Validating the Prediction Profiler with Disallowed Combinations - A Case Study

Jeremy Ash, Caleb King, Laura Lancaster, Ryan Lekivetz, Joseph Morgan, Yeng Saanchi

JMP Statistical Discovery

DATAWorks 2023

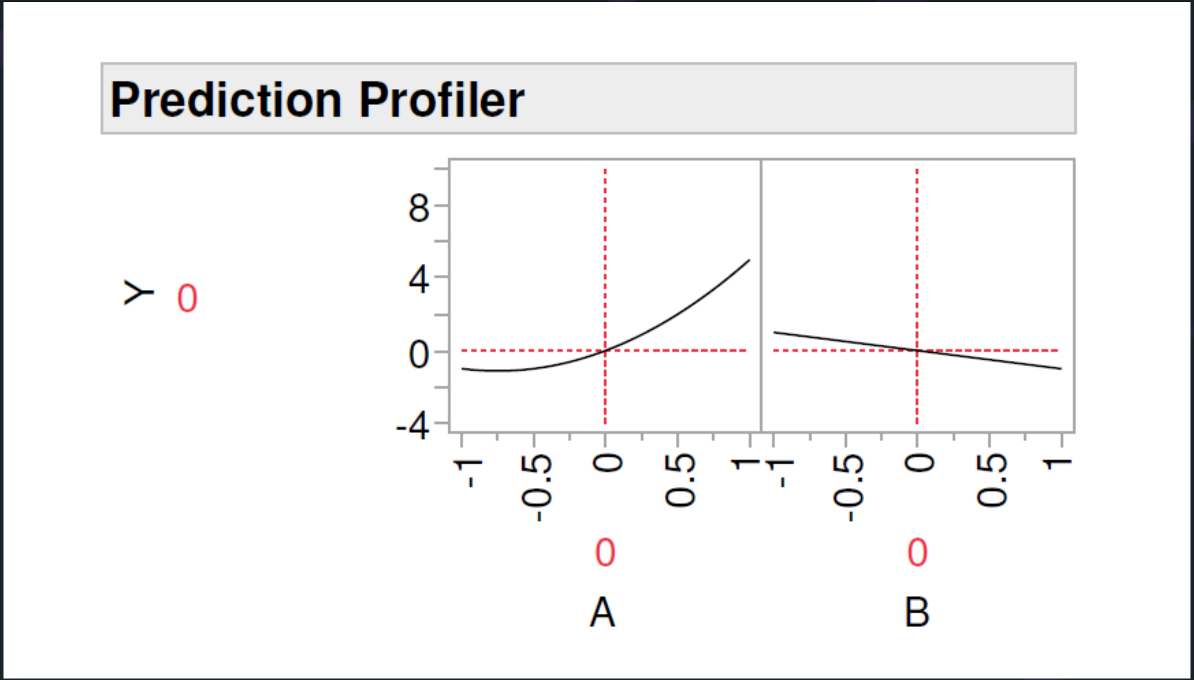
JMP STATISTICAL DISCOVERY FROM SAS

Prediction Profiler

- Visually explore the relationships between multiple factors and responses
- Set of univariate plots for each factor that show predicted response(s) for settings of the factors
- Interact with plots to change factor values
- Often used in conjunction with desirability function
- “Maximize desirability” is a very important feature

