Designs on the Web

Peter J Cameron

School of Mathematical Sciences
Queen Mary, University of London
London E1 4NS, U.K.
p.j.cameron@qmul.ac.uk

Joint work with R. A. Bailey, P. Dobcsányi,
J. P. Morgan and L. H. Soicher

http://www.designtheory.org

The DTRS project

We are in the process of developing a web-based Design Theory Resource Server (DTRS) for combinatorial and statistical design theory. One critical element is our *External Representation of Designs* which will be used to store designs and their properties in a standard platform-independent manner (external means external to any software). This will allow for the straightforward exchange of designs between various computer systems, including databases and web servers, and combinatorial, group theoretical and statistical packages. The external representation will also be used for outside submissions to our design database.

A beta-release of the *External Representation* with documentation is now available.

What is a design?

We have a set T of treatments which we wish to compare, and a set Ω of plots or experimental units to which the treatments may be applied. A *design* is a function $F : \Omega \to T$ (so that $F(\omega)$ is the treatment applied to plot ω).

If T and Ω are completely unstructured, simply use each treatment the same number of times (as near as possible).

The existence of structure on T and Ω complicates matters!

Treatment and plot structures

For example, one treatment may be a placebo, or the "standard" treatment against which the others are to be compared; or the treatments may have a number of factors which can be applied at different levels (fertiliser dose, watering, planting time, etc.) These are important but for now we disregard them. If T is homogeneous, we can replace the function F by a partition of Ω (and assign one treatment to each part).

Plot structure arises because the set of plots is not homogeneous. For example, we may have several plots on each of a number of farms in different geographical areas; soil fertility or moisture content may vary across a field; one plant may shade another; experiments over a period of time are subject to daily or seasonal effects; and drugs may have carry-over effects.

Block designs

The simplest plot structure is a system of blocks, where plots in a block are more alike than those in different blocks (as in the farms example). Thus, the blocks form a partition of Ω. So a *block design* consists of a set Ω with two partitions, corresponding to treatments and blocks.

The block partition is given, and we have to choose the treatment partition so that information can be recovered about the treatments as efficiently as possible – this means that the variances of the estimators of treatment differences should be as small as possible.

In the worst case, if we applied each treatment on only one farm, we couldn't separate treatment effects from geographical effects.

Which block design?

If the number of treatments is equal to the number of plots in a block, we could ensure that each treatment occurs once in each block. This is a *complete* block design, and is clearly the best we can do. Usually, constraints don't allow this.

Typically, the block size k is smaller than the number v of treatments, so that the block design is *incomplete*. In this case, if there is a block design which is *balanced* (so that any two treatments occur together in a block exactly λ times), then it is the best design to use, with respect to all practical criteria. This why balanced incomplete-block designs (BIBDs, or 2-designs) are so important. But often it is the case that no such design exists, and indeed there may be no design which is uniformly best. In this case the choice of design might depend on other factors.

Representations of block designs

$\mathbf{R}_1$: A set of plots with two partitions (as above).

$\mathbf{R}_2$: Bipartite graph, vertices are points (treatments) and blocks, joined if the treatment appears in the block, labelled with the number of times it appears.

$\mathbf{R}_3$: Set of points, each block is a multiset of points, so the blocks form a multiset of multisets.

$\mathbf{R}_3'$: Dual to $\mathbf{R}_3$.

$\mathbf{R}_4$: Bipartite graph with vertices and edges labelled. Edge labels as above; vertex labels denote the number of repetitions of blocks and/or points.

The design is called *binary* if no treatment occurs more than once in a block (so edge multiplicities are all 1).

Example

Suppose we have six plots, $1, 2, 3, 4, 5, 6$, which fall into two blocks $\alpha = \{1, 2, 3, 4\}$ and $\beta = \{5, 6\}$. We have two treatments A and B, and apply A to plots $1, 3, 5$ and B to plots $2, 4, 6$. These two partitions form representation $\mathbf{R}_1$. It has 8 automorphisms.

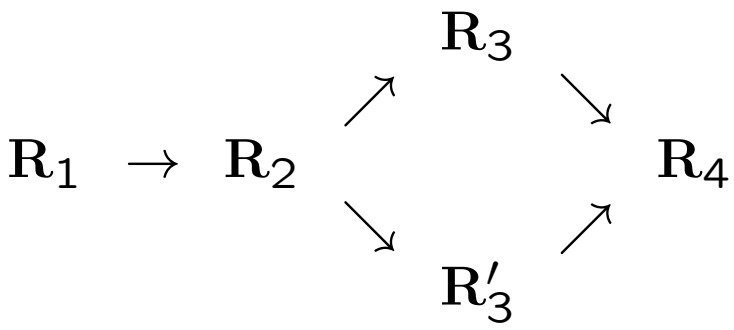
$\mathbf{R}_2$: Complete bipartite graph on vertices $\{A, B\}$ and $\{\alpha, \beta\}$, with multiplicies 2 on $A\alpha$ and $A\beta$ and 1 on the other two edges. This has 2 automorphisms.

$\mathbf{R}_3$: Point set $\{A, B\}$, blocks $[A, A, B, B]$ and $[A, B]$. Again 2 automorphisms.

$\mathbf{R}_3'$: Point set $\{\alpha, \beta\}$; block $\{\alpha, \alpha, \beta\}$ with multiplicity 2. Trivial automorphism group.

$\mathbf{R}_4$: Complete bipartite graph with vertex α labelled 2, and A and B labelled 1; edges $A\alpha$ labelled 2, $B\alpha$ labelled 1. Trivial group.

