Chapter 2

Alan R. Vazquez

Excercises

```
import numpy as np
```

Many of the exercises in this book use the Python `oapackage` (Eendebak and Vazquez 2019). You may need to install the package first by

```
# pip install oapackage
```

At the start of most of the programs, you need to import the package with `import oapackage`. We like to abbreviate lengthy names using

```
# import oapackage as oap
```

So whenever you see `oap` in the book or in the programs, you now know where it refers to.

Once installed and imported, you can use all kinds of convenient functions for the characterization and enumeration of the arrays. A few additional functions are provided in `oafunctions.py`. We use `import oafunctions as oaf` in our programs.

1. The function `strength` in `oafunctions.py` takes an array as input and returns the strength of the array. Have a look at this function and explain how the function works. After that, answer the following questions:

- What would a strength of 1 imply? What a strength of 0?
- Does a strength-4 array also have strength 3?
- Can you detect a strength greater than 4 with the program?

2. In many exercises, you need to read files with orthogonal arrays to do something with them. The files all have a special structure that permit easy reading by the function `oap.readarrayfile`. In particular, this is how we arranged all the files:

- First line: 3 numbers specifying c:# columns of a single array, r: # rows of a single array and a:# orthogonal arrays in the file.
- Line 2: ID of the first array.
- Line 3 up to $(2 + r)$: the first array.
- ⋮
- Line $2 + a(r + 1) - r$ up to $1 + a(r + 1)$: array #a.
- Last line: -1.

We assume that the symbols in any column of an array are coded $0, 1, \ldots, (s - 1)$, where s is the number of different symbols in that column.

The file "result-12.2-2-2-2-2-2.oa" contains two different OA(12, 2^6, 2). Check the file's structure by opening it in an editor. You read the file with

```
#N12n6 = oap.readarrayfile(`result-12.2-2-2-2-2-2.oa')
```

The command returns a tuple called `N12n6` with two so-called `array_link` objects. Don't worry! They don't bite. We will gently introduce them later on. For now, turn, say, the second of them into a `numpy` array with

```
# A = np.array(N12n6[1])
```

and determine the strength using the function of the previous exercise.

3. Consider pure-level arrays with level numbers $s = 2, 3, 4$.

- Make a table with the smallest possible run sizes for arrays with strength $2 \leq t \leq 4$.
- What is the smallest possible run size for an OA$(N, 4 \times 3 \times 2^4, 3)$?

4. Construct the following OAs by hand or show that the array does not exist.

- $OA(16, 4^2 2; 2)$.
- $OA(16, 4^2 2^2; 3)$.
- $OA(16, 4^1 2^3; 2)$.
- $OA(16, 4^1 2^3; 3)$.
- $OA(18, 3^2 2^2; 2)$.

6. Case study one that needs an orthogonal array. $OA(9, 3^4; 2)$

7. Case study two that needs an orthogonal array. $OA(18, 6^1 3^3; 2)$

8. Case study three that needs an orthogonal array?

9. Foldover of a three-level design. Show that you start from strength two and you arrive at a strength three array.

10. Split a two-level fractional factorial design. Split according to a basic column and to a generated column and see if you get an OA of strength 2.

11. Excercise on collapsing in Section 2.3.

12. Another type of collapsing: Strong OAs

References

Eendebak, Pieter, and Alan Vazquez. 2019. "OApackage: A Python Package for Generation and Analysis of Orthogonal Arrays, Optimal Designs and Conference Designs." *Journal of Open Source Software* 4: 1097. https://doi.org/10.21105/joss.01097.