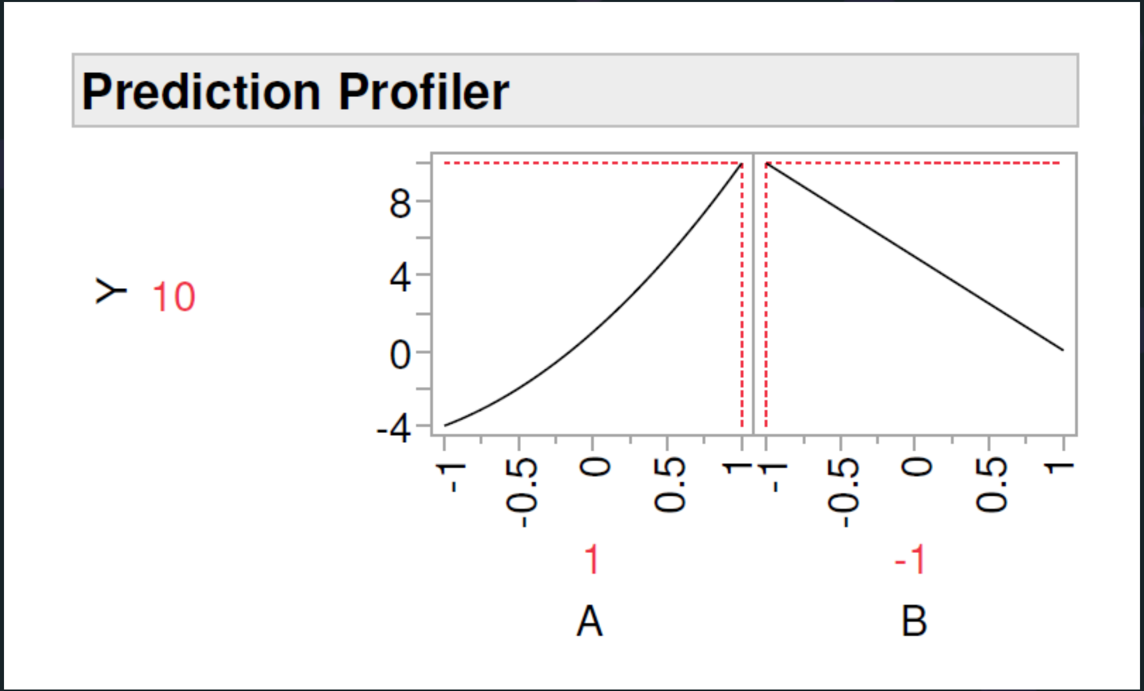
Prediction Profiler

$$Y = 3 * A - 1 * B + 2 * A^2 - 4 * A * B.$$



Prediction Profiler

-Response goal (related to desirability) to maximize response

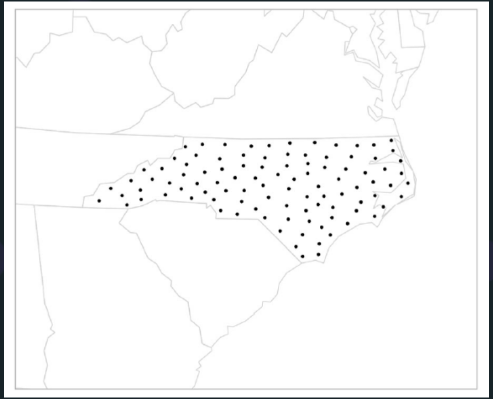


Disallowed Combinations

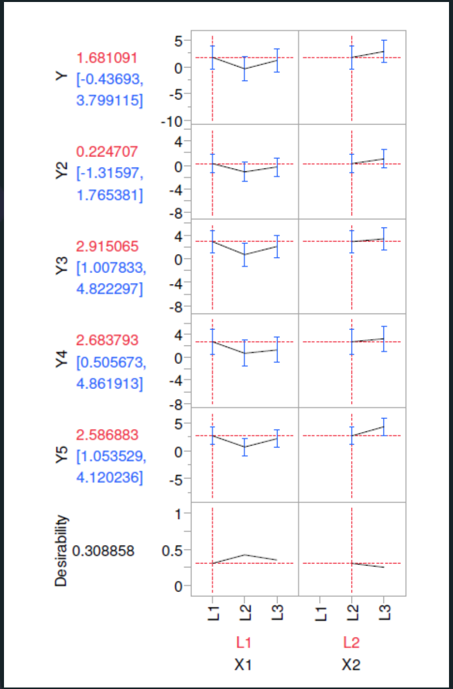
- Boolean expression that evaluates to true if a given design point is not in the design space and false for a design point that is in the design space.
- Typically used to create a design with no disallowed combinations (i.e., only design points in the allowable region)

Examples:

- A * B > 0.5 | A * B < -0.5
- A = “old machine” & B = “new part”
- outside of a polygon



Disallowed Combinations



- 5 responses, Y1-Y5
- 3-level Categorical X1 & X2
- Disallowed Combination

  X1 = “L1” & X2 = “L1”

Our Task

Test the prediction profiler with disallowed combinations

Team

- Developers and test engineers with different backgrounds: advanced degrees in statistics, computer science, operations research, and bioinformatics
- This team had been investigating “maximize desirability”

What is software testing?

