Fachbereich II – Mathematik - Physik - Chemie

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN

University of Applied Sciences

Ulrike Grömping

Tutorial for designing experiments using the R package RcmdrPlugin.DoE

Anleitung zur Versuchsplanung mit dem R-Paket RcmdrPlugin.DoE (englischsprachig)

Reports in Mathematics, Physics and Chemistry

Berichte aus der Mathematik, Physik und Chemie

Reports in Mathematics, Physics and Chemistry

Berichte aus der Mathematik, Physik und Chemie

The reports are freely available via the Internet:
http://www1.beuth-hochschule.de/FB_II/reports/welcome.htm

04/2011, November 2011

© 2011 Ulrike Grömping
Tutorial for designing experiments using the R package RcmdrPlugin.DoE
Anleitung zur Versuchsplanung mit dem R-Paket RcmdrPlugin.DoE
(englischsprachig)

Abstract

This tutorial is meant for users who are familiar with basic design of experiment concepts and want to use the GUI interface provided by *R* package **RcmdrPlugin.DoE** for creating simple designs, adding response values to them, and doing basic analyses of the results. It is not a comprehensive user's guide to package **RcmdrPlugin.DoE**, and it is neither an introduction to design or analysis of experiments. Nevertheless, the author could not resist including some remarks on adequate choice of designs or analysis methods.

Accompanying material:
http://prof.beuth-hochschule.de/fileadmin/user/groemping/downloads/Tutorial_DoEGUI_exampleData.zip

Tutorial for designing experiments using the *R* package RcmdrPlugin.DoE

Version 1

1 About this tutorial

This tutorial explains usage of the *R* package **RcmdrPlugin.DoE** for GUI-based design of experiments (DoE) in *R*. It targets readers who already have a basic understanding of DoE but do not necessarily know much about *R*; nevertheless, readers are assumed to be able to start an *R* session. Textbooks such as Box, Hunter and Hunter (2005) or Montgomery (2001) are recommended for details on DoE. Some comments on advantages and disadvantages of designs or analyses are included in this tutorial, especially for aspects which are frequent causes of trouble in the author's experience.

   This tutorial has been finalized with *R* 2.14.0 using *R* Commander (**Rcmdr**) version 1.7-2, **RcmdrPlugin.DoE** version 0.11-5, based on **DoE.base** version 0.22-8, **FrF2** version 1.2-10, **DoE.wrapper** version 0.8-6, **BsMD** version 0.6-5.2, **lhs** version 0.5, **AlgDesign** version 1.1-7 and **rsm** version 1.40. Data files for all examples are provided on the author's website as comma separated value files (in European notation, i.e. with decimal comma and semicolon as separator); some example files are also provided as *R* workspaces. It is assumed throughout that *R* Commander is configured to show commands in the script window.

1.1 Structure

The second section gives a brief introduction into how to start *R* commander and the **RcmdrPlugin.DoE** plugin and how to load and save *R* workspaces and program files, the third section walks readers through all important steps of creating, inspecting and exporting a design, of (re-)importing response data and analyzing a design, based on a simple screening design. After this general overview, Section 4 points out how to work with the GUI efficiently by using the appropriate help buttons and the store, load and reset form buttons on design generation dialogs. Section 5 then looks at design creation in more detail, demonstrating creation of most design types with simple examples:

- A blocked regular fractional factorial design as a follow-up experiment to the screening design used in Section 3 is produced.
- A full factorial experiment from the literature is reproduced and its analysis is sketched.

Subsequently,

- a general orthogonal array
- and a D-optimal design for the same experimental problem

are created and – using the appropriate subset of data from the full factorial – also analysed. Last but not least, two designs for quantitative factors,

- a central composite design and
- a latin hypercube design,

are created, the latter without any analysis considerations. The subsequent Section 6 provides an example for repeated measurements, long and wide versions of designs, and

for the aggregation functions that can be useful in this context. Section 7 shows how to create and handle a Taguchi parameter design with the software, based on a literature example. The final Section 8 briefly explains how to use the DoE GUI as a starting point for getting into *R* command line programming.

1.2 Example data files

All files mentioned here can be downloaded at http://prof.beuth-hochschule.de/fileadmin/user/groemping/downloads/Tutorial_DoEGUI_exampleData.zip. The following data files are available:

- **`Screen.example.withresp.csv`** for Section 3

- **`Screen.example.repeat.wide.withresp.csv`**:
  potato cannon example with individual measurements in wide format;
  usable for Section 6

- **`FullFactorial3.4.withresp.csv`**
  (computer run times example, for Section 5.2)

- **`oa.3.4.withresp.csv`** (computer run times example, for Section 5.3)

- **`Dopt.3.4.withresp.csv`** (computer run times example, for Section 5.4)

- **`inner.rda, outer4.rda, paramDesign.InjectMold.long.rda`**
  `paramDesign.InjectMold.long.withresp.csv` and
  `paramDesign.InjectMold.withresp.HSS.csv` (all for Section 7)

- **`InjectMoldLong.rda`** with the long format injection molding experiment as a combined experiment of both inner and outer array factors and a suitable response file **`InjectMoldLong.withresp.HSS.csv`** (both for Section 7)

The design variables are also included in the **`csv`** files in order to avoid any mixups. The designs themselves are generated throughout this tutorial. For users who want to skip some of the design generation, **`rda`** files and **`csv`** files without responses are also available for all setups (for example, **`Screen.example.rda`**, **`Screen.example.csv`**). The **`rda`** files can easily be combined with the **`csv`** files with responses into files for analysis purposes using the import tools described in Section 3.5.

1.3 Terminology/notation

RcmdrPlugin.DoE is henceforth also called "the software" or "the DoE GUI". Its usage is explained using various worked examples; it is recommended that readers work through these themselves while getting used to the software, particularly for Section 3. Initially, every step is explained in detail, including many figures that show menus or dialog windows in different stages. In later examples, steps analogous to the ones that were previously explained are not explained again; only the additional steps are illustrated by figures. Apart from figures, menu items are provided in the notation (e.g.)

Design → Import → Change working directory …

The above line tells readers to first select the top level *Design* menu, within that the *Import* menu and within that the menu item "Change working directory …". Occasional programming code lines are always provided in **`courier font`**, likewise is computer output.

2 Basic Tasks

2.1 Starting *R* Commander and its DoE Plugin

Within a new *R* session, the command

```
require(RcmdrPlugin.DoE)
```

opens the *R* commander with its plugin **RcmdrPlugin.DoE** included. In an open *R* commander session, you can check that **RcmdrPlugin.DoE** is available by looking for the *Design* menu (cf. Figure 1). As long as the *R* commander has not been previously used within the existing *R* session, the above command works. After the *R* commander has already been used but has been closed, the command

```
Commander()
```

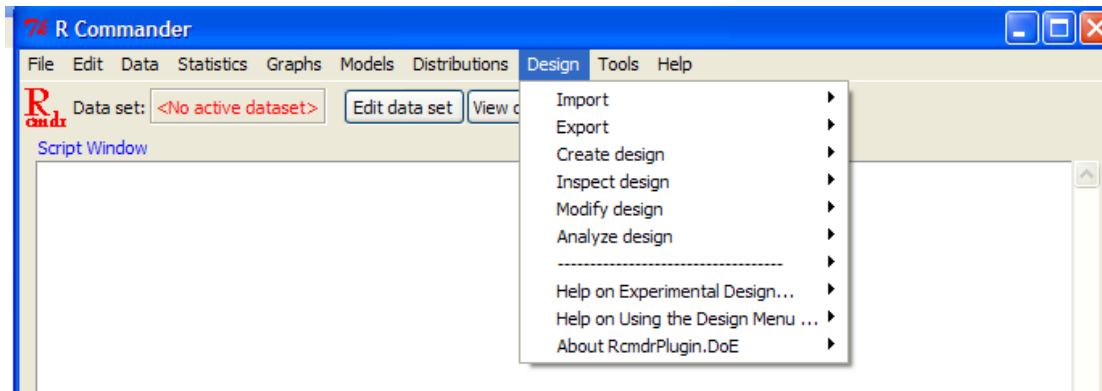opens a new *R* commander session, which will also include the *Design* menu, if it was included before.

Figure 1: The *R* commander menu with expanded "Design" menu

If you want to load the Plugin **RcmdrPlugin.DoE** into an open *R* commander session that does not yet include the *Design* menu, load the plugin from the menu item

    Tools → *Load Rcmdr plugin(s) …*

Details on (a much older version of) the *R* Commander can also be found in Fox (2005).

2.2 Storing and loading *R* workspaces, code and output

The *Import* and *Export* sub menus of the *Design* menu (cf. Figure 1) contain a few utilities for general *R* related tasks. These are partly unchanged replications from the *File* or *Data* menu of *R* Commander (e.g. the *Change working directory* item of both sub menus); other items have been modified to suit experimenters' needs (e.g. *Load workspace* in *Import* sub menu).

    An *R* workspace (file with suffix `rda` or `Rdata`) contains all kinds of objects that have been generated in an *R* session, for example data frames[1] that contain an experimental plan and possibly some response values or lists of stored dialog settings (cf. Section 4.3). Specifically, note that the code history or the output generated in a session are NOT part of the *R* workspace; these must be separately stored, if desired. The dialog invoked by

    Design → *Import* → *Load R workspace …*

---

[1] "data frame" is the *R* expression for the data table with all its columns.

loads an *R* workspace and makes the first data frame of class **design**[2] encountered in the workspace listing the active data set in *R* Commander (shown below the menu bar at the top of the *R* Commander window). The

  Design → *Export*

menu allows to save the content of the script window (file name should have the suffix **R**), the content of the output window or the *R* workspace with all its content (file name should have the suffix **Rdata** or **rda**). It is also possible to separately save a particular data frame of class **design**, i.e. a specific experiment, using the dialog invoked by the menu item

  Design → *Export* → *Export experiment …*

which is grayed out, unless there is an experiment in the *R* workspace.

2.3 Using general *R* Commander facilities

Apart from the *Design* menu, analysis of experimental data is also supported by the *Statistics*, *Graphs* and *Models* menus. The *File* and *Data* menus are useful for more advanced file handling and data management tasks. For example, the *File* menu allows to open a stored script file, while data file import from some foreign language formats is handled from within the *Data* menu, which also allows to add new calculated variables to the data frame.

3 Step by step through using the Design menu

In this chapter, a simple 2-level screening design is created, inspected, exported, populated with response data and analyzed. Before the functionality is explained, the example experiment is presented in some detail.

3.1 The example experiment

The example has been published on the internet (Mayfield 2007) and deals with the layout of a so-called potato cannon. It is reproduced here for demonstration purposes. Emphasis is on the technical aspects – the author would have chosen a different design for the question at hand, but is of course nevertheless grateful to Philip Mayfield for the detailed published example.

The experiment investigates a so-called potato cannon that works according to the following principle: the cannon is powered by an air chamber set under pressure and then released by a valve which gets triggered by battery-driven electricity. The air sets a wad into motion which will move the golf ball through a barrel of a certain length into the air. The angle of the barrel can also be modified.

- Experimental goal: find settings for the experimental factors such that a golf ball (or potato or …) consistently travels a far distance (the farther the better).
- Eight experimental factors:
    - Air volume (size of air chamber)
    - Pressure
    - Valve (two different ones from the same manufacturer whose valves are known to be quite variable)

---

[2] Data frames of class **design** are designs created by the design creation functions of one of the packages **DoE.base**, **FrF2**, **DoE.wrapper** or **RcmdrPlugin.DoE**, or possibly other packages whose authors may have adopted this structure; most of the analysis facilities from the *Design* menu are for such designs only.

- o Voltage (one or three 9V batteries)
- o Barrel length (4 or 6 feet)
- o Angle (45 or 60 degrees)
- o Wad type (paper or cloth)
- o Ball type (white=expensive, pink=cheap)
- Execution: twelve different experimental runs have been set up; at each setup, four shots have been conducted, and the "distance traveled" by the golf ball has been measured for each shot.
- Response variable: Interest is in the distance the golf ball traveled (in feet).
  The way the experiment has been executed implies that the individual shots are repeat runs only but no proper replications (for example, the cannon has not been re-assembled or re-aligned between shots). Therefore, it is not allowed to treat them as independent runs, and the analysis is conducted for their averages and standard deviations. The main response variable for the purpose of this tutorial is the average distance traveled (called *y*).

3.2 Creating a design

The menu opened by

　　Design → Create design

gives access to various design creation dialogs (cf. Figure 2), structured by topics.
The menu item

　　Design → Create design → Screening design…

opens the dialog shown in Figure 3. The designs created by that dialog are meant for screening many factors, all at 2 levels, within a small experimental array. These designs make most sense, if it is expected that many of the screened factors will be unimportant and future experiments will concentrate on few of the screened factors only.
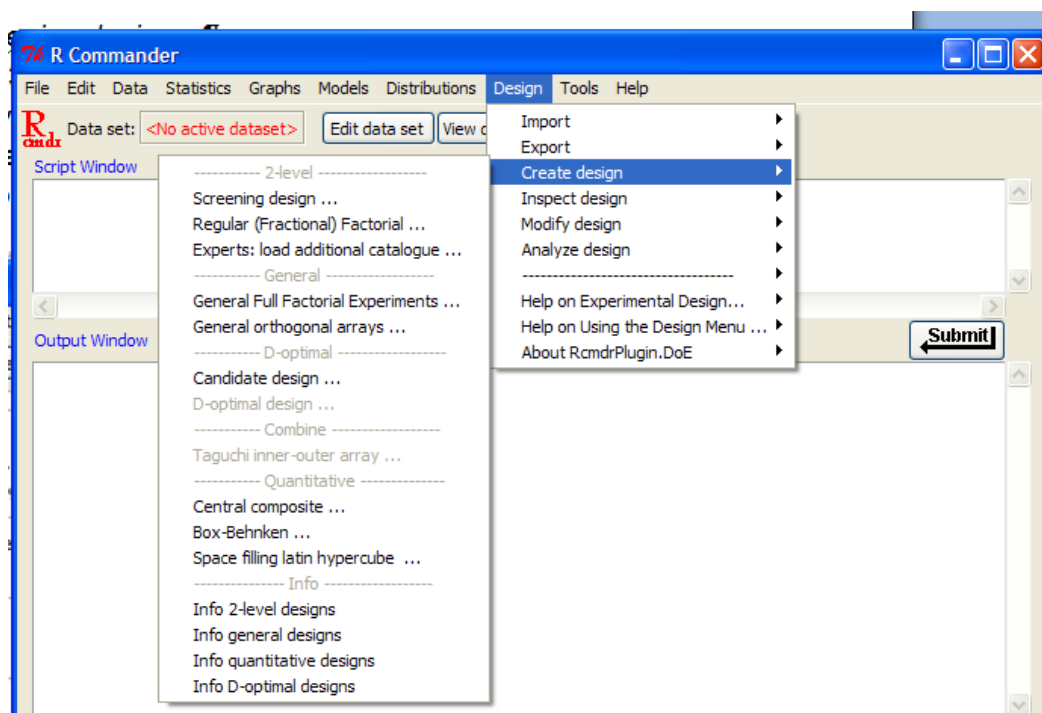

Figure 2: *Create design* menu

Usage of the "Create 2-level screening design" dialog is similar to many other dialogs and will serve as a model for the others. We will generate the potato cannon example design in 8 factors and 12 runs, as introduced in Section 3.1. The design is based on two settings for each factor, with the goal of maximizing the distance traveled by golf balls shot by a potato cannon (cf. Section 3.1).

Figure 3: The "Create 2-level screening design" dialog (default settings)

On the "Base Settings" tab (cf. Figure 3), the name of the design, its number of runs and number of factors can be specified. Figure 3 shows the default settings. We generate the design `Screen.example` with 8 factors in 12 runs by changing the entries for "Name of new design", "Number of runs" and "Number of factors" accordingly. Because of the structure that was used by Mayfield (2007), matching the correct response data to experimental runs is simplified by checking the box for "12 run design in Taguchi order". Further details on the "Base Settings" tab:

- Center points are not included (these would be unusual for screening designs).

- and the design is neither replicated. (It was discussed above that there are repeated measurements; their direct treatment with **R** is shown in Section 6. In this chapter, their mean and standard deviation are directly entered as responses.)

- Randomization is done with the seed that was randomly generated by the program. If you want to make sure to be able to reproduce exactly the same randomization later, it may make sense to record that seed (it can also be retrieved from the stored design object, but not conveniently so).

Figure 4: The "Factor Details" tab of the "Create 2-level screening design" dialog

Figure 5: "Factor Details" tab entries for the potato cannon example

Moving to the "Factor Details" tab (cf. Figure 4) allows to customize the factor details. For an initial try-out of design creation it may be sufficient to leave this tab untouched, which would generate a design with default factor names and levels. For the final design, it will at least be advisable to assign factor names, often also levels for each factor. For some cases, it may be sufficient to have common levels for all factors, e.g. "current" and

"proposed". The comment field for each factor allows to include descriptive text. This text is only relevant if the "Export" tab is used for exporting the design.

In the example used here, we will work with individual factor levels for each factor. Therefore, the box for "common factor levels" is unchecked, which activates the text boxes for individual factor level entries, and the factor details are filled in factor by factor. Moving along with the Tab key makes text entry smooth and easy. Figure 5 shows the Factor details tab with all fields filled in. Pressing the OK button on the outer part of the dialog (regardless which tab is active) produces the design and writes the command line into the **R** commander script window (cf. Figure 6).
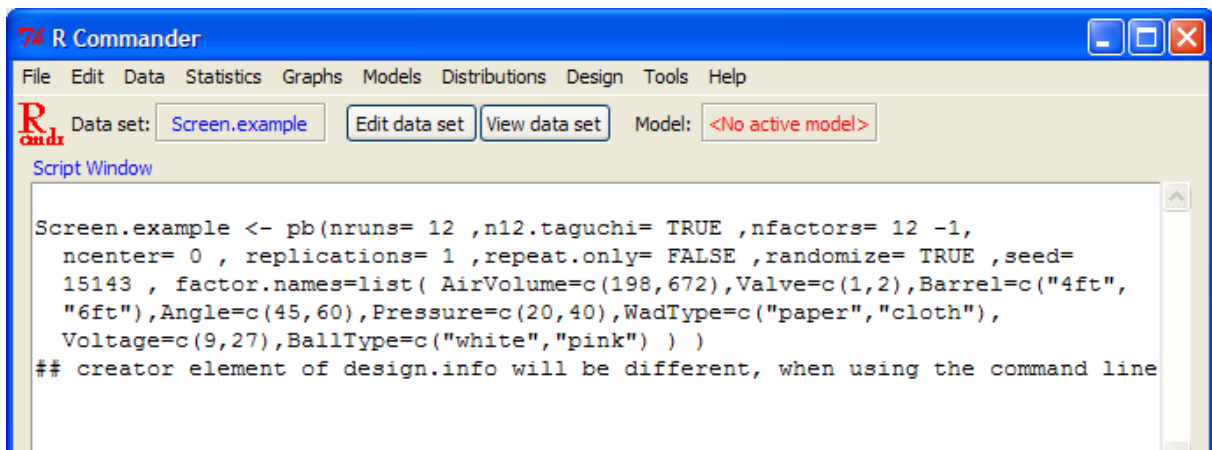
Figure 6: Logged command in the **R** commander Script Window[3]

3.3 Inspecting the created design

Before using a freshly-created design for experimentation, it should be inspected in detail in order to make sure it does what was intended. The design can be inspected by pressing the "View data set" button or by using items from the

Design → Inspect design

menu (cf. Figure 7). The item "*Display active design …*" allows to display the design in the output window (choosing between actual run order and standard order). Other menu items show a brief summary, plot or tabulate selected factors or display the `design.info` attribute of the design. The latter is mainly for experts but also allows to identify the seed used in design generation, which may sometimes be convenient if an exact same design is to be reproduced.

