Part II – Decision Systems for Engineering Design
Lecture 2: Sampling Plans and Knowledge Discovery

Robin Purshouse
University of Sheffield

Spring Semester 2023/24

1 Lecture overview

This lecture introduces approaches for aiding decision-makers' understanding of the relationships between design variables, parameters, and performance criteria for an engineering design problem. These approaches are known as *sampling plans* since they involving evaluating samples drawn from the design space of the problem (together with samples of the parameter space, in cases where parameters are defined).

In this lecture, we will cover a number of approaches to the design of sampling plans:

- Random sampling;
- Space-filling designs;
- Other approaches to sampling.

We conclude the lecture with an introduction to knowledge discovery approaches that can be incorporated into a decision system to help a decision-maker learn about the relationships in their problem.

2 Introduction

Consider a problem where design $\mathbf{x}$ from design space $\mathcal{D}$ is to be evaluated against performance criteria $\mathbf{z}$. For simplicity, we will assume that the design space is continuous and we will assume there are no further input parameters of interest.

As in many engineering design problems, the vector of evaluation functions $\mathbf{f}(\mathbf{x})$ is assumed to be a closed box, where the only insight we can gain into the relationship $\mathbf{x} \rightarrow \mathbf{z}$ is to take samples from $\mathcal{D}$ and evaluate them using $\mathbf{f}(.)$.

We denote the sth sample by $\mathbf{x}^{(s)}$ and our task is now to define a sampling plan to generate the set of samples $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(S)}\}$ where S is the total number of samples. Ideally, we would sample the design space as many times as possible; however, in practice, the total will be constrained by the resources at our disposal. These resources could include:

- Wall clock time (i.e. the actual time available to perform the task);
- Core hours (i.e. the amount of computing resource available);
- Bench hours (i.e. the amount of laboratory resource available for physical experiments);
- License numbers (i.e. how many computer experiments can be run in parallel).

Note that in all the discussion that follows, the design variables have each been *normalised* on the range $[0 \ 1]$, which guards against scaling issues (e.g. when measuring distances in design space). Normalisation is achieved using the upper and lower values defined for each variable:

$$\bar{x}_i^{(s)} = \frac{x_i^{(s)} - x_i^L}{x_i^U - x_i^L} \tag{1}$$

where $\bar{x}_i^{(s)}$ is the normalised value of sample s, and x_i^U and x_i^L represent the upper and lower bound respectively for the ith design variable.

Going forward, we drop the $\bar{x}$ notation for ease of reading; however, we will assume the design space is normalised in all cases.

3 Random sampling

The most straightforward approach to building a plan is to sample from each design variable over its full range according to a uniform probability distribution. An example set of results for such a strategy is shown in Figure 1, for a problem with two design variables x_1 and x_2 for which 10 samples have been generated using a pseudo-random number generator. From looking at the figure, it is clear that the samples are not very representative of the overall design space – with no representation at all of the quadrant of the design space with $x_1 < 0.5$ and $x_2 < 0.5$. If we took these results to a decision-maker then he or she would be lacking any information about the performance of designs in this quadrant—quite an unsatisfactory result!

With just two design variables, it is of course obvious from inspection that the sampling is deficient and a different random plan could be instantiated to search for a better result. However, this issue will not be so easily detected when there are many more design variables in the problem, and performing repeated random sampling to find a better result will also be inefficient.

The under-representation of areas of the design space as a result of random sampling led to the development of better techniques which can provide improved coverage of the overall design space—these techniques are known as *space-filling designs*.

