.design` from package **DoE.wrapper** or at `bbd` from package **rsm** .

Inputs on Tab Base Settings

name of design must be a valid name. The design itself is created under this name in the R workspace.

number of factors must always be specified. It must be an integer number from 3 to 7. The number of factors must match the number of entries on the Factor Details tab.

block name name for a block variable (NULL, if no blocks)). Blocks are possible for 4 and 5 factors only (ignored otherwise).

seed can be provided for reproducing an existing design.

randomization can be switched off in exceptional situations. Usually, one would not want to do that.

Author(s)

Ulrike Groemping

References

~put references to the literature/web site here ~

See Also

See also `bbd.design` for the function that does the calculations and `bbd` for the underlying function from package **rsm**.

`Menu.BsMDPlot` *Help on using the menu for 2-level Bayesian Screening analysis*

Description

Using the menu for 2-level Bayesian Screening analysis

Using the menu for 2-level Bayesian Screening analysis

Select the response that is to be analysed.
Select the factors to include in the screening - usually use all factors, but exclude dummy factors for pb type designs.
Select the degree of interactions to be included in the model.
Indicate, whether only a plot is requested or also a printout of model probabilities. The default is to also print the model probabilities.

Author(s)

Ulrike Groemping

See Also

See also `BsProb.design` and `plot.BsProb` for the underlying functions.

`Menu.ccd` *Menu for generating central composite designs for quantitative factors*

Description

With this menu, central composite designs for quantitative factors can be generated. The menu calls function ccd.design from R-package DoE.wrapper.

Usage

```
Menu.ccd()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with the attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry “Export design” from the “Design” menu, or the full R workspace can be stored using the appropriate entry in the “File” menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under “Data Management”.

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See also `FrF2` for the function defining the cube portion of the design `ccd.augment` for the function that augments the cube portion with the star points, also function `ccd` from package **rsm** for the function that underlies function `ccd.augment`. Finally, look at `Menu.Quantitative` for the available designs for quantitative factors available in this software.

`Menu.ccdTab1` *Basic information for Central Composite Designs*

Description

Basic information for Central Composite designs

Brief statistical background

Central composite designs can have three (`alpha=1`) or five levels in all factors and are useful for fitting second order polynomials to experimental data based on several quantitative response variables. They can be used in a process of sequential experimentation, experimenting first with the cube portion and later adding (a) star block(s).

For more information, look at the help file for function `ccd.design` from package **DoE.wrapper** or at `ccd` from package **rsm**.

Inputs on Tab Base Settings

name of design must be a valid name. The design itself is created under this name in the R workspace.

Determine the cube portion An existing 2-level fractional factorial, preferrably of at least resolution V, can be selected, or a new design for the cube portion can be generated;
if the number of center points is to be different between cube and star portion of the design, the cube portion must be generated with center points.

number of center points The number of center points in the cube portion is ideally defined within the cube itself. In that case, the number given refers to the center points in the star portion. If the cube portion has no center points, a single number of center points is applied to both the cube and the star portion.
Alternatively, a separate number can be given to specify different numbers of center points for cube and star portion, e.g. `0, 5` for the unfortunate situation where the cube portion has already been run without any center points (not recommended!).)

alpha determines how far inside (alpha<1) or outside (alpha>1) of the cube the star points are positioned. The special values `orthogonal` (default) or `rotatable` define calculated alpha values with interesting statistical properties. However, it must of course be checked whether these make practical sense. For example, if the orthogonal alpha turns out to be 2 but experimentation beyond 1.5 does not make sense, the number should be defined manually.

seed can be provided for reproducing an existing design.

randomization can be switched off in exceptional situations. Usually, one would not want to do that. Note that the checkbox here refers to the star portion only. For the cube portion, there is also such a checkbox.

Author(s)

Ulrike Groemping

References

~put references to the literature/web site here ~

See Also

See also `FrF2` for the function defining the cube portion of the design `ccd.augment` for the function that augments the cube portion with the star points, also function `ccd` from package **rsm** for the function that underlies function `ccd.augment`.

`Menu.changecontr` *Help on using the menu for changing contrasts*

Description

Help on using the menu for changing contrasts

Help on using the menu for changing contrasts

This menu allows to change contrasts for one or more factors. Contrary to R-Commanders native contrast changing function, it simultaneously treats the matrix `desnum` for the `design` object, so that the integrity of the class `design` object is maintained. Furthermore, it uses the contrast functions from base R rather than those from package **car**.

One or more factors can be selected from the list on the left and moved to the list on the right. On OK, all factors from the RHS list are assigned the chosen contrast.

Note that the default contrast setting is "Treatment (dummy) coding", regardless of the current factor coding.

Author(s)

Ulrike Groemping

See Also

See also `change.contr` for the function that does the work

`Menu.colremove` *Help on using the menu for deleting columns*

Description

Help on using the menu for deleting columns

Help on using the menu for deleting columns

This menu allows to permanently remove certain types of columns (NOT: block factor column or columns with experimental factors) from the design.

If you want to keep a response column in the data frame but not treat it as a response variable any more, use the select / deselect responses menu instead.

Author(s)

Ulrike Groemping

See Also

See also `class-design` for the underlying function

`Menu.contour` *Contour plots for linear models with quantitative regressors*

Description

Contour plots for linear models with quantitative regressors

Overview

This dialog allows to specify pairs of numeric regressors for which the response surface is to be displayed. Each display can be a contour plot with or without image colours, a 3D-perspective plot or an image plot.

For looking at several plots simultaneously, numbers of rows and columns can be specified. On OK, all plots are generated. One should always choose the number of rows and columns such that all requested plots fit on one display, otherwise the early plots will be overwritten by the later plots.

The functions behind the functionality are `contour`, `persp` and `image`.

The second tab on the dialogue allows to specify at which values variables not in the plot are to be fixed. The default is the average for numeric variables, the first level for factors. (

Inputs

number of rows and number of columns These are used in the `mfrow` function for creating the row and column layout for arranging several plots.
Within an R session, modifications are retained for further calls of the dialog.

Select plot type Contour plots show contours of constant height, 3D perspective plots show the surface in mesh form; both come with or without coloring. The image plots show colours only, without contours. It is mainly a matter of taste which plots to use.
Within an R session, modifications are retained for further calls of the dialog.

Select pairs for plotting The default choice plots pairs for all numeric factors. This can be too many plots for one page in case of many factors.
For the top radio button, pairs can be conveniently formed from two groups of variables:
1: each pair within group 1
2: each pair within both groups 1 and 2 (but not pairs *between* groups)
3: each pair that involves any factor from group 1
4. each pair between groups 1 and 2 (but no pair *within* any group)
Initially, all variables are in group 1. They can be moved between groups by selecting them with the mouse and moving them with the arrow buttons.

In case of the bottom (manual selection) radio button choice, move selected 2-factor pairs between the available list (those are not selected) and the selected list by selecting them with the mouse (multiple selections possible) and moving them with the arrow buttons.
These choices have to be redone with each call to the dialog.

Modify slice positions Each response surface varies the values of two numeric variables, keeping all other variables fixed. These fixed values are also called the slice positions. The default slice positions are the average for numeric variables and the first level for factors. These can be modified on the second tab of this dialog. Within an R session, modifications are retained for further calls of the dialog w.r.t the same model.

Author(s)

Ulrike Groemping

See Also

See also `contour` for the functions that do the actual plotting

`Menu.display` *Using display design menu*

Description

This help file describes displaying the active design in actual or standard run order.

Details

This help file describes displaying the active design in actual or standard run order.

Displaying in actual run order is the default. It is useful for feasibility considerations, for identifying unwanted systematic patterns, or for identifying run order effects in the responses.

Displaying in standard run order is useful, because it helps to identify structure for some types of design (mainly for factorial designs, not so exciting e.g. for latin hypercube designs).

Author(s)

Ulrike Groemping

`Menu.Dopt` *Menu for generating D-optimal designs*

Description

With this menu, D-optimal designs can be generated. The menu calls function Dopt.design from R-package DoE.wrapper. Optionally, the candidate designs menu can be called before calling this menu in order to create a candidate design.

Usage

```
Menu.cand()
Menu.Dopt()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with the attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry "Export design" from the "Design" menu, or the full R workspace can be stored using the appropriate entry in the "File" menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See also `fac.design` and `oa.design` for the functions uasble for defining the candidate points of the design, `Dopt.design` for the function that creates the D-optimal design, also function `optFederov` from package **AlgDesign** for the function that underlies function `Dopt.design`.

