§3_D-efficient_designs.pdf

GENERATION OF *D*-EFFICIENT DESIGNS

[Accompanying material for: Katrin Auspurg & Thomas Hinz (2015): Factorial Survey Experiments. Sage Series: Quantitative Applications in the Social Sciences No. 175. Thousand Oaks, CA: Sage]

Material that is provided along with this introduction (§3_material):

- The complete coding to produce the fairness of earnings example
  - as an SAS program (also called “SAS job”): SASjob_earningsexample.sas
  - as a pdf: SASjob_earningsexample.pdf; this file can also be employed after pasting the commands into the SAS program editor. The word document might be helpful for readers who do not yet have experience with SAS or a program license in case they want to get a quick overview of the coding needed to run the macros.
- The data set that will be produced by the macro is provided in Stata format: **fractionalized120.dta** (within the **§4_material → Folder “Stata”**) and can be used to produce the text vignettes or to compare the efficiency with alternative random samples (for more information see §4_material: → §4_vignettetexts.pdf).

3.1 SAS Macros

In the following we give short instructions on how to use the SAS macros for *D*-efficient experimental designs and how to interpret the SAS output (**some recommendations of other software packages are given in Section 3.2**). Again the fairness of earnings example is used to illustrate the single steps. A detailed documentation of the macros is provided in Kuhfeld (2010), and readers might also look for updates on the SAS web pages.

3.1.1 Installation of Macros

Note that at least with the desktop version of SAS all codes will work only if you first **installed the relevant macros**, as these are not part of the core package of SAS but are user-written packages (author: Warren Kuhfeld). The macros are available for free on the official website of SAS.[1] “Installing” entails no more than downloading the packages into the right program folder (where all macros have to be placed). For more details see Kuhfeld (2010) or the information on the SAS website.

3.1.2 Programming Editor and %Mktex Macro

Once the macros are installed, one can use them within the standard **SAS programming editor**. The default setting is that all data files that are produced (like the file containing the *D*-efficient vignette

---

[1] http://support.sas.com/resources/papers/tnote/tnote_marketresearch.html → Folder “Macro Code (ZIP)” [last time visited: September 30, 2014]. When working with web-based SAS versions, like SAS on demand or the SAS student version, one should check in advance if the macros are available. In our experience it is possible to use them in current versions (last check: September 2014) of SAS on demand, but not with the student version (as the macro builds upon some tools not provided with this program version). However, once again we recommend checking the SAS website for up-to-date information.

sample) are stored into the "Work-library" as part of the working memory – meaning all these files are lost when exiting the program. One has to export them to another library before exiting or, even better, **specify another library as a working directory using the LIBNAME command** (LIBNAME path to working directory).

Using the macros is easy as one does not need much programming code.[2] The following code would be sufficient to **start an algorithm search** for a *D*-efficient fraction of 120 vignettes ("runs") with an orthogonalization of some specific two-way interactions out of a $3^2 2^3 4^2$-full factorial (i.e. a vignette universe built up from seven different dimensions: two dimensions with three levels, three dimensions with two levels, and two dimensions with four levels).

```
%mktex (3 3 2 2 2 4 4, interact x1*x2 x1*x3 x5*x6, n=120);
```

%mktex calls the macro, and within the parentheses the desired design specifications are given. The numbers indicate the levels for all seven dimensions. Detailed options, like the interactions one wants to orthogonalize or the desired size of the sample ("design"; in this case: *N*= 120 vignettes), are separated by commas, while the semicolon denotes the end of the command. In the example, beneath the main effects the two-way interactions between the 1st and 2nd, the 1st and 3rd, and the 5th and 6th dimensions will get orthogonalized (i.e. in the sample not only all dimensions but also these interaction terms will show minimal correlations and multicollinearities with each other as well as with the main effects of dimensions, meaning that later on in data analyses the effects of all these parameters will be identifiable).[3] To request a Resolution V design (i.e. orthogonalization of *all* two-way interactions), one could simply use the subcommand `interact=@2` instead:

```
%mktex (3 3 2 2 2 4 4, interact=@2, n=120);
```

It is possible to specify **further options**, like the maximum search time the algorithms should take or a minimum (satisficing) *D*-value that has to be reached.[4] The search process used by the algorithms has some randomness built in. To make it replicable one has to specify a starting number that is used; otherwise the computer's real time clock (which is the default) will be used. This starting number can be set with the option "set seed = number", with "number" being any number one might choose (for an example using "815" as a seed see Section 3.1.3). Note that when not using this option, you will always get a somewhat different design when the macro is re-run. We recommend using the option, as replicability is a desirable feature of all research designs. Kuhfeld (2010) gives thorough explanations for additional options of these macros and related macros (which allow one, for instance, to look for the minimum numbers of vignettes that are needed to build up the desired design).