For example, the "Summarize active design" menu item produces the output shown in Figure 8 in the **R** commander output window. The summary output strongly depends on the type of design. For a screening design, it shows the run number and the factor levels only.

---

[3] A remark on the code shown in Figure 6: There is a specialty about the function `pb` behind 2-level screening designs: Contrary to all other designs, here the number of factors (`nfactors`) should always be set to `nruns – 1` in order to include unused columns into the design for analysis purposes. The DoE GUI automatically does this. The actual number of required factors is hidden in the option factor.names, where there is a specification for each actual factor. In the final design, the dummy factors, here e1, e2, e3 with "e" indicating error, can only be recognized by their name (cf. Figure 8 below).
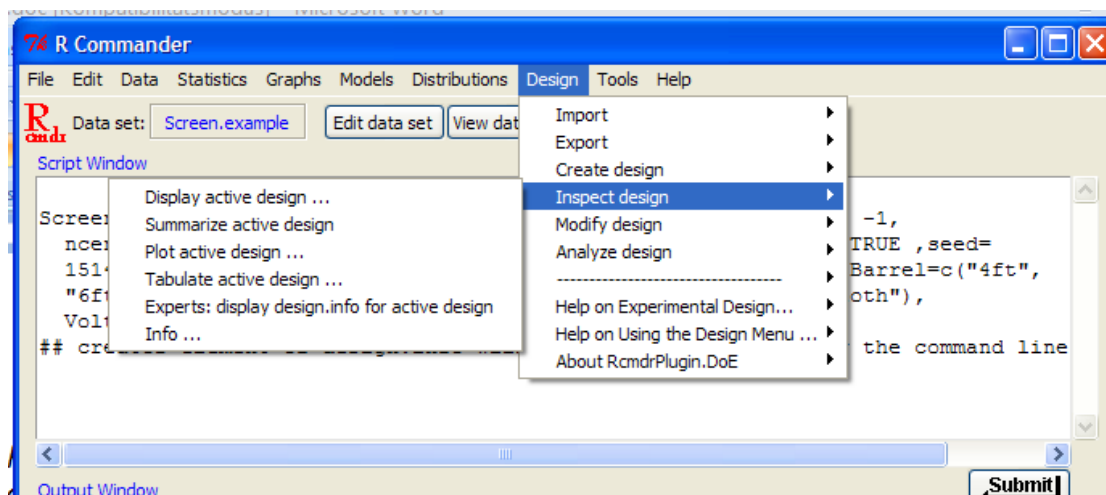
Figure 7: The *Inspect design* menu
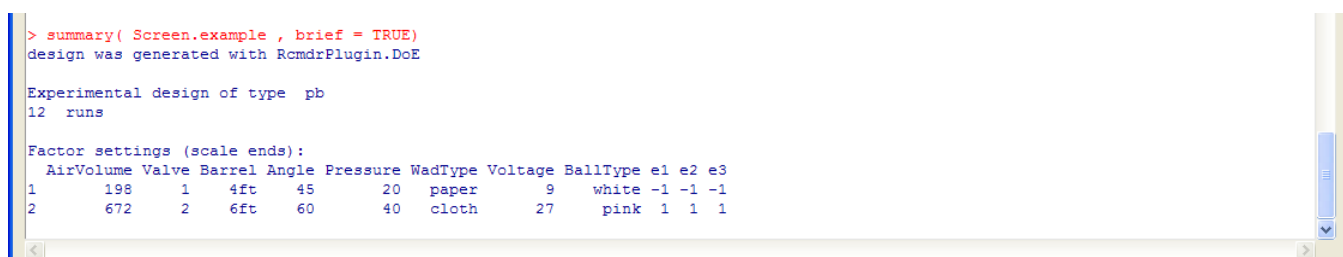
Figure 8: Output of design summary
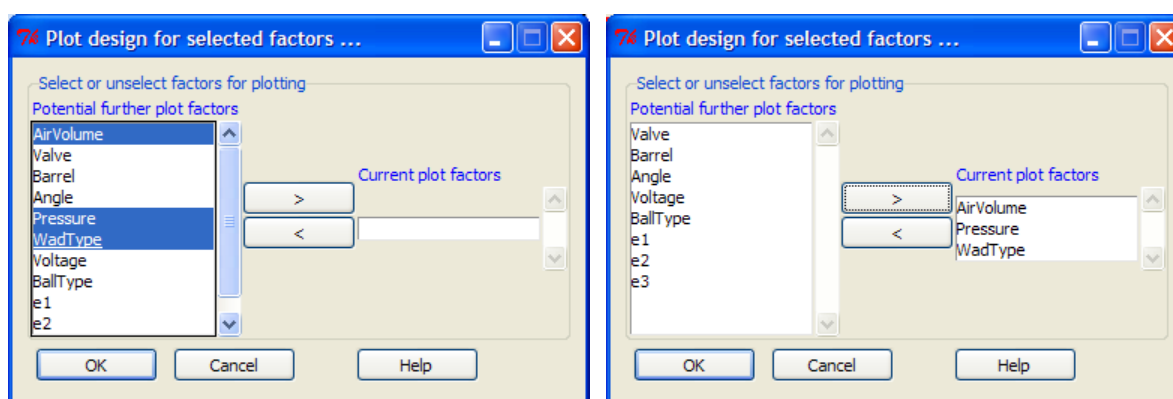
Figure 9: The dialog for plotting the active design

Plotting the active design even before collecting experimental data is useful for visualizing the numbers and structure of combinations occurring for a group of factors: In the dialog opened by the

    Design → Inspect design → Plot active design…

menu item, it is necessary to decide on the design factors to be plotted (cf. Figure 9). For the currently active screening design, a mosaic plot is produced which is useful for three or at most four factors at a time. For example, selecting factors *AirVolume*, *Pressure* and *WadType* for plotting (cf. Figure 9; pressing the arrow to the right in the left-hand side state moves the selected factors to the right-hand-side list), the OK button produces the mosaic plot in Figure 10, which shows that all combinations of factors occur at least once, and four of the combinations occur twice as often as the other four (this is the same for any three-factor combination for this 12 run design). Note that it will often be necessary to fetch the

open graphics window to the top of the screen after producing the graph. This is one of the nuisances that come with this free tool.

Analogously to and with the same purpose as plotting, it is also possible to tabulate frequencies of factor combinations instead; these would be displayed in the output window, like the summary information.
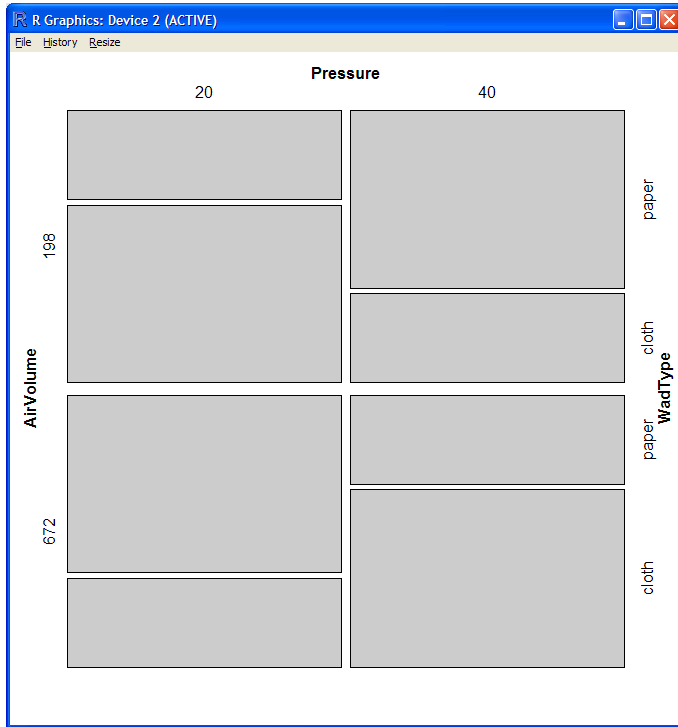
Figure 10: Graphics window with mosaic plot for the three selected factors from Figure 9

3.4 Polishing and exporting the created design

Depending on the result of design inspection, one will perhaps want to make some adjustments. Within an unclosed **R** commander session, re-opening the design generation dialog will remember the latest settings, except for the seed which will always be changed (you can manually change it to a given number again; if necessary, do that directly before pressing the OK button in order to make sure that no automatic refresh will change it again).

In our example, suppose we are now satisfied with the inspected design. The idea is then to export the design so that we can create and edit a sheet for the experimenter who will run the experiment and enter the response data outside of **R**. Exporting can be done either from within the design creation menu on the Export Design tab (and could also have been done immediately), or it can be done from the separate menu item

Design → *Export* → *Export experiment …*

This opens the dialog shown in Figure 11 (settings already modified).

The chosen "all formats" radio button creates the files `Screen.example.html`, `Screen.example.csv` and `Screen.example.rda`, i.e. files named according to the entry given for export file names and of each sort on offer. All export requests export an **R** workspace (`rda` file; cf. also Section 2.2) that contains only the chosen design. Depending on the settings, one or both of the other file types is also created. For these, the decimal separator entry is relevant; for many users, default will work best. For the author as a German user with German computer settings (comma decimal with semicolon

as separator) but *R* set to using decimal points rather than commas, the comma decimal separator is requested for creating a data file with numbers with decimal commas separated by semicola so that the files can be easily opened by the computer's spreadsheet software. The storage directory can be modified – this will modify the *R* working directory for everything, in line with what the button says.
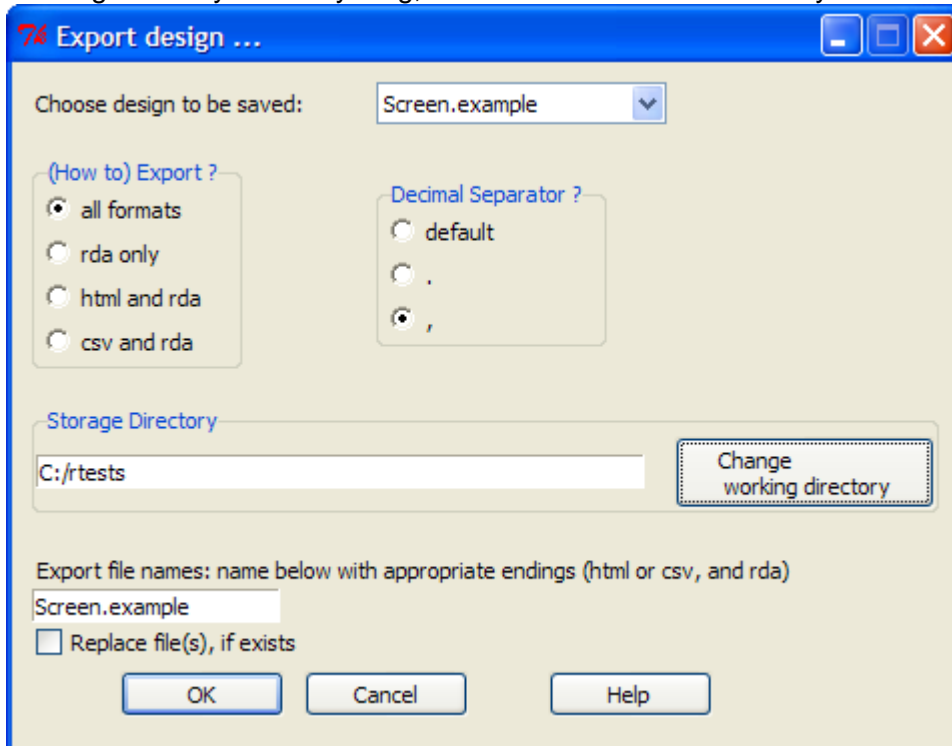
Figure 11: Export design dialog with user-specified settings
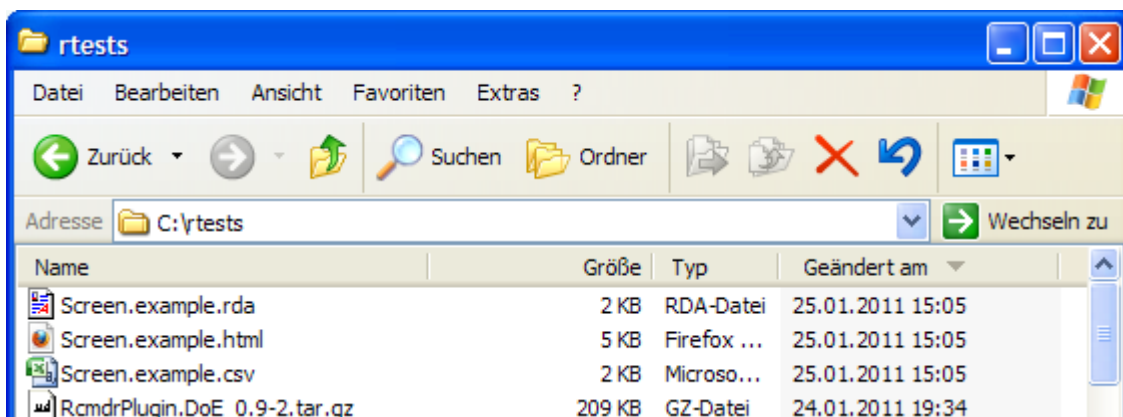
Figure 12: Files produced by the exporting routine (German Windows version)

After pressing OK, the three files can be found in the requested directory (cf. Figure 12). It is a matter of taste whether one uses the **html** file (with rows colored white or grey in order to minimize the risk of getting mixed up between neighbouring rows) or the pure **csv** file for preparing a data collection sheet. It will often be advisable to edit one of these files and provide additional more structured information to the experimenter. After the experiment has been conducted, however, it is usually easiest to use the exported **csv** file for providing the response values to *R* (cf. next section). Great care is needed in making sure that each response value is matched to the correct experimental row – whenever the original exported **csv** file is used for giving the data to *R*, even after reordering that file (all

columns together!), everything should work smoothly, since matching is done based on run numbers. Whenever response values are given to **R** with a `csv` file other than the original one, make sure that the rows in the `csv` file are in the same order as the rows in the experimental file (double-checking won't hurt!). In any case, it should normally not be necessary to reorder rows, because the rows are exported in run order and should be kept that way.

3.5 (Re-)importing response data

For each run of the potato cannon experiment, the average distance traveled and the standard deviation of the distances of the 4 repeat shots are available for analysis. These have been entered into the `csv` file outside of **R**, and the file has been stored under the new name **screen.example.withresp.csv**, much like it could have been done as the result of an experiment.

If the experimental plan without responses is the active data set in **R** Commander, it is possible to add the response values to this data frame via the

Design → Modify Design → Add response variable(s)…

menu item, which opens a dialog for adding a calculated response or response data from a `csv` file. The dialog is very similar to the one shown in Figure 13 for the probably more frequent situation, where both the experimental data and their response values have to be loaded into a fresh **R** Commander session. This dialog can be accessed via the

Design → Import → Re-import experiment from `csv` and `rda` files …

menu item.

Figure 13: Dialog for importing response data from a `csv` file

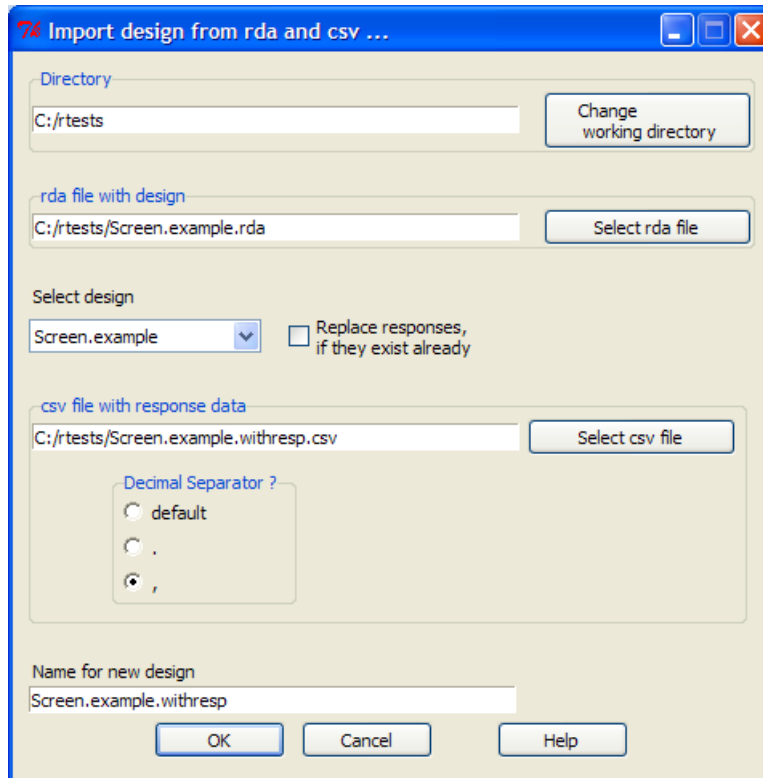The dialog shown in Figure 13 requests specification of both the `csv` file with responses and the unmodified `rda` file (cf. Figure 13); both files must be in the same

directory. Apart from the obvious selections of files and design names, the decimal separator must be chosen in accordance with the nature of the **csv** file: some European **csv** files have a decimal comma combined with a semicolon as field separator, while most US **csv** files have a comma field separator and a decimal point. The settings shown in Figure 13 work for the author's German computer. After pressing OK, the new design with responses is the active design, and this design can be analyzed.