Menu.DoptTab1 *Basic information for creation of D-optimal Designs*

Description

Basic information for creation of D-optimal designs; this menu has been created in August 2010 and is still experimental. Please report issues for helping improve the software.

Brief statistical background

D-optimal designs are useful for generating the design points such that they are efficient for estimation of the parameters of a pre-specified model.

For more information, look at the help file for function `Dopt.design` from package **DoE.wrapper** or at `optFederov` from package **AlgDesign**.

Inputs on Tab Base Settings

name of design must be a valid name. The design itself is created under this name in the R workspace.

Determine the candidate design An existing design can be selected, or a new design can be generated;
the candidate design is a large design from which the D-optimal design is selected.

Specify model formula of the model for which an optimum design is sought

Specify constraints constraint condition(s) on the design variables; for example, if several factors refer to sizes, it may not be possible to combine all settings (e.g. under space restrictions);
See `Syntax` and `Logic` for an explanation of the syntax of general and especially logical R expressions.

nRepeat number of independent repeats for optimization;
increasing this number increases search time, but also the chances to find a global optimum

seed can be provided for reproducing an existing design.

randomization can be switched off in exceptional situations. Usually, one would not want to do that. Note that the checkbox here refers to the optimization process only. For the candidate set, this can also be decided on the respective separate menu.

Author(s)

Ulrike Groemping

See Also

See also function Dopt.design for the function from **DoE.wrapper** that creates the design, which in turn is based on function optFederov from package **AlgDesign**.

Menu.EffectsPlots *Help on using the menu for (half) normal effects plots*

Description

Help on using the menu for (half) normal effects plots

Help on using the menu for (half) normal effects plots

Select the response that is to be analyzed.
Decide with the check buttons,
whether a half or full normal plot is requested,
whether effects are labelled with their names or with codes (usually A, B etc.)
and whether only significant effects are to be labelled
Change the value for alpha (level of significance) to your liking.

Author(s)

Ulrike Groemping

See Also

See also DanielPlot for the underlying function

Menu.ExpImp *Exporting and importing Designs*

Description

This help file describes facilities for exporting and importing designs.

Details

The *Import* sub menu of the *Design* menu allows to change the working directory, as a general convenience feature. Furthermore, it enables importing an R workspace with more than one object in it - this facility makes the first class design object in the work space the active data set, and it allows to import a design from the R workspace into R-Commander (=make it the active data set). Furthermore, a previously-exported design can be imported again, using both an edited csv-file (created from a stored html- or csv-file) and an rda-file. It is assumed that the csv file contains new responses, while the rda-file contains all the structural information on the design. (If the csv file contains responses ONLY, these must be in the correct order. If run order information is also

contained in the csv file in the correct format, the rows of the file are correctly assigned to the rows of the design, even if reordered.)

The *Export* menu allows to save a complete R workspace or to export an individual experimental design. Furthermore, it also allows to save the generated code scripts (script window) and output (output window). Exporting an individual experimental design to csv and/or html is often useful, as experimental data are usually more easily entered using software like Excel. If the files are handled in a controlled way, response data that are entered as additional columns to an exported `csv`- or `html`-file can be re-imported later, using the re-import dialog accessible from the import menu. For more detail, cf. `export.design` for the exporting and `add.response` for the re-importing based on a `csv`-file and an `rda`-file.

Other features are not linked from the Design menu: If an experimental design is available in R format (e.g. by exporting it to rda, as mentioned in the previous paragraph) but responses are stored in a format different from `csv`, it is also possible to re-import the response data into the design. If the design without responses is the active design (e.g. after loading the `rda` file using the *Load R workspace* menu item from the *Import* sub menu of the *Design* menu), the response data can be imported using the appropriate menu item of the *Import data* sub menu of the *Data management* menu. Subsequently, the response data can be added to the design using the *Add response variable(s)...* item of the *Modify design ...* sub menu of the `Design` menu.

It is also possible of course to import experimental data generated elsewhere that do not contain the structural information available with class `design` by using the usual import facilities from the *Data management* menu of R-Commander. However, for such experimental data, the usability of special features of package **RcmdrPlugin.DoE** is severely restricted.

Author(s)

Ulrike Groemping

`Menu.exportTab` *Using the export menu*

Description

This help file describes the design exporting options. Default choices are no exporting for design generation forms or exporting an html file and an rda file, when separately calling the export routine.

Details

design information At the top of the form, the design name is shown (provided on the first tab of the design generation form) or can be chosen from the designs available in the R work space.

(How to) export radio buttons determine the type(s) of export file(s). Choices are that
no export file is generated (export tabs on design generation menus),
OR all file types listed below are generated,
OR only an `rda` file is generated,
OR an `rda` file and an `html` file are generated,
OR an `rda` file and a `csv` file are generated.
If Microsoft Excel is to be used in providing the actual experimenters with an annotated file

for experimentation, usage of an `html` file is recommended. After the experiment has been carried out, the data can be stored as `csv` and reimported into R, where they can be processed using the structural information from the design, if the stored `rda` file is also provided.

decimal separator radio buttons determine the decimal separator to be used; relevant, if the data contain decimal numbers; the default uses the `OutDec` option of the R system, the other choices explicity determine the decimal separator. If chosen correctly, the html file can be opened in Excel with decimal numbers read as such; furthermore, decimal separator "," is taken to also mean separator ";", i.e. `csv` files also use the matching separator (decimal separator comma goes with separator semicolon and corresponds to `read.csv2`, decimal separator point goes with separator comma and corresponds to `read.csv`).

The Storage Directory can be changed with the selection button. For the separate export menu, the chosen directory will become the new working directory; this is not the case for the Export tab in design creation menus.

Provide the file name without ending in the file names field. Files with this name and the appropriate endings are produced, as chosen by the "(How to) export" radio buttons.

The replace checkbox can be checked, if you want to replace previously created files. In this context, note that the rda file exports an R workspace that ONLY includes the generated design and NOT the complete R workspace content.
If you want to store the complete R workspace, use the appropriate menu item on the R commander file menu (usually the left-most menu).

Note

The purpose of exporting to `html` or `csv` format is the possibility to edit response data outside of R, e.g. for persons without access to R who conduct the experiment.

The resulting edited data file can be stored in `csv` format and re-imported into R. If re-importing is done with the appropriate menu item in the *Import* sub menu of the *Design* menu, providing the stored `rda` file along with the edited data file, all structural information stored with the design is available.

If users prefer to store the edited data file in a format different from `csv`, the data can be re-imported in two steps: First, the edited data file must be imported with R-Commanders *Data management* menu. Afterwards, the response information can be added to the design using the menu item *Add response variable(s) ...* from the sub menu *Modify design* of the menu `Design`.

Author(s)

Ulrike Groemping

See Also

See Also function `export.design` from package **DoE.base** for the function that actually does the exporting

`Menu.fac` *Menu for generating full factorial designs*

Description

This menu is for generating full factorial designs. The menu calls function function fac.design from package DoE.base and has been opened internally using the function shown in the usage section, which normal users will never have to deal with.

