DoEgen: A Python Library for Optimised Design of Experiment Generation and Evaluation

DoEgen is a Python library aiming to assist in generating optimised Design of Experiments (DoE), evaluating design efficiencies, and analysing experiment results.

In a first step, optimised designs can be automatically generated and efficiencies evaluated for any mixture of factor-levels for numeric and categorical factors. Designs are automatically evaluated as function of number of experiment runs and the most efficient designs are suggested. In particular DoEgen provides computation of a wide range of design efficiencies and allows to import and evaluate externally generated designs as well.

The second part of DoEgen assists in analysing any derived experiment results in terms of factor importance, correlations, and response analysis for best parameter space selection.

Written by Sebastian Haan (Sydney Informatics Hub, The University of Sydney)

Table of Contents

Definitions

An Experiment Design is typically defined by:

- Number of Factors: the parameters or variates of the experiment
- Number of Runs: the number of experiments
- Levels: The number of value options for each factor, which can be either numeric values (discrete or continuous) or categorical. Discrete levels for continuous factors can be obtained by providing the minimum and maximum of the factor range and the number of levels. The more levels, the more "fine-grained" the experiment will evaluate this factor, but also more experimental runs are required.

The goal of optimising an experimental design is to provide an efficient design that is near-optimal in terms of, e.g., orthogonality, level balance, and two-way interaction coverage, yet can be performed with a minimum number of experimental runs, which are often costly or time-consuming.

Functionality

If you would like to jumpstart a new experiment and to skip the technical details, you can find a summary of the main usage of DoEgen in [Case Study Use Case].

Currently, the (preliminary) release contains several functions for generating and evaluating designs. Importing and evaluating external designs is supported (e.g. for comparison to other DoE generator tools). DoE also implements several functions for experiment result analysis and visualisation of parameter space.

The main functionalities are (sorted in order of typical experiment process):

- Reading Experiment Setup Table and Settings (Parameter Name, Levels for each factor, Maximum number of runs, Min/Max etc)
- Generating optimised design arrays for a range of runs (given maximum number of runs, and optional computation-time constraints, see `settings_design.yaml`).
- Evaluation and visualisation of more than ten design efficiencies such as level balance, orthogonality, D-efficiencies etc (see Design Efficiencies for the complete list).
- Automatic suggestion of minimum, optimal, and best designs within a given range of experiment runs.
- Import and evaluation of externally generated design arrays.
- Experiment result analysis: Template table for experiment results, multi-variant RMSE computation, best model/parameter selection, Factor Importance computation, pairwise response surface and correlation computation, factor correlation analysis and Two-way interaction response plots.
- Visualisation of experiment results.

Installation And Requirements

Requirements

- Python >= 3.6
- SWIG >=3.0.12
- OApackage
- xlrd
- XlsxWriter
- Numpy
- Pandas
- PyYAML
- scikit-learn
- matplotlib
- seaborn

The DoEgen package is currently considered experimental and has been tested with the libraries specified in `requirements.txt`.

The OApackage requires an installation of SWIG, which can be found at https://www.dev2qa.com/how-to-install-swig-on-macos-linux-and-windows/or can be installed via conda

```
conda install swig
```

After installing `swig` and `numpy`, DoEgen can be installed either with

```
python setup.py build
python setup.py install
```

or using pip

```
pip install geobo
```

Note that OAPackage can be also installed manually by following installation instructions and documentation for OApackage (tested with OApackage 2.6.6), which can be found at https://pypi.org/project/OApackage/.

User Templates

1) The factor (parameter) settings of experiment are defined in an experiment setup table (see `Experiment_results_template.xlsx`). A new excel setup template table can be also created with `create_setupfile.py`. Each factor is on a new row and specified by `Parameter Name`, `Parameter Type` , `Level Number`, `Minimum`, `Maximum`

2) After the experiment is run, the results have to be filled in an experiment result table (see `Experiment_results_template.xlsx`). A new excel result template table can be also created with `create_resultfile.py` The result table allows to fill in multiple output properties (Y_label: output target to be predicted) and experiment positions. The results have to be provided in the table with the following columns:

- `Nexp`: Run# of experiment, need to match Run# in Experiment setup and design.
- `PID`: Identifier# of label of location (point) in experiment (e.g. if experiment is run at different locations simultaneously).
- `Y Label`: Identifier# or label of Y-Variate (target property that has to be predicted or evaluated, e.g. Rain and Temperature). This allows to include multi-output models with distinct target properties. Note that currently each Y variate is evaluated separately.
- `Y Exp` The experiment result for Y
- `Y Truth` (optional) if the true value available is available for Y. This is required to calculate the RMSE and to select best parameter space.
- Not currently considered (yet) in result stats computation: `Std Y Exp`, `Std Y Truth`, `Weight PID`

	A	B	C	D	E
1	Parameter Name	Parameter Type	Level Number	Minimum	Maximum
2					

Figure 1: Experiment Setup Table Header.