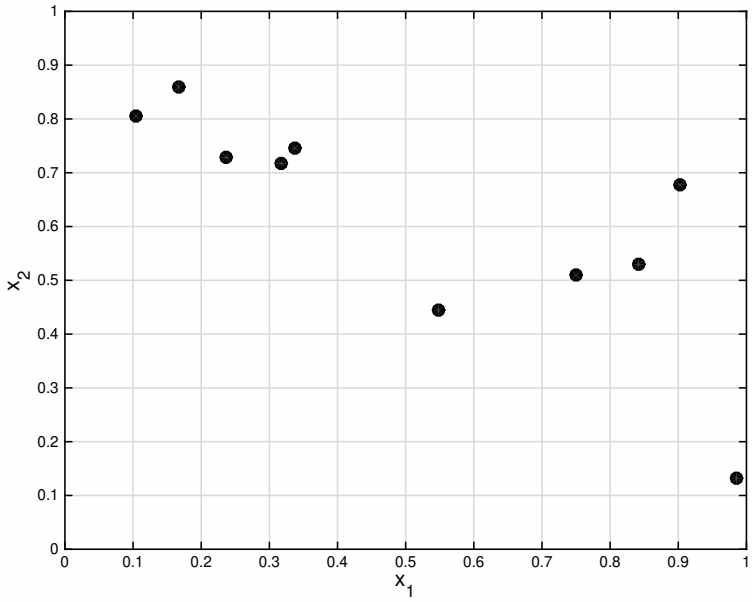


Figure 1: Random sampling of a design space with two design variables and 10 samples.

4 Space-filling designs

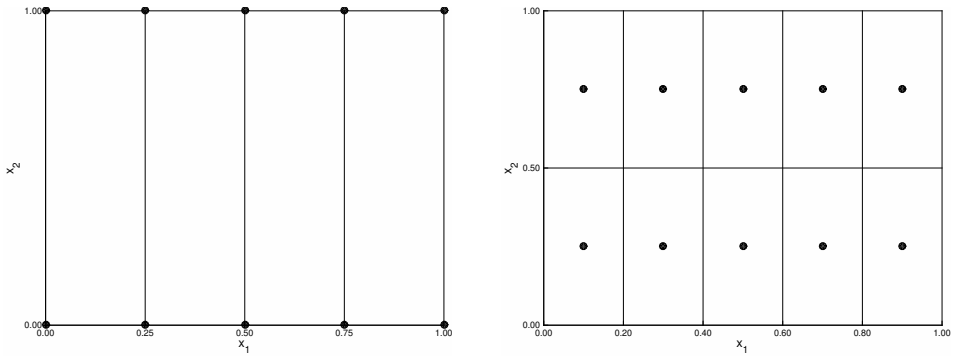
4.1 Full factorial designs

One option for generating a space-filling $\mathbf{X}$ is to partition the design space into a set of hypercubes, and then define a sample at each of these locations—either within each hypercube or on the corners of each hypercube. This approach is known as *full factorial* design.

Consider again a design space with two design variables x_1 and x_2 and 10 samples available. There are only two ways to generate 10 samples from a full factorial design: either by splitting x_1 into 5 and x_2 into 2, or vice versa. In the example shown in Figure 2, we opt for the former partition. Both types of design (corners or centre) are shown in the figure.

Both plans shown in Figure 2 are space-filling. However the situation is not entirely satisfactory. Firstly, we were constrained to a 5-by-2 or 2-by-5 scheme if we wanted 10 samples; secondly the sampling is not optimally informative since we have large numbers of replications of the same values for each design variable.

An immediate option here in the case of the centre-based plan is to randomly perturb each sample within the hypercube in which it is enclosed. The results of such a scheme are shown in Figure 3. This is an improvement in terms of removing the replications of design variable values but, since the hypercubes are so large, we start to encounter some of the issues with coverage of the design space that we suffered



(a) Hypercube corner sampling (b) Hypercube centre sampling

Figure 2: Full factorial plans for a problem with two design variables, with 10 samples available. Note the difference in the size of the hypercubes between the two full factorial approaches.

with the completely random plan.