Usage

```
Menu.fac()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with some attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry "Export design" from the "Design" menu, or the full R workspace can be stored using the appropriate entry in the "File" menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See Also `fac.design` for the function behind this menu, `Menu.General` for choice between full factorial designs and orthogonal main effects designs.

`Menu.FacDetails2Tab` *How to use the factor details tab for 2-level and quantitative designs*

Description

This help file describes how to input factor details for 2-level and quantitative designs

Inputs on tab Factor Details for all such menus

Common factor levels can be checked to define a common set of factor levels that is used for all factors. The first level corresponds to -1 or the lower bound of the range of a quantitative factor, the second level to +1 or the upper bound of the range of a quantitative factor.

When unchecking the checkbox, levels can be individually adjusted for each factor. WARNING: Re-checking the box after individual level changes looses all settings.

Modify factor details for selected factor is the frame in which changes can be made. A factor can be selected by clicking on the arrow below `Select` (and also by clicking on an item in the listboxes below the frame). Repeatedly jumping through the frame with the `Tab` key automatically cycles through the factors one by one.

The entry `Comment or label` is included into an exported html file as annotation for the factor. This may be useful for conducting the experiment.

Buttons for changing the factor order If factor details have been entered and it is discovered later that the factors are needed in a different order (e.g. because of the alias structure of a regular 2-level fractional factorial design), the buttons `Move Up` and `Move Down` can be used for changing the order of the factors.

Hard to change factors can be specified for regular 2-level fractional factorial designs, if special choices have been activated by the checkbox below the tab. This ascertains that the experiment can be run with as few as possible changes in the first (number to be specified) factors.

This should only be used if absolutely unavoidable, as it worsens the information on the affected factors.

Author(s)

Ulrike Groemping

`Menu.FacDetailsGenTab` *How to use the factor details tab for general designs*

Description

This help file describes how to input factor details for general designs

Inputs on tab Factor Details for all such menus

Modify factor details for selected factor is the frame in which changes can be made. A factor can be selected by clicking on the arrow below `Select` (and also by clicking on an item in the listboxes below the frame). Repeatedly jumping through the frame with the `Tab` key automatically cycles through the factors one by one.

The number of levels MUST BE ENTERED for each factor. (Jumping out of the entry box without entering a valid number of levels creates an error message popup.) The levels themselves are then automatically set to the integer numbers from one to the number of levels. Different levels (separated by blanks) can be manually entered. It is recommended to use the automatic levels for preliminary checks on available designs and their properties, but to enter meaningful factor levels for experiments that will actually be run.

The entry `Comment or label` is included into an exported html file as annotation for the factor. This may be useful for conducting the experiment.

Buttons for changing the factor order If factor details have been entered and it is discovered later that the factors are needed in a different order (e.g. because of the alias structure of a regular 2-level fractional factorial design), the buttons `Move Up` and `Move Down` can be used for changing the order of the factors.

Manually specify column numbers for array is a checkbox for the orthogonal array menu that is available if a specific design has been selected by ID (from a drop-down menu on the Base Settings tab). If this box is checked, the factor details include an additional column for the column numbers that needs to be filled with a valid column number for each factor. The number of levels is then automatically derived.

Specifying column numbers can be useful, if an expert user has investigated the properties of a design and wants to obtain a better-suited design than the default one. For example, when accomodating two 2-level factors, one 4-level factor and one 8-level factor in `L64.2.5.4.10.8.4`, it is better to use columns 2 and 3 than the default columns 1 and 2 for the two 2-level factors. With the latest version of the software, this optimization can also be done automatically using the setting "min3" for Automatic Optimization on the "Base Settings" tab; nevertheless, because of performance reasons, it may be wise to specify known good column numbers, if available. Note that explicit specification of column numbers overrides any optimization request.

Author(s)

Ulrike Groemping

`Menu.facTab1` *Basic information for full factorial designs*

Description

This help file describes usage of the basic information tab of the full factorial design menu

Brief statistical background

Full factorial designs consist of all possible combinations of factor levels, i.e. the number of runs is the product of all numbers of factor levels, for example 24 for an experiment with two 2-level factors and one 6-level factor. Of course, their size grows fast with the number of factors. If a full factorial design is not feasible, orthogonal main effects designs or (manually-generated) combinations of smaller designs may be a reasonable option.

Inputs on Tab Base Settings

name of design must be a valid name. The design itself is created under this name in the R workspace.

number of runs is a consequence of the specifications on the Factor Details tab. It is displayed for information purposes only; its value is only valid if the Factor Details tab contains entries for all factors.

number of factors must always be specified. The number of factors must match the number of entries on the Factor Details tab.

replications is the number of times each experimental run is conducted. If larger than 1, each run is conducted several times. If the checkbox next to the number of replications is checked, it is assumed that the experiment involves repeated measurements for one setup of the experimental run; if it is not checked, the experimental run itself is replicated with everything relevant newly set up (much more valuable than repeated measurements, unless the key driver of variability is in the measuring step). If the check box is not checked, the experiment will be randomized separately for each round of replications (first all first runs, then all second runs etc.).

number of blocks is the number of equally-sized blocks of homogeneous units into which the overall number of runs is to be subdivided. Note that the number of blocks must be compatible with the numbers of levels of the experiment: it must be the product of one or more primes that the numbers of levels factor into. For example,
a design with three factors at 2,5 and 5 levels can have 5 blocks or 10 units each, no other blocking is possible for this design because it would confound blocks with factor main effects;
a design with three factors at 2, 6 and 6 levels can have 2, 3, 6, 4 or 12 blocks, because all these numbers of blocks can be obtained from the three 2s and two 3s the numbers of levels factor into without confounding a main effect.
An error message will be given whenever an impossible number of blocks or a number of blocks that would require aliasing of blocks with main effects is used; the design is generated with a warning message whenever the block factor is aliased with any two-factor interaction among the design factors.

randomization settings should normally not be changed; you can provide a seed if you want to exactly reproduce a randomized design created in the past. Unchecking the randomization box will produce a non-randomized experiment. This is usually NOT recommended.

Author(s)

Ulrike Groemping

See Also

See Also `fac.design` for the function that does the calculations and `Menu.General` for overall help on the general factorial design menu.

Menu.FrF2level *Menu for generating regular fractional factorial 2-level designs*

Description

With this menu, regular fractional factorial 2-level designs can be generated. The menu calls function FrF2 from package FrF2 and has been opened internally using the function shown in the usage section, which normal users will never have to deal with.

Usage

```
Menu.FrF2level()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode (https://socialsciences.mcmaster.ca/jfox/Misc/Rcmdr/installation-notes.html). It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

If the checkbox "Activate Special Choices" is ticked, the user form gets additional tabs for requesting estimable 2-factor interactions, for block or split-plot details, and for hard-to-change factors. Also, some of the standard tabs get more content than otherwise. If you are not an expert in the statistical details of experimental design, it is normally best to not activate these additional choices. The standard choices are sufficient for many purposes.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with some attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry "Export design" from the "Design" menu, or the full R workspace can be stored using the appropriate entry in the "File" menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2FrF`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See Also FrF2 for the function behind this menu, catlg for the design catalogues underlying this menu, and Menu.2level for choice between regular fractional factorial designs and screening designs, and DoEGlossary for the DoE-related terminology used in this software or its documentation.

Examples

```
## running the menu with default settings corresponds to
Design.1 <- FrF2(8, 4)
## the stored code from the menu will look more complicated because it also lists
##     the defaults for all the other arguments
```

Menu.FrF2levelTab1 *Basic information for regular (fractional) factorial 2-level designs*

Description

Basic information for regular (fractional) factorial 2-level designs

Brief statistical background

These designs are created with function FrF2 from package **FrF2**. Look up the documentation there for statistical details. Users who are not familiar with the terminology around regular fractional factorial designs (e.g. the concept of resolution) can browse the DoEGlossary.

Inputs on Tab Base Settings

name of design must be a valid name. The design itself is created under this name in the R workspace.

number of runs must be a power of 2. If this is omitted by unchecking the checkbox next to it, the function looks for the smallest possible design that fulfills all requirements.

number of factors must always be specified. It must be smaller than the number of runs. Resolution IV designs are possible, if this number is smaller than half the number of runs.

replications is the number of times each experimental run is conducted. If larger than 1, each run is conducted several times. If the checkbox next to the number of replications is checked, it is assumed that the experiment involves repeated measurements for one setup of the experimental run; if it is not checked, the experimental run itself is replicated with everything relevant newly set up (much more valuable than repeated measurements, unless the key driver of variability is in the measuring step). If the check box is not checked, the experiment will be randomized separately for each round of replications (first all first runs, then all second runs etc.).

blocks is the number of blocks. Blocks are needed, if a factor relevant to the outcome but not of interest in itself is to be corrected for (e.g. batch of material). If you just give a number here, the program takes care of all details itself. If you are an expert and consider it necessary, you can provide more information on the special tab on blocking that is visible if special choices have been activated.

randomization settings should normally not be changed; you can provide a seed if you want to exactly reproduce a randomized design created in the past. Unchecking the randomization box will produce a non-randomized experiment. This is usually NOT recommended.