Before starting to analyze the data, the design is displayed in Figure 14, using the dialog opened by

Design → Inspect design → Display design…

with the standard order radio button chosen. In this way, the design can be easily compared to the published design on the website.


```
Output Window                                                                    Submit

> print( Screen.example.withresp , std.order=TRUE)
   run.no.in.std.order run.no Airvolume Valve Barrel Angle Pressure WadType Voltage BallType e1 e2 e3     mean     sd
5                   1      5       198     1    4ft    45       20   paper       9    white -1 -1 -1  85.521 15.620
2                   2      2       198     1    4ft    45       20   cloth      27     pink  1  1  1 127.875 15.620
6                   3      6       198     1    6ft    60       40   paper       9    white  1  1  1 208.000  5.774
4                   4      4       198     2    4ft    60       40   paper      27     pink -1 -1  1 203.771 16.166
10                  5     10       198     2    6ft    45       40   cloth       9     pink -1  1 -1 313.813 15.011
8                   6      8       198     2    6ft    60       20   cloth      27    white  1 -1 -1  78.667  5.292
12                  7     12       672     1    6ft    60       20   paper      27     pink -1  1 -1 140.046  5.292
3                   8      3       672     1    6ft    45       40   cloth      27    white -1 -1  1 466.771 16.166
1                   9      1       672     1    4ft    60       40   cloth       9     pink  1 -1 -1 389.958  4.619
7                  10      7       672     2    6ft    45       20   paper       9     pink  1 -1  1 167.979  5.292
11                 11     11       672     2    4ft    60       20   cloth       9    white -1  1  1 166.021  5.292
9                  12      9       672     2    4ft    45       40   paper      27    white  1  1 -1 424.479 15.011
NOTE: columns run.no.in.std.order and run.no are annotation, not part of the data frame
```
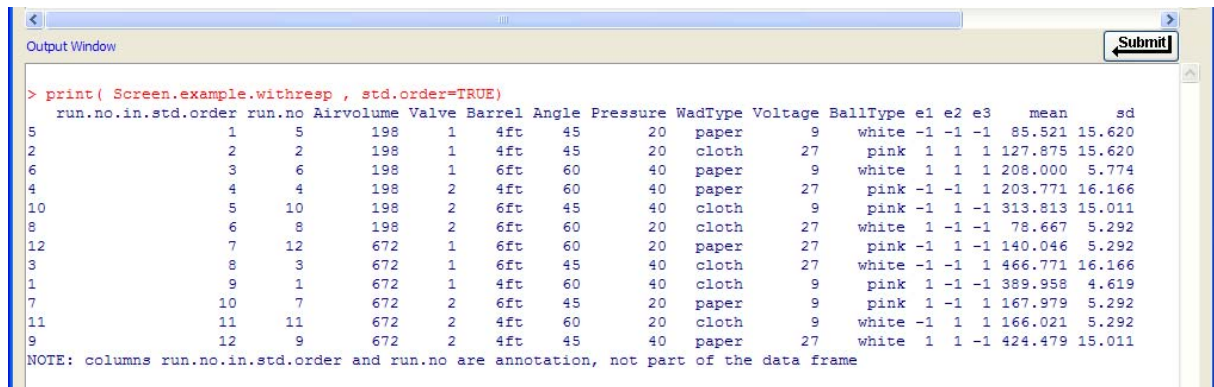
Figure 14: Design with added responses, displayed in standard order

Note that the software per default considers all numerical columns that are not design factors as responses, apart from the columns **name** and **run.no**. Any non-desired response can be removed by using the dialog opened by the

Design → Modify design → Select / deselect response variables…

menu item (which leaves the variable in the data frame but no longer as a response) or the dialog opened by the

Design → Modify design → Remove column(s)…

menu item.

3.6 Adding calculated responses

Sometimes, transformed values of the responses are needed. While models can have calculated responses, some of the simple plotting facilities cannot. It is therefore useful to add a calculated response to the data frame.

The dialog for adding response variables (cf. previous section) can be used for that purpose, by entering e.g. the *R* expression **log(mean)** in the field labeled "R object that contains response(s)", provided column **mean** is available in the *R* workspace (a future version may also look up column names in the active data frame). For most situations, it will be more appropriate to first create a calculated column using menu item

Data → Manage Variables in Active Data Set → Compute New Variable …

Here, the logarithm of the mean can e.g. be added under the name **ln.mean** (using formula **log(mean)** like before). Subsequently, it can be made a response in the dialog invoked by the menu item

Design → Modify design → Select / deselect response variables…

3.7 Analyzing experimental results

For the 12 run potato cannon experiment, there are three natural analysis steps:

- a half-normal plot for assessing effect significance
- alternatively or additionally, a linear model analysis which uses the three degrees of freedom from the dummy factors (`e1` to `e3`) for estimating error variance
- and main effects plots for easy effect interpretation.

Half Normal Plot for mean, alpha=0.1

A = AirVolume , B = Valve , C = Barrel , D = Angle , E = Pressure , F = WadType , G = Voltage , H = BallType , J = e1 , K = e2 , L = e3
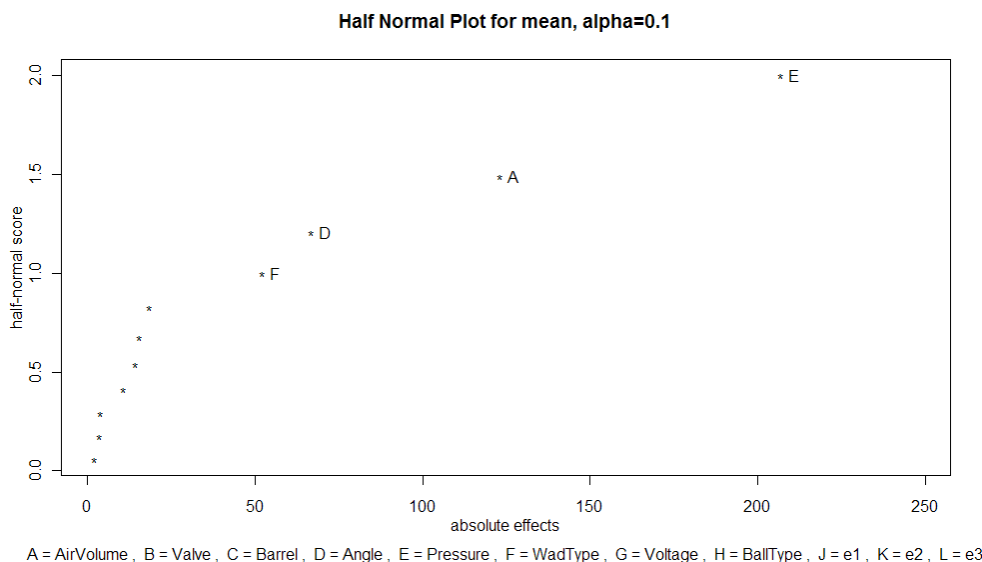
Figure 15: Half normal effects plot for the mean shot width in the potato cannon experiment

The half-normal plot (Figure 15) shows that there are four factors with significant main effects on average shot width at the liberal significance level of 0.1: E=Pressure, A=Air Volume, D=Angle and F=Wad Type. These four look distinctly apart from the remaining ones so that the screening appears to have worked out for these data. The plot in Figure 15 was created from the dialog opened by menu item

   Design → Analyze Design → Effects plot…

keeping the default settings. Note that it is very important that the main effects of the three dummy factors (`e1`, `e2`, `e3`) are included into the half normal plot, which is the reason why the software included these factors in design creation, even though only eight factors were specified.
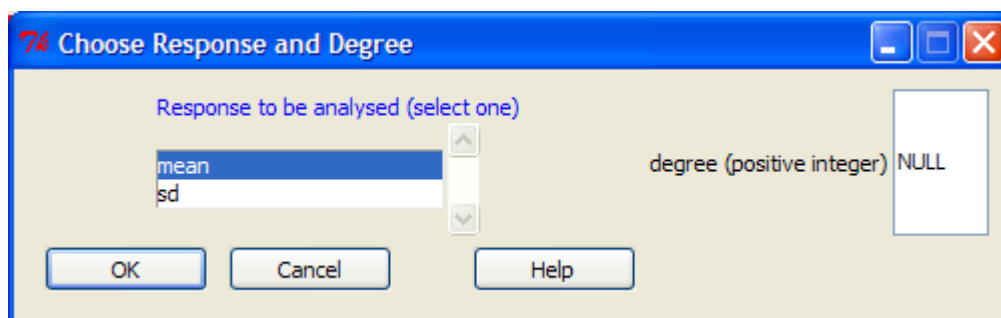

Figure 16: Menu for the default linear model

Alternatively, it makes sense to use a linear model with only the main effects of real factors included – here the three dummy factors (`e1` to `e3`) must be excluded and are thus subsumed in the random error.
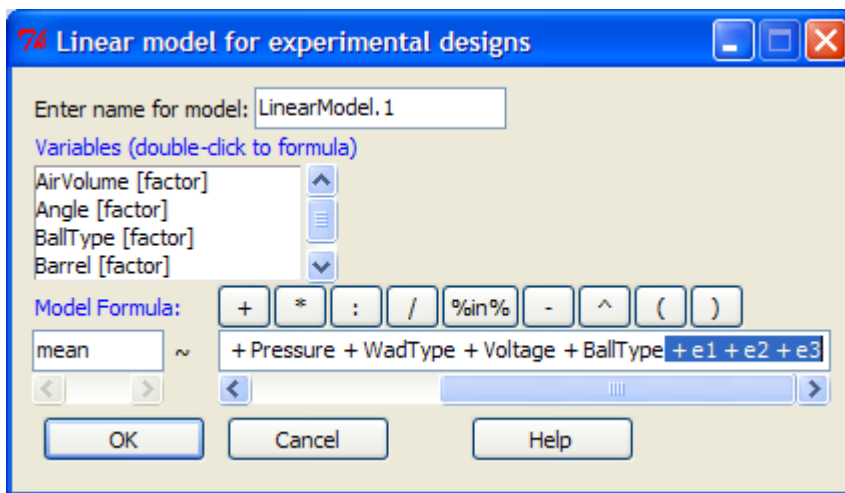

Figure 17: Menu for modifying the default linear model, dummy factors e1 to e3 to be deleted

The menu item

 Design → Analyze design → Default linear model…

opens a dialog for selecting a response and optionally specifying a polynomial degree (cf. Figure 16); pressing OK opens another dialog (Figure 17) with a reasonable default linear model which can be modified as desired. The default model (degree kept as NULL in Figure 16) depends on the type of design. For most design types, it is a quadratic model with all factors, for screening designs like the one considered here, it is a linear main effects model with all factors. For the potato cannon screening design, the dummy regressors should be removed from the model (as was mentioned before, cf. also Figure 17). After deleting the dummy regressors and pressing OK, the command for creating and summarizing the linear model is displayed in the script window and the results are shown in the output window of **R** commander (cf. Figure 18). The linear model identifies the same four factors as the active factors that were already found in the half normal effects plot. Note that this strategy only works reasonably well if there are several dummy variables – ideally, one should use both the effects plot and the linear model analysis.

For effect interpretation, it is most convenient to look at plots, in this case main effects plots. There are two ways to do so: With menu item

 Design → Analyze Design → Main Effects and Interaction Plots…

from the 2-level section, main effects plots all on the same scale can be generated, even without an explicit prior linear model analysis. The dialog (cf. Figure 19) allows selection of factors to be displayed in main effects plots or interaction plots (not appropriate for this design). "Length of abbreviations" refers to the length factor levels are shortened to. Here, the longest factor level label is 5 characters long; as all labeling fits onto the graph without overlapping with this length, abbreviation length has been set to 5 (instead of the default 4). The resulting main effects plots for the eight factors of this design are shown in Figure 20. Obviously, high air volume, 45 instead of 60 degrees angle, above all else high

pressure, and cloth instead of paper, improve the distance. The other factors have much smaller effects.

```
Output Window                                                    Submit

Call:
lm.default(formula = mean ~ AirVolume + Valve + Barrel + Angle +
    Pressure + WadType + Voltage + BallType, data = Screen.example.withresp)

Residuals:
      1       2       3       4       5       6       7       8       9      10
 -8.387   4.885   8.387  -6.957  10.460  -4.885   6.957  -6.957   4.885 -10.460
     11      12
 -8.387  10.460

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  231.075      4.583  50.423 1.72e-05 ***
AirVolume1    61.467      4.583  13.413 0.000896 ***
Valve1        -5.287      4.583  -1.154 0.332219
Barrel1       -1.862      4.583  -0.406 0.711701
Angle1       -33.331      4.583  -7.273 0.005364 **
Pressure1    103.390      4.583  22.561 0.000191 ***
WadType1      26.109      4.583   5.697 0.010722 *
Voltage1       9.193      4.583   2.006 0.138514
BallType1     -7.168      4.583  -1.564 0.215736
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.87 on 3 degrees of freedom
Multiple R-squared: 0.9962, Adjusted R-squared: 0.986
F-statistic: 97.78 on 8 and 3 DF,  p-value: 0.001539
```
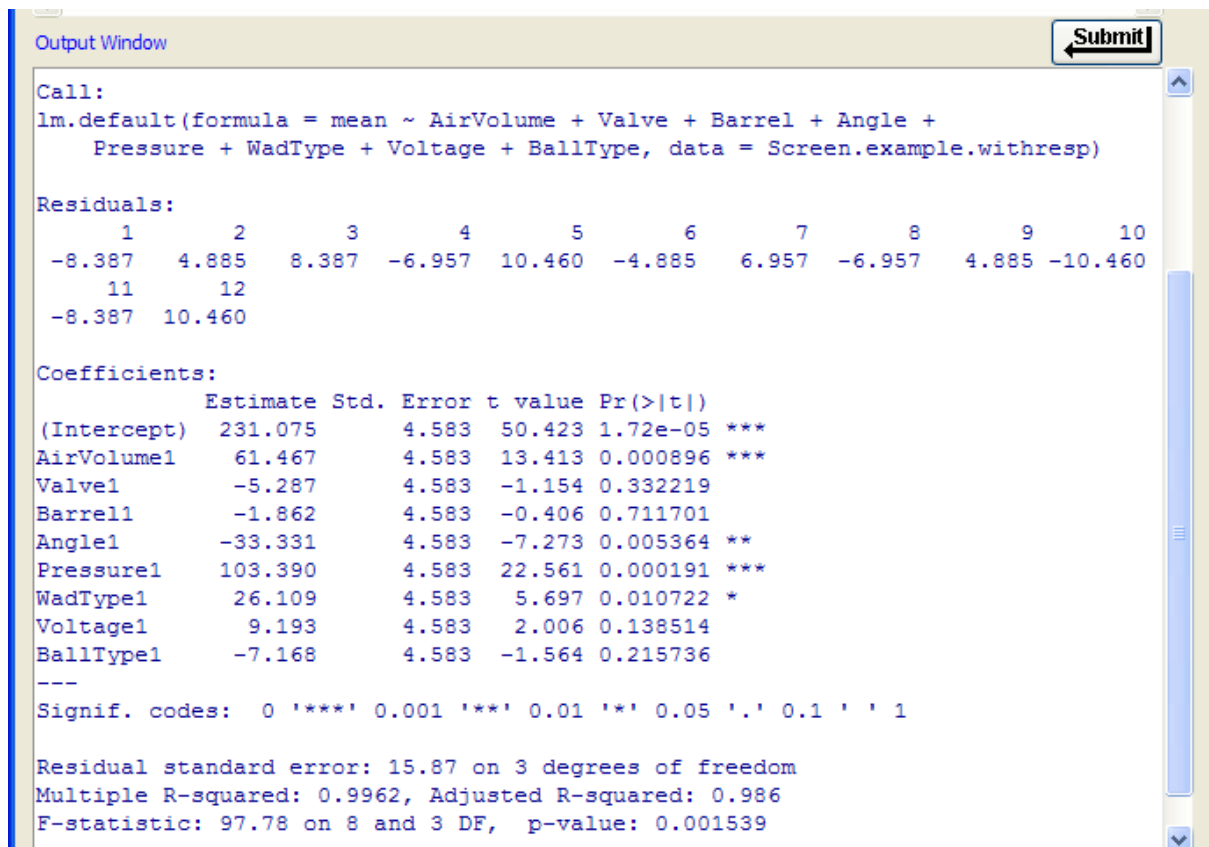
Figure 18: Output of linear model analysis for main effects
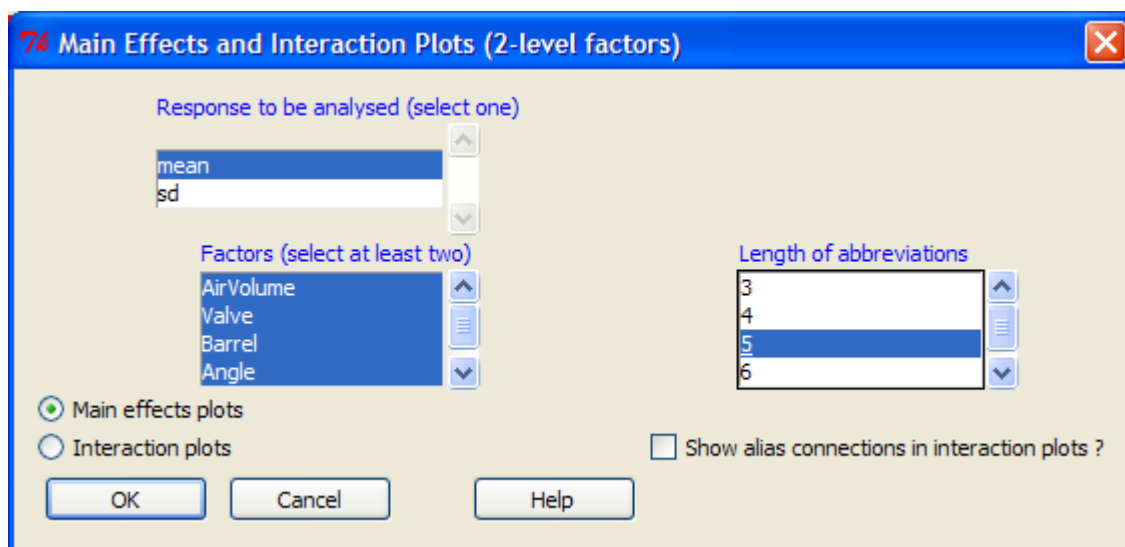
Figure 19: Dialog for main effects and interaction plots of 2-level factors
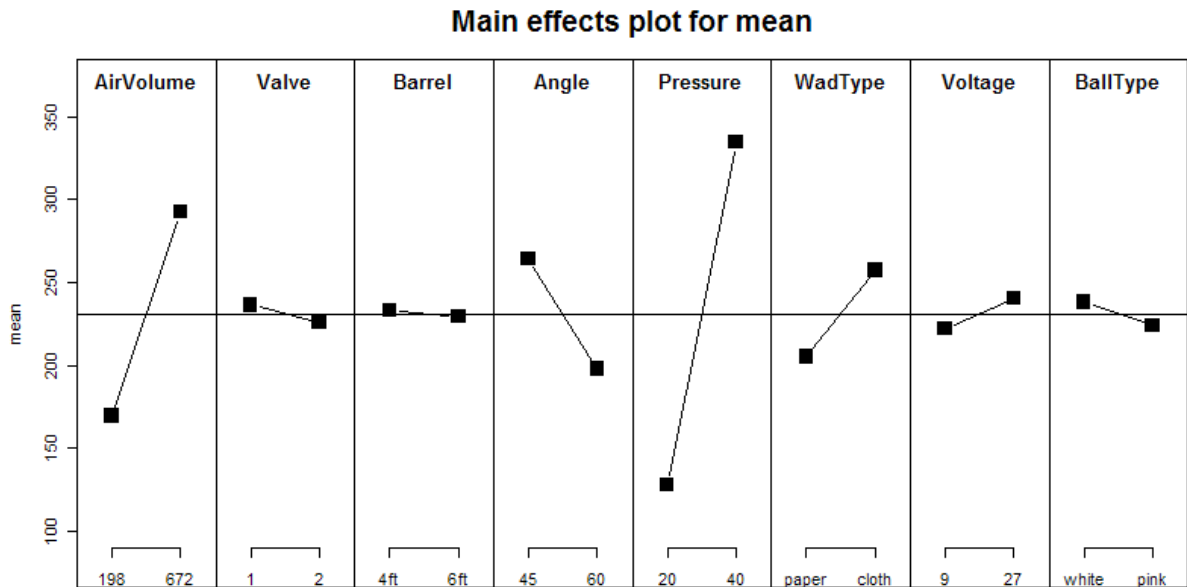
Main effects plot for mean

Figure 20: Main effects plots for all factors

After a linear model analysis, it is also an option to display effects plots for the model, using the

  Models → Graphs → Effect Plots

menu item, which immediately plots all effects of the fitted model without any possibilities for customization. The plots for the linear model of Figure 18 are shown in Figure 21; they have the advantages

- to also be valid for unbalanced data situations, since they are adjusted for the other effects present in the linear model,

- to show confidence bands

and the disadvantages

- to be not customizable

- to be on different scales.