---

[2] Or as Kuhfeld frames it: "just tell it [the %mktex-macro] what you want, and it figures out a good way to do it […]. This macro should always produce a very good design with minimal effort for even the most unsophisticated users" (Kuhfeld, 2010, p. 1018). However, as explained in detail in our textbook, one should be very careful when selecting the design features (like parameters that should be identifiable) and always thoroughly check the resulting design before going on with field work.

[3] In order to estimate the effects, there also have to be enough valid ratings for all of the vignettes in the sample; see Chapters 3 and 5 of the textbook.

[4] Which will, however, rarely be necessary as standard designs are produced within a few minutes.

3.1.3 Exclusion of Implausible and Illogical Cases

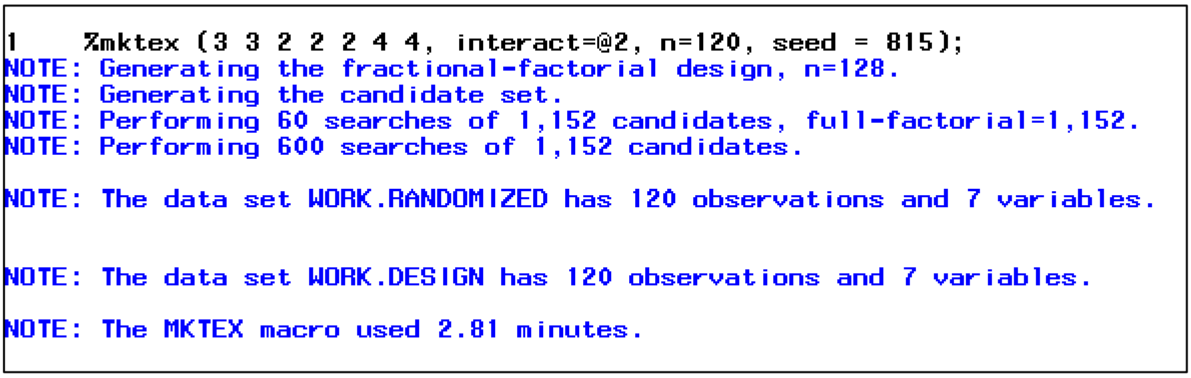
Implausible or "bad" cases have to be excluded before running the %mktex macro from the candidate set for the sample. This is done by first specifying these cases in a container (which is labeled "impl" in the following sample code, but one might just use any name), and then using the option "restriction=containername" in the %mktext macro. For instance, one can exclude the joint occurrence of the first level of dimension 1 with the second or third level of dimension 2 by running the following commands:

```
* specification of the restriction (i.e. the cases to be excluded);

%macro impl;
     bad =      x1=1 & x2 > 2
     %mend;

*Generation of the D-efficient design under this restriction of
excluding the 'bad' cases;


%mktex (3 3 2 2 2 4 4, restrictions = impl, interact=@2, n=120, seed =
815);
```

3.1.4 Information Provided in the SAS Log and Output Windows

When running the macro, **summary information on the search process and its success** is provided within the ***SAS log*** window (see the log file shown in Figure 3-1; for more detailed information, see Kuhfeld, 2010, p. 1017 et seq.). It starts with some information on the initial candidate set used for the algorithms. This candidate set can have more observations ("runs") than are requested for the final design. The further lines inform us about the number of algorithm search steps for better designs, the size of the full factorial, and finally the two output data files that are produced. The file "RANDOMIZED" (which is here simply stored into the working directory ("WORK") as no other library was specified) displays the different vignette cases in a random order; the file "DESIGN" contains exactly the same sample but without random vignette sorting. Lastly, the time the macro needed to create these data sets is shown. In our experience, for common FS designs and using the high working memories of current PCs, it will rarely take the macro more than ten minutes to find an optimal design.