Design Properties allow to set a minimum resolution (III, IV or V+); for most purposes, IV is a good solution, i.e. it might be a good idea to increase the default choice.
In addition, a choice can be made between
MA = maximum resolution and minimum aberration, which implies usage of the design with the overall least severe degree of aliasing among effects,
OR
MaxC2 = maximum number of clear 2-factor interactions, which means that the design is chosen for which the most 2-factor interactions are not aliased with any other main effect or 2-factor interaction.

Note that, when choosing MaxC2, it is particularly important to also consider setting resolution to IV, as otherwise the design will very likely be of resolution III only, even if there are good resolution IV designs with many clear 2-factor interactions around.

Information Buttons are available for looking at interim information on the suitable designs: The button `View available designs` allows to look at the properties of the designs in the catalogue, the button `Show best 10 designs ...` prints more detailed information on the best 10 available designs for the currently specified settings.

Note that you may want to retrieve the still open menu from somewhere in case the latter button is used, since this button releases the grab on the menu in support of scrollability of the printed results. I hope to find a better solution for this in the future.

Expert Choices If the special choices are activated, it is possible
to specify a set of generators,
to specify a catalogue of designs different from the default one (if one is in the workspace; with package **FrF2.catlg128** installed, a catalogue from that package can be loaded into the workspace using the menu item "Experts: load additional catalogue ...")
or to pick a specific design name from the current catalogue of designs.
Radio buttons activate one of these possibilities. Details for the consequences of these choices can be found in the help file for function `FrF2`.

Author(s)

Ulrike Groemping

See Also

See also `FrF2` for the function that does the calculations, `catlg` for the catalogue of designs that this functionality is based on, and `Menu.2level` for the distinction between screening designs and regular fractional factorial designs.

`Menu.FrF2levelTabEstimable`
Estimable 2fis for regular (fractional) factorial 2-level designs

---

Description

Estimable 2fis for regular (fractional) factorial 2-level designs

Warnings

Not all combinations of inputs are compatible with each other. You may avoid unnecessary work by reading the advice below. In case of violating the restrictions below, the specification of estimable 2fis will take precedence over other specifications (you will be warned, if this happens; on re-opening the dialog, you will then have the chance to adjust the settings).

Blocked designs (number of blocks larger than 1 on the "Base Settings" tab) and specification of an estimable model are not compatible with each other in the current version of the software. Thus, you have to decide on using blocks OR specifying estimable 2-factor interactions. (Nevertheless, 2 or 4 blocks can be specified as detailed in the section "Brief statistical background" below.

Generators or a specific design (radio buttons on the "Base Settings" tab) can neither be specified together with estimable 2-factor interactions.

Finally, it is neither possible to simultaneously specify hard-to-change factors on the "Factor Details" tab and estimable 2-factor interactions.

Brief statistical background

This tab allows to specify 2-factor interactions (2fis) that are of special interest and must be estimable from the experiment. Per default, it is assumed that the design must be at least of resolution IV.

There are two different approaches to estimability: If it can realistically be assumed that all the 2fis that are not of special interest are negligible, it is sufficient to request that the specified 2fis are clear of aliasing with EACH OTHER. Otherwise, i.e. if one is not prepared to assume that all the unselected 2fis are negligible, it is more appropriate to request that the specified 2fis are clear of aliasing with ANY 2fis (both with each other and the other 2fis). Even under this approach, the design need not be resolution V, since the unselected 2fis may be aliased with each other.

If a design with certain interactions to be estimable is to be run in more than four blocks, there is currently no automated way to produce it, as it is not permitted to specify a number of blocks together with selecting 2fis to be estimable. A blocked design with 2 or 4 blocks can be obtained by explicitly including one (2 blocks) or two (4 blocks) blocking factors.
For the case of 4 blocks, it is important to include the interaction between the two blocking factors into the selected interactions.
For the case of 2 blocks only, the blocking factor is treated like the other factors, i.e. if its interactions with other factors are not among the selected ones, they may be aliased with other 2fis (and even with main effects, if resolution III is permitted in exceptional situations), which is usually considered adequate.

The design is created with function FrF2 from package **FrF2**. The special functionality for estimable 2fis is described here. Users who are not familiar with the terminology around regular fractional factorial designs (e.g. the concept of resolution) can browse the DoEGlossary.

Inputs on Tab Estimable Model

Mode of requesting estimable 2-factor interactions The radio button switches off the request for specific interactions to be estimable or requests one of the two implemented modes that have

been discussed above.

Mimimum resolution is per default IV and can be set to III by checking the box. Usually, the default is more appropriate.

Select 2-factor interactions In case of the manual selection radio button choice, move selected 2-factor interactions between the available list (those are not selected) and the selected list by selecting them with the mouse (multiple selections possible) and moving them with the arrow buttons.
For some classical structures (compromise designs by Addelman (1962) or Ke, Tang and Wu (2005), cf. also `estimable.2fis`), it is more comfortable, and in case of distinct designs (bottom radio button for mode of requesting estimable interactions) also often much faster, to use the pre-specified strucures for two groups of factors; the type of structure can be selected from the drop-down, and the factors can be moved between the two groups using the arrow buttons (each factor must belong to one of the groups).

Limit search time If the bottom radio button for the mode of requesting estimable 2fis is chosen, the search for a design can take a long time. In this case, it is possible to increase (or decrease) the search time after which the search is stopped even if no appropriate design was identified. Searches with extensive requests can easily time out, especially if they are not implemented using the pre-specified group structure. Expert users may be able to utilize the structure of their problem for finding an appropriate design, using the command-line language rather than the menus.

Author(s)

Ulrike Groemping

See Also

See also `FrF2` for the function that does the calculations and `estimable` for the statistical detail of the "Estimable Model" functionality within this function.

Menu.General *General factorial designs*

Description

This menu covers full factorial designs orthogonal main effects designs for cases for which not all factors are at two levels. Furthermore, Taguchi-parameter designs are covered. This help file is about how and when to apply these.

Quantitative or Qualitative

Both types of design are suitable for quantitative and qualitative factors alike. If you have quantitative factors only, you may want to consider the special menu for these.

Full factorial designs

Full factorial designs are straight-forward to generate and are generated by function `fac.design` from package **DoE.base**. The number of runs needed for a full factorial experiment is the product of the numbers of levels of all factors. This may be more than is feasible. In such situations, the orthogonal main effects plans may be helpful. If interactions are also of interest, it may be useful to combine several plans or to pay attention to specific properties of orthogonal arrays (automatic support for such possibilities is currently poor and will be improved in the future).
Full factorial designs can be run in blocks. This is advisable, whenever the experimental units are not homogeneous, but smaller groups of units (the blocks) can be made reasonably homogeneous (e.g., batches of material, etc.).

Orthogonal main effects plans

If a full factorial experiment is too large, an orthogonal main effects plan may be useful. As long as there are no interactions between the factors represented by columns of the array, all such arrays work well, provided they are large enough for stable estimation. Some arrays also work well in the presence of interactions or even allow estimation of interactions for special subsets of variables. However, there is no automated support for selection of an array that has desirable properties. It may therefore be useful to specifically select an array the properties of which are known to the experimenter.

Warning

Important: For all factorial designs, the experiment must conduct all experimental runs as determined in the design, because the design properties will deteriorate in ways not easily foreseeable, if some combinations are omitted.

It must be carefully considered in the planning phase, whether it is possible to conduct all experimental runs, or whether there might be restrictions that do not permit all combinations to be run (e.g.: three factors, each with levels "small" and "large", where the combination with all three factors at level "large" is not doable because of space restrictions). If such restrictions are encountered, the design should be devised in a different way from the beginning. If possible, reasonable adjustments to levels should ensure that a factorial design becomes feasible again. Alternatively, a non-orthogonal D-optimal design can take the restrictions into account. *Unfortunately, this functionality is not yet implemented in this GUI.*

Taguchi inner-outer array designs - also called parameter designs

With the menu item *Create Taguchi inner-outer array*, two **existing** designs can be combined into a crossed inner-outer array design. For more detail, see the literature and the help in the Taguchi design menu.