Interactions cannot be interpreted within a Taguchi L12 experiment, and conclusions from main effects plots or a main effects linear model run the risk of being biased by active two-factor interactions (however, bias risk is smaller than with regular resolution III designs). Once four active main effects have been identified – they stick out very clearly here –, their interactions can also be inspected by restricting the linear model to these four factors with their two-factor interactions (1 df for intercept, 4 df for the main effects, 6 df for the interactions, 1 df remains for error), assuming so-called effect heredity (no interactions for factors without main effects). This can be done in the default linear model from the *Design* menu, or in the linear model always available in *R* Commander under
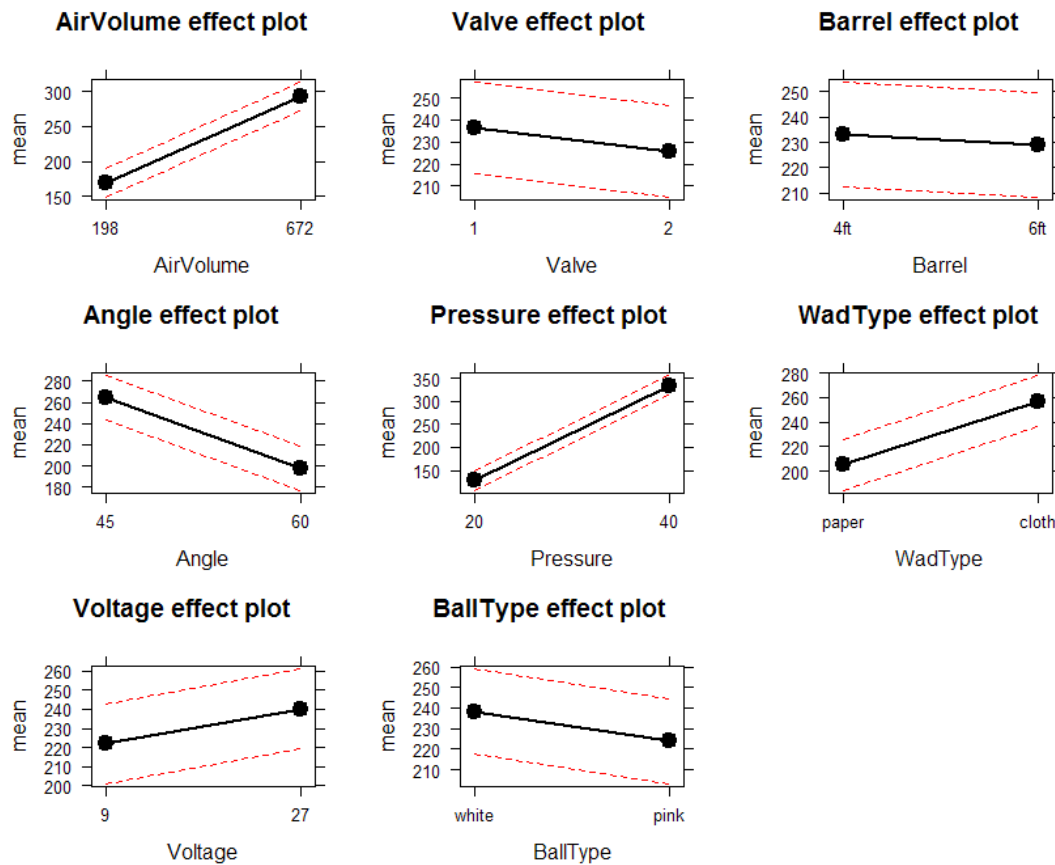
  Statistics → Fit → Linear model…

Figure 21: Effects plots from the main effects model for the potato experiment
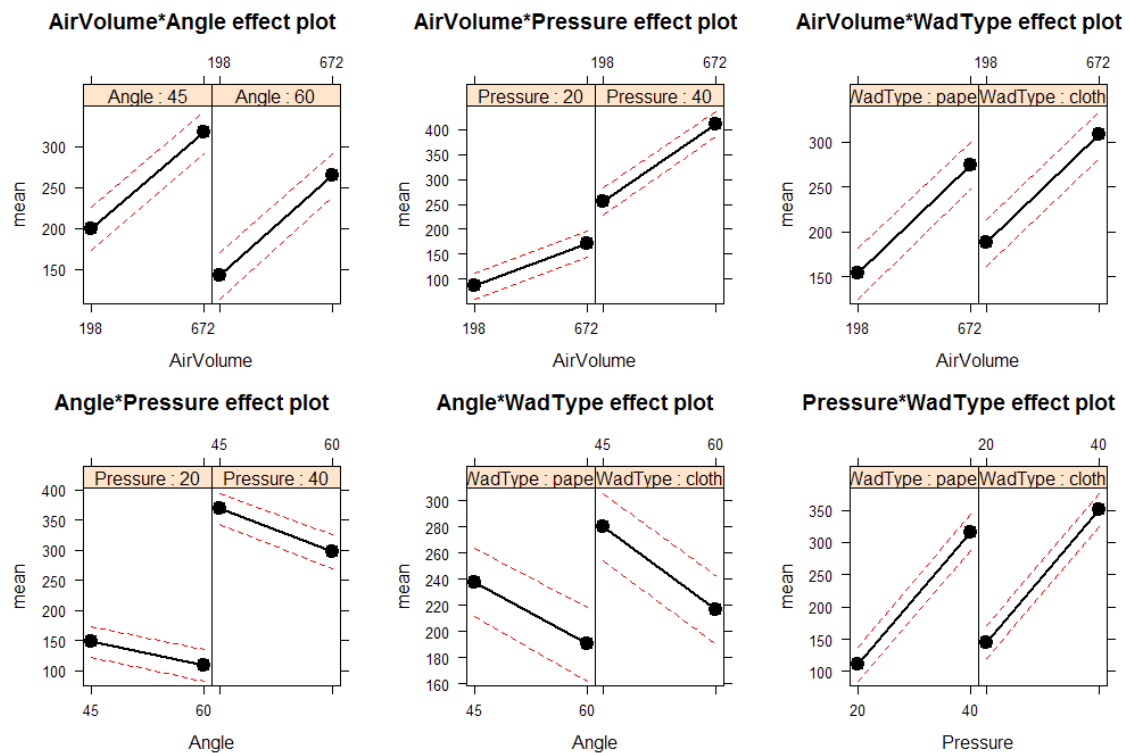CAUTION: scales differ between plots

Figure 22: Interaction plots for the potato cannon experiment from linear model with interactions on the four factors identified before

The linear model dialog opened by that menu item always comes up with the currently active model, which is quite helpful for exploring the effect of changes. Once a model has been decided upon, its main effects and interaction plots can be displayed (again as effects plots for the linear model, cf. Figure 22). (Unfortunately, it is not possible from the GUI to overlay these on one graph, which the author would prefer.) From the linear model summary (not shown), only the Pressure by Angle interaction is statistically significant; as there is only one error degree of freedom, resulting from one duplicated factorial combination, this assessment has a weak foundation in the data.

The safest way, after conducting an initial screening experiment like the one we just saw, is to concentrate on the variables identified as active and to run a further in depth experiment with them. Such an experiment will be set up (but not run) in Section 5.1.

4   Features for increasing work efficiency

Before looking at further examples, some features of the DoE GUI are explained in order to enable impatient readers to become more independent and experiment with dialogs more. This section can also be skipped by readers who would rather take a guided look at further design creation and analysis facilities immediately.

4.1 Help buttons

On all design creation dialogs, there is a help button on the outer frame that gives overall help on dialog usage, as well as a Tab Help button on (almost) each tab that provides help on the specifics of using that tab.

4.2 (Re-)Send commands from the script window

Sometimes, it can save a lot of time to send a quick command from the script window. This is always possible by selecting text in the script window and pressing the submit button. It is also possible to edit scripts in the script window, and to write additional commands. Section 5.2 shows an example, where graphical output is adjusted to users' needs by submitting a brief line of code in a way that is not possible based solely on the menus.

4.3 Store and Load form buttons

Imagine you are a quality engineer and in the middle of filling in the factor details of a 20 factor experiment, just typing in the details for the 17th factor, when your boss calls you to an emergency in the plant that cannot even wait for the last three factors to be handled. It would be extremely annoying to lose the work done up to this point. Therefore, the design generation menus of the software contain the buttons "Store form" and "Load form".

Whenever the "Store form" button is pressed, the current dialog settings are stored within the *R* workspace. If the form entries are to be used in a later *R* session, it is crucial to store the *R* workspace as a whole; this can be done using the "File" menu of *R* commander or from the appropriate menu item within the

Design → Export

menu. In a later *R* session, after loading the stored *R* workspace again, the "Load form" button of the design generation dialog (e.g. in Figure 5) retrieves the previous dialog entries. The "Load Form" button can also retrieve the menu state from designs that were created by the software (for *R* experts: the `creator` element of the `design.info` attribute contains all necessary information).

For example, after the *R* session was closed, we want to modify a small detail in the design created in the previous section, which was stored in directory "C:\rtests". The following steps are taken:

➢ A fresh *R* session is started.

➢ **RcmdrPlugin.DoE** is loaded (cf. Section 2.1).

➢ Since the directory "C:\rtests" will be used during the whole *R* session, the working directory is changed:

> *Design → Import → Change working directory …*

(can also be done from the *R* Commander file menu).

➢ Now, load the stored *R* workspace with the menu item

> *Design → Import → Load R workspace …*

(cf. Figure 23).
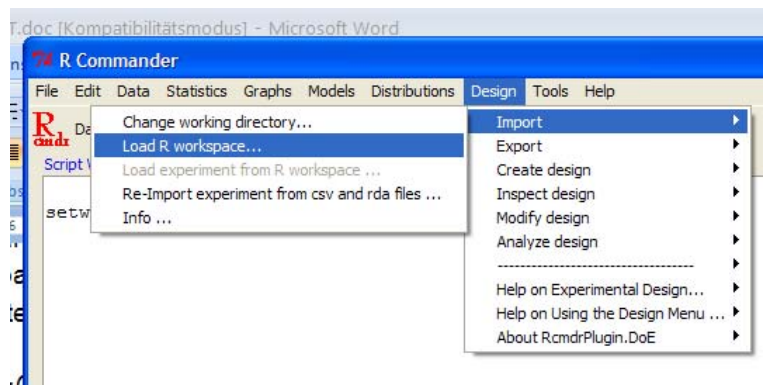
Figure 23: Loading an *R* workspace

In Figure 23, there is no experiment in the *R* workspace. Therefore, "Load experiment from *R* workspace…" is greyed out in the "Import" menu. After choosing "*Load R workspace…*" menu item, a typical file open dialog is opened. Assume we load the stored workspace `Screen.example.rda` (which was created by the dialog of Figure 5). As this workspace only contains one data set, this data set automatically becomes the active data set (cf. Figure 24, blue text in the top left area below the menu bar). Otherwise it may be necessary to load the desired design from the *R* workspace using menu item

> *Design → Import → Load experiment from R workspace …*

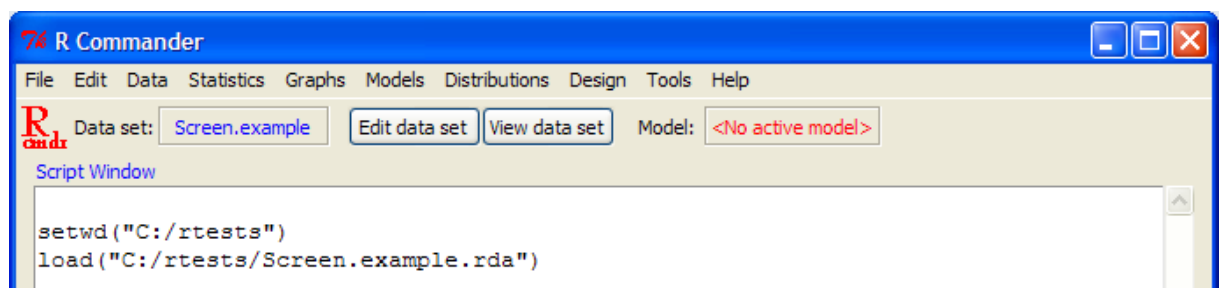or clicking on the currently active data set.

Figure 24: The active data set is now Screen.example

Now, the menu item

> *Design → Create design → Screening design …*

brings up the menu from Figure 3 (except for the seed for randomization, which is always changed). Pressing the "Load form" button opens the dialog shown in Figure 25 (there may of course be more than one entry to choose from). Pressing OK changes all settings except for the randomization seed to the stored ones. These can then be modified as desired. If the seed of the original design is to be retained – for example because the first experimental results have already been obtained and modifications are for annotation purposes only – make sure to change the seed to the recorded one as the final step before pressing OK.

> *CAUTION: If you really have collected data already and want to change design details for annotation purposes, make sure to keep a version of the original design in case something goes wrong! (This kind of care should be applied with any software!)*

Figure 25: Dialogue for retrieving stored design form

5 *Further examples of design creation and analysis*

Many aspects of design creation are the same for most types of design. Some additional aspects occur with special designs. For giving a more diverse picture, this section illustrates creation of further designs and also shows some analysis examples. Attention is mainly on aspects not previously covered. For example, each design creation dialog has an "Export" tab; since these work more or less like the "Export design" dialog that was discussed before, they are not at all discussed here. It is not intended with this tutorial to comprehensively document the capabilities of the software.

5.1 Regular fractional factorial 2-level designs

The dialog for creating regular fractional factorial 2-level designs (cf. Figure 26) is the most intricate one in the software: checking the "Activate Special Choices" box would even open further possibilities on the "Base Settings" tab and include an additional "Estimable Model" tab. This tutorial cannot cover all possibilities of creating fractional factorial 2-level designs with this dialog. The dialog is also more restricted than command line usage of the underlying function **FrF2**. For example, it does not (yet) offer split-plot designs.

Figure 26: Dialog for regular fractional factorial 2-level designs, Base Settings tab

Figure 27: Inputs for the follow-up experiment for the potato cannon screening, blocked with two blocks and three within-block repeat.only replications

A blocked design with little confounding

First, a blocked regular fractional factorial design is created, opening the dialog shown in Figure 26 through the menu item

Design → Create design → Regular(Fractional) Factorial …

Here, a blocked resolution V design in 16 runs is created for experimenting with the four factors found important in the potato cannon screening experiment (it would be possible to do five further runs for extending the previous 12 run design to a full factorial in the four factors of interest with one extra run, but conducting a new design is much safer, since the additional four factors from the eight factor experiment are under control this way). Assuming that the experiment will take two days, the number of blocks is set to 2; the idea is that conditions will vary somewhat between days.

Technically, first of all, the name of the design is entered (FollowUp), the number of runs is set to 16, the number of factors to 4. The 12 run design was conducted with 4 repeated measurements for each run, presumably because of suspected random variability between shots. Thus, a total of 48 shots of the cannon were done for the 12 run experiment – these were not independent, i.e. the experimental setup was not redone completely between shots. For the 16 run design, a few repeated shots will certainly also be useful. One might decide to do three shots for each run, which also implies a total of 48 shots. Instead of giving the means and standard deviations to *R* only, like in Section 3, we can also collect the repeated data separately and have them handled in *R*. This is achieved by telling the software to include 3 repeat.only replications (cf. Figure 27). Section 6 will discuss how such repeat measurements can be handled by the software.

The design that resulted from the dialog in Figure 27 is summarized by

Design → Inspect design → Summarize active design

Table 1 shows the result. This design has very little confounding: it is a full factorial with only the four-factor interaction confounded with blocks. This will allow detailed analysis of the four interesting variables.

A resolution IV design in 16 runs without blocking

For illustration of confounding patterns, another example is considered: the unblocked 16 run array with all eight of the potato cannon factors and without repeated measurements (i.e. treated as before) would have been an alternative to the 12 run screening experiment. Table 2 shows the alias structure of the design, according to which the two-factor interactions are heavily aliased with each other, but not with main effects. This is a great benefit of the four additional runs in comparison with the 12 run design discussed in Section 3, which is the reason that the author would have preferred this design over the 12 run Taguchi design if at all feasible.

Table 1: Summary of 4-factor follow-up design

```
   Experimental design of type  FrF2.blocked
   16  runs

   blocked design with  2  blocks of size  8
   each run measured  3  times (no proper replication)
   Factor settings (scale ends):
     AirVolume Angle Pressure WadType
   1       198    45       20    paper
   2       672    60       40    cloth

   Design generating information:
   $legend
   [1] A=AirVolume B=Angle      C=Pressure  D=WadType

   $`generators for design itself`
   [1] full factorial

   $`block generators`
   [1] b1=ABCD

   no aliasing of main effects or 2fis among experimental factors

   Aliased with block main effects:
   [1] none
```

Table 2: Summary of alternative design

```
> summary( Alternative , brief = TRUE)
Call:
FrF2(nruns = 16, nfactors = 8, blocks = 1, alias.block.2fis = FALSE,
    ncenter = 0, MaxC2 = FALSE, resolution = NULL, replications = 1,
    repeat.only = FALSE, randomize = TRUE, seed = 4218, factor.names =
list(AirVolume = c(198,
        672), Valve = c(1, 2), Barrel = c("4ft", "6ft"), Angle = c(45,
        60), Pressure = c(20, 40), WadType = c("paper", "cloth"),
        Voltage = c(9, 27), BallType = c("white", "pink")))

Experimental design of type  FrF2
16  runs

Factor settings (scale ends):
  AirVolume Valve Barrel Angle Pressure WadType Voltage BallType
1       198     1    4ft    45       20   paper       9    white
2       672     2    6ft    60       40   cloth      27     pink

Design generating information:
$legend
[1] A=AirVolume B=Valve      C=Barrel     D=Angle      E=Pressure  F=WadType
[7] G=Voltage    H=BallType

$generators
[1] E=ABC F=ABD G=ACD H=BCD

Alias structure:
$fi2
[1] AB=CE=DF=GH AC=BE=DG=FH AD=BF=CG=EH AE=BC=DH=FG AF=BD=CH=EG
AG=BH=CD=EF
[7] AH=BG=CF=DE
```

5.2 A full factorial design

The response data for this section are in file **FullFactorial3.4.withresp.csv**.