4.2 Latin hypercube sampling

The popular approach known as *Latin hypercube sampling* (LHS) extends the perturbation approach by ensuring uniform coverage of each design variable (although not necessarily their interactions). In the LHS scheme, if S samples are required then each design variable is partitioned into S regions of equal length. The interactions of these regions across design variables form hypercubes. An LHS plan is then achieved by randomly perturbing the *ordering* of the partition for each variable. Sampling is then performed in hypercubes defined by the randomised ordering.

As an example, consider again the situation where 10 samples are required from a two-dimensional design space. Both design variables are partitioned into regions of equal length, with the partitions denoted by indices: $1, 2, \ldots, 10$. The indices are then randomly shuffled. For example, we might achieve:

- $s^{(x_1)} \sim \{5, 6, 1, 7, 9, 2, 4, 3, 8, 10\}$
- $s^{(x_2)} \sim \{10, 6, 2, 5, 3, 1, 4, 9, 7, 8\}$

We now place a sample in each hypercube defined by the pairs of randomised indices, i.e. $\{5, 10\}$, $\{6, 6\}$, and so on. The result is shown in Figure 4; reasonable space-filling has been achieved.

4.3 Space-filling metrics

Note that not all plans produced by LHS are space-filling. An example of a LHS plan with poor space-filling properties is shown in Figure 5. Whilst the plan achieves good

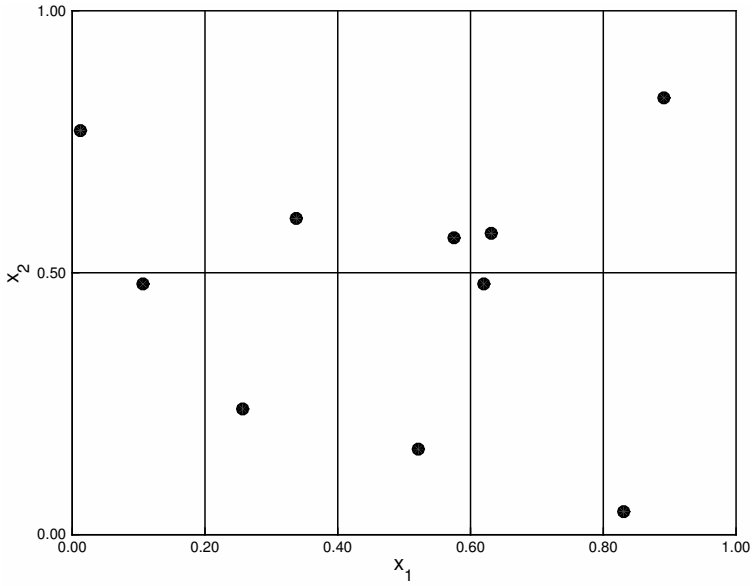


Figure 3: Perturbed full factorial plan for a design space with two design variables and 10 samples available.

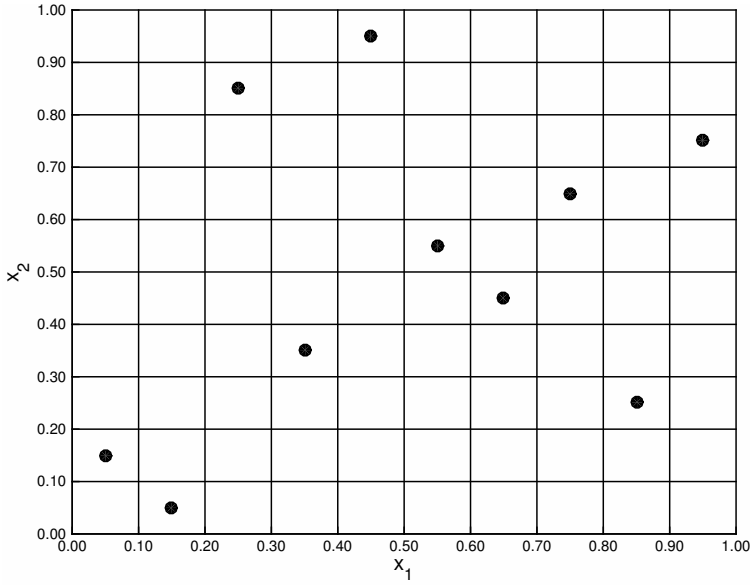


Figure 4: Latin hypercube sampling of a design space with two design variables and 10 samples available.