Author(s)

Ulrike Groemping

References

Box G. E. P, Hunter, W. C. and Hunter, J. S. (2005) *Statistics for Experimenters, 2nd edition*. New York: Wiley.

See Also

See Also pb for the function behind the screening designs, FrF2 for the function behind the regular fractional factorial designs, and catlg for a catalogue of regular fractional factorial designs, and DoEGlossary for a glossary of terms relevant in connection with orthogonal 2-level factorial designs.

Menu.IAPlot *Help on using the menu for 2-level main effects and interaction plots*

Description

Using the menu for 2-level main effects and interaction plots

Using the menu for 2-level main effects and interaction plots

Select the response that is to be analysed.
Select the factors whose main effects or interactions are to be plotted (at least two factors are needed even for main effects).
Decide with the radio buttons, whether a main effects plot or an interaction plot is needed.
Select the abbreviation length for factor names from a list of lengths (Often the default will be OK).
For interaction plots, indicate whether confounding should be shown in the plot. The default is to show confounding, if confounded interactions are present.

Interpreting the alias information in the interaction plots

If alias information is shown, each plot has a number. Plots with identical numbers are based on the same effect estimate, i.e. they are confounded. This means that the displayed interaction is the sum of all the confounded interactions - they can either work into the same direction or into opposite directions, or most of them are negligible and one is dominating or ...

Author(s)

Ulrike Groemping

See Also

See also MEPlot and IAPlot for the underlying functions for main effects and interaction plots.

`Menu.import` *Using the import menu*

Description

This help file describes importing a design that is already in the R workspace but is not the currently active data set.

Details

This help file describes importing a design that is already in the R workspace but is not the currently active data set. The dialog is only available if at least one (object of class) design exists in the R workspace.

Select the design from the drop-down list of available designs, and press OK.

For importing an unloaded R work space first, use the *Load R workspace* menu item.

Author(s)

Ulrike Groemping

`Menu.importrdacsv` *Re-importing an exported design with the R-Commander*

Description

This help file describes how to re-import an exported design that exists as an rda and a csv file, when using the R-Commander.

Details

A design has usually been exported in order to input the response data outside of R. This is also the recommended strategy.

The re-importing supports the addition of responses that are available in a csv file to the class `design` data frame in the saved rda file. Usage of this menu also works, if the design is already present in the R workspace. In this case, it is not necessary to first load an rda file.

Simply provide a valid file name for the combined design,
for convenience, change the working directory,
select the rda file,
if necessary, select the correct design from the drop-down,
select the csv file,
adjust the decimal separator to the one used in the csv file,
and press OK.

NOTE: If the response data are not in a csv file, they can be imported from all kinds of other file formats using from R commanders data management menu. The menu *Design* offers the sub menu *Modify Designs* with the *Add responses* menu item for this purpose.

Author(s)

Ulrike Groemping

See Also

See also function `add.response` for the function that does the work

`Menu.Inspect` *Inspecting the active design*

Description

This help file describes the menu for inspecting the active design

Details

This menu offers possibilities for inspecting a designs properties. It is not meant to analyse data but to look at the structure of a design.

The menu item *Display active design* prints the current design.
The menu item *Summarize active design* prints a summary of the current design, based on the `summary` method for class `design`.
The menu item *Plot active design ...* plots selected experimental factors of the current design, based on the `plot` method for class `design`, applied to the design stripped of any responses.
The menu item *Tabulate active design ...* tabulates selected experimental factors of the current design.
The menu item *Experts: display design.info for active design* displays the design.info attribute of the current design (cf. `design` for a general description of class `design` and its attributes).

Plotting and tabulating are particularly interesting for designs with qualitative factors, and when looking at three or four factors at a time, because this highlights factor combinations that do not occur or do not occur equally often. Ideally (generalized resolution IV), each projection onto three factors is a full factorial, which is equivalent to zero words of generalized length 3 (the analogous request for projections onto four factors corresponds to generalized resolution V).

Note that the menu items are available only, if the active data frame is a class `design` object.

Author(s)

Ulrike Groemping

Menu.lhs *Menu for generating latin hypercube samples for quantitative factors*

Description

With this menu, latin hypercube samples for quantitative factors can be generated. The menu calls function lhs.design from R-package DoE.wrapper.

Usage

```
Menu.lhs()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with the attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry “Export design” from the “Design” menu, or the full R workspace can be stored using the appropriate entry in the “File” menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under “Data Management”.

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See Also `lhs.design` for the function behind this menu and `Menu.Quantitative` for the available designs for quantitative factors used in this software or its documentation.

`Menu.lhsTab1` *Basic information for latin hypercube samples*

Description

Basic information for latin hypercube samples

Brief statistical background

Latin hypercube designs try to fill the multidimensional space spanned by several quantitative response variables. Space filling is attempted by different methods (the simplest and least effective of which is a random sample. For more information, look at the help file for function `lhs.design` from package **DoE.wrapper**.

Inputs on Tab Base Settings

radio buttons allow to choose among different design types. The default choice (`optimum`) is reasonable, if it is feasible. `random` should only be used for a quick random look, of course no space-filling optimality can be expected by a random selection. For the other variants and the optimality criterion, see the help of the underlying function.

name of design must be a valid name. The design itself is created under this name in the R workspace.

number of runs must be provided and can be any positive integer.

number of factors must always be specified. Although not very reasonable, it is possible to choose the number of factors larger than the number of runs.
The number of factors must match the number of entries on the Factor Details tab.

number of decimal places can be provided for rounding (as in function `round`).

seed can be provided for reproducing an existing design.

Author(s)

Ulrike Groemping

See Also

See Also `lhs.design` for the function that does the calculations and `lhs-package` for the package used in that function.

`Menu.linearModelDesign`
RcmdrPlugin.DoE Linear Model Dialog for experimental data

Description

This dialog is used to specify a linear model to be fit by the `lm` function.

Usage

```
Menu.linearModelDesign(response = NULL)
```

Arguments

`response`	response to apply the linear model to

Details

The left model-formula box specifies the response variable to be used in the model; it may be a variable name or an expression evaluating to the response variable, such as `log(income)`.

The right model-formula box specifies the right-hand (i.e., predictor) side of the model. If you are interested in the details, see `lm`.

You can type directly in the model formula boxes. Alternatively, double-clicking the left mouse button on a variable in the variable-list transfers it to the left-hand side of the model (if it is empty) or to the right-hand side. You can also enter operators and parentheses using the buttons above the formula.

For regular 2-level fractional factorial experiments, initially one often wants to look at all interactions up to a certain degree (e.g. 2 or 3). This is possible by selecting the degree of the model from

the drop-down menu. For (half) normal plots of the effects, choose the degree high enough that there are `nruns-1` effects in the plot.

If the active model is from this box or could have come from this box, the initial values of the left-hand-side and right-hand-side of the formula are retained from the previous model.

Author(s)

Ulrike Groemping, adapted to this situation from menu by John Fox

See Also

`lm`, `FrF2`

`Menu.loadcatlg` *Using the menu for loading catalogues*

Description

This help file describes loading a catalogue from a loaded catalogues package. Currently, the only such package is FrF2.catlg128.

Details

This help file describes loading a catalogue package is **FrF2.catlg128**, if that package is installed.

The menu is only available if package **FrF2.catlg128** is installed.

Select the design from the drop-down list of available designs, and press OK.

If R package **FrF2.catlg128** is not installed, it has to be installed outside of the R-commander.

Author(s)

Ulrike Groemping

`Menu.mExport` *Using the export menu*

Description

This help file describes the menu items of the export menu.

Details

This help file describes the menu items of the export menu.

It is possible to

change the working directory

save the commands from the script window

save the output from the output window

save the R workspace

export the active experimental design to `rda` format and - if desired - to a csv file and/or an html file.

Author(s)

Ulrike Groemping

`Menu.mImport` *Using the import menu*

Description

This help file describes the menu items of the import menu.

Details

This help file describes the menu items of the import menu.

It is possible to

change the working directory

Load a stored R workspace

Load a design from the R workspace

and to add response data from a csv file to an experimental design that is already in the workspace or stored as an rda file.

Importing data - of course also experimental data - from other file formats is possible from R commanders data management menu.

Author(s)

Ulrike Groemping

Menu.model *Using the linear model menu*

Description

This help file describes usage of the linear model menu.

Usage