Sometimes, two levels are not enough for some of the factors. For any factorial situation, it is theoretically possible to use a full factorial design, i.e. to conduct an experiment with every factor combination – often, this will be prohibitive in terms of the number of runs, but for smaller situations it can be feasible.

For example, Tsao and Margolin (1971) reported a full factorial experiment on computer run times with four factors in three levels each (used here with data as reported by Mellor-Crummey 2005):

A	Page Replacement Algorithm	LRUV	FIFO	RAND
B	Deck Arrangement	GROUP	FREQY	ALPHA
C	Problem Program	Small	Medium	Large
D	Memory Pages	24P	20P	16P

Creation and data entry

A full factorial design in four 3-level factors requires 81 runs. The menu item

Design → Create design → General full factorial …

brings up the dialog in Figure 28, which has already been edited to show the desired name of the design and the number of factors. The "Factor Details" tab must then be edited to hold all the level information (cf. Figures 29 and 30). Per default, levels 1 to number of levels are set, which is particularly useful for initial try-outs because it saves a lot of typing. The figure shows how the levels are entered with blanks between them. After the information has been completed, the design is generated, exported (files **FullFactorial3.4.csv** and **FullFactorial3.4.rda**), the experiment is conducted, response data are input into the **csv** file (file: **FullFactorial3.4.withresp.csv**), and responses are re-imported with the steps described in Section 3.
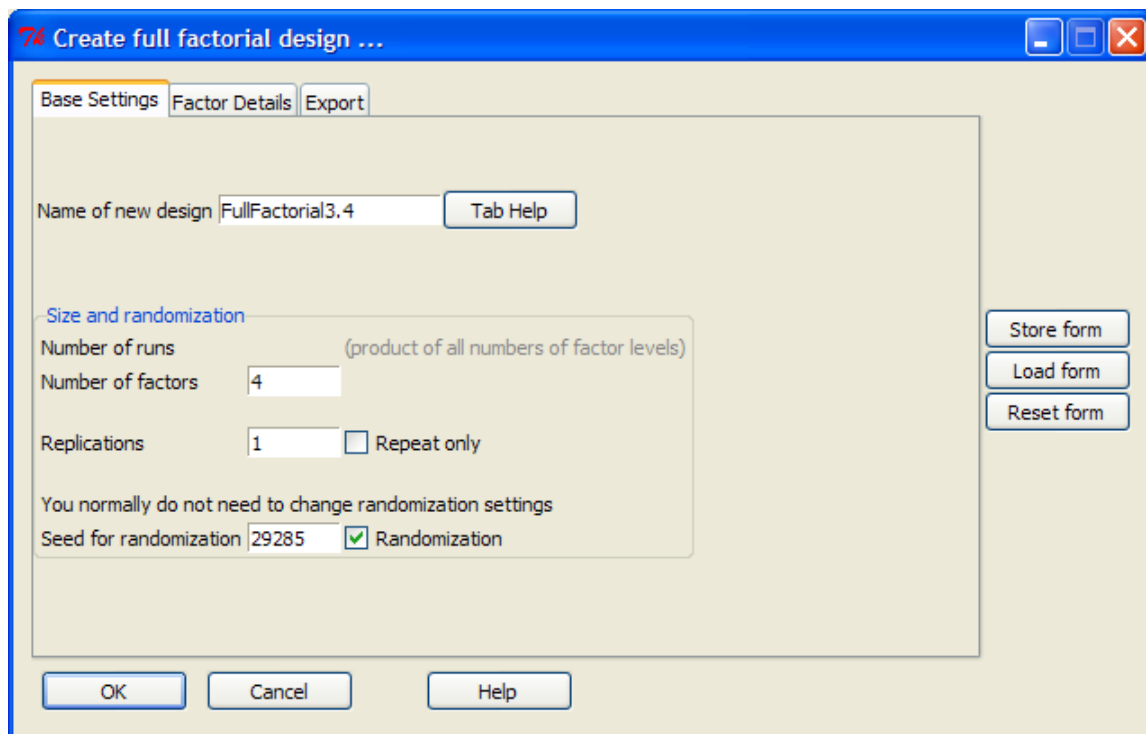

Figure 28: Dialog for full factorial designs (the actual seed at design generation was 5503)

Figure 29: Factor details tab for full factorial designs, editing in progress
(the completely edited dialog is shown in Figure 30)

Figure 30: Factor details for the experiment

As the data have been originally analyzed as base 10 logarithmic run time, this is also the response used here. The logarithmic data are already part of the `csv` file. (It would also be possible to add a calculated response after loading the data into the software, as was explained in Section 3.6; for this, the user must know the appropriate syntax for the requested calculation, e.g. log10(time) for the base 10 logarithm of time.)

Data analysis

Main effects and interaction plots for this design can be created from the

> *Design → Analyze design → Main effects and interaction plots …*

menu item under the general factorial designs. Figure 31 shows the dialog with selections for an interaction plot of factors `Algorithm` and `Program` for the response `log10time`. For a full factorial design, the direct creation of main effects and interaction plots from mean response values is safe, because the design can estimate all effects without any confounding. The caution at the top of the dialog warns that this is not always the case.
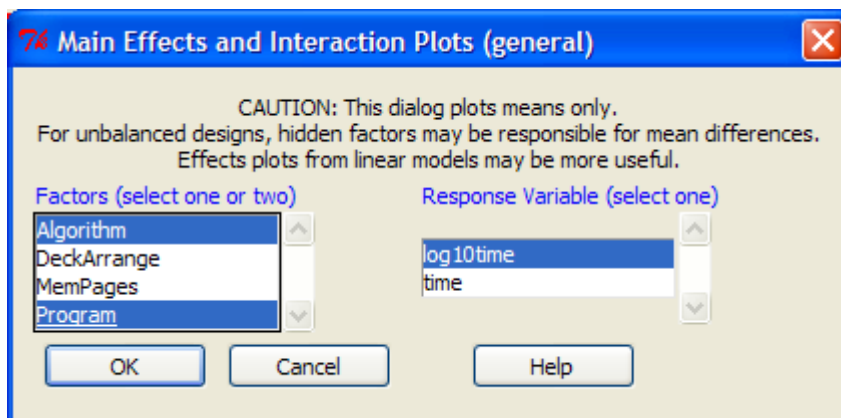

Figure 31: Dialog for general main effects and interaction plots
(selection of two factors will create an interaction plot)

General main effects and interaction plots are per default created one by one. Nevertheless, one may want to combine them onto one page – this cannot be done within the menus, but with a very simple command line intervention (cf. Section 4.2 for how to send script commands to *R*): The commands

```
par(mfrow=c(1,4))
```

and

```
par(mfrow=c(2,3))
```

respectively, issued directly before creating all the respective graphs, do the trick for Figures 32 and 33. For example, after issuing the first of these commands for a one row and four column layout, the four main effects plots can be sequentially generated by choosing each factor in turn in the dialog of Figure 31.
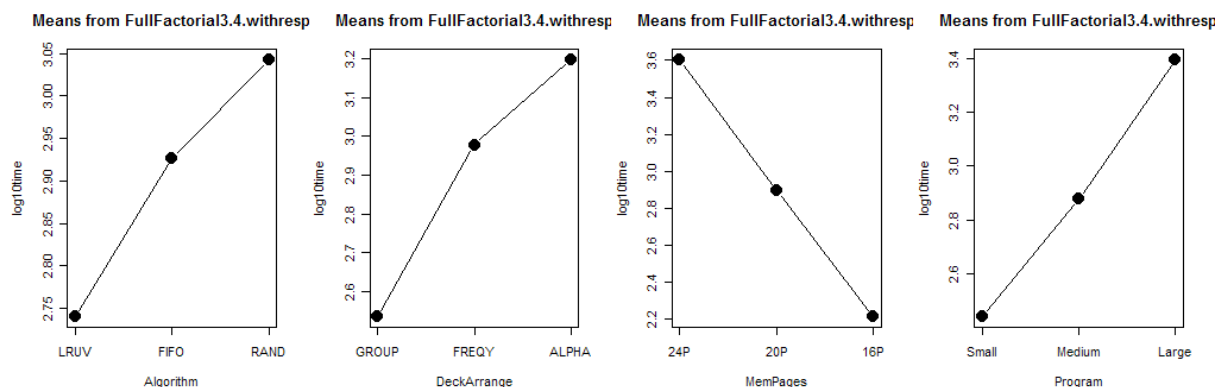
Figure 32: Main effects plots for log (base 10) run time (caution: each plot on own scale!)
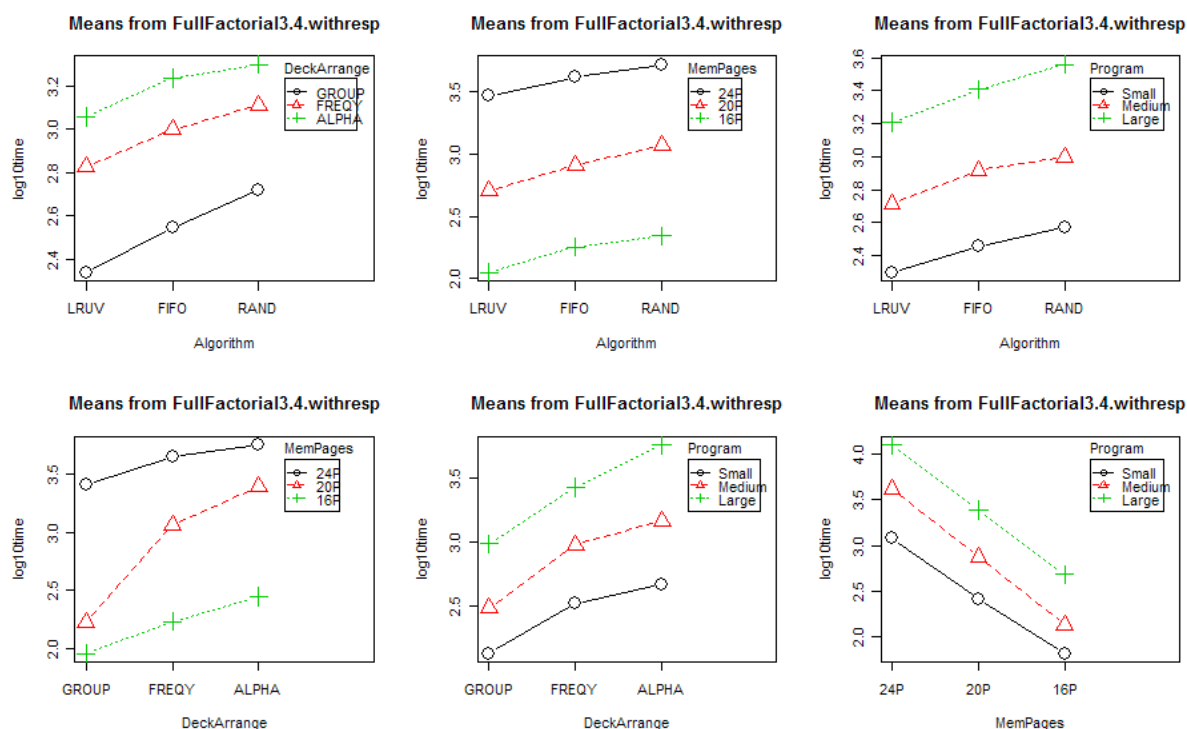

Figure 33: Interaction plots for log run time (caution: each plot on separate scale)

Plotting response means is one of the simplest analysis techniques, but can be misleading in case of unbalanced data. It is generally safer – although neither foolproof – to plot effects from linear models. This can also be done in **R** Commander, but can be quite dissatisfactory because of very crowded graphs for larger models; Figure 36 will show effects plots from a linear model analysis from a reduced model. First, the default linear model analysis is investigated.

Figure 34 shows creation of a linear model with all main effects and two-factor interactions for this design. This model has a very high R^2 (about 98%) and – analyzed with the *Models* menu of **R** commander – with menu item

   Models → Hypothesis tests → Anova Table …

(select type II sums of squares) declares all main effects and the two-factor interactions of Deck Arrangement with Program and Deck Arrangement with Memory Pages significant (cf. Table 3; Program with Memory Pages was almost significant, too). This is in line with the visual impression from interaction plots.
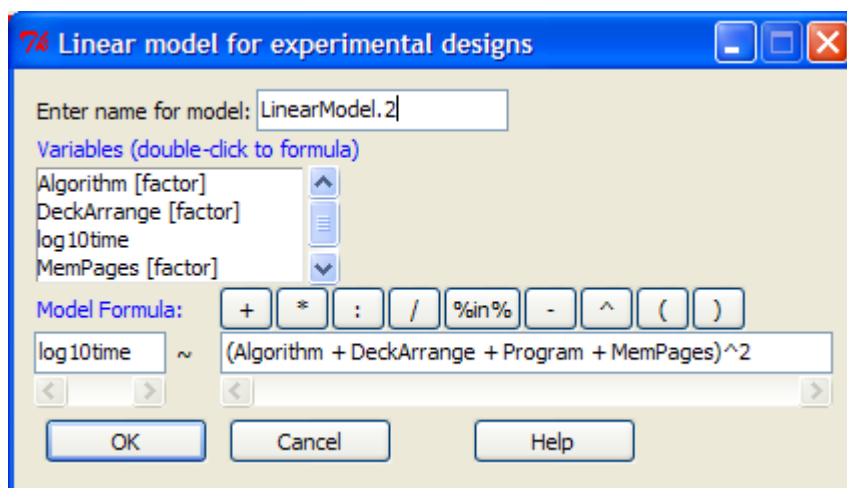
Figure 34: The default linear model with response `log10time`

Table 3: Anova Table for linear model with all 2-factor interactions

```
> Anova(LinearModel.2, type="II")
Anova Table (Type II tests)


Response: log10time
                        Sum Sq Df    F value     Pr(>F)
Algorithm                6.665  2    52.3868 8.563e-13 ***
DeckArrange             32.695  2   256.9852 < 2.2e-16 ***
Program                 65.174  2   512.2697 < 2.2e-16 ***
MemPages               137.924  2  1084.0790 < 2.2e-16 ***
Algorithm:DeckArrange    0.279  4     1.0967   0.36898
Algorithm:Program        0.106  4     0.4176   0.79511
Algorithm:MemPages       0.183  4     0.7209   0.58189
DeckArrange:Program      0.844  4     3.3153   0.01776 *
DeckArrange:MemPages    10.128  4    39.8035 1.108e-14 ***
Program:MemPages         0.650  4     2.5529   0.05086 .
Residuals                3.053 48
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that it would also have been possible to analyze the linear model on the base 10 log scale without having a logarithmic variable available. This could have been done by simply changing the a-priori calculated `log10time` to `log10(time)` in the response field of the linear model dialog, cf. Figure 34.

As was already shown above, the effects from a linear model can be displayed with much less effort than shown above for main effects and interaction plots: The menu item

Models → Graphs → Effect plots

does the trick. It has been mentioned that these look very crowded for the full model. After adjustment of the model to include only the (almost) significant interactions (cf. Figure 35), the plots are at least readable (see Figure 36). Nevertheless, they are not as easy to read

as the ones in Figures 32 and 33, which are applicable, whenever the data are balanced w.r.t. the effects that are displayed and higher order effects can be assumed negligible or balanced out.
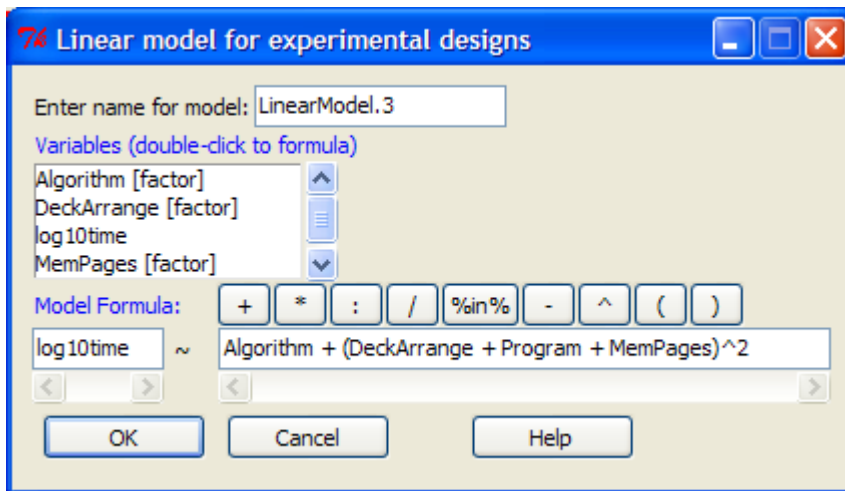

Figure 35: Model without two-factor interactions with p-values higher than 0.1 (in the restricted model, all factors are significant at the 5%-level)
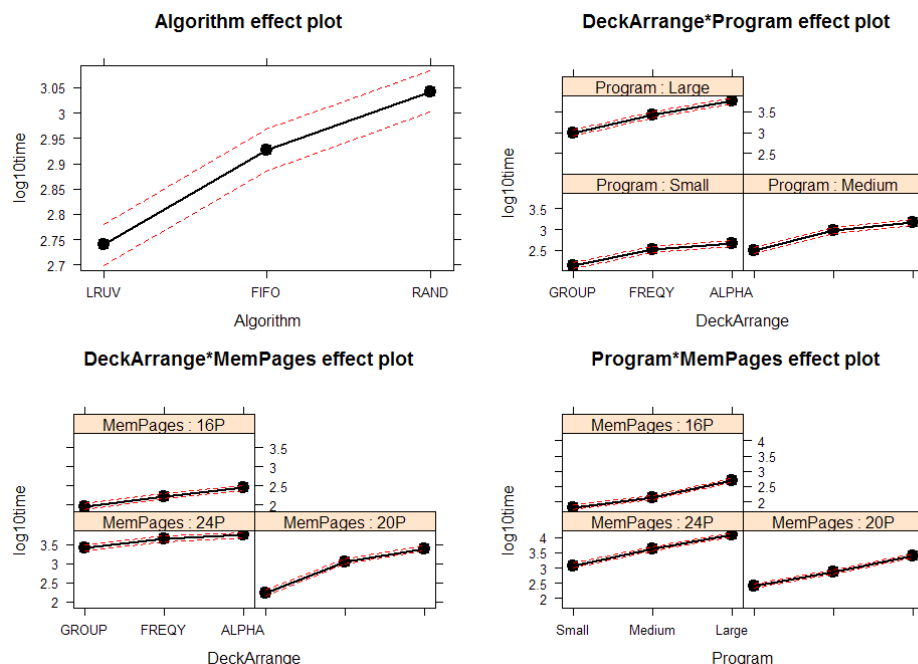
Figure 36: Effect plots from the linear model with some interactions excluded

5.3 A general orthogonal array

The response data for this section are in file **oa3.4.withresp.csv**.

Often, a full factorial design is not affordable. In such cases, a general orthogonal array can be used that can at least maintain orthogonality of main effects.

Creation, inspection and data entry

Such an array can be implemented with the dialog invoked by the menu item

Design → Create design → General orthogonal arrays …

Within that dialog, after inputting the number of factors and the factor detail information (at least the number of levels), it is possible to look at the available orthogonal arrays. Figure 37 and Table 4 show how to look up orthogonal arrays with up to 40 runs for the situation of the previous section. It can be seen that an experiment with four 3-level factors can be conducted in orthogonal arrays with 9, 18, 27 or 36 runs, when restricting attention to designs with up to 40 runs. We will now look at creation and inspection of the 18 run experiment, based on L18.2.1.3.7, an orthogonal array with 18 runs, one factor with 2 levels and seven factors with 3 levels. The dialog will use the desired L18.2.1.3.7, if it is selected in the "Specific array" drop-down menu in Figure 37 (otherwise, the smallest possible array, the L9.3.4, would be used). Another way to make sure the design has more than 9 runs is to enter a positive number in the text box for "Minimum residual df"; in this example, the model uses 9 degrees of freedom, so that any requirement for residual degrees of freedom disallows 9 runs.

Figure 37: Looking up available orthogonal arrays with up to 40 runs for the factors required on the Factor Details tab (which is completely analogous to e.g. Figure 30; level details need not be filled in for the "Show available designs" button to work)

Furthermore, Figure 37 shows the option "min3" for automatic optimization. This means that column allocation within the chosen array is optimized in order to minimize confounding among any triple of factors (i.e. of main effects with two-factor interactions). While choice of this option can substantially improve a design if relatively few columns from a larger array are used, specifying the option can also involve substantial search time and can even be prohibitive in some situations. In this particular case, the option runs fast, but does not yield any improvement versus naïve usage of the first four 3-level columns. The resulting design (using the seed 25768, as shown in the dialog) can be inspected using the

Design → Inspect design → Summarize active design

menu item, which yields Table 5. The summary shows – among other things – that there are two generalized words of length 3, i.e. the design is of resolution III, which has a similar meaning as for regular fractional factorial 2-level designs (interested expert readers are referred to Grömping 2011 for mathematical background). This information makes it useful to look at confounding among three factor projections by looking at tables or mosaic plots. Plotting the three-factor projection of the three factors Algorithm, Deck Arrangement and Program using the menu item

Design → Inspect design → Plot active design

shows relatively mild confounding for this triple of factors (cf. Figure 38): for each pair of levels of two factors, one of the three levels of the third factor does not occur at all, for example the **RAND** algorithm with deck arrangement **GROUP** does not occur with a **Large** program. Creation of analogous plots for all other triples of factors would show comparable pictures for these.

Table 4: Available orthogonal arrays for accommodating four 3-level factors in up to 40 runs