	A	B	C	D	E	F	G	H
1	Nexp	PID	Y Label	Y Exp	Y Truth	Std Y Exp	Std Y Truth	Weight PID
2								

Figure 2: Experiment Result Table Header.

Running Tests

To verify that DoEgen works, you can run the example experiment

```
$ python -m doegen.init_tests
$ python -m doegen.doegen test/settings_design_test.yaml
$ python -m doegen.doeval test/settings_expresults_test.yaml
```

Documentation

Please do not modify `README.md`. Instead make any changes in the master documentation file `MANUAL.md` (uses pandoc markdown syntax) and then convert to the inferior Github markdown flavor (note that the new github-flavored markdown format gfm option does not correctly solve figure caption and resize options):

```
pandoc -f markdown -t markdown_github MANUAL.md -o README.md
```

and to pdf:

```
pandoc -V geometry:margin=1.2in MANUAL.md -o docs/MANUAL.pdf
```

or as standalone html:

```
pandoc MANUAL.md -o MANUAL.html
```

Main Modules and Usage

Design Generation

Design generation with `doegen.py`: Main model for generating optimised designs and computation of efficiencies. Settings are specified in settings yaml file `settings_design.yaml`. If the yaml and .xlsx template files are not yet in your working directory (e.g. after first doegen installation), you can create in the the yaml and excel template files with

```
$ python -m doegen.init_config
```

Before running `doegen.py`,two things have to be the done:

1) fill in experiment setup table (see template provided `Experiment_setup_template.xlsx` or example in `test/` folder)
2) provide settings in settings file (see `settings_design.yaml`)

Now you are ready to run the design generation

```
$ python -m doegen.doegen settings_design.yaml
```

This will produce a number of files for different experiment run length (see folder `test/results/DesignArray_Nrun...`):

- The optimised design array `EDarray_[factor_levelels]_Nrun.csv`.
- A table of design efficiencies `Efficiencies_[factor_levelels]_Nrun.csv`
- Table of Canonical Correlation Coefficients `Table_Canonical_Correlation.csv`
- Table of two-way Interaction balance `Table_Interaction_Balance.txt`
- Table of Pearson correlation coefficients between all factor pairs `Table_Pearson_Correlation.csv`
- Plot of pairwise correlation including regression fit `pairwise_correlation.png` (see example plot below)

Besides the default optimisation (based on function `doegen.deogen.optimize_design`), DoEgen also allows the to construct full orthogonal designs using the function `doegen.doegen.gen_highD`, which is based on OApackage orthogonal arrays and extensions. However, this works only for special cases with limited number of factors and design levels. Thus, it is currently not fully automated but might assist advanced users to construct optimal designs.

Design Selection

DoEgen will select by default three designs based on the following citeria:

1) minimum Design with the criteria:

- number of runs >= number of factors + 1
- center balance > 95%
- level balance > 95%
- Orthogonal Balance > 90%
- Two Level interaction Balance > 90%
- Two Level Interaction Minimum One = 100%

2) optimal Design with the criteria:

- center balance > 98%
- level balance > 98%
- Orthogonal Balance > 95%
- Two Level interaction Balance > 95%
- Two Level Interaction Minimum One = 100%

3) best design which is based on best score that is sum of efficiencies above and includes a small penalty for runsize relative to maximum runsize

This will deliver (see folder `test/results/`):

- Overview summary of the three designs and their main efficiencies: `Experiment_Design_selection_summary.txt`
- Three tables (`Designtable_minimum/optimal/best...csv`) for the there suggested designs that are converted in the actual level values
- An overview of the efficiencies is plotted as function of exp run and saved in `Efficiencies_[factor_levels].png`

In case the user wants to select another design for a different run size, one can covert the design array into a design table with the function `doegen.deogen.array2valuetable()`.

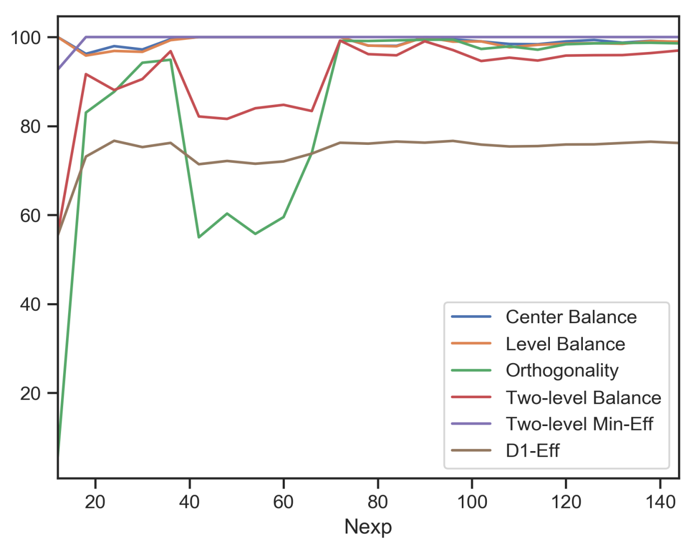

Figure 3: Example overview plot of the main efficiencies (from 0=worst to 100=best) as function of number of experiments.