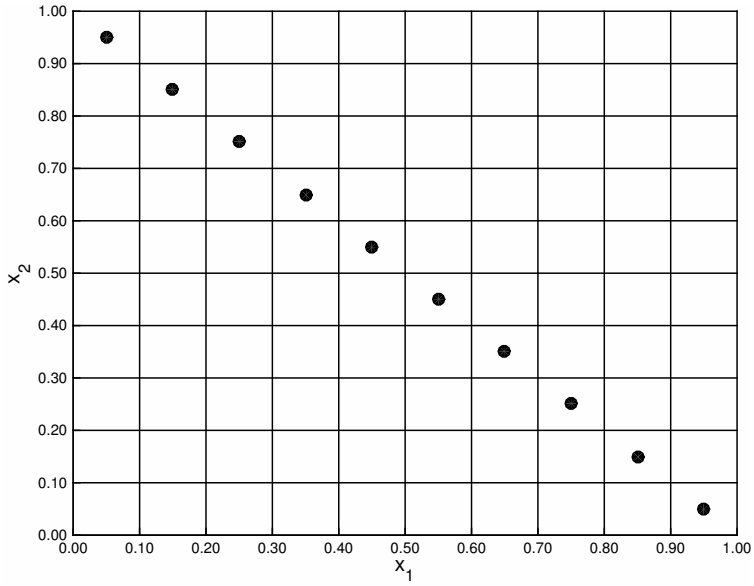


Figure 5: A Latin hypercube sampling plan with poor space-filling properties

coverage of the design variables individually, it leaves large areas of the search space untouched. In two dimensions, the issue with space-filling performance is obvious but, as for the random plan, it may not be so easy to see such problems in a higher dimensional design space.

To help identify the space-filling capabilities of an LHS plan (or indeed any form of sampling plan), quantitative metrics are available. Here, we introduce one such metric known as 'Φ_q', where smaller values indicate better space-filling properties.

In order to compute Φ_q for a sampling plan $\mathbf{X}$ we need to first consider the distances between all the pairs of samples in the plan. 'Distance' can be defined in a number of different ways, but Euclidean distance is a common choice and is used here. Focusing just on *unique* distances d, we rank all these unique values in ascending order: $d_1, d_2, \ldots, d_\mu$, where μ is the total number of unique distances observed in the plan. We also count the pairs of samples associated with each distance: $J_1, J_2, \ldots, J_\mu$. We are now in a position to compute the Φ_q metric using Equation 2:

$$\Phi_q(\mathbf{X}) = \left(\sum_{j=1}^{\mu} J_j / d_j^q \right)^{1/q} \tag{2}$$

where q is a tuneable parameter that dictates how much the smaller distances in the equation will dominate the final value for the metric. Larger values of q will lead to domination by the smaller distances—choices for q typically vary between 1 and 100; in these notes we will choose $q = 5$.

In Table 1, we calculate the value of Φ_q for the two Latin hypercube sampling plans shown in Figure 4 and Figure 5. The value for the random plan is seen to be smaller

Sampling plan, $\mathbf{X}$	Space-filling metric, $\Phi_5(\mathbf{X})$
Random LHS plan	8.53
Main diagonal LHS plan	11.0

Table 1: Space-filling metrics for the two LHS sample plans shown in Figures 4 and 5

than for the diagonal plan, indicating a design with superior space-filling properties as expected. Using Φ_q as a metric, we could devise an optimization procedure to search through the space of plans to find a suitable plan with good space-filling properties.

5 Other approaches to sampling

Latin hypercube sampling is a very popular approach for generating a sampling plan that can help provide insight into the relationships between inputs and outputs in an engineering design problem. LHS is also popular across a wide range of other fields. However, a variety of other approaches are available for generating a sample plan. In this final part of the lecture, we briefly introduce two other well-known alternatives.