```
17   designs found

                 name nruns                        lineage

5             L9.3.4      9

20        L18.2.1.3.7     18     3~6;6~1;:(6~1!2~1;3~1;)

22        L18.3.6.6.1     18

39          L27.3.13      27         3~9;9~1;:(9~1!3~4;)

40        L27.3.9.9.1     27

76        L36.2.16.3.4    36        2~16;9~1;:(9~1!3~4;)

80       L36.2.11.3.12    36     3~12;12~1;:(12~1!2~11;)

81 L36.2.10.3.8.6.1      36

83  L36.2.9.3.4.6.2      36

85       L36.2.4.3.13    36 3~12;12~1;:(12~1!2~4;3~1;)

87  L36.2.3.3.9.6.1      36

89 L36.2.2.3.12.6.1      36 3~12;12~1;:(12~1!2~2;6~1;)

90  L36.2.2.3.5.6.2      36

92  L36.2.1.3.8.6.2      36     3~7;6~3;:(6~1!2~1;3~1;)

94        L36.3.13.4.1    36 3~12;12~1;:(12~1!3~1;4~1;)

95       L36.3.12.12.1    36

96         L36.3.7.6.3    36
```

Table 5: Summary of 18 run orthogonal array of the four factors

```
design was generated with RcmdrPlugin.DoE


Experimental design of type  oa
18  runs


Factor settings (scale ends):
  Algorithm DeckArrange Program MemPages
1     LRUV        GROUP    Small      24P
2     FIFO        FREQY   Medium      20P
3     RAND        ALPHA    Large      16P
Generating Orthogonal Array:
[1] L18.2.1.3.7
Selected Columns:
[1] 2 3 4 5
Numbers of generalized words of lengths 3 and 4:
   3    4
  2.0 1.5
```
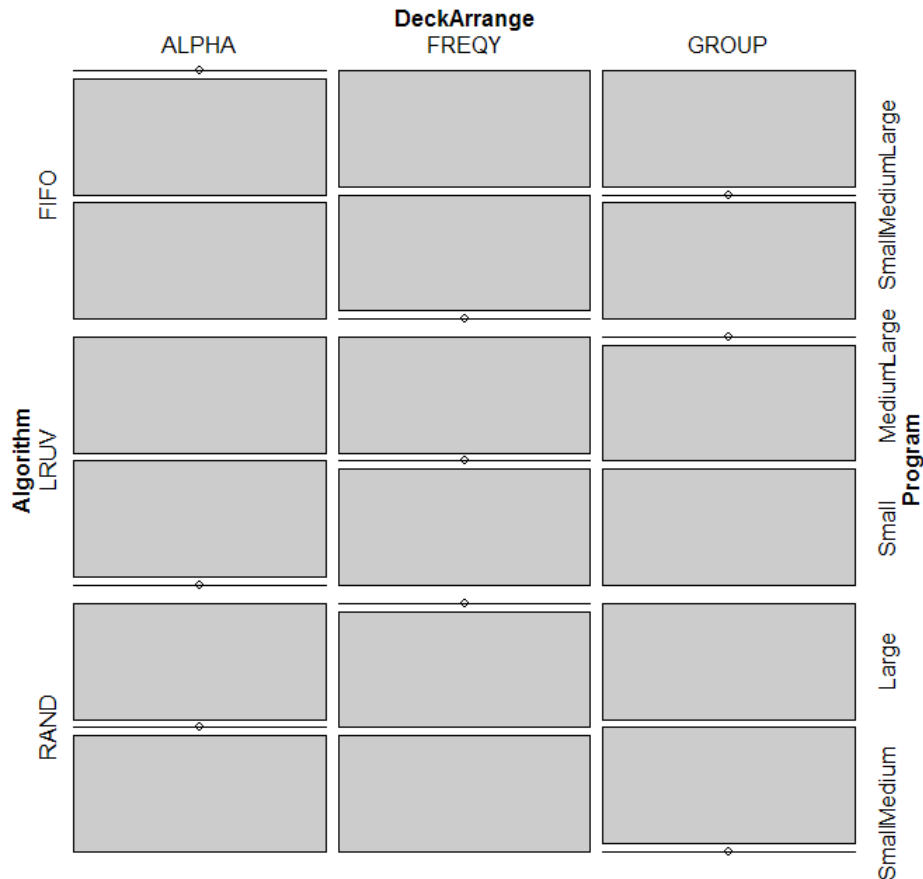

Figure 38: Mosaic plot for inspecting the 18 run orthogonal array

Data analysis

As data from the full factorial are available, we can pretend that the 18 run design was actually run by selecting the actually observed response values for its 18 runs from those of the full factorial (this is how `oa3.4.withresp.csv` was created from `oa3.4.csv`). Adding the response data to the design is done as usual (export design, add response

data to `csv`, re-import). The design can now be analyzed w.r.t. main effects, since it has eight main effect degrees of freedom. There are a total of 6*4 = 24 degrees of freedom for two-factor interactions. These can of course not all be accommodated within an 18 run array and are also confounded with the main effects. The 18 run design is analyzed with a main effects model only – this is also the default for such designs and can be easily obtained via the menu item

Design → *Analyze design* → *Default linear model...*

Once the linear model has been created, its results can again be displayed through the general **R** commander menu item

Models → *Graphs* → *Effect plots*

For a main effects model, the generated effect plots are useful and readable as they come (cf. Figure 39) and look reasonable and in line with previous results. It cannot be expected that the small array can provide conclusions on interaction effects.
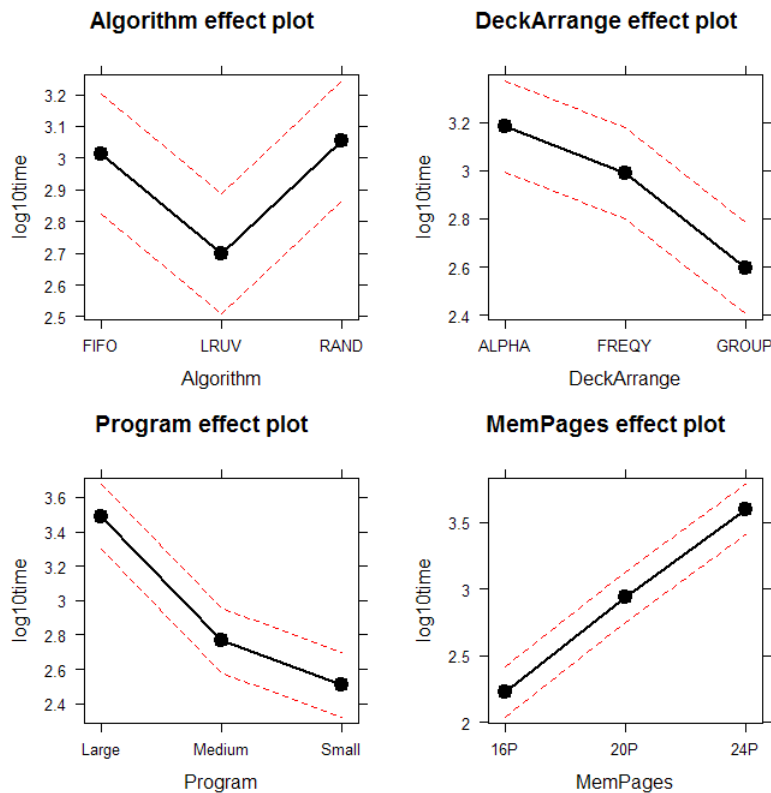

Figure 39: Effects plots from the linear main effects model of the 18 run design

5.4 A D-optimal design

The response data for this section are in file **Dopt3.4.withresp.csv**.

Now suppose that the experimenter wants an optimal design for estimating the second order model which has one parameter for the overall mean, 8 main effect parameters and 24 two-factor interaction parameters, i.e. a total of 33 parameters. Hence, at least 33 runs have to be used, less will not be accepted by the software. Assume we want a 40 run design (i.e. 7 extra degrees of freedom for error) that optimally estimates this model.

Creation and data entry

Before starting to create a D-optimal design, it is necessary to pick a candidate design. The menu item

  Design → Create design → Candidate design…

allows to pick candidate designs from existing designs or existing data frames (need not be designs) and also allows to create a candidate design in various ways. After the previously created design **FullFactorial3.4** has been made the candidate design, menu item

  Design → Create design → D-optimal design…

invokes the dialog for creating the desired D-optimal design (cf. Figure 40). Again, response values can be found by picking the appropriate values from the full factorial responses.

Figure 40: Creation of a D-optimal design from the full factorial candidate set

Data analysis

Again the data can be analyzed with a linear model. Here, because balance is not even guaranteed for main effects, simple main effects and interaction plots from the *Design* menu can be misleading and should not be used. The ANOVA table of the default quadratic linear model declares all main effects and one interaction statistically significant. Reducing the model to significant effects and plotting these yields the graph in Figure 41.

Table 6: ANOVA table for D-optimal design

```
Anova Table (Type II tests)
Response: log10time
                     Sum Sq Df  F value    Pr(>F)
Algorithm            0.4535  2  16.8077   0.002125  **
DeckArrange          3.1024  2 114.9729 4.432e-06  ***
MemPages            11.5154  2 426.7530 4.855e-08  ***
Program              5.2547  2 194.7373 7.313e-07  ***
Algorithm:DeckArrange 0.0300 4   0.5561   0.701968
Algorithm:MemPages   0.0086  4   0.1585   0.952865
Algorithm:Program    0.0317  4   0.5876   0.682409
DeckArrange:MemPages 0.6545  4  12.1282   0.002890  **
DeckArrange:Program  0.1244  4   2.3055   0.157795
MemPages:Program     0.0549  4   1.0175   0.459794
Residuals            0.0944  7
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```
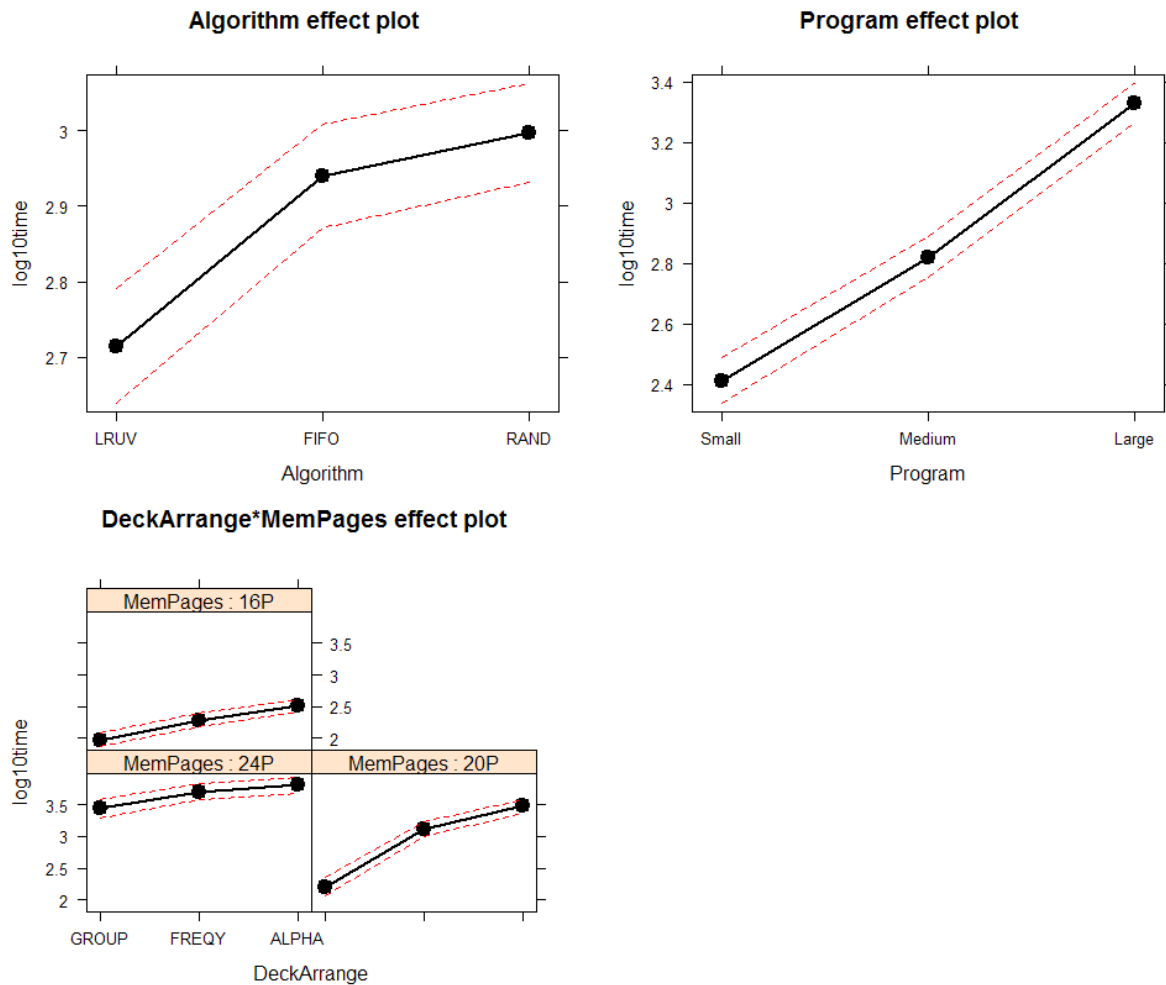

Figure 41: Effects plots from fitted linear model for the D-optimal design

Results from the D-optimal design are roughly comparable to those from the 18 run design (the positions of factor levels in the plots are different); this larger design is also able to recover the interaction between **MemPages** and **DeckArrange** that was found in the full factorial experiment.

Accomodating constraints

It is sometimes necessary to observe certain constraints in experimentation, because not all combinations are possible – for example, there may be size constraints so that the combination of two factors at large level with a third (environment) factor at small level is not possible. The ability to handle all kinds of constraints is one of the big advantages of D-optimal designs. Figure 42 illustrates inclusion of constraints with the run time example shown above, assuming (purely for demonstration purposes) that Algorithm RAND cannot be combined with 24P MemPages and DeckArrangement GROUP. If character strings are needed within the constraint, make sure to use either double quotes only, or single quotes only, but not a mix of both. Users have to be able to write their condition in *R* syntax; syntax help can be accessed from the dialog.

Figure 42: Creating a D-optimal design with a constraint

5.5 A central composite design

When experimenting with quantitative factors, one often strives for an optimum. After initial experiments with 2-level factors and potential steepest ascent improvements, one finally wants to locate an optimum based on a second order approximation to the response behavior.

Creation and data entry

For modeling the response surface, central composite designs can be used. Within the DoE GUI, they are created as an augmentation to a "cube" consisting of a fractional factorial 2-level design with center points.[4] The dialog for their creation (cf. Figure 43) is invoked by the menu item

Design → Create design → Central composite …

It does not have a factor details tab, as the factor details are defined within the cube portion of the design, which is either already available or is created from within the central composite design creation dialog. If called from within that dialog, the dialog for fractional factorial designs starts with modified defaults, leaving it to the software to find the smallest design of at least resolution V (called resolution V+ in the dialog) for the number of factors to be investigated (cf. Figure 44). After successful creation of the cube portion, a message box (to be confirmed by clicking OK) tells the user to continue the creation of the central composite design. Now, the number of center points must be chosen. This can be one number, which is applied to both the star portion and the cube portion, if the cube portion does not yet have center points and otherwise to the star portion only. If cube and star portion are supposed to have different numbers of center points and the cube does not have center points yet, two numbers can be given, separated by a comma (number for cube block, number for star block). Furthermore, the value for alpha can optionally be changed; the default is a reasonable general purpose value, but may be infeasible in some situations.