Design Efficiencies

DoEgen computes more than ten efficiencies and saves them as .csv file for each generated design array. All indicators, except for the canonical correlations, have a range from 0 (worst possible) to 1 (optimal):

- Center Balance: 100% [1 - Sum(Center-Deviation)/Array Size], i.e. the average center balance over all factors.
- Level Balance: Defined as 100% [1 - Sum(Imbalance)/Array Size], the average level balance over all factors.
- Orthogonality: Defined as 100% [1 - Orthogonality], i.e. the average orthogonality over all factor pairs.
- Two-way Interaction Balance: Similar to level balance but for pairwise factor balance.
- Two-way Interaction with at least one occurrence: 100% [1 - Sum(Not at least one pairwise factor occurrence)/number of pairwise combinations]; 100% if all factor-level pair combinations occur at least once.
- D-Eff: D-Efficiency (model includes main term and quadratic).
- D1 Eff: only main terms
- D2 Eff: main, quadratic, and interaction terms
- A-Eff: A-efficiency (main term and quadratic)
- A1-Eff: only main terms
- A2-Eff: main, quadratic, and interaction terms
- Acor_can_avg: average canonical correlation efficiency
- Acor_can_max: maximal canonical correlation coefficient

For further inspection, `doegen.deogen.evaluate_design2` creates also the following tables and plots:

- Table of Canonical Correlation
- Table of Pearson Correlation (same as above if normalised discrete variables)
- Table of Two-way Interaction Balance
- Cornerplot of pairwise factor relation with Y

Experiment Result Analysis

Experiment Result Analysis with `doeval.py`: The experiment results have to be provided in a result table with the format as specified in #user-templates, and specifications in the `settings_expresults.yaml` file. Then run

```
$ python -m doegen.doeval settings_expresults.yaml
```

This will create the following stats tables and plots (see folder `test/expresults/` as example):

- A valuation of the factors in term of "importance", which is defined by the maximum change (range) in the average Y between any factor levels. Results are visualized in bar plot and saved as csv, including, min, max, std deviation across all levels
- Computes RMSE between experiment result and ground truth; results saved as csv.
- Ranks list of top experiments and their parameters based on RMSE
- Computes average and variance of best parameters weighted with RMSE; saved to csv file
- An overview plot of all the correlation plots between Y and each factor (see function `plot_regression`)
- Moreover it will plot Y value for each pairwise combination of factors (see function `plot_3dmap`), which allows the user to visualise categorical factors

Use Case Study

Here we demonstrate a typical use case where we would like to first generate and select an optimal experiment design. Then subsequently after running the experiment we would like to answer the question which is the best parameter space and what parameters are important. Our case study is