5.1 Orthogonal arrays

For some engineering design problems, technical specialists will have some understanding of the important interactions between the different design variables. In this case, the information can be used to focus the sampling plan on capturing these interaction effects—with *orthogonal arrays* (OAs) forming a particularly useful class of method. An orthogonal array of 'strength' p captures interaction effects between up to p variables at a time. This is because all the different combinations of levels of the variables included are guaranteed to occur across p columns of the sampling plan the same number of times.

Orthogonal arrays are generally used where the number of levels is quite small (few plans exist for 10-level designs, for example[1]). As an illustration, consider an engineering design problem with three variables, x_1, x_2 and x_3, where each variable is partitioned around the value 0.5 (after normalisation). The 2-strength orthogonal array that captures the interaction effects between pairs of these variables is shown in Table 2. Note how when we select any two variables (i.e. columns in the sampling plan), the four combinations of levels for those variables – $\{< 0.5, < 0.5\}$, $\{< 0.5, > 0.5\}$, $\{> 0.5, < 0.5\}$, $\{> 0.5, > 0.5\}$—all appear the same number of times (once), enabling us to see how these two variables work together.

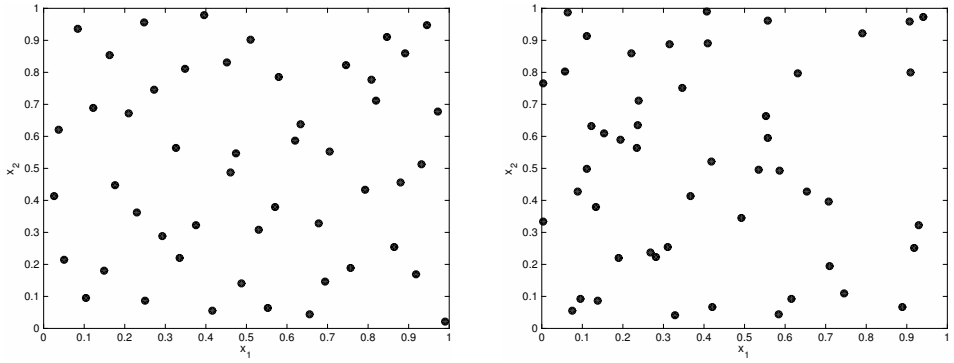
5.2 Sobol sequences

A weakness of the LHS approach is that, when a sampling plan has been determined for S number of samples (e.g via analysis of competing sampling plans using metrics

[1] See `http://neilsloane.com/oadir` for a useful OA library

x_1	x_2	x_3
< 0.5	< 0.5	< 0.5
< 0.5	> 0.5	> 0.5
> 0.5	< 0.5	> 0.5
> 0.5	> 0.5	< 0.5

Table 2: Orthogonal array-based sampling plan of strength 2 for three design variables explored over 2 levels



(a) Sobol sequence (b) Mersenne Twister sequence

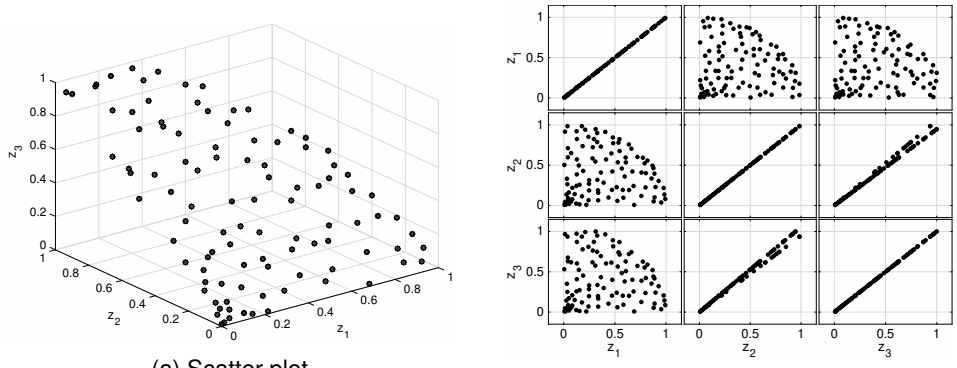
Figure 6: Sampling plans for two design variables with 50 samples using quasi-random (Sobol) and pseudo-random (Mersenne Twister) generators.