Response data can be added to the design in the usual way. For central composite designs, sequential experimentation is particularly relevant. Very often, the cube portion of the experiment is run in the beginning – ideally already with some center points. Only after some interim analysis, the star portion of the design is also run. It is recommended to separately create the cube first, add responses to it and analyse it; if necessary, it can be augmented with a star portion later. However, if the complete experiment has already been created in one go, one can add cube portion responses to the complete design in the usual way, by assigning missing values (NA) to the not-yet-known response entries. Interim analyses are then possible on the cube portion of the design created by the menu item

Design → Modify design → Remove star portion from central composite design

If the star point runs are later also completed, complete response data can simply be added again to the original central composite design.

---

[4] *In the typical sequence of experimentation, the cube with some center points would already exist and would be expanded into a central composite design in case curvature is detected. This is the reason, why cube and star portion are always treated as separate blocks. It may sometimes happen that a central composite design is immediately created, but is to be analyzed after the cube portion of the experiment including center points has been run. For such situations, the menu item*

Design → Modify design … → Remove star portion from design

allows to create a reduced 2-level fractional factorial with center points (the cube portion alone) which can be analysed with the simple analysis menus on offer for 2-level experiments.

Figure 43: Dialog for creating a central composite design
Click on "Create new 2-level design" button for newly creating the cube portion

Figure 44: Dialog for creating the cube portion of a central composite design
The number of center points for the cube can be specified here or later.

Data analysis

After adding response values to the design (cf. also previous section for specific issues with sequential experimentation), the experimental data can be analyzed with the software. For all response surface analyses, a response surface model should be created, using menu item

    Design → Analyze design → Response surface model …

Classical linear model analysis from the *Design* menu or from the *Statistics* menu of **R** Commander is also available, and does also allow creation of response surface plots; however, steepest ascent analysis cannot be conducted, unless a proper response surface model has been used.[5] Figure 45 shows the response surface plot dialog invoked by menu item

    Design → Analyze design → Response surface plots …

after adding a dummy response variable to a simple non-customized central composite design with four factors and 4 center points in each block. Plotting multiple response surface graphs at once – e.g. multiple contour plots – requires

- specification of the number of rows and columns for a graphical layout
- and choice of pairs of experimental variables for which to display the response surface at fixed levels of the respective other experimental variables.
- The values at which the other experimental variables are fixed, can be chosen on the tab "Modify slice positions". Per default, levels for unused factors are fixed at the center.
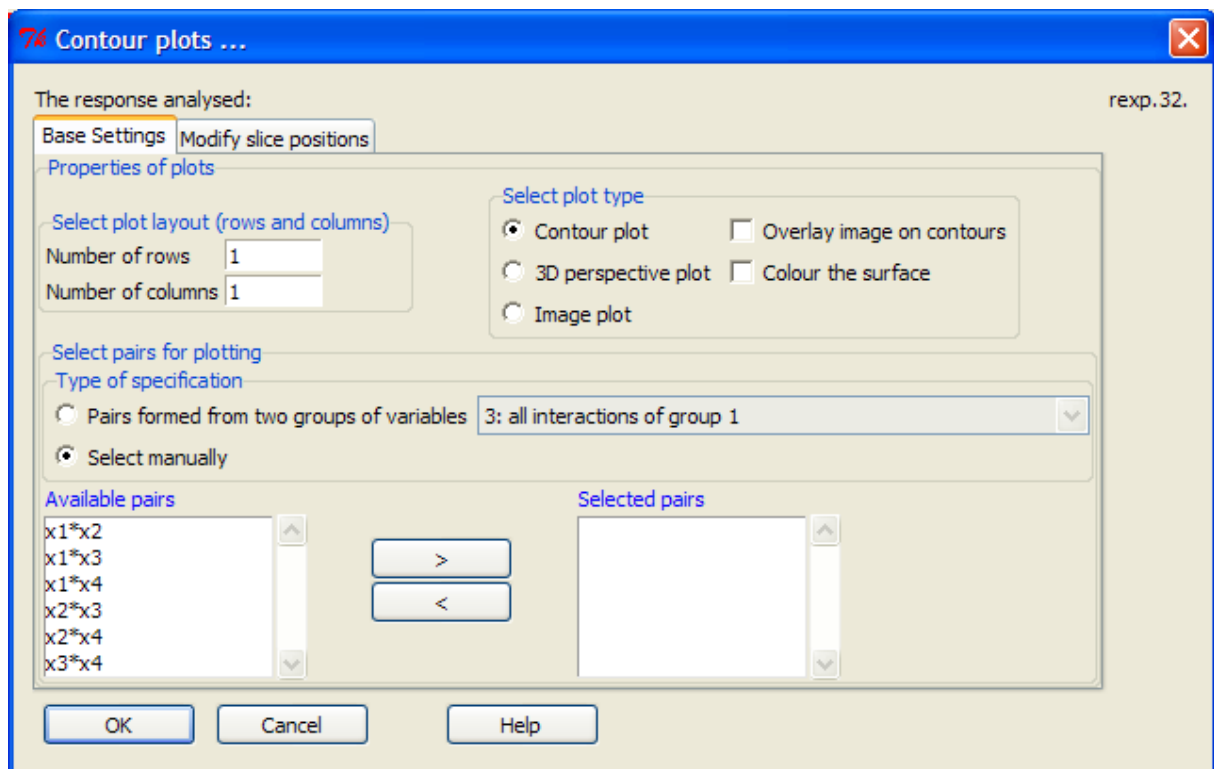
Figure 45: Dialog for creating response surface plots

---

[5] The dialog for main effects and interaction plots for general designs is also available, but only offers visualization of block differences, which may be of secondary interest but are usually not of primary interest.

For example, for the four experimental variables x1 to x4, response surface plots for all six pairs of variables can be shown in 2 row and 3 column layout, specifying number of rows and number of columns accordingly and moving all pairs from "Available pairs" to "Selected pairs", with the "Select manually" radio button active.

5.6 A latin hypercube design for computer experiments

For computer experiments, it often does not increase experimentation cost to have many different factor levels, and at the same time replicates do not convey new information. Therefore, space-filling designs like latin hypercube designs have been proposed for experimenting with quantitative factors in computer experiments. Some such designs can also be generated with the DoE GUI using the menu item

Design → Create design → Space filling latin hypercube …

The dialog for these designs (cf. Figure 46) asks for the method of design generation and for the number of decimal places of factor levels – the factors have as many levels as there are experimental runs, and they are equally distributed between scale ends; the scale ends can be specified on the Factor Details tab of the dialog (default: minimum=0, maximum=1). Currently, only designs from *R* package **lhs** are implemented in the DoE GUI – in the future, some designs from package **DiceDesign** may be added. Pressing OK with unchanged default settings produces an array with 20 runs in four factors immediately. Note that the radio button choice "optimum" can sometimes take some time. On the other hand, the "Type of lhs" radio button should usually not be set to "random".
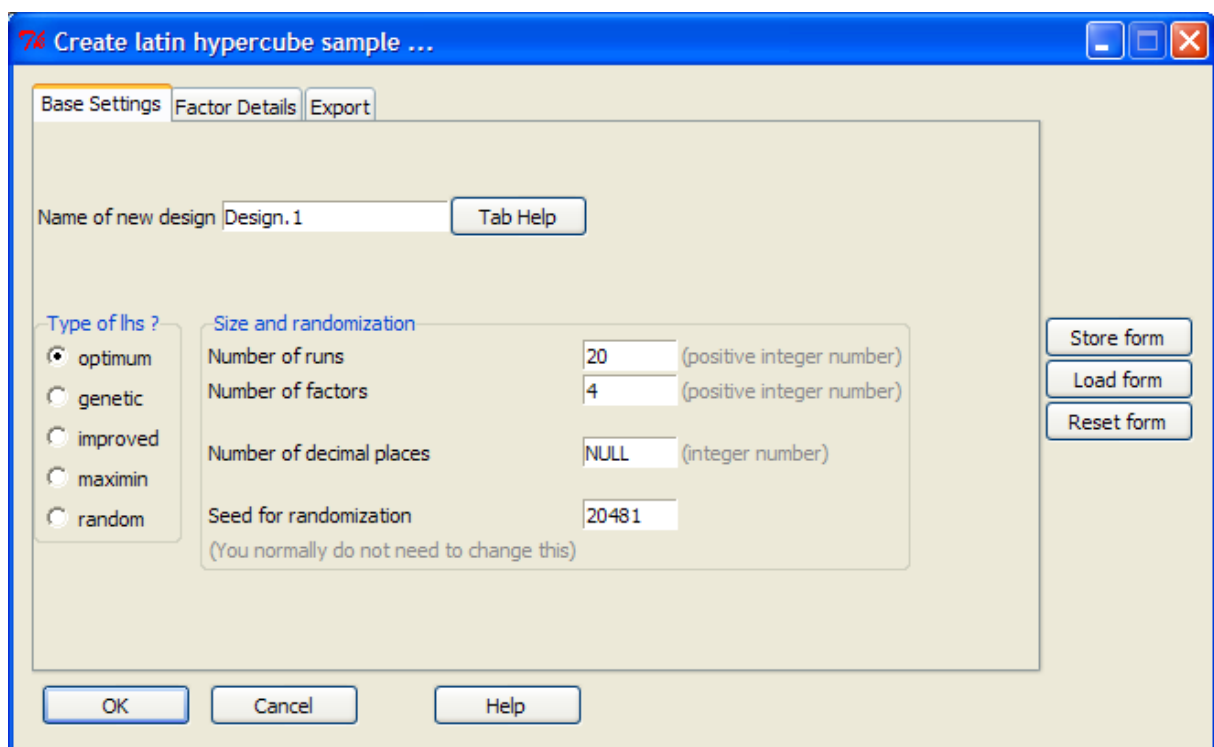
Figure 46: Dialog for creating latin hypercube designs

6 Repeated measurements, long and wide versions, and aggregation

The potato cannon experiment created in Section 3 had 4 repeated measurements per run, which were entered into the software as means and standard deviations. This design is now re-entered into *R* with individual measurements, in order to look at long and wide versions and aggregation, which will also be needed for the next section. The dialog in Figure 47 creates a screening design like in Section 3.2, but with four repeat only replications – i.e. four replications and the `repeat.only` checkbox checked. In the resulting experimental plan, all 4 shots for each run directly follow underneath each other, i.e. the design has 48 rows (12 runs with 4 shots each). This is called the long format (many rows).
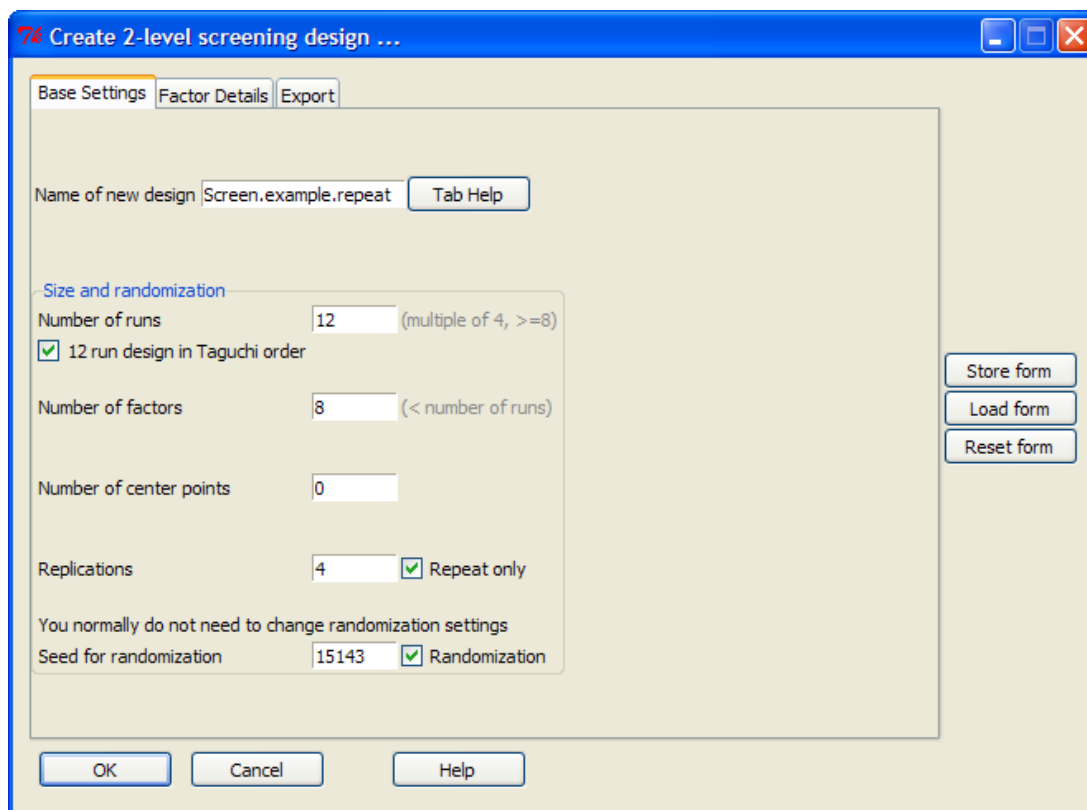
Figure 47: Create potato cannon experiment with repeated measurements

Verify the long format by

    Design → Inspect design → Display active design

For working with the design, it may be more convenient to have repeat shots next to each other in the same row. This can be achieved by changing from long to wide format with menu item

    Design → Modify design → Change from long to wide format

which is available whenever a long format design is the active data set. After applying the long-to-wide change, the wide format design looks as shown in Table 7.

The wide (or long, if preferred) design can be exported, populated with response values and re-imported exactly as shown in Section 3. Once this has been done, the response can be aggregated in several ways (after bringing it to wide format, if necessary), using the menu item

6 Repeated measurements, long and wide versions, and aggregation

Design → Modify design → Aggregate design …

The dialogue for aggregation (cf. Figure 48) offers selection of a response (named as in the long format) and an aggregation function (e.g. mean). A column with the aggregated response values will be added to the design and becomes a response of the design. Originally, the individual response values are also available, after the first aggregation, they are not considered responses any longer. However, remember that numeric variables can always be selected or deselected as responses, even the individual measurements.

Table 7: Wide format design prepared for individual measurements displayed in standard order

```
> print( Screen.example.repeat.wide , std.order=TRUE)
   run.no.in.std.order run.no AirVolume Valve Barrel Angle Pressure WadType
7                    1      7       198     1    4ft    45       20   paper
4                    2      4       198     1    4ft    45       20   cloth
8                    3      8       198     1    6ft    60       40   paper
1                    4      1       198     2    4ft    60       40   paper
9                    5      9       198     2    6ft    45       40   cloth
5                    6      5       198     2    6ft    60       20   cloth
2                    7      2       672     1    6ft    60       20   paper
11                   8     11       672     1    6ft    45       40   cloth
12                   9     12       672     1    4ft    60       40   cloth
6                   10      6       672     2    6ft    45       20   paper
10                  11     10       672     2    4ft    60       20   cloth
3                   12      3       672     2    4ft    45       40   paper
   Voltage BallType e1 e2 e3 y.1 y.2 y.3 y.4
7        9    white -1 -1 -1  NA  NA  NA  NA
4       27     pink  1  1  1  NA  NA  NA  NA
8        9    white  1  1  1  NA  NA  NA  NA
1       27     pink -1 -1  1  NA  NA  NA  NA
9        9     pink -1  1 -1  NA  NA  NA  NA
5       27    white  1 -1 -1  NA  NA  NA  NA
2       27     pink -1  1 -1  NA  NA  NA  NA
11      27    white -1 -1  1  NA  NA  NA  NA
12       9     pink  1 -1 -1  NA  NA  NA  NA
6        9     pink  1 -1  1  NA  NA  NA  NA
10       9    white -1  1  1  NA  NA  NA  NA
3       27    white  1  1 -1  NA  NA  NA  NA
NOTE: columns run.no.in.std.order and run.no are annotation, not part of the data
frame
```
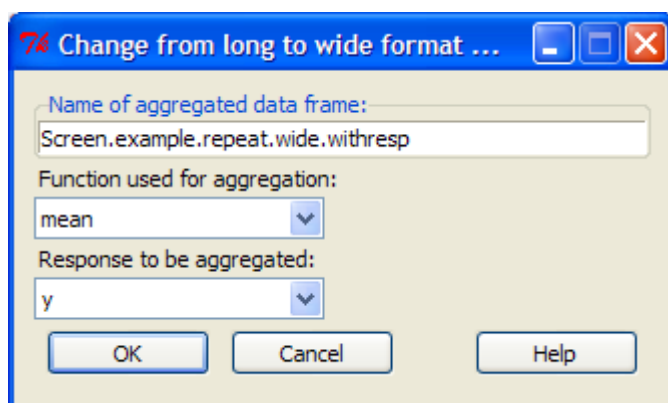

Figure 48: Dialog for aggregating responses for wide format designs

7 Taguchi parameter designs

This topic is illustrated with an injection molding experiment from the literature (cf. Table 11.33 from the book by Hedayat, Sloane and Stufken 1999[6]). In that experiment, seven control factors control the behavior of an injection molding process, which has to work well in the presence of three noise factors. The response is percentage shrinkage of the molded product. The design consists of an 8 run inner array for the seven control factors, crossed with a four run outer array for the three noise factors. The design can be analyzed as one large array or by aggregating over the outer array for each run of the inner array. Both approaches will be shown here, and the outcome of their analysis compared; many statisticians are a bit wary of the parameter design approach, and its risks will be illustrated in Section 7.2.

7.1 Creation and data entry

The design is generated with the software with the following steps
1. The inner array (`inner`) is created as a regular fractional factorial with 8 runs and 7 factors with factor names
    o   CycleTime,
    o   MoldTemp,
    o   HoldPress,
    o   CavityThick,
    o   InjectSpeed,
    o   HoldTime,
    o   GateSize

    using the menu item

    Design → Create design → Regular (Fractional) Factorial…

    Levels are "-1" and "+1" for all factors; factors CavityThick, InjectSpeed and HoldTime have their levels swapped in the dialog in order to match the published design structure. The seed is 30537.
2. The outer array (`outer4`) is created as a regular fractional factorial with 4 runs and 3 factors with factor names
    o   PercRegrind,
    o   Moisture,
    o   AmbientTemp

    using the same menu item; here, AmbientTemp has its levels swapped, and the design is not randomized.
3. The two designs are combined by the menu item

    Design → Create design → Taguchi inner/outer array…

Creation of fractional factorial 2-level designs has already been covered, and this case is identical to other cases. Note that it is recommended to separately store the inner and outer array, because this makes it easy to do subsequent modifications.