Figure 3-1 Sample of a SAS Log File

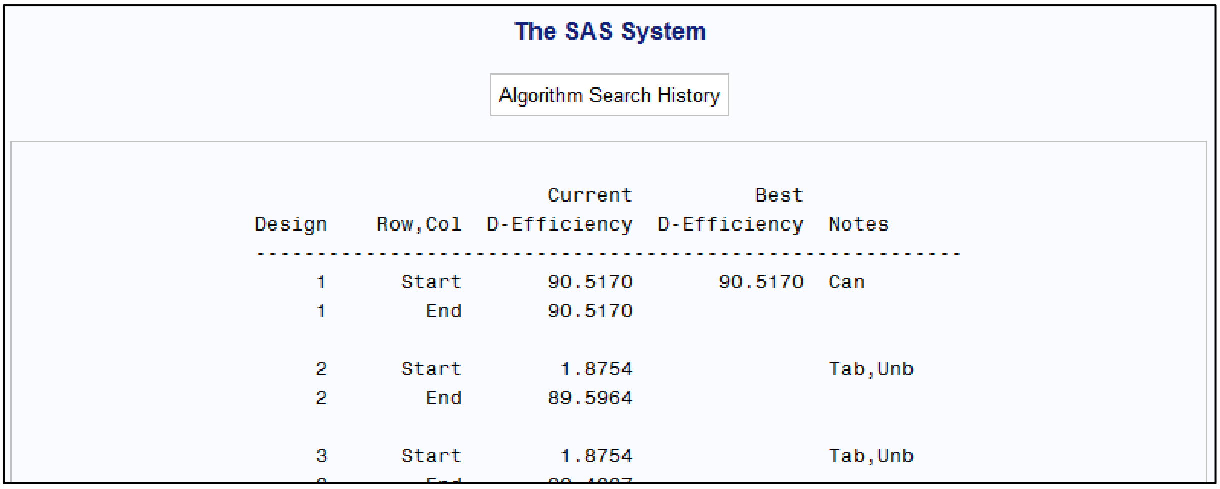


```
1    %mktex (3 3 2 2 2 4 4, interact=@2, n=120, seed = 815);
NOTE: Generating the fractional-factorial design, n=128.
NOTE: Generating the candidate set.
NOTE: Performing 60 searches of 1,152 candidates, full-factorial=1,152.
NOTE: Performing 600 searches of 1,152 candidates.

NOTE: The data set WORK.RANDOMIZED has 120 observations and 7 variables.


NOTE: The data set WORK.DESIGN has 120 observations and 7 variables.

NOTE: The MKTEX macro used 2.81 minutes.
```

Sometimes the macro does not come to a result or does not even start; when this happens you will find **error messages** in the log file that elucidate the reason. The most likely reason for the macro not working is probably that you specified not enough vignettes to make it possible for all the desired parameters to be orthogonalized (message: **“More parameters than run”**). Other messages, in contrast, do not stop the macro and are in general no problem as far as the macro comes to a final result (which means that a table with efficiency measures is shown in the output window, as described below). For instance, in the case of strong restrictions to the design, like the exclusion of many implausible cases, one often gets the “**NOTE: The candidate set initialization failed**”. However, the macro is still able to continue and produce a meaningful vignette sample.

In addition, **more detailed information on the search process** is provided in the ***SAS outcome window*** (also called the ***Result viewer)***. Here the different algorithms used by the macro and the search history are documented. For instance, in the example shown in Figure 3-2 the first algorithm used is the “PROC OPTEX” procedure based on the “Fedorov algorithm” that exchanges data rows (“runs”) of the candidate set and keeps them in case they improve *D*-efficiency. After that initial algorithm other ones are used that, for instance, check whether swapping the levels of the design can further improve the *D*-efficiency. Next, it reveals which of the different algorithms worked best in terms of *D*-efficiency. Finally, based on this approach, the macro attempts to enhance *D*-efficiency even further by using more iterations and further level mutations (“refinements”). The output of this search and refinement history is organized in tables, as can be seen in the record of the start of the search history for our earnings example displayed in Figure 3-2. Each “Design” number (see the first column) corresponds to a complete iteration, based on a different initialization. In columns two to four the initial and final efficiency of each iteration step is shown. The “Best D-Efficiency” column only states those *D*-efficiency values that outperform prior ones, while “Notes” provides details on the algorithms that were used. In addition, when working with restrictions (such as cases that should get excluded) this column displays the note “conforms” once a design first conforms to the restrictions (a detailed list on further possible notes and their meaning can be found in Kuhfeld, 2010, p. 1054).