The structure



The arrows correspond to homomorphisms between the automorphism groups. The first homomorphism measures the non-binarity of the design; the others measure repeated blocks or points.

For the DTRS project, at present we consider only binary designs. We have chosen to use representation $\mathbf{R}_3$, which is almost universal among mathematicians. Thus, a block design for us consists of a set of points, and a list of blocks, each block a subset of the point set; but repeated blocks are allowed.

External Representation of designs

The External Representation (which is not yet complete) is in XML format. We believe that this is a standardised and well-established format which should be increasingly recognised in the foreseeable future.

A design in XML format is human-readable to a limited extent, but is primarily intended for exchange between programs, databases, etc. The document has a syntax which can be checked; it also makes some mathematical assertions about the designs it contains, which can also in principle be checked.

The External Representation document contains a specification of the syntax together with extensive documentation.

An example

This is a list of designs containing a single design, the Fano plane, in XML.

```
<list_of_designs
 design_type="block_design" no_designs="1"
 pairwise_nonisomorphic="true"
 xmlns="http://designtheory.org/xml/ns">
    <block_design b="7" id="ls-t2v7k3lambda1-1" v="7">
        <blocks ordered="true">
            <block><n>0</n><n>1</n><n>2</n></block>
            <block><n>0</n><n>3</n><n>4</n></block>
            <block><n>0</n><n>5</n><n>6</n></block>
            <block><n>1</n><n>3</n><n>5</n></block>
            <block><n>1</n><n>4</n><n>6</n></block>
            <block><n>2</n><n>3</n><n>6</n></block>
            <block><n>2</n><n>4</n><n>5</n></block>
        </blocks>
    </block_design>
</list_of_designs>
```

At the first level of indentation, between the opening and closing tags `block_design`, we have indeed specified the design: it has $v = 7$ points, $b = 7$ blocks, and the seven blocks are listed (the first one is $[0, 1, 2]$, and there are tags to identify each of 0, 1, 2 as natural numbers). There is also a somewhat meaningless identification string for the design.

Specifying a block design

The specification of a block design is shown below. Properties followed by a question mark are optional. Remember that all designs are assumed to be *binary*, so that a block is a set of points (rather than a multiset). This is an example of RNC, a compact and friendly way to specify the syntax of an XML document. The previous example included only the required fields.

```
block_design = element block_design {
    attribute id { xsd:ID } ,
    attribute v  { xsd:positiveInteger } ,
    attribute b  { xsd:positiveInteger } ,
    blocks ,
    point_labels ? ,
    indicators ? ,
    combinatorial_properties ? ,
    automorphism_group ? ,
    resolutions ? ,
    partial_balance_properties ? ,
    statistical_properties ? ,
    info ?
}
```

Indicators

Indicators include `repeated_blocks`, `resolvable`, `equireplicate`, `constant_blocksize`, `t_design`, `connected`, `variance_balanced`, `efficiency_balanced`, `partially_balanced`, `cyclic`, `one_rotational`.

They take the value "true", "false", "unknown" or sometimes "not applicable".

There may be also some extra information; e.g. for `t_design`, the indicator gives the maximum value of t.

Other tags

Combinatorial properties include a range of t-design properties likely to be of interest to mathematicians. These include Steiner system, square design, etc. We also include information about α-resolvability, t-wise balance, etc.

Under *automorphism group* we store generators for the automorphism group, various properties (transitivity, primitivity, multiple transitivity, stratifiability, etc.), and the cycle types of elements together with a representative element for each cycle type.

If a design is *resolvable*, we can in principle store representatives of the isomorphism classes of resolutions (two resolutions being isomorphic if one is obtained from the other by applying an automorphism of the design).

Other tags, continued

Among the *statistical properties*, we include pairwise and canonical variances, how the design performs on a number of optimailty criteria, the efficiency factors of the design, and robustness properties. (Statistical robustness of a block design is defined as its ability to maintain desirable statistical properties under loss of individual plots or entire blocks. A catastrophic result of such a loss is that the design become disconnected. Less catastrophic are losses in the information provided by the design, as measured by various optimality criteria.)

If a design is *partially balanced*, then there is a unique coarsest association scheme with respect to which this is the case, called the *balancer* of the design; we will store this association scheme and some of its properties and refinements.

Lists of designs

The outermost tag is `list_of_designs`. Here we can store such information as "these are, up to isomorphism, all designs with such-and-such properties" (or maybe "these designs are pairwise non-isomorphic but we don't know if the list is complete"), or "these designs were constructed by X using software Y". Other comments can also be added. Each design in the list is described by the earlier specification.

Any document which is exchanged by systems following this specification should be a `list_of_designs`.

Where next?

We would like your feedback on what we have done. Have we omitted something crucial?

We intend to extend this specification, first to handle non-binary block designs, and then to handle more general types of designs (row-column designs, factorial designs, semi-Latin squares, etc.)

We are also producing software to make it easy to read and write correctly specified designs, and to interface with combinatorial and statistical software (in the first instance GAP and R).

www.designtheory.org

One of the main features of the website will be a database of designs, which can be searched according to the criteria in the specification and input or output in XML format.

Leonard Soicher is producing a GAP share package for computing with designs; it will have the ability to output designs matching our specification. This software has already been used in a couple of research projects, including finding optimal regular graph designs.

Take a look at the *Encyclopaedia of Design Theory* in the library (and please feel free to contribute to it!).

Finally, please sign up to the mailing lists, and get involved in the discussion about how the project should proceed.