The third step is done with the dialogue shown in Figure 49: The design name has been adapted, the appropriate arrays have been chosen, and it has been decided to

---

[6] Hedayat et al. quote Engel 1992 for these data and mention that they use the data with "corrections" proposed by Steinberg and Bursztyn 1994; note, however, that the data from the book differ from those in Steinberg and Bursztyn in two details; as the results look more convincing for the version given in the book, the data are used in that form.

generate the design in the wide version that is used when considering means or variances or SN ratios of the outer array data w.r.t. the inner array factors.
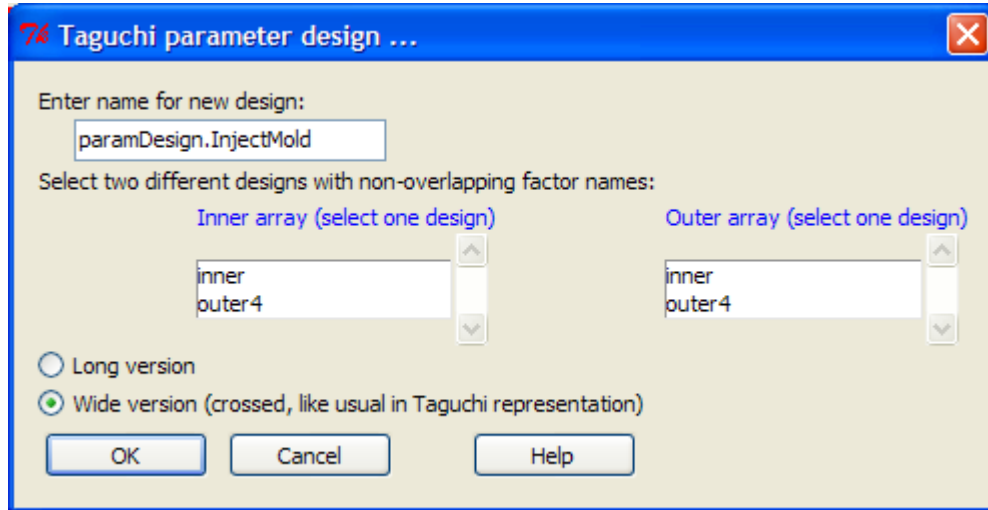

Figure 49: Dialog for creating a Taguchi parameter design[7]
(Note: A long design can be reformatted to wide, but not vice versa;
the corresponding long design is also provided as accompanying material;
however, analysis facilities for long designs are currently very limited)

Table 8: The injection molding data
(order of factors and runs different from Hedayat, Sloane and Stufken (1999),
but design identical; output from software has been slightly edited to fit page)

```
CycTi   MT    HP    CavTh   InjSp   HTi    GS     y.1 y.2 y.3 y.4
-1     -1    -1    -1      -1      -1     -1     2.2 2.1 2.3 2.3
 1     -1    -1     1       1      -1      1     3.0 3.1 3.0 3.1
-1      1    -1     1      -1       1      1     0.5 3.1 0.4 2.8
 1      1    -1    -1       1       1     -1     4.0 1.9 4.6 2.2
-1     -1     1    -1       1       1      1     2.5 0.3 2.7 0.3
 1     -1     1     1      -1       1     -1     2.1 4.2 1.0 3.1
-1      1     1     1       1      -1     -1     2.0 1.9 1.8 2.0
 1      1     1    -1      -1      -1      1     2.0 1.9 1.9 1.8


     Outer array:
       Moisture PercRegrind AmbientTemp
   1       -1            -1           -1
   2        1            -1            1
   3       -1             1            1
   4        1             1           -1
```

Again, response data are added to the design by exporting to a spreadsheet (`csv`), adding the data, and re-importing to *R*. The data in standard order are shown Table 8. The four response values for each run of the inner array can then be aggregated into a mean, standard deviation and/or SN ratio.

---

[7] Note: `outer4` has been chosen instead of `outer`, because `outer` cannot be exported because it is also the name of a function; presumably, this will be fixed in a later version of the software (but does not have very high priority).

7.2 Data analysis

The half-normal effects plots for average response and log of standard deviation (Figure 50, untransformed looks very similar) show that none of the factors significantly affects the mean, while Holding Time strongly affects the standard deviation (lower holding time = lower standard deviation). The main effects plots (cf. Figure 51) show that some factors may nevertheless be active for the main effects – a half normal plot does not work well if half or more of the effects are active.

In addition, the design is of resolution III, and all main effects are heavily confounded with two-factor interactions. Certainly, we seem to learn from this experiment that Holding Time should be short in order to reduce variability of the outcome. The big issue with the 8 runs of the inner array is the severe confounding they suffer from. Column F is confounded with AG, BC and DE. Thus, one can never be quite certain about correct interpretation. This design is an impressive example of that fact, as will be seen now.

Half Normal Plot for y.mean, alpha=0.1

A = CycleTime , B = MoldTemp , C = HoldPress , D = CavityThick , E = InjectSpeed , F = HoldTime , G = GateSize

Half Normal Plot for ln.s, alpha=0.1

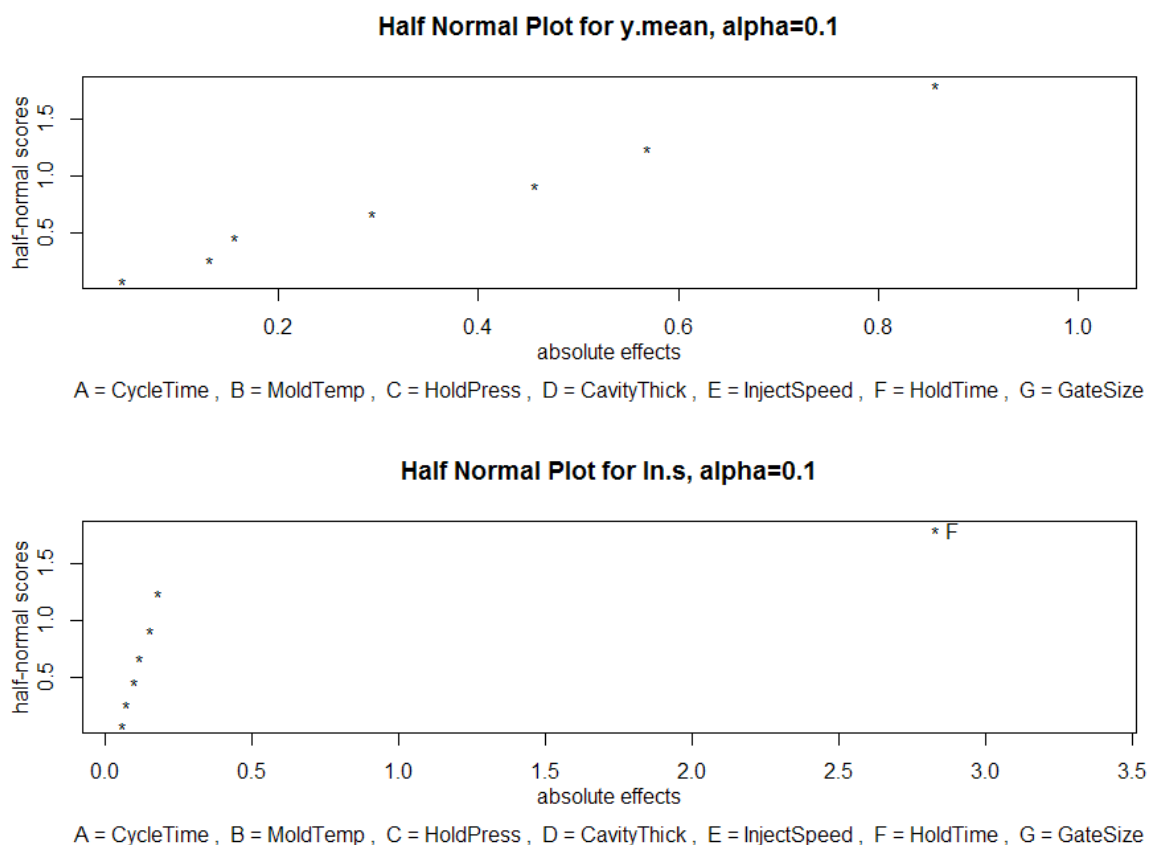A = CycleTime , B = MoldTemp , C = HoldPress , D = CavityThick , E = InjectSpeed , F = HoldTime , G = GateSize

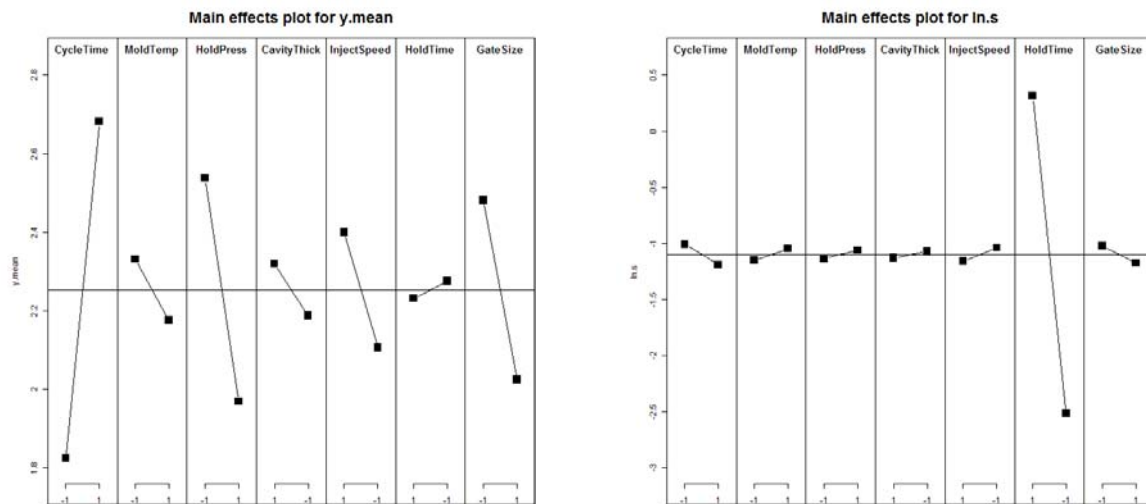Figure 50: Half normal plots for average and ln(standard deviation) of response

Figure 51: Main effects plots for average and ln(standard deviation)
in injection molding experiment

For the following simple analysis to hold, it is necessary to assume that the experiment was not conducted in split-plot form – i.e. preparing each setting for the inner array once and then conducting all runs of the outer array with it – but was conducted as a properly randomized experiment of all 32 combined runs. Note that this was probably not the case. This makes the analysis itself questionable, but not the message about incorrect interpretation in the previous analysis. The point of an analysis of one large array is to estimate all interactions between control and noise factors and use control factors that interact with noise factors for robustification. The crossed array allows unbiased estimation of all interactions between control and noise factors; however, one has to rely on the somewhat ambitious assumption that there are neither interactions among control factors nor interactions among noise factors (since all interactions within each group of factors are confounded with main effects). If we want to analyze the design as though it had been run as a 32 run design, we first need to generate the design as a Regular (Fractional) Factorial; this can be accomplished – with some expertise – by requesting all interactions between control and noise to be estimable on the Estimable Model tab of the relevant dialog (combined design available as `InjectMoldLong.rda`). After populating that design with response values in the usual manner (response data available in `InjectMoldLong.withresp.HSS.csv`; let us call the enriched *R* data frame `InjectMoldLong.withresp`), we can use the GUI directly for a half normal plot (cf. Figure 52)[8]. Amazingly, the graph looks very different from what was found for the wide array, where there were no effects marked for the mean and only factor F (Holding Time) for the standard deviation. With the combined analysis, the interactions between mainly

---

[8] In principle, the analysis can also be run on the parameter design in long version (`paramDesign.InjectMold.long.withresp`, not provided). However, since parameter designs are usually run as split-plot designs, the GUI does not support this analysis, because it is not usually statistically adequate. For a responsible expert decision to nevertheless analyse a parameter design in long format, users have to use command line programming. Figure 52 can be obtained by the following command:

```
DanielPlot(lm(y ~ (.)^2, paramDesign.InjectMold.long.withresp),
    half=TRUE, code=TRUE)
```

factors D and E and the noise factors H and J indicate that robustification of the process can be expected from playing with the settings of factors D and E rather than factor F. Hence, it appears that the large effect of factor Hold Time (F) in Figures 50 and 51 – also diagnosed as a large dispersion effect in Engel 1992 – is in fact presumably due to the interaction of factors Cavity Thickness and Injection Speed with which Hold Time is confounded. Steinberg and Bursztyn (1994) discussed this in more detail, also reflecting on why the effects of factors Cavity Thickness and Injection Speed cancel out in Engel's (1992) original analysis of the experiment.
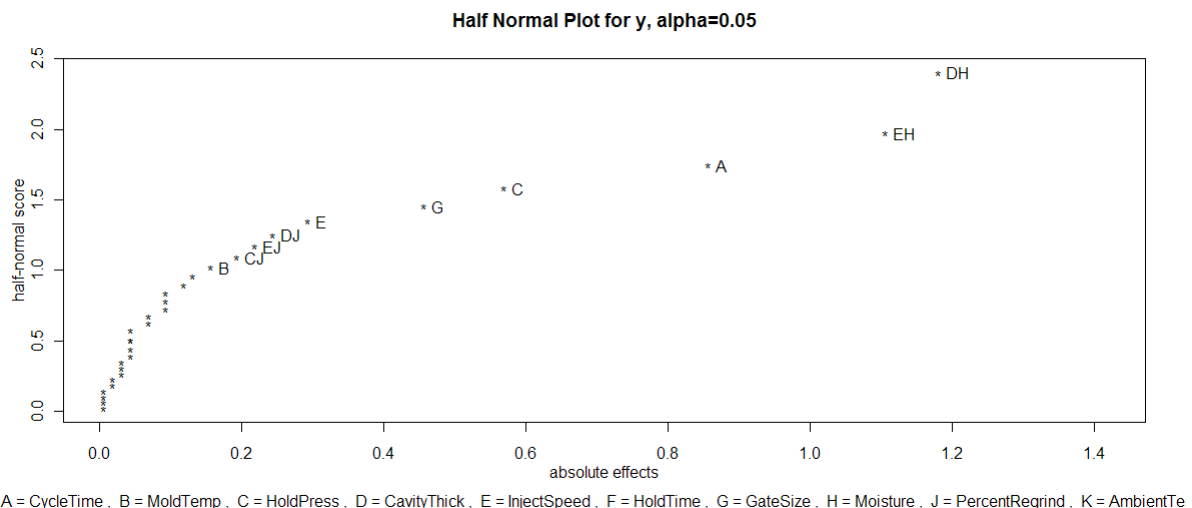

Figure 52: Half normal plot for the combined analysis of 32 run injection molding experiment

Note that the half normal plot of Figure 52 is presumably not quite adequate, as the experiment has most likely been conducted as a split-plot design, doing all outer array runs for each inner array run in direct sequence without resetting inner array factors for full experimental variability. Thus, the whole plot factors, i.e. the main effects, have larger intrinsic variation than the split-plot factors (noise factors) and interactions of split-plot factors with whole plot factors.

Furthermore, the message of this section should not be taken to mean that a combined array is always superior to a crossed array in line with the Taguchi approach. Dispersion effects arising from interactions between noise factors and control factors can be detected very well using combined arrays. However, there may also be proper *dispersion* effects that do not arise from interactions; these may be more easily detected with crossed arrays.

8 Using the stored code for command line programming

Per default, the software shows a history of all commands used in the script window. These can be stored for documentation purposes, or they can be used as a basis for further analyses.

Users can edit the script window as they like. Commands can also be resubmitted from there. Users who are willing and able to do command line programming themselves can use this feature for getting familiar with the commands and their usage – however, be aware that the commands generated by the dialogs are often not in the simplest possible

form, because programming economy has treated many cases alike and thus has included options in their default form that could also have been omitted.

It may be wise to store the script file and/or the content of the output window during or after an *R* commander session; this can be done via the

Design → Export → Save script as …

and/or

Design → Export → Save output as …

menu items. This way, the point-and-click history is in some way documented, and the code can e.g. serve as an example for repeating a design generation or an analysis with slightly modified settings. For example, after a full factorial design has been created, with all factor details entered, it might be the case that a decision is made to save experimental runs and go for an orthogonal array only. Instead of redoing all the factor details entries, it would be less work to create the orthogonal array with standard settings and to then change the factor details by copying the entry for the `factor.names` option from the command for creation of the full factorial to the command for creation of the orthogonal array. Such chances may eventually make users realize that command line programming is in the long run more economical. Certainly, the flexibility of the packages for command line programming is much higher than that of the DoE GUI. For the occasional user, GUI usage may remain the best alternative, however.

9 References

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

Engel, J. (1992). Modeling variation in industrial experiments. *Applied Statistics* **41**, 579-593.

Fox, John (2005). The *R* Commander: A Basic-Statistics Graphical User Interface to *R*. *Journal of Statistical Software* **14**, Issue 9, 1-42.

Grömping, U. (2011). Relative projection frequency tables for orthogonal arrays. Report 1/2011, Reports in Mathematics, Physics and Chemistry, Department II, Beuth University of Applied Sciences Berlin.

Hedayat, A.S., Sloane, N.J.A. and Stufken, J. (1999). *Orthogonal arrays. Theory and applications*. With a foreword by C. R. Rao. Springer Series in Statistics. Springer, New York.

Mayfield, P. (2007). Potato Cannons and the Taguchi L12 – a Design of Experiments Case Study. http://www.sigmazone.com/Taguchi_L12_SpudGun.htm Accessed 07 Nov 2011.

Mellor-Crummey, J. (2005). Lecture notes of Lecture 528. http://www.cs.rice.edu/~johnmc/comp528/lecture-notes/Lecture18.pdf Accessed 07 Nov 2011.

Montgomery, D.C. (2001, 5th ed.). *Design and Analysis of Experiments*. Wiley, New York.

Steinberg, D. and Bursztyn, D. (1994). Dispersion effects in Robust-Design Experiments with Noise Factors. *J. Quality Technology* **26**, 12-20.

Tsao, *R*.F. and Margolin, B.H. (1971). Multifactor Paging Experiment: II Statistical Methodology. in Freiberger, W., Ed. *Statistical Computer Performance Evaluation*, Academic Press, NY, 135-162.

9 References

R packages:

Barrios, E. (2009). **BsMD**: Bayes Screening and Model Discrimination. Version 0.6-5.2.

Carnell, *R*. (2009). **lhs**: latin hypercube samples. Version 0.5.

Fox, J. (2011). **Rcmdr**: *R* Commander. Version 1.7-2.

Franco, J., Dupuy, D. and Roustant, O. (2011). **DiceDesign**: Designs of Computer Experiments. Version 1.1.

Grömping, U. (2011). **DoE.base**: Full factorials, orthogonal arrays and base utilities for DoE packages. Version 0.22-8.

Grömping, U. (2011). **FrF2**: Fractional Factorial designs with 2-level factors. Version 1.2-10.

Grömping, U. (2011). **DoE.wrapper**: Wrapper package for design of experiments functionality. Version 0.8-6.

Grömping, U. (2011). **RcmdrPlugin.DoE**: *R* Commander Plugin for (industrial) Design of Experiments. Version 0.11-5.

Wheeler, B. (2010). **AlgDesign**: Algorithmic Experimental Design. Version 1.1-7.

all In *R* development core team (2011). *R*: A language and environment for statistical computing. *R* Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.*R*-project.org/.