such as Φ_q), the space-filling properties of that plan cannot be assumed to hold if less than S samples end up being evaluated (e.g. because there are less resources available than initially believed). An alternative approach in this situation is to use *Sobol sequences*. Sobol sequences are quasi-random number generators that have good space-filling properties in comparison to pseudo-random number generators. These properties are also invariant with respect to S.

A comparison between a sampling plan for a Sobol sequence in comparison to a uniform random number generator is shown in Figure 6. Note the more even coverage of the Sobol scheme.

6 Knowledge discovery

In the preceding sections we have looked at how to generate a sampling plan $\mathbf{X}$ that can be evaluated to provide a set of sampled performance criteria $\mathbf{Z}$. Assuming for the time-being that no further samples will be taken, the task now is to understand the relationships within the problem that have been exposed by the sample pairs $\{\mathbf{X}, \mathbf{Z}\}$ and communicate these to the decision-maker. Essentially this is a **knowledge discovery** problem.



(a) Scatter plot

(b) Matrix scatter plot

Figure 7: Scatter plots for an example performance criteria set $\mathbf{Z}$ with three criteria.

6.1 Visualisation

A variety of generic multivariate data visualisation approaches can be used to help visualise the relationships in the sample set. The most popular of these are *scatter plots* and *parallel coordinates* plots.

6.1.1 Scatter plots

A scatter plot displays data as points in a two-dimensional Cartesian space. It is a very effective visualisation method for data that is naturally two-dimensional, but has poor scalability as the dimensionality of the data rises. Scatter plots can also be plotted in three-dimensions, although readability typically depends on the angle of orientation of the plot. Scatter plots can be combined with glyphing (see below) to increase the number of dimensions that can be represented.

Example scatter plots displaying a three-dimensional performance space are shown in Figure 7.

6.1.2 Parallel coordinates

In the parallel coordinates visualisation method, each dimension is represented by a parallel line, with spacing between each line. A particular data point is then drawn as a polyline, where the vertices of the polyline are the values for the data point corresponding to each dimension. Parallel coordinates scale better than scatter plots, but require a judicious choice of the ordering of the axes (since a dimension can only be adjacent to at most two other dimensions).

An example parallel coordinates plot is shown in Figure 8, using the same data as for Figure 7.

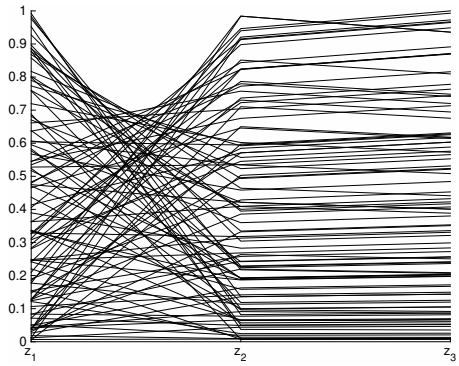


Figure 8: Parellel coordinates plot of the same performance criterion data as Figure 7

6.1.3 Glyphs

In the glyphing approach, each dimension is associated with a particular *feature* in the visualisation—for example, colour or shape of a data point symbol. An example is shown in Figure 9 for the design of a water distribution system, in which a three-dimensional scatter plot of three performance criteria (leakage, system failure index and capital cost) is enhanced with information about three further objectives: water age is indicated by the colour of the points; fire flow deficit is indicated by the orientation of the data points (which are pointed to enable orientation to be observed), and operating cost is represented by the size of the data point.