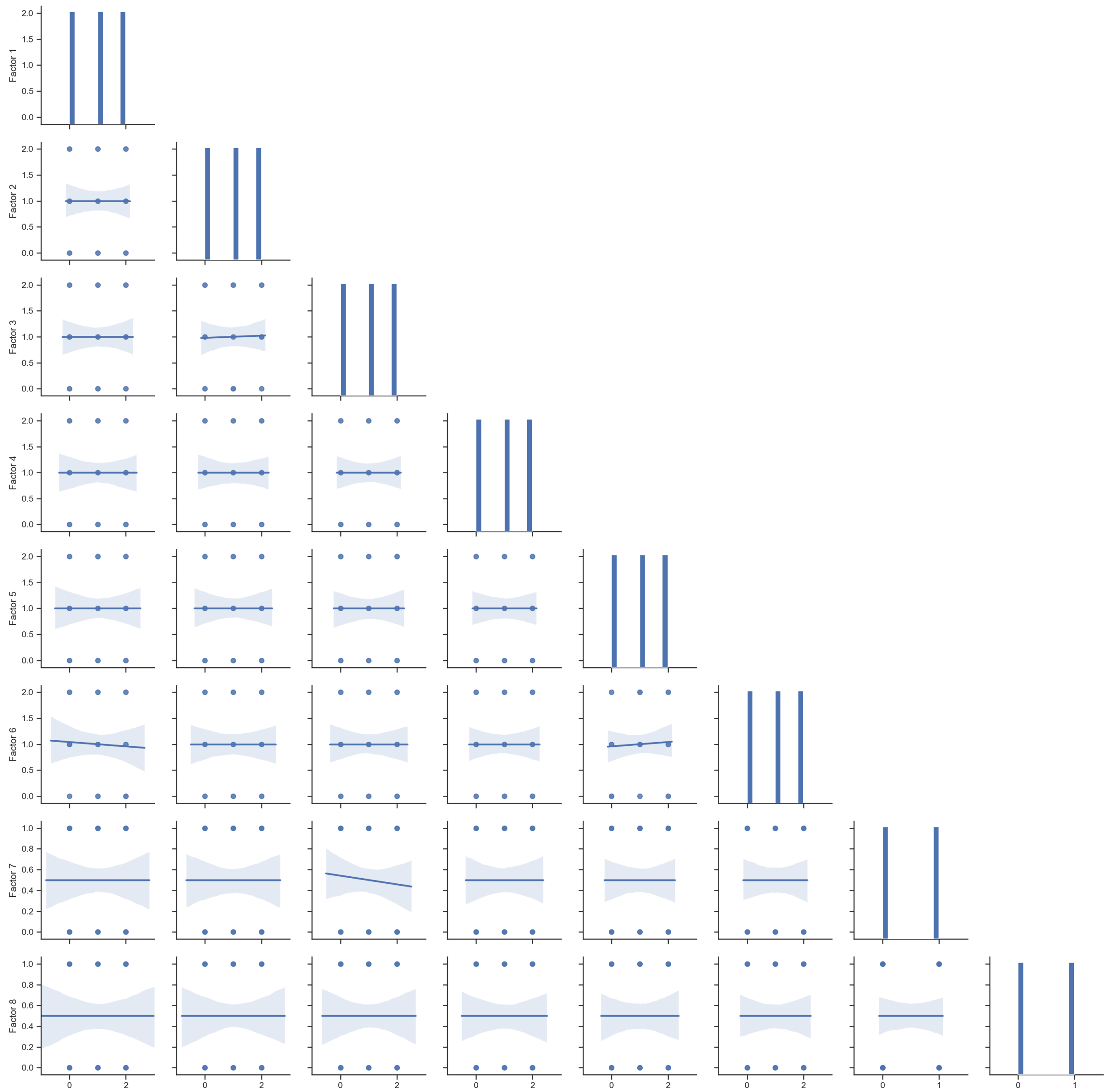

Figure 4: Pairwise factor correlation plot of an example 8 factor design array with a mix of 3- and 2-level factors. The lines and blue shadows correspond to the linear regression fit and its uncertainty. Two pairs are 100% orthogonal if the linear regression line is horizontal. The diagonal bar charts show the histogram of level values for each factor (perfect level balance if histogram is flat).