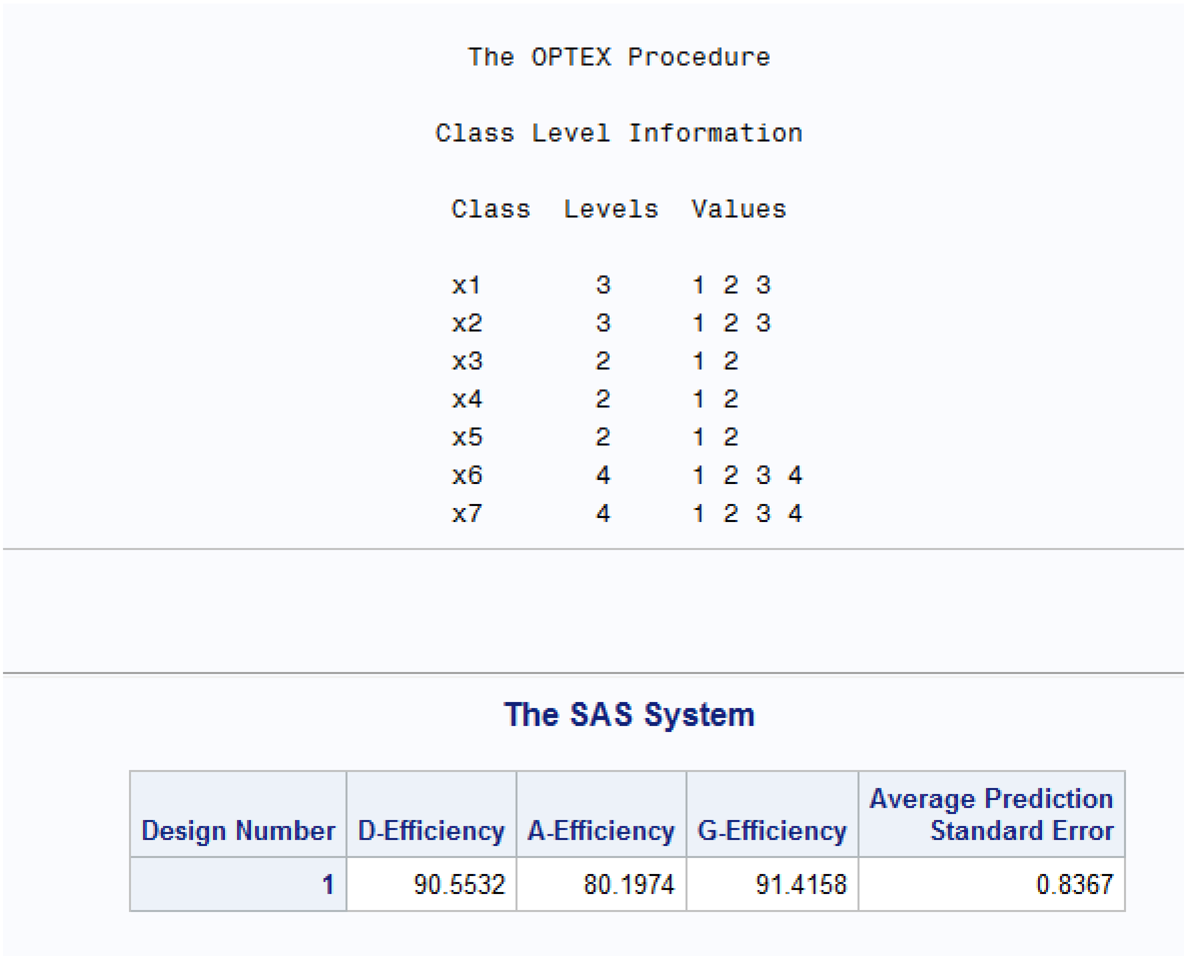
Figure 3-2 SAS Output in the “Results Viewer” (First Lines of Search History)



```
The SAS System

Algorithm Search History

                         Current          Best
Design    Row,Col   D-Efficiency   D-Efficiency   Notes
----------------------------------------------------------
     1      Start        90.5170        90.5170   Can
     1        End        90.5170

     2      Start         1.8754                  Tab,Unb
     2        End        89.5964

     3      Start         1.8754                  Tab,Unb
```

At the end, when everything worked well and the algorithm converged at an optimum *D*-efficiency value, the “PROC OPTEX” procedure is automatically activated, which means that the levels of all dimensions (“factors”) and the final *D*-efficiency get displayed, as can be seen in Figure 3-3.