The most famous, or perhaps infamous, example of a glyphing approach is *Chernoff faces*; in this approach, each dimension is assigned to one of the features of a face (e.g. distance between eyes, radius of eyes, slant of eyebrows). A data point is then represented as a face. An example is shown in Figure 10. There are numerous objections to this technique, such as psychological biases associated with particular facial characteristics and the cognitive ability required to identify relationships across a series of faces. However it is quite an interesting method, even if one perhaps not quite appropriate for engineering decision-making!

6.2 Dimensionality reduction

Rather than simply attempting to visualise the 'raw' data generated by the sampling plan, it may be possible to use data mining techniques to identify lower-dimensional structure in the data that is relevant to decision-making. There are a large number of methods that can be used, including:

- **Principal components analysis:** finds an orthogonal transformation of the data, where each transformed dimension in turn contains as much variance from the original data as possible. If the variance is mostly contained in the first few transformed dimensions, then scatter plots may be used to visualise the relationships. Principal components analysis was introduced in Lecture 5 of Part I of the module.

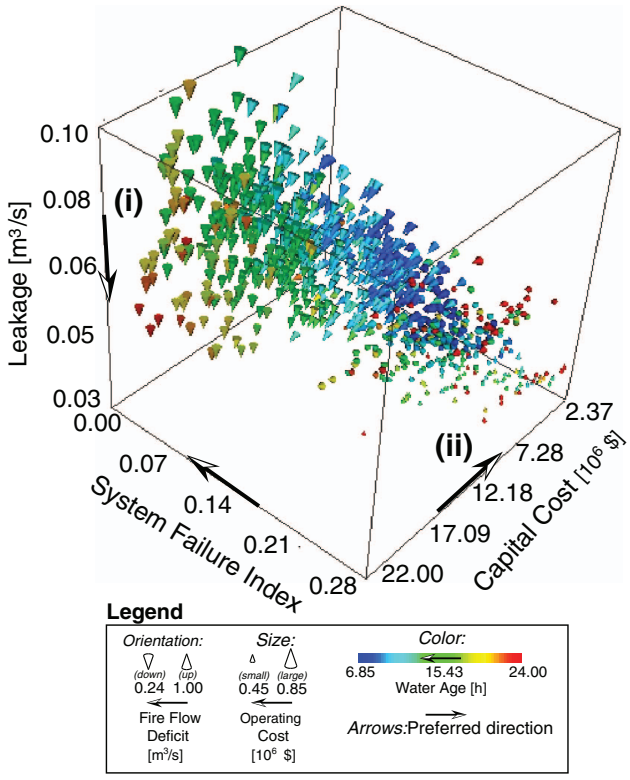


Figure 9: Scatter plot with glyphing. Taken from Fu et al.'s *Optimal Design of Water Distribution Systems Using Many-Objective Visual Analytics*, 2013