Our Adopted Definition

“Testing is the process of executing a program with the intent of finding errors.”

G. Myers, The Art of Software Testing, Wiley, 1979

Our Challenge

- How to determine appropriate coverage of all possible uses of the feature?
- How to determine the appropriate “oracle” for comparing and evaluating test results?

Where are the bugs?

“Bugs lurk in corners and congregate at boundaries.”

B. Beizer, Software Testing Techniques, Van Nostrand Reinhold, 1983

Our Testing Approach

Combinatorial Testing via Covering Arrays

- Use of covering arrays so that for a system with *m* inputs, a strength *t* covering array ensures that all possible combinations for any set of *t* inputs will occur at least once in the suite of test cases
- Why Covering Arrays?
  - Cost-efficient
  - Selection problem – what to test
  - Enable finding of failures due to interactions between multiple factors
  - Disciplined approach to testing vs. "let's just test more"

Data Pedigree

- What do you do with limited data sets?
- Need to anticipate how they might be used
- Our Solution:
  - – Combine data set generation and test case selection using combinatorial testing
  - – Consider both test case selection and data set generation factors
    - Data set generation – 15 inputs
    - Test case selection – 3 inputs

Factors and Levels

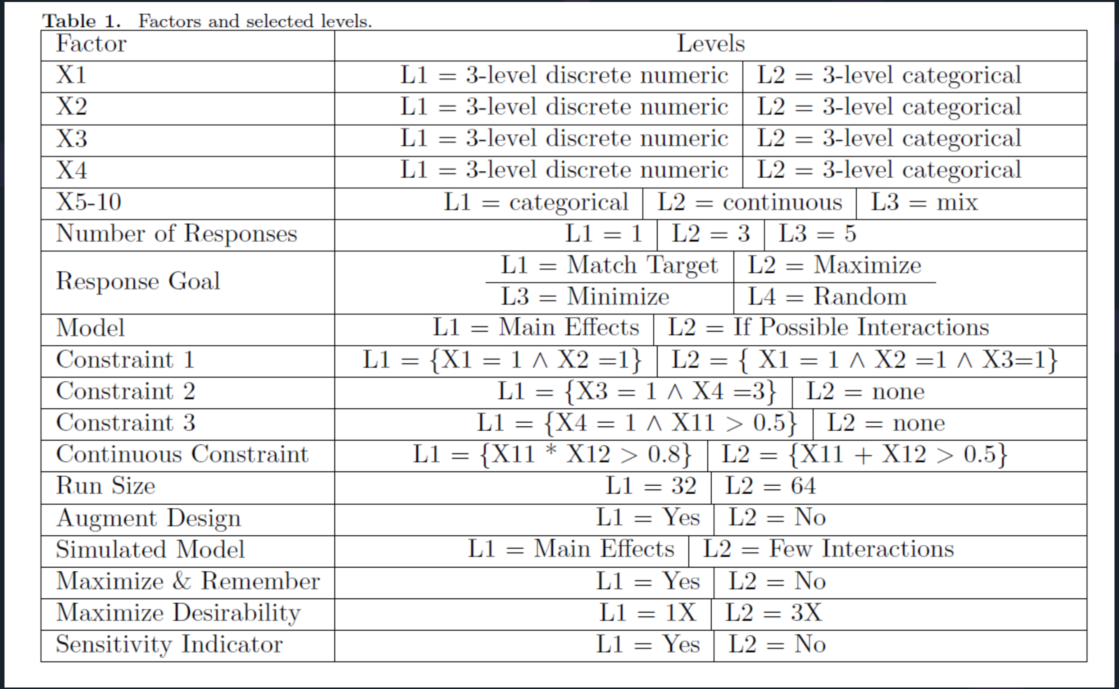


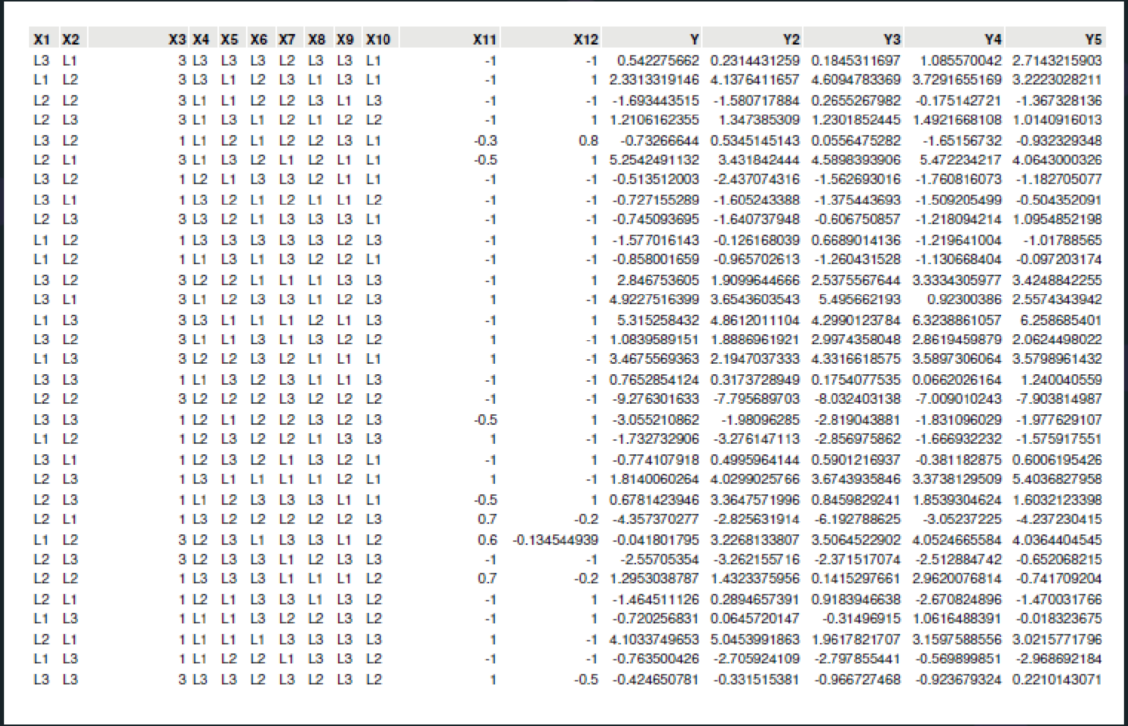
Table 1. Factors and selected levels.

Factor	Levels		
X1	L1 = 3-level discrete numeric	L2 = 3-level categorical	
X2	L1 = 3-level discrete numeric	L2 = 3-level categorical	
X3	L1 = 3-level discrete numeric	L2 = 3-level categorical	
X4	L1 = 3-level discrete numeric	L2 = 3-level categorical	
X5-10	L1 = categorical	L2 = continuous	L3 = mix
Number of Responses	L1 = 1	L2 = 3	L3 = 5
Response Goal	L1 = Match Target	L2 = Maximize	
	L3 = Minimize	L4 = Random	
Model	L1 = Main Effects	L2 = If Possible Interactions	
Constraint 1	L1 = $\{X1 = 1 \wedge X2 = 1\}$	L2 = $\{X1 = 1 \wedge X2 = 1 \wedge X3 = 1\}$	
Constraint 2	L1 = $\{X3 = 1 \wedge X4 = 3\}$	L2 = none	
Constraint 3	L1 = $\{X4 = 1 \wedge X11 > 0.5\}$	L2 = none	
Continuous Constraint	L1 = $\{X11 * X12 > 0.8\}$	L2 = $\{X11 + X12 > 0.5\}$	
Run Size	L1 = 32	L2 = 64	
Augment Design	L1 = Yes	L2 = No	
Simulated Model	L1 = Main Effects	L2 = Few Interactions	
Maximize & Remember	L1 = Yes	L2 = No	
Maximize Desirability	L1 = 1X	L2 = 3X	
Sensitivity Indicator	L1 = Yes	L2 = No	