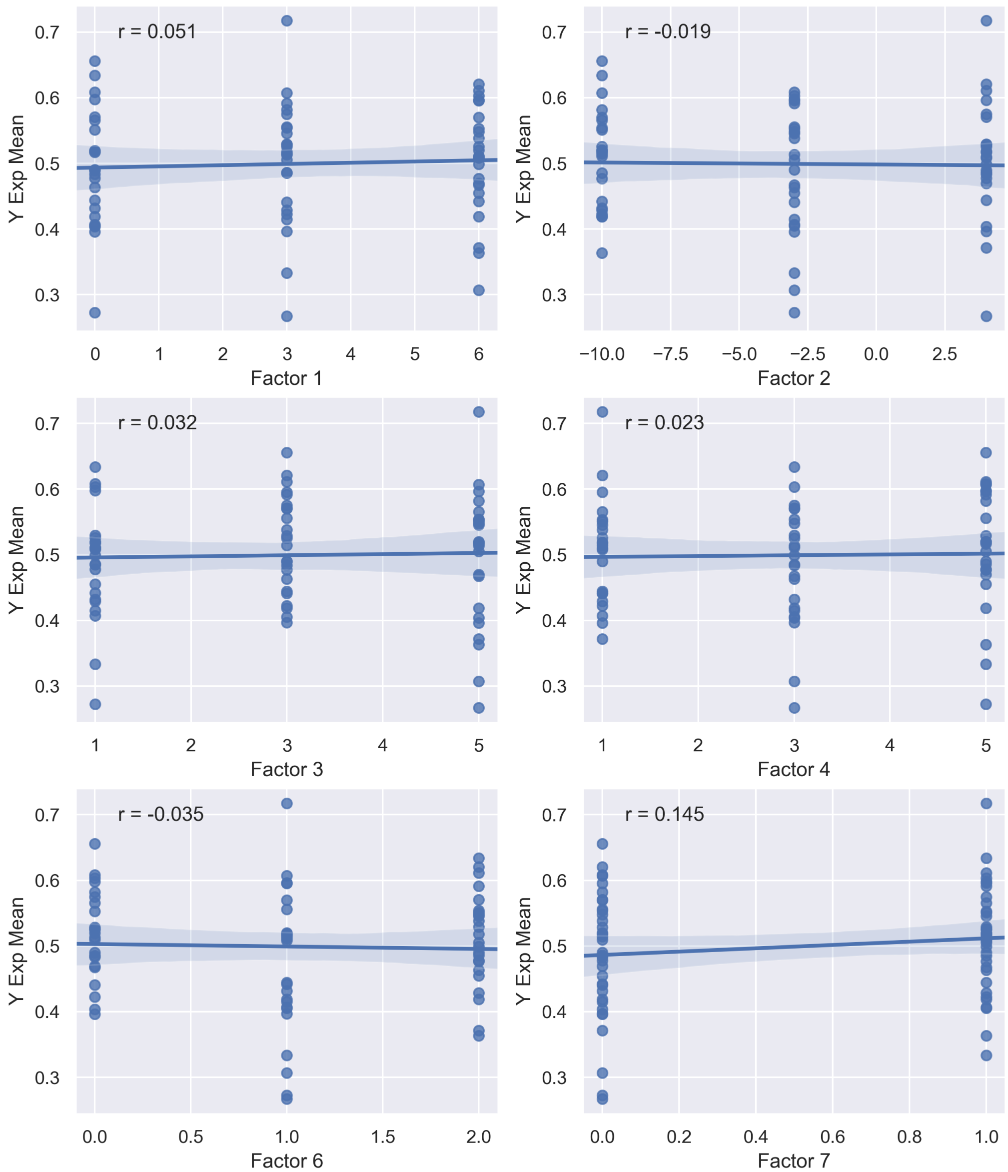

Figure 5: Overview plot of X-Y Correlation for each factor as function of their level values. On top the linear regression coefficient `r` is shown along the linear regression fit and its uncertainty (line and shadow).

Figure 6: Cornerplot of pairwise factor relation with Y. The color(bar) indicates the value of Y.

given by the test example, which consists of 8 factors (parameters) that are specified in the experiment setup table `Experiment_setup_test.xlsx`.

Parameter Name	Parameter Type	Level Number	Minimum	Maximum
Factor 1	Continuous	3	0	6
Factor 2	Continuous	3	-10	4
Factor 3	Discrete	3	1	5
Factor 4	Continuous	3	1	5
Factor 5	Categorical	3		
Factor 6	Discrete	3	0	2
Factor 7	Discrete	2	0	1
Factor 8	Categorical	2		

Figure 7: Test Experiment Setup Table with 6 discrete and 2 categorical factors. Each factor can have a certain number of levels (values), which are here either 3 or 2

The first goal is to generate an efficient design with only a fraction of the entire parameter combination (in our case the full factorial would be $3^6 \times 2^2 = 2916$). The maximum number of experiments (in this case we choose 150) is set in the file `settings_design_test.yaml`, which also specifies input and output directory names, as well as the maximum time for optimising one run (in this case 100 seconds per design optimisation). This configuration will generate and optimize a range of experiments with different design run sizes from 12 to 150, in steps of 6 runsizes (since the lowest common multiple of our mix of 2 and 3 factor levels is 6). Note that the user can also choose a different stepsize, which can done by setting the value in the setting parameter `delta_nrun`. Now we are all setup to start the experiment design generation and optimisation script, which we do by running the script doegen.py with the settings file as argument:

```
$ cd DoEgen
$ python -m doegen.doegen test/settings_design_test.yaml
```

This will generate for each runsize an optimised design array and a list of efficiencies and diagnostic tables and plots (see Design Generation for more details). To simplify the selection of the generated experiment designs, DoEgen suggests automatically three designs: 1) one minimum design (lowest number of runs at given efficiency threshold), 2) one optimal design, and 3) one best design (either equal or has larger experiment run number than optimal design). In our case the three design are selected for run numbers 30 (minimum), 72 (optimal), 90 (best). Since the optimal design has basically almost the same efficiencies as the best design (see figure below) but at a lower cost of experiment runs, we choose for our experiment the optimal design, which is given in the table `Designtable_optimal_Nrun72.csv`.

Now it is time to run the experiment. In our example, we produce just some random data for the 72 experiments with 10 sensor locations (PID 1 to 10) and one output variable Y (e.g. temperature). To analyse the experiment, the results have to written in a structured table with the format as given in `experiment_results_Nrun72.xlsx` (see description in figure below).