Figure 3-3 SAS Output in the “Results Viewer”: Information on Final Design



```
                 The OPTEX Procedure

              Class Level Information

               Class   Levels   Values

               x1         3     1 2 3
               x2         3     1 2 3
               x3         2     1 2
               x4         2     1 2
               x5         2     1 2
               x6         4     1 2 3 4
               x7         4     1 2 3 4
```

The SAS System

Design Number	D-Efficiency	A-Efficiency	G-Efficiency	Average Prediction Standard Error
1	90.5532	80.1974	91.4158	0.8367

In our example, a *D*-efficiency value of 90.55 is reached. Keep this information on *D*-efficiency! It is one parameter that one should report to inform readers on the employed sample in a study. You can just save a log of the output window (see the options provided after a right click on this window) or copy the table and paste it into another document, such as a text document. In the table, some alternative efficiency measures are also shown. However, as indicated, we recommend that *D*-efficiency is the focus, as this measure is the most common and flexible one (i.e. it is not dependent on the coding used for the dimensions). As already mentioned, the final vignette sample that holds this efficiency is provided in the dataset “Design” (with using numeric codes for the different levels of the seven different vignette dimensions, which are labeled “x1” to “x7”). In addition, the same dataset but with a randomized order of the vignettes (“runs”) is provided (dataset “Randomized”). All these datasets can be found in the current working directory (which is the library “Work” if no other library is specified).

3.1.5 Blocking Vignettes to Decks

When running the **%mktblock macro** afterwards, the design gets **allocated to different decks** (here called “blocks”), again orientated toward a maximum orthogonality and level balance within the single decks (blocks). The following code can be employed to split our sample of 120 vignettes into six different decks of 20 vignettes each (the macro will automatically work with the dataset in use, which is still the *D*-efficient sample of 120 vignettes as long as no other dataset was opened in the meantime):

```
%mktblock (nblocks = 6);
```

In contrast to the former data sets, the resulting dataset “Blocked” also contains an indicator variable (“block”) for the six different decks. Additional data files are automatically produced and stored in the same library (“Work” if no other library is specified). These files provide some statistical information on the sample and decks (e.g. they contain matrices of the frequencies of different levels and correlations between dimensions). Similar information is also provided in the output window.

3.1.6 Exporting Data to Other Data Formats like Stata

All files can simply be exported to other statistical packages such as Stata or SPSS (see the instructions after a right click on the data files), but you will probably only need the Blocked-data (or Randomized data if not working with fixed decks) as all statistical information (like correlation tables) can easily be produced with other statistical software packages and is probably more economically stored in output windows than in separate data files.

3.2. Further Software Tools

In addition to the SAS macros, the *Ngene* software for designing choice experiments and the *Conjoint Value Analysis* (CVA) module of the *Sawtooth* program provide very comprehensive tools for the construction of optimal experimental designs, while there are also some user-written packages for the freeware software package R that allow searches for *D*-efficient designs (like the AlgDesign package written by Bob Wheeler; see Wheeler 2004). Learning how to use these programs should be no challenge for any researchers who are already familiar with syntax commands in statistical software packages. As there is always a lot of programming work going on, readers might also check themselves if even more sophisticated packages have become available. A website collecting information on enhancements and R packages for experimental designs can be found at http://cran.r-project.org/web/views/ExperimentalDesign.html (last visited in April 2014).

References:

Kuhfeld, W. F. (2010). *Marketing research methods in SAS. Experimental design, choice, conjoint and graphical techniques*. SAS Institute: Cary, NC.

Wheeler, R.E. (2004). *efficient.rounding. AlgDesign*. The R project for statistical computing. http://cran.r-project.org/web/packages/AlgDesign/AlgDesign.pdf; last visited: October 2014).