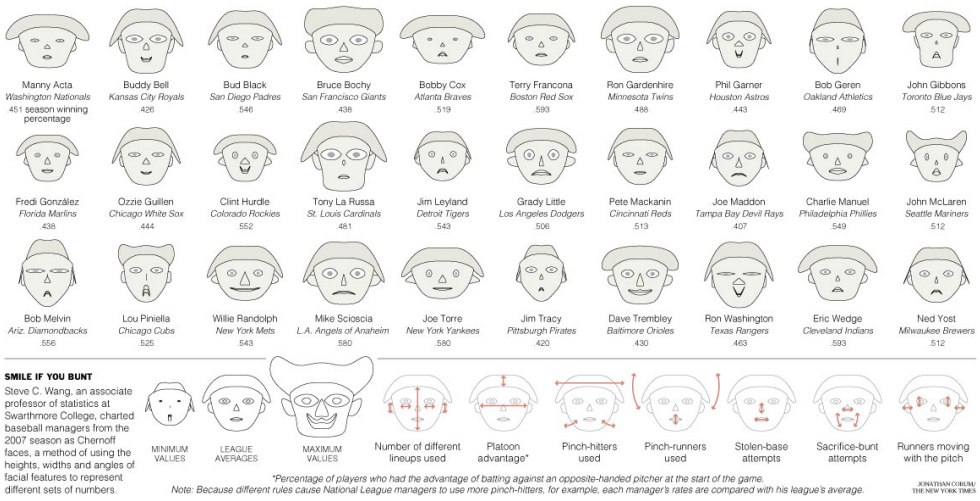


Figure 10: Chernoff faces representation of tactics used by US baseball team managers. Taken from Correll's *Ross-Chernoff Glyphs, Or: How Do We Kill Bad Ideas in Visualization?*, 2018

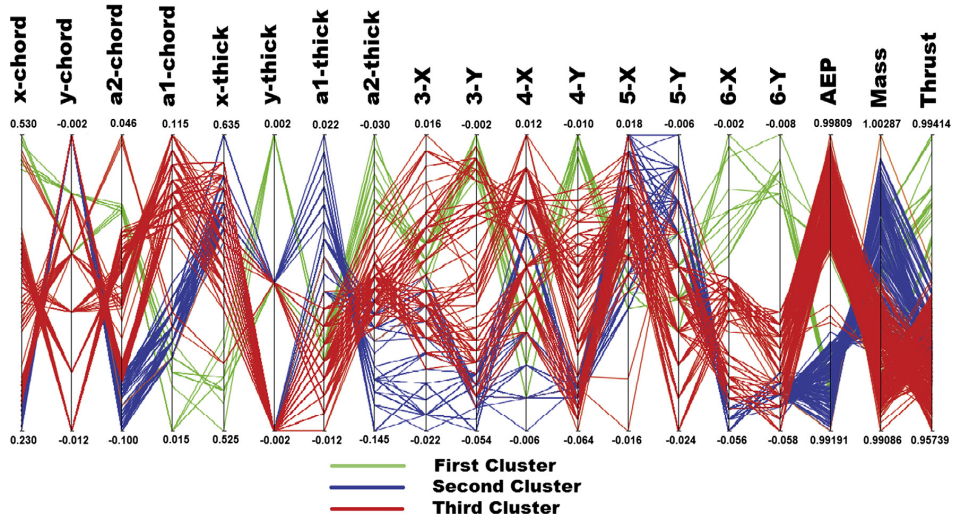


Figure 11: Parallel coordinates plot with clustering. Taken from Fischer et al.'s *Multi-objective optimisation of horizontal axis wind turbine structure and energy production using aerofoil and blade properties as design variables*, 2014

- **Clustering:** finds groups of data points that are similar, according to a given similarity metric (e.g. Euclidean distance in normalised design space). A variety of clustering approaches exist, which typically work by starting with all data points in one cluster and progressively breaking the cluster up, or by starting with all data points as their own clusters and then progressively merging the clusters together. In decision systems applications, clustering can be very effective for identifying groups of promising designs that have similar attributes in the design space, or represent similar trade-offs in the performance space. An example is shown in Figure 11, where a clustering algorithm was used to identify geometrically similar designs that were then grouped using colours on a parallel coordinates plot covering both the design space and performance space.

6.3 Further reading

- Bandaru S., Ng A.H.C., Deb K., Data mining methods for knowledge discovery in multi-objective optimization: Part A - Survey. *Expert Systems with Applications* 2017;70:139–159.
- Fischer G.R., Kipouros T., Savill A.M. Multi-objective optimisation of horizontal axis wind turbine structure and energy production using aerofoil and blade properties as design variables. *Renewable Energy* 2014;62:506–515.
- Fu G., Kapelan Z., Kasprzyk J.R., Reed P., Optimal design of water distribution systems using many-objective visual analytics. *Journal of Water Resources Planning and Management* 2013;139:624–633.
- Pronzato L., Müller W.G. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 2012;22:681–701.