To run the experiment analysis script, settings such as for input output directory names are given in the settings file `settings_expresults_test.yaml`, and we can now run the analysis script with

```
$ python -m doegen.doeval test/settings_expresults_test.yaml
```

This analysis produces a range of diagnostic tables and result plots for each output variable Y (in our case we have only one Y). One of the question of this example use case is to identify what factors are important, which is given in the figure `Ybarplot.png`. The "importance" basically indicates how much a factor changes Y (defined by the maximum average change in Y between any levels). This has the advantage to identify also important factors that have either a low linear regression coefficients with Y

```
RESULTS OVERVIEW:
--------------------------------
Minimum Exp Design Runsize: 30
Optimal Exp Design Runsize: 72
Best Exp Design Runsize:    90
--------------------------------


-------------------------------------------------------------
Efficiency           | Min Design | Opt Design | Best Design |
-------------------------------------------------------------
Center Balance       |   97.207   |   100.000   |   100.000  |
Level Balance        |   96.667   |   100.000   |   100.000  |
Orthogonality        |   94.227   |   99.148    |   99.378   |
Two-Way Interact Bal|   90.556   |   99.206    |   99.048   |
D Efficieny          |   0.000    |   0.000     |   0.000    |
D1 Efficieny         |   75.262   |   76.242    |   76.261   |
-------------------------------------------------------------
```

Figure 8: Result Overview of Experiment Design Generation and the three suggested choices. The most important criteria for a good design are orthogonality (100% means that all factor pairs are 100% orthogonal to each other), level/center balance (100% is best) and two-way interaction balance (100% is best). We also want to make sure that at each pairwise interaction occurs at least one (100% Two-Level Min Efficiency). D-efficiency maximises the determinant of the information matrix $|X^T X|$, which corresponds to minimizing the generalized variance of the parameter estimates for a pre-specified model X. Here, D1-efficiency defines the model with only the main effects, while D-efficiency includes also all quadratic terms in the model X. Typically D1-efficiency should be larger than 60%, while D-efficiency only increases if number of experiments is much larger than the number of model terms. In this case study we consider only D1-efficiency given that we want to minimize the number of experiments.

Designtable_optimal_Nrun72

Nexp	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8
1	0.0	-10.0	1	1.0	L1	0	0	L1
2	0.0	-3.0	3	3.0	L1	1	1	L2
3	3.0	-10.0	5	5.0	L2	1	0	L1
4	6.0	4.0	1	1.0	L3	0	0	L2
5	0.0	4.0	5	3.0	L2	0	0	L1

Figure 9: Header with first 5 rows of the optimal design with 72 experiments

Nexp	PID	Y Label	Y Exp	Y Truth	Std Y Exp	Std Y Truth	Weight PID
1	1	1	0.979961	0.874907			
1	2	1	0.140151	0.66966			
1	3	1	0.585378	0.466926			
1	4	1	0.707357	0.671836			
1	5	1	0.110776	0.058323			
1	6	1	0.125913	0.739387			
1	7	1	0.05169	0.954659			
1	8	1	0.693776	0.11485			
1	9	1	0.139383	0.518314			
1	10	1	0.516987	0.307063			
2	1	1	0.653136	0.906968			
2	2	1	0.856592	0.485269			
2	3	1	0.968248	0.580183			

Figure 10: Header with first rows of the experiment result table for 72 experiments. Note that the `Nexp` number has to match the experiment design table `Nexp`. Each experiment (label `Nexp`) can have multiple locations or points (identifier# `PID`), e.g., if experiment is run at different locations simultaneously. In addition, it is possible that one has multiple output Y-variates, labeled with identifier `Y :abel` (target property that has to be predicted or evaluated, e.g. Rain and Temperature). The column `Y Exp` holds the experiment result for Y while the column `Y Truth`holds the ground truth value, which is required to calculate the RMSE and to select best parameter space.

(see r values in plot `Expresult_correlation_X.png`) or are categorical. Such insight can be valuable to determine, e.g., which factors should be investigated in more detail in a subsequent experiment or to estiamate which factors have no effect on Y.

