EFFECTIVE TEST DESIGN USING COVERING ARRAYS
Definition A

A covering array $\text{CA}(N; t, k, v)$ is an $N \times k$ array such that the $i$-th column contains $v$ distinct symbols. If a $\text{CA}(N; t, k, v)$ has the property that for any $t$ coordinate projection, all $v^t$ combinations of symbols exist at least once, then it is a $t$-covering array (or strength $t$ covering array). A $t$-covering array is optimal if $N$ is minimal for fixed $t$, $k$, and $v$. \(^1\)
Definition B

A covering array is a tool you can use to optimize your testing to get enough information without using too many resources. Covering arrays help with this by optimally sampling from the input space, thus maximizing your chances of finding defects while minimizing time-resources-cost.

- Use covering arrays to balance your resources and manage your risk.
- You must know what is enough information, and you must know what are too many resources.
- GIGO – You must understand what you are testing.
- Design test plans to use covering arrays from the start.
What goes into a covering array?

- **Factor:** An input, something to test.
- **Level:** The possible settings for a factor.
  Note that factors can be continuous. To handle continuous values, divide the continuous values into partitions of equivalent data.
- **Strength:** How many factors to cover. A strength 2 covering array provides a test for every possible pair of factor-levels. Strength 3 provides a test for every possible combination of 3 factor-levels.
WHAT AM I TESTING  EXAMPLE: DETERMINING FACTORS AND LEVELS

You can change the styles for JMP’s report tables: shading rows, etc. I would like to test all these styles to make sure they print properly. If there is a problem with one of the settings, I just need to try each one.

But, if there is a problem caused by the interaction of settings, I probably won’t find it that way.

• 6 options
• Each setting is on or off
• Each option can be set independently
• That means I have six 2-level factors
A full factorial is 64 runs: 6 options, each with 2 levels. $2^6 = 64$

Resources: If I weren’t performing 64 separate tests here, what else could I be doing?

Many software problems are caused by 2-way interactions. ²

A strength 2 covering array will give me a test plan that covers all possible 2-way interactions.

The decision on how much to test depends on available resources and the level of risk involved.
FILL IN THE DIALOG  FACTORS, LEVELS, AND STRENGTH

![DOE - JMP Pro](image)

### Covering Array

#### Factors

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underline Headings</td>
<td>Categorical</td>
<td>On</td>
</tr>
<tr>
<td>Shade Headings</td>
<td>Categorical</td>
<td>On</td>
</tr>
<tr>
<td>Column Borders</td>
<td>Categorical</td>
<td>On</td>
</tr>
<tr>
<td>Row Borders</td>
<td>Categorical</td>
<td>On</td>
</tr>
<tr>
<td>Shade Alternate Rows</td>
<td>Categorical</td>
<td>On</td>
</tr>
<tr>
<td>Shade Cells</td>
<td>Categorical</td>
<td>On</td>
</tr>
</tbody>
</table>

**Specify Factors**

Add a factor by clicking the Add Factor button. Double click on a factor name or level to edit it.

[Continue]
I have a table with 6 test runs.
I go and perform each test.
I mark each run as Pass (1) or Fail (0) in the Response column.

<table>
<thead>
<tr>
<th>Response</th>
<th>Underline Headings</th>
<th>Shade Headings</th>
<th>Column Borders</th>
<th>Row Borders</th>
<th>Shade Alternate Rows</th>
<th>Shade Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>2</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>3</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>4</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>5</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>6</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
</tbody>
</table>
The analysis script provides a fast way to tally the results.

I had 5 passes and 1 failure.

The analysis script can tell me immediately that one or more of three pairwise combinations is causing a problem.

With 6 runs, I discovered a problem I would not have found had I tested each setting by itself.
What if I have 5 options (factors),
but one is only possible
if another is set to a particular setting (level)?

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paint Type</td>
<td>Categorical</td>
<td>Blue, Red</td>
</tr>
<tr>
<td>Material</td>
<td>Categorical</td>
<td>Iron, Aluminum</td>
</tr>
<tr>
<td>Time to Set</td>
<td>Categorical</td>
<td>2 hours, 3 hours, 4 hours</td>
</tr>
<tr>
<td>Oven</td>
<td>Categorical</td>
<td>On, Off</td>
</tr>
<tr>
<td>Temperature</td>
<td>Categorical</td>
<td>High, Low</td>
</tr>
</tbody>
</table>
• Similar to data filter.
• Select the combination that’s disallowed.
• Here, if the oven is off, I can’t select a temperature. So I select Off and both temperature settings.
- This design gives me 9 runs.
- Note that anywhere the Oven is Off, there is no Temperature setting – it’s missing.
The script equivalent to the previous filter is saved when you create the table. You can initially input the disallowed combinations this way, with Use Disallowed Combinations Script instead of the filter.
• Design your test from the beginning using covering arrays.
• Know what you’re testing.
• Use covering arrays to balance resources and risk.
REFERENCES


Contact me at:
Melanie.Drake@jmp.com