Experimental Designs for Taste Testing

R.D. Ball
HortResearch, P.B. 92169
Auckland
New Zealand
r.ball@hort.cri.nz

Abstract

Two important considerations for experimental designs for taste testing are order and carry-over effects. Ideally each sample would be presented equally often in each order and would be preceded equally often by every other sample. Williams (1949) gives a number of designs which achieve this but only for complete block designs. We describe methods for incomplete block designs which are approximately balanced for carry-over effects. Using the cyclic structure of cyclic designs we obtain designs balanced for order effects and nearly balanced for carry-over effects. For non-cyclic designs a computer intensive discrete optimisation is necessary.

1 Experimental design for taste testing

In sensory analysis a number of taste panelists are chosen to taste a number of different samples (i.e. products or treatments). Panelists may be asked to rate each sample according to various attributes such as sweetness or acidity and/or overall acceptability. Two commonly used methods of presentation are: monadic: where samples are presented one at a time and all questions answered before presenting the next sample and multiple: where all samples are presented together and may be retasted for each question. Multiple presentation may give more immediacy of product comparisons. For multiple presentation the design requirements are slightly different although the methods can be adapted. Henceforth we consider only monadic presentation.

Additionally it is necessary to keep the number of samples tasted by each panelist in a session quite small to avoid problems with fatigue and burning (e.g. kiwifruit, pineapple) or intoxication (e.g. alcoholic drinks).

The maximum possible information on within panelist treatment comparisons is desired. This is obtained by choosing an efficient incomplete block design, which is often possible using cyclic designs. If the treatments are tasted in different orders the results could be affected by the order of presentation (order effect) and by the previous treatment or treatments (carry-over effect).
the results could be affected by the order of presentation (order effect) and by the previous treatment or treatments (carry-over effect).

Order and carry-over effects have been found to be important, though which is most important varies with product and attribute e.g. [16].

Ideally one would like the treatment effects to be orthogonal to order effects i.e. every treatment would be tasted equally often in every order. Immediately preceding a treatment is either one of the r treatments or no treatment for the first treatment in a block. Ideally one would also like the treatment effects to be orthogonal to carry-over effects i.e. that every treatment be preceded equally often by every other treatment. Designs which achieve this are called carry-over designs, cross-over designs or neighbour balanced.

The standard model is:

\[ y_{ijk} = \mu + b_j + o_k + v_{i(j,k)} + t_{i(j,k-1)} + \epsilon_{ij} \]

where the first order carry-over effect is the expected difference in response (after correcting for order) caused by the previous treatment in the block. Other models are possible, such as those allowing for different use of the scale by different panelists, e.g. Mead [15] allows for an additional multiplicative scaling factor for each panelist, carry-over effect could be a function of the value of other attributes, and responses and continuous covariates could be subjected to general non-linear transformation methods such as ACE [3], AVAS [18] as in [14], or taste responses could be evaluated in prior experiments with controlled levels of compounds of interest and transformations determined parametrically as in [5].

The use of carry-over designs for taste testing are discussed in [6, 12, 17]. These authors use Latin square designs or designs of [19].

2 Incomplete block designs, nearly balanced for order and carry-over effects

Since we require incomplete block experiments we can’t use Latin squares or the designs of Williams[19]. To generalise it is necessary to relax the requirements. Rather than requiring order and first order carry-over effects to be balanced with respect to treatments we only require that they be nearly balanced.

The rationale for this is similar to that of Mead [15] who advocates the use of nearly balanced designs in place of BIBs. Like BIBs, cross-over designs are combinatorial freaks, a design with the desired parameters usually doesn’t exist. With modern computing however BIBs are not really needed. Efficient designs that are nearly balanced are just as good for practical purposes and present no problems for analysis. Similarly, for practical purposes designs nearly balanced for order and carry-over effects will suffice.

Efficient nearly balanced incomplete blocks designs can be generated by one of several computer programs, e.g. those written by David Whittaker, Nye John and
Bennet McElwee. One of these uses cyclic designs, which we shall use as a starting point for constructing designs nearly balanced for order and carry-over effects.

2.1 Cyclic designs

A cyclic design is determined by its initial blocks. If there are \( v \) treatments represented by the numbers 0, 1, ..., \( v - 1 \), then the full design is generated by repeatedly adding 1 to each element of the initial block and reducing mod \( v \). Each new block is added to the design until the the new block is equivalent to the initial block as a set.

For example the initial block \((013)\) with \( v = 4 \) generates the design with blocks \((013),(120),(231),(302)\).

With several different initial blocks designs with efficiency close to the maximum possible can usually be obtained. For incomplete block designs the advantage of using cyclic designs is that it cuts the number of possible designs down to a manageable number which can be enumerated, but which still contains many efficient designs.