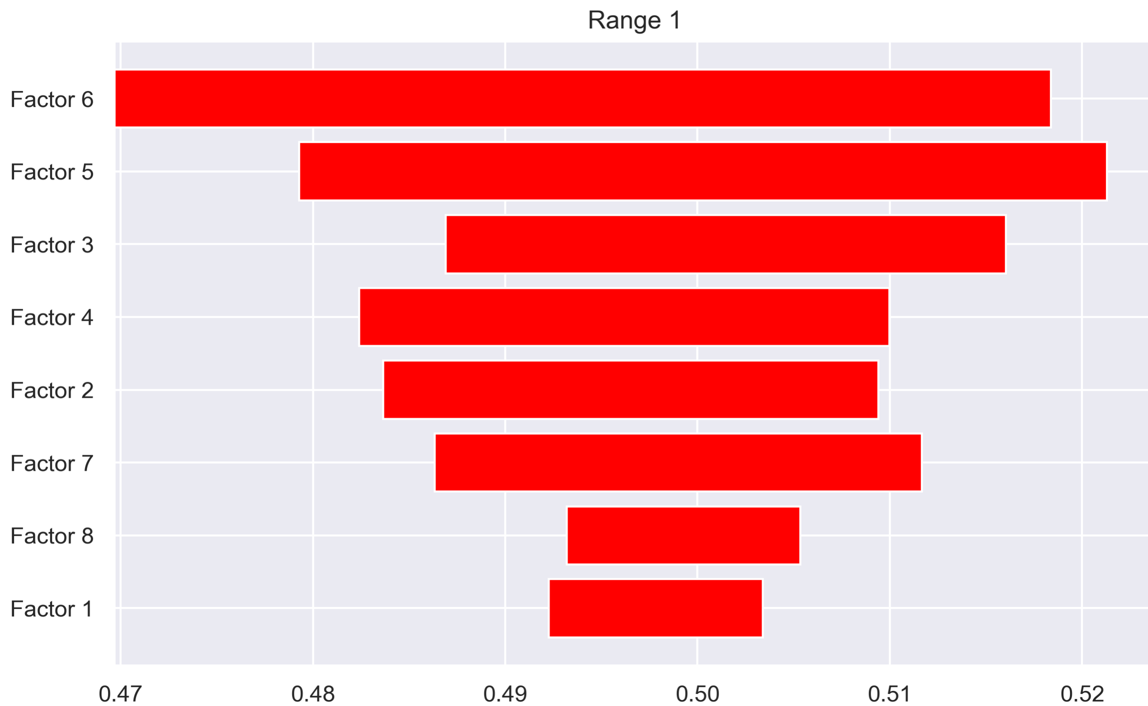

Figure 11: Factor Importance ranked from maximum to lowest change (range) in Y

Another important question is what are the best parameter values based on the obtained experiment results so far? This question can be answered by computing the Root-Mean-Square-Error between experiment results and ground truth (or alternatively the likelihood if the model predictions include also uncertainties). Table `Experiment_1_RMSE_Top10_sorted.csv` provides an overview of the top 10 experiments sorted as function of their RMSE. Moroever we can calculate the (RMSE-weighted) average of each factor for the top experiments as shown in bar plot below.

Furthermore, multiple other diagnostics plots such as factor-Y correlation and pairwise correlation maps are generated (see Experiment Result Analysis for more details).

Comparison to Other DoE Tools

The aim of DoEgen is to provide an open-source tool for researchers to create optimised designs and a framework for transparent evaluation of experiment designs. Moreover, DoEgen aims to assist the result analysis that may allow the researcher a subsequent factor selection, parameter fine-tuning, or model building. The design generation function of DoEgen is build upon the excellent package `OApackage` and extends it further in terms of design efficiency evaluation, filtering, automation, and experiment analysis. There are multiple other tools available for DoE; the table below provides a brief (preliminary) summary of the main advantages and disadvantages for each tool that has been tested.

Feature	SAS JMP	pyDOE2	OApackage	DoEgen
Open-Source	no (paid)	yes	yes	yes

Feature	SAS JMP	pyDOE2	OApackage	DoEgen
Design Optimisation Score	very good	limited	good	good
Optimal Runsize Finder	no	no	no	yes
Design Efficiency Eval	yes	no	limited	yes
Exp Result Analysis	yes	no	no	yes
Development Stage	advanced	early	moderate	very early

Literature

OApackage: A Python package for generation and analysis of orthogonal arrays, optimal designs and conference designs, P.T. Eendebak, A.R. Vazquez, Journal of Open Source Software, 2019

pyDOE2: An experimental design package for python

Dean, A., Morris, M., Stufken, J. and Bingham, D. eds., 2015. Handbook of design and analysis of experiments (Vol. 7). CRC Press.

Goos, P. and Jones, B., 2011. Optimal design of experiments: a case study approach. John Wiley & Sons.

Kuhfeld, W.F., 2010. Discrete choice. SAS Technical Papers, 2010, pp.285-663.

Zwerina, K., Huber, J. and Kuhfeld, W.F., 1996. A general method for constructing efficient choice designs. Durham, NC: Fuqua School of Business, Duke University.

Cheong, Y.P. and Gupta, R., 2005. Experimental design and analysis methods for assessing volumetric uncertainties. SPE Journal, 10(03), pp.324-335.

JMP, A. and Proust, M., 2010. Design of experiments guide. Cary, NC: SAS Institute Inc.

Attribution and Acknowledgments

Acknowledgments are an important way for us to demonstrate the value we bring to your research. Your research outcomes are vital for ongoing funding of the Sydney Informatics Hub.

If you make use of this code for your research project, please include the following acknowledgment:

"This research was supported by the Sydney Informatics Hub, a Core Research Facility of the University of Sydney."

Contributors

We would like to thank Dietmar Muller (School of Geophyics, University of Sydney) for suggesting the need for this library, Danial Azam (School of Geophyics, University of Sydney) for testing DOEgen on real-world cases, Christopher Howden (SIH, University of Sydney) for statistical consultancy, literature suggestions, and documentation review, and Joel Nothman for the code review.

DoEgen has benefited from the OApackage library OApackage for the design optimisation code and we would like to thank the researchers who have made their code available as open-source.