```
Menu.model()
```

Details

This help file describes usage of the linear model menu. The menu is only available if the active data set is a design with at least one response.

Select the intended response in the menu (it can also be changed afterwards, but less conveniently).

The default NULL for the degree will use the default polynomial degree from function `formula.design`.

The menu opens the standard model selection menu for linear models that is available in R-Commander, providing a convenient starting formula.

Author(s)

Ulrike Groemping

Menu.Modify *Modifying designs*

Description

This help file describes the menu for modifying designs

Details

Freshly-generated designs do not have any responses. For very small designs, response values can be added by using the data editor within the `R-commander`. For larger designs, it is preferrable to do all editing outside of `R` and re-combine response data with the structural information using the `import` menu. It is also possible to add a response from a vector or data frame within `R` by using the menu item *Add response variable(s)*.

Default analyses for experimental designs either work with all responses or with the first response of the design. It can therefore be advantageous to delete responses from the list of responses. This can be done with the menu item *Select/Deselect response variables*. At a later stage, one might want to redefine previous response variables as responses. This is also possible via that menu.

Contrary to response deselection, which leaves the variable in the design, one may want to permanently remove a column (response or other variable, but not experimental factor or block factor),

for example because some artificial data for playing are no longer needed with real response data available. This can be done with the *Remove column(s)* menu item.

Qualitative design factors come with prespecified contrast coding, which influences how analysis results appear. If the default contrasts are not the desired ones, they can be changed from the *Change contrasts* menu item. Note that this is preferrable to the built-in function from R-Commander, because it preserves the integrity of the class `design` object.

Some designs come in long format (repeated measurements, some parameter designs) but should be brought into wide format and aggregated for conducting simple analyses. The reformatting can be done with the menu item *Change from long to wide format*, the subsequent aggregation with the menu item *Aggregate design*.

The menu item *Fold design* (not yet implemented) will allow to augment a 2-level factorial designs with a second portion as large as the first one by reversing the levels of some or all factors.

The menu item *Add center points* allows to add center points to 2-level designs with only quantitative factors and certain further restrictions (e.g. no splitplot design).

The menu item *Add star portion for central composite design* allows to add a starpoints with center points to regular (fractional) factorial 2-level designs with only quantitative factors and certain further restrictions (e.g. no splitplot design, no estimability requirements).

The menu item *Augment lhs design* allows to add additional points to latin hypercube sample designs according to different types of augmentation rules available in R-package **lhs**.

Note that command line use of the packages underlying this R-Commander routine is more flexible and allows more tuning.

Author(s)

Ulrike Groemping

`Menu.oa` *Menu for generating orthogonal main effects designs*

Description

This menu is for generating orthogonal arrays for experiments for which not all factors have 2 levels. The menu calls function function oa.design from package DoE.base and has been opened internally using the function shown in the usage section, which normal users will never have to deal with.

Usage

```
Menu.oa()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander

under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

Why not 2-level factors only

It is possible within this menu to handle situations with 2-level factors only. However, most of the time, treating experiments in 2-level factors with special functions for this purpose is both more convenient and yields more efficient designs. Thus, it is recommended to use the menu for 2-level designs, if all factors are at 2 levels only. At the screening stage, where you are not yet sure which factors are important, it is very often reasonable to conduct the experiment with each factor at only 2 levels!

Exceptions: special resolution V designs in 128, 256 or 2048 runs

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with some attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry "Export design" from the "Design" menu, or the full R workspace can be stored using the appropriate entry in the "File" menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See Also `oa.design` for the function behind this menu, `Menu.General` for choice between full factorial designs and orthogonal main effects designs.

`Menu.oaTab1` *Basic information for orthogonal main effects designs*

Description

Basic information for orthogonal main effects design Menu

Details

It is useful to look at available designs for deciding which specific design should be selected. For example, when trying to automatically optimize a design, it may be more useful to start from a design with relatively few factors. If a design for two 2-level factors, one 4-level factor and one 8-level factor is sought, you may e.g. use the array `L64.2.5.4.10.8.4` or the array `L64.2.53.4.1.8.1`, which will make quite a difference in terms of run time for the optimization choice "min34".

The designs `L128.2.15.8.1`, `L256.2.19` and `L2048.2.63` are irregular resolution V designs for the 2-level factors.

Brief statistical background

The orthogonal main effects designs are of different types. They all work well if there are indeed no interactions between factors. Some of them have complete aliasing between main effects and two-factor interactions at least for some factors. It is therefore advisable to check the design before actually conducting the experiment with respect to its potential analysis options and biases.

Note that it is usually preferable to create an experiment with solely 2-level factors from the special menu for 2-level situations (exceptions: resolution V nonregular arrays in 128, 256 or 2048 runs, cf. Details section). If there is just one factor at more than 2 levels, it may also be useful to simply cross this factor with an otherwise 2-level design.

If only relatively few of the columns are used, it is possible with some orthogonal arrays to also estimate interactions, or at least to estimate main effects unbiasedly even in the presence of interactions. This may e.g. be possible for some of the arrays in 2, 4 and 8 or 16 level factors (that have arisen from regular fractional factorials). Automatic optimization can help finding such designs.

It is highly recommended to diagnose the structure of the design before using it for experimentation, e.g. using the "Summarize design ..." item in the "Inspect design" menu.

Inputs on Tab Base Settings

name of design must be a valid name. The design itself is created under this name in the R workspace.

number of factors must always be specified. The number of factors must match the number of entries on the Factor Details tab.

specific array can be selected from the drop-down list; this implies that this particular array is used for generating the design; the array name indicates its number of runs and the maximum possible numbers of factors with various numbers of levels. For example, the array `L12.2.2.6.1` can accomodate 2 factors at 2 levels each and one factor at 6 levels

Automatic optimization can be selected from the drop-down list. The default is no optimization ("none"). "min3" requests minimization of generalized words of length 3 (cf. `generalized.word.length`), while "min34" requests further optimization among several equally-good length 3 designs w.r.t. length 4 words. Automatic optimization may sometimes take very long or use up too many resources; usability will be better for designs with fewer factors (cf. also details section).

minimum number of runs can be selected from dropdown, but is not needed

minimum residual df is per default 0 and can be set to any positive integer number; it specifies the minium number of extra runs over and above what would be needed for a model with main effects for all factors; for example, when using the design L12.2.2.6.1 for two 2-level factors and one 6-level factor, the model with all main effects requires `1+2*(2-1)+1*(6-1)=8` degrees of freedom, i.e. there are four extra degrees of freedom for pure error or lack of fit

replications is the number of times each experimental run is conducted. If larger than 1, each run is conducted several times. If the checkbox next to the number of replications is checked, it is assumed that the experiment involves repeated measurements for one setup of the experimental run; if it is not checked, the experimental run itself is replicated with everything relevant newly set up (much more valuable than repeated measurements, unless the key driver of variability is in the measuring step). If the check box is not checked, the experiment will be randomized separately for each round of replications (first all first runs, then all second runs etc.).

randomization settings should normally not be changed; you can provide a seed if you want to exactly reproduce a randomized design created in the past. Unchecking the randomization box will produce a non-randomized experiment. This is usually NOT recommended.

Inspect available designs based on requests on the number of runs (from, to) and decisions whether or not child `arrays` are to be shown, and whether to only display designs that can accomodate factors with the factor level settings from the "Factor Details" tab. The button will create a printout of the available designs, based on the work horse function `show.oas`. The menu remains open and needs to be maximized again after looking at the displayed designs.

Author(s)

Ulrike Groemping

See Also

See Also `oa.design` for the function that does the calculations and `Menu.General` for overall help on the general factorial design menu.