For taste testing cyclic designs have several further advantages. Suppose we have an efficient cyclic design. Because of the cyclic property every treatment already occurs equally often in each order. (One of the desired properties). This will remain the case if we permute the initial blocks and re-generate the design cyclically. Since the efficiency of a design is not affected if individual blocks are re-ordered, we can retain efficiency and order balance with any set of permutations of the initial blocks. By optimising over the set of permutations we can usually achieve near balance for first order carry-over effects. To assess the number of times each treatment is preceded by each other treatment it is sufficient (by the cyclic property again) to consider the number of times treatment 0 is preceded by treatment \( i \). Furthermore because of the cyclic property each treatment is automatically preceded equally often by the empty treatment.

For a given set of initial blocks optimisation can proceed by minimising the objective function \( c(D) = \text{range}(S) + \text{stdev}(S) \), where \( S = n_i : 1 \leq i < v \) where \( n_i \) is the number of times treatment 0 is preceded by treatment \( i \) in the design \( D \).

Number of precedences of treatment 0 in cyclically generated blocks

<table>
<thead>
<tr>
<th>permuted initial block</th>
<th>treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>(0 1 2 3)</td>
<td>0 0 0 3</td>
</tr>
<tr>
<td>(1 0 3 2)</td>
<td>2 1 0 0*</td>
</tr>
<tr>
<td>(3 2 1 0)</td>
<td>0 1 0 2</td>
</tr>
<tr>
<td>(0 2 1 3)</td>
<td>1 0 2 0</td>
</tr>
<tr>
<td>(0 2 3 1)</td>
<td>1 0 2 0</td>
</tr>
<tr>
<td>(1 2 3 0)</td>
<td>0 0 1 2*</td>
</tr>
<tr>
<td>(4 1 2 0)</td>
<td>0 1 1 1*</td>
</tr>
</tbody>
</table>

Total for chosen blocks | 2 2 2 3 |

* indicates chosen blocks for optimal design
2.2 Optimising and testing in Splus

```r
> m_matnx (c(0,1,2,3,0,1,2,3,0,1,2,4), nc=4,byrow=T)  # 1st precedences
> des_optimal.reorder.cyclic(m,n=5,
max.iter=100,nswaps=3)
iter 0 objective : 11.86
iter 1 objective : 4.258
iter 2 objective : 2.957
iter 11 objective : 1.5
> des
[1,] 1 0 3 2
[2,] 1 2 3 0
[3,] 4 1 2 0
> des.full.expand.cyclic(des,5)
```

2.3 When a cyclic design is not possible

When a cyclic design is not possible, or when partial sets are needed in the cyclic design a discrete optimisation is necessary. An objective function must be formulated to choose a trade-off between the importance of order and carry-over effects.

An objective function using a weighted sum of mean squares of columns of matrices $O[i,j] = \text{number of times treatment } i \text{ occurs in column } j$ and $P[i,j] = \text{number of times treatment } i \text{ is preceded by treatment } j.$ was used to obtain the design below. At each iteration a certain number of random swaps is done on each of a number of rows. As a strategy to avoid being trapped at a local minimum these numbers were increased if no improvement was obtained in a certain number of iterations, and reset to 1 if a new best design was found.

2.4 Example for product improvement

The basic design (for product improvement using response surface methodology) was a three factor design with data points corresponding to faces and corners of a unit cube, with face points pushed out to the unit sphere and three centre-points, giving a total of 8 edges + 6 faces + 3 centre-points = 18 treatments. Six samples were presented to each of 48 panelists.

A cyclic design with initial blocks (0 3 6 9 12 15), (0 1 5 9 10 14), (0 1 2 3 13 15), (0 2 4 7 10 11) generated by the cyclic designs program was used, with treatments 15,16,17 representing the 3 centrepoints. The first two initial blocks generated partial sets of 3 and 9 blocks.

After 50,000 iterations a design was found in which each treatment was preceded by every other treatment 0,1, or 2 times, but only preceded twice by at most one treatment.
> \texttt{table(cbind(des48x6.full.opt[,-6]),des48x6.full.opt[,-1])}

\begin{verbatim}
  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
0 0 2 0 1 1 1 1 0 1 2 0 0 0 1 1 0 1 1
1 1 0 1 1 1 1 1 0 0 1 1 0 1 0 1 1
2 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0
3 1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 1
4 0 1 0 0 0 1 1 0 1 1 1 1 1 1 2 1
5 1 1 1 1 0 2 0 1 0 1 1 1 1 1 0 0 1
6 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1
7 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1
8 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 1
9 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1
10 0 1 1 1 1 1 0 1 1 1 1 0 1 2 1 1 0 1 0
11 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1
12 1 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1
13 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1
14 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 1
15 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1
16 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1
17 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0
\end{verbatim}

> \texttt{table(col(des48x6.full.opt),des48x6.full.opt)}

\begin{verbatim}
  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
0 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
6 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
\end{verbatim}

\section*{Acknowledgments}

I would like to thank Professor Nye John and David Whitaker for introducing me to cyclic designs and discussing their work on optimal designs, when we were all members of the former DSIR Applied Mathematics Division, and John Maindonald for many useful discussions.

\section*{References}