License

Experiment_1_RMSE_Top10_sorted

	Nexp	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8	Y Exp Mean	Y Exp Std	RMSE
45	46	3.0	-10.0	3	1.0	L3	0	1	L1	0.4226	0.3121	0.2540
27	28	3.0	4.0	3	3.0	L2	1	0	L2	0.3967	0.3688	0.2583
24	25	6.0	-3.0	1	5.0	L3	2	0	L2	0.4550	0.3097	0.2814
44	45	6.0	-3.0	5	3.0	L3	2	0	L2	0.5479	0.3025	0.3003
43	44	0.0	4.0	3	3.0	L1	2	0	L2	0.5707	0.2950	0.3019
49	50	3.0	-10.0	1	1.0	L1	2	1	L2	0.4288	0.3383	0.3104
36	37	0.0	-10.0	5	1.0	L3	0	1	L1	0.5656	0.2906	0.3215
25	26	3.0	-10.0	5	5.0	L3	0	0	L2	0.5819	0.3253	0.3326
16	17	6.0	-3.0	3	1.0	L1	1	0	L1	0.5956	0.2460	0.3328
23	24	3.0	4.0	1	3.0	L1	2	1	L1	0.5293	0.2846	0.3349

Figure 12: Picture of Table `Experiment_1_RMSE_Top10_sorted.csv` which shows the factor values of the top 10 experiments based on their RSME values.

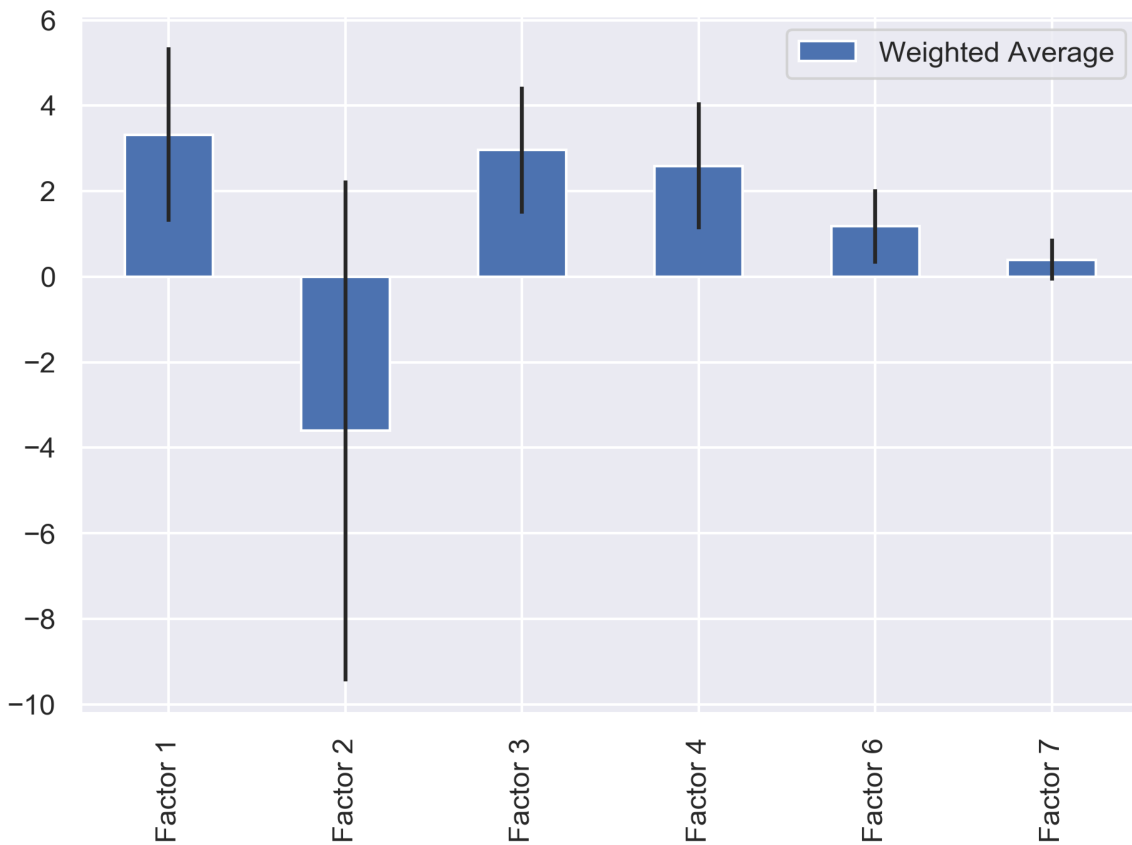

Figure 13: Factor values of the top 10 experiments based on their RSME values. The bar heights indicate the top factor's average value and the dark lines their standard deviation. Note that the average and their standard deviation are computed with the weights $RMSE^{-2}$.