`Menu.Optimal` *Optimal designs*

Description

Optimal designs start from the assumption that a certain model structure is suitable for the problem at hand and optimize the design points for estimation of this model structure.

Optimal designs

R package **AlgDesign** offers optimal designs. This GUI currently offers the so-called D-optimal designs for standard situations only (no augmentation, no blocking, little tuning of parameters). D-optimal designs are most widely-spread but not necessarily best for all purposes. The implementation of D-optimal designs relies on a candidate data set.

D-optimal designs are liked because they are very flexible in terms of run size. Furthermore, they are the only designs for which it is very easy to incorporate constraints. Whenever the specified run size offers enough degrees of freedom and the candidate set contains enough variety for estimating the model, a D-optimal design can be found. However, a design which is optimal for the chosen run size can be very bad in absolute terms; like always, miracles are not possible.

Furthermore, note that there are many situations in (industrial) experimentation, for which assuming a model structure is ambitious if not inadequate. For such situations, other types of designs should be used.

Usage

As was already mentioned, the implementation of D-optimal designs in this GUI requires a candidate design, which can be any data frame, e.g. a full factorial design, an orthogonal array design or a latin hypercube design. The menu item *Candidate design ...* offers shortcuts for selection or creation of candidate designs.

The menu item *D-optimal design ...* allows creation of a D-optimal design, using the active data set (possibly restricted to exclude inadequate combinations) as the candidate set.

Author(s)

Ulrike Groemping

References

Atkinson, A.C. and Donev, A.N. (1992). *Optimum experimental designs*. Clarendon Press, Oxford.

Federov, V.V. (1972). *Theory of optimal experiments*. Academic Press, New York.

Wheeler, R.E. (2004). *Comments on algorithmic design*. Vignette accompanying package **AlgDesign**. `../../AlgDesign/doc/AlgDesign.pdf`.

See Also

See also optFederov and Dopt.design

Menu.param *Help on using the menu for Taguchi inner-outer array designs*

Description

Using the menu for Taguchi inner-outer array designs

Usage

```
Menu.param()
```

Using the menu for Taguchi inner-outer array designs

The menu is available only, if there are at least two designs.

Provide a name for the new design.
Choose an inner array and an outer array.
Decide on long or wide format.

WARNING: Currently, the functionality works well for standard situations only. Whenever there are variables other than the factors in either of the designs (even a block variable), problems are likely to occur at least for the wide version designs. This will be worked on, but not top priority.

Author(s)

Ulrike Groemping

See Also

See also param.design for the underlying function

Menu.pb2level *Menu for generating non-regular fractional factorial 2-level designs*

Description

With this menu, non-regular (=screening) orthogonal 2-level designs can be generated. The menu calls function function pb from package FrF2 and has been opened internally using the function shown in the usage section, which normal users will never have to deal with.

Usage

```
Menu.pb2level()
```

Prior Remark for Windows users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode. It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

User form structure

This user form is structured as a notebook with several tabs, and each tab has its own specific help button. It is recommended to work through the tabs from left to right, although it is possible to switch between tabs back and forth.

The buttons to the right of the tabs store or load form settings (cf. next section) for the complete form with all tabs.

What happens when pressing OK?

On OK, the menu will create an experimental design as an R data frame with some attributes (`desnum`, `run.order` and `design.info`). The list `design.info` contains, among other things, the element `creator`, which contains the stored form.

If requested on the Export tab, the R workspace with ONLY the experimental design will be saved at the specified location with the ending `rda`, and an Excel-readable html-file or a csv-file will be generated, depending on the users choice. Alternatively, it is possible to permanently store an R workspace with ONLY the design later using the menu entry "Export design" from the "Design" menu, or the full R workspace can be stored using the appropriate entry in the "File" menu of R commander. The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Storing and loading form settings

The settings of the form (all tabs) can be saved at any time with the button "Store form", which generates an object (of class `menu.design2pb`). The button "Load form" can be used for loading these settings into the form again in order to continue working on the entries. The purpose of this functionality: work can be safely interrupted, or a finished design can be modified after a team discussion or ...

The stored inputs will be an object within the R session. If they are to be kept for future R sessions, the R workspace must be stored on disk or on another storage medium (file type `RData` or `rda`). This can be done from the file menu of the R commander (usually the leftmost menu). The menu entry for loading the stored R workspace in a future session can be found under "Data Management".

Within an R session, the latest stored form inputs are normally loaded automatically on next usage of the menu. If you want to restart fresh, the button "Reset form" sets everything to default again.

As pointed out above, if a design is actually generated by pressing the OK button, the menu settings are saved within the design object. The button "Load form" knows how to load the form settings from designs, so that it is not necessary to separately store the form settings in this case.

Author(s)

Ulrike Groemping

See Also

See Also pb for the function behind this menu and Menu.2level for choice between regular fractional factorial designs and screening designs and DoEGlossary for the DoE-related terminology used in this software or its documentation.

Menu.pb2levelTab1 *Basic information for 2-level screening designs*

Description

Basic information for 2-level screening design Menu

Brief statistical background

Most of the screening designs are so-called Plackett-Burman designs. Exceptions occur, where Plackett and Burman did not suggest a design or where the Plackett-Burman designs coincide with the regular fractional factorial designs which are not so suitable for screening purposes. For more information, look at the help file for function pb from package **FrF2**.

Inputs on Tab Base Settings

name of design must be a valid name. The design itself is created under this name in the R workspace.

number of runs must be provided and must be a multiple of 4.

the check box below the number of runs is relevant for 12 run designs only: if checked, the 12 run design is in the arrangement as provided by Taguchi, otherwise it is a Plackett-Burman design. (The two are equivalent to each other, but comparison is tedious.)

number of factors must always be specified. It must be smaller than the number of runs. The function always creates a design with the number of factors one less than the number of runs. Those factors not used in the experiment are named e1, e2 etc. and serve for creating half-normal plots of the effects. The number of factors must match the number of entries on the Factor Details tab.

replications is the number of times each experimental run is conducted. If larger than 1, each run is conducted several times. If the checkbox next to the number of replications is checked, it is assumed that the experiment involves repeated measurements for one setup of the experimental run; if it is not checked, the experimental run itself is replicated with everything relevant newly set up (much more valuable than repeated measurements, unless the key driver of variability is in the measuring step). If the check box is not checked, the experiment will be randomized separately for each round of replications (first all first runs, then all second runs etc.).

randomization settings should normally not be changed; you can provide a seed if you want to exactly reproduce a randomized design created in the past. Unchecking the randomization box will produce a non-randomized experiment. This is usually NOT recommended.

Author(s)

Ulrike Groemping

References

~put references to the literature/web site here ~

See Also

See Also pb for the function that does the calculations and Menu.2level for overall help on the 2-level design menu.

Menu.pickcube *Pick the cube portion from a central composite design*

Description

This help file describes how to use the menu for picking the cube portion from a central composite design.

Details

Specify the name of the new cube only design (default: name of active design with .cubeonly appended), and press OK. The resulting reduced design can be used for effects plots of the cube portion etc. This is especially useful, if the star point response values are not yet available, and preliminary analyses are to be run.

NOTE: If the intention is sequential experimentation (as is often the case), it is usually more desirable to first specify a standalone cube with center points and analyze that design, and to only append a star portion later, if needed.
While subsequent reduction of a central composite design to its cube portion does work for many purposes, the resulting design is not as clean as a cube portion created from scratch; for example, augmenting it with a star portion again will fail. (This could of course be fixed, but this fix has low priority.)

WARNING: Package **RcmdrPlugin.DoE** has been subjected to some testing, but must still be considered a beta version. It is strongly recommended to **always keep a master copy of your valuable experimental data safely stored without touching it**. (This would even hold for a final version of any software, but is of course far more important for a beta version.

Author(s)

Ulrike Groemping

See Also

See also function pickcube for the function that does the work

Menu.plot *Help on using the menu for plotting selected factors*

Description

Help on using the menu for plotting selected factors

Help on using the menu for plotting selected factors

This menu allows to select experimental factors for plotting.

The menu is particularly useful for viewing the structure of general orthogonal arrays. In mosaic plots, up to three or four factors can be easily viewed together.

Author(s)

Ulrike Groemping

See Also

See also the `plot` method for class `design` for the underlying function

Menu.QuantDesignAnalyze *Analyze designs for quantitative factors*

Description

This is a brief explanation on the menu for analysis of designs for quantitative factors.

Available analysis options

The menu item *Response surface model ...* creates a response surface analysis with first order (FO) terms, and potentially also two-factor interactions (TWI) and/or pure quadratic (PQ) terms. The resulting object has class `rsm` and can be postprocessed with the subsequent menu items.

The menu item *Steepest slope ...* supports moving the experimental range towards a more promising area, based on a first order model or on a second order model with a saddle point.

The menu item *Plot response surface ...* supports creation of contour, perspective or image plots of a response surface model, and can also be used for linear models with quantitative variables.

It is of course also possible to use other analysis facilities within the R-commander, or to use the command-line facilities offered in package `rsm` and elsewhere.

Author(s)

Ulrike Groemping

References

Box G. E. P, Hunter, W. C. and Hunter, J. S. (2005) *Statistics for Experimenters, 2nd edition*. New York: Wiley.

`Menu.Quantitative` *Designs specifically tailored to quantitative factors*

Description

This GUI covers classical response surface designs (central composite and Box-Behnken) as well as so-called latin-hypercube samples for factors that can and should be set to many different levels. This help file is about when to apply which of these.

Central composite designs

Central composite designs can be generated from scratch or as an extension to 2-level full or fractional factorials generated with the 2-level dialogue or with function `FrF2`. Their rationale is explained `here`. They usually have five different levels per experimental factor.

Box-Behnken designs have only three levels for each factor. They are explained `here`. They can not be generated by augmenting an existing design.

Latin hypercube designs - if used with optimization which is strongly recommended - try to fill the experimental space with points in an efficient way. Note that they are NOT trying to optimize the design w.r.t. a specific statistical model. Such optimal plans are available outside the R Commander with package `AlgDesign`. Simple versions of these have already been implemented into the R Commander.

Author(s)

Ulrike Groemping

References

Box G. E. P, Hunter, W. C. and Hunter, J. S. (2005) *Statistics for Experimenters, 2nd edition*. New York: Wiley.

See Also

See Also `ccd.design` and `ccd.augment` for the functions behind the central composite designs, `bbd.design` for the function behind the Box-Behnken designs, and `lhs.design` for the function behind the latin hypercube samples.

Menu.responses *Help on using the menu for response selection*

Description

Help on using the menu for response selection

Help on using the menu for response selection

This menu allows to select numerical variables to become responses or to deselect current responses so that they remain on the data frame but loose their response status.

If you want to permanently remove a response altogether, use the remove columns menu instead.

Author(s)

Ulrike Groemping

See Also

See also `class-design` for the underlying function

Menu.rsm *RcmdrPlugin.DoE response surface model Dialog for experimental data*

Description

This dialog is used to specify a response surface model to be fit by the `rsm` function.

Usage

```
Menu.rsm(response = NULL, factor.names = NULL)
```

Arguments

`response`	response to apply the response surface model to
`factor.names`	character vector of factor names to apply the response surface model to

Details

The LHS model-formula box specifies the response variable to be used in the model; it may be a variable name or an expression evaluating to the response variable, such as `log(income)`.

The RHS model-formula box specifies the right-hand (i.e., predictor) side of the model. If you are interested in the details, see `rsm`. The formula consists of first order (FO), quadratic (PQ) and interaction (TWI) terms. Individual variables can be omitted from the formula or re-entered by editing the dialog. It is also possible but usually not necessary to enter extra variables. An existing blocking variable is per default included.
If the analysis is coded, the variables must be given as `x1`, `x2`, and so forth, otherwise by their original names.
For redoing from within this dialog, manually empty the RHS model-formula box; when clicking the FO, TWI, PQ and SO while some factors are selected in the variables box, the respective terms are entered to the model.

Author(s)

Ulrike Groemping, adapted to this situation from menu by John Fox

See Also

`rsm`, `rsmformula`, `code.design`

Menu.rsmmodel *Using the response surface model menu*

Description

This help file describes usage of the response surface model menu.

Usage

```
Menu.rsmmodel()
```

Details

This help file describes usage of the response surface model menu. The menu is only available if the active data set is a design with at least one response.

Select the intended response in the menu (it can also be changed afterwards, but less conveniently).

The default degree is 2; degree 1.5 means inclusion of interactions but not of quadratic terms.

The menu opens a response surface model selection menu, providing a convenient starting formula.

Author(s)

Ulrike Groemping

Menu.steepest *Using the steepest slope menu*

Description

This help file describes usage of the steepest slope menu.

Usage

```
Menu.steepest()
```

Details

This help file describes usage of the steepest slope menu for response surface models. The menu is only available if there is an rsm model available with at least one response.

It is necessary to select the model (as rsm models cannot be active models in R Commander). The default degree is 2, but 1 is also possible.

The checkbox provides a switch between classical steepest ascent in first order models (start in the middle) or models that are almost linear with far out optima (start in a saddle point). The latter option is reasonable for second order models, for which the Hessian reveals that there is still room for improvement = the stationary point is a nearby saddle point, it is more appropriate to try increasing (decreasing) the response starting at that saddle point and walking into both directions.

Author(s)

Ulrike Groemping

Menu.summarize *Using the summarize menu*

Description

This help file describes summarizing a design that is already in the R workspace but is not necessarily the currently active data set.

Details

This help file describes summarizing a design that is already in the R workspace but is not necessarily the currently active data set. The menu is only available if at least one (object of class) design exists in the R workspace.

Select the design from the drop-down list of available designs. If a plot or a table is wanted, check the respective check box. On pressing OK, a printed summary and - if requested - a simply-structured table are generated, and - again if requested - the design is plotted. For details on the functions behind these activities, see `summary.design`, `table`, and `plot.design`.

The tables and plots from this menu will only be useful for very small designs. For obtaining more tailored tables, using the "Contingency Tables" dialogue from the `Analyze` menu. For different types of plots, the `Graphics` menu has some offerings.

Author(s)

Ulrike Groemping

`Menu.tab` *Help on using the menu for tabulating selected factors*

Description

Help on using the menu for tabulating selected factors

Help on using the menu for tabulating selected factors

This menu allows to select experimental factors for tabulating (with command `table`).

The menu is particularly useful for viewing the structure of general orthogonal arrays. Up to three or four factors are most easily viewed together.

Author(s)

Ulrike Groemping

See Also

See also `table` for the underlying function

`Menus` *Help on using R-Commander menus*

Description

Important instructions for windows users for making the menus work well

Important Usage Note for Windows Users

The most important technical point about this menu and the R-Commander in general:
The R GUI should be installed with the Single Document Interface (SDI) mode instead of the default MDI mode. John Fox, the author of R-Commander, has described how to change MDI to SDI, if R has been installed in the wrong mode (`https://socialsciences.mcmaster.ca/jfox/Misc/Rcmdr/installation-notes.html`). It is strongly discouraged to run the R-Commander under the MDI mode, as it happens very frequently that windows suddenly disappear. They usually have iconized only and can be retrieved from the taskbar; nevertheless, this behavior can be very annoying and can be avoided by using R in SDI mode.

Author(s)

Ulrike Groemping

`PlotMeansDoE.menu` *Menu for main effects and interaction plots for general factorial designs*

Description

The menu for main effects and interaction plots for general factorial designs allows to plot individual main effects plots (one factor selected) or interaction plots (two factors selected).

Details

The menu calls the **RcmdrMisc** function `plotMeans` for creating simple main effects or interaction plots for factorial designs. It is available whenever the active data frame is an experimental design with at least one response that has at least one factor (not a quantitative variable but a factor with categories).

Arithmetic means for any numeric variable can be plotted separately for the levels of one factor (main effects plot) or the level combinations of two factors (interaction plot). An interaction plot created by the menu always makes the factor with fewer levels the legend factor. (This can only be changed by modifying the command in command line programming.)

For designs with two-level factors only, the analogous menu for 2-level factors can do several plots at once.

Author(s)

Ulrike Groemping

See Also

See also `plotMeans` for the underlying function

Index