Test Suite

- $2^4 * 3^2 * 4 * 2^8 * 2^3$ = 1,179,648 possible test cases
- Strength 2 covering array has all pairwise combinations covered in 13 runs
- All 8 combinations of the 3 profiler options are covered

One Test Case from Test Suite



X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	Y	Y2	Y3	Y4	Y5
L3	L1	3	L3	L3	L3	L2	L3	L3	L1	-1	-1	0.542275662	0.2314431259	0.1845311697	1.085570042	2.7143215903
L1	L2	3	L3	L1	L2	L3	L1	L3	L1	-1	1	2.3313319146	4.1376411657	4.6094783369	3.7291655169	3.2223028211
L2	L2	3	L1	L1	L2	L2	L3	L1	L3	-1	-1	-1.693443515	-1.580717884	0.2655267982	-0.175142721	-1.367328136
L2	L3	3	L1	L3	L1	L2	L1	L2	L2	-1	1	1.2106162355	1.347385309	1.2301852445	1.4921668108	1.0140916013
L3	L2	1	L1	L2	L1	L2	L2	L3	L1	-0.3	0.8	-0.73266644	0.5345145143	0.0556475282	-1.65156732	-0.932329348
L2	L1	3	L1	L3	L2	L1	L2	L1	L1	-0.5	1	5.2542491132	3.431842444	4.5898393906	5.472234217	4.0643000326
L3	L2	1	L2	L1	L3	L3	L2	L1	L1	-1	-1	-0.513512003	-2.437074316	-1.562693016	-1.760816073	-1.182705077
L3	L1	1	L3	L2	L1	L2	L1	L1	L2	-1	-1	-0.727155289	-1.605243388	-1.375443693	-1.509205499	-0.504352091
L2	L3	3	L3	L2	L1	L3	L3	L3	L1	-1	-1	-0.745093695	-1.640737948	-0.606750857	-1.218094214	1.0954852198
L1	L2	1	L3	L3	L3	L3	L3	L2	L3	-1	1	-1.577016143	-0.126168039	0.6689014136	-1.219641004	-1.01788565
L1	L2	1	L1	L3	L1	L3	L2	L2	L1	-1	-1	-0.858001659	-0.965702613	-1.260431528	-1.130668404	-0.097203174
L3	L2	3	L2	L2	L1	L1	L1	L3	L3	-1	1	2.846753605	1.9099644666	2.5375567644	3.3334305977	3.4248842255
L3	L1	3	L1	L2	L3	L3	L1	L2	L3	1	-1	4.9227516399	3.6543603543	5.495662193	0.92300386	2.5574343942
L1	L3	3	L3	L1	L1	L1	L2	L1	L3	-1	1	5.315258432	4.8612011104	4.2990123784	6.3238861057	6.258685401
L3	L2	3	L1	L1	L3	L1	L3	L2	L2	1	-1	1.0839589151	1.8886961921	2.9974358048	2.8619459879	2.0624498022
L1	L3	3	L2	L2	L3	L2	L1	L1	L1	1	-1	3.4675569363	2.1947037333	4.3316618575	3.5897306064	3.5798961432
L3	L3	1	L1	L3	L2	L3	L1	L1	L3	-1	-1	0.7652854124	0.3173728949	0.1754077535	0.0662026164	1.240040559
L2	L2	3	L2	L2	L2	L3	L2	L2	L2	-1	-1	-9.276301633	-7.795689703	-8.032403138	-7.009010243	-7.903814987
L3	L3	1	L2	L1	L2	L2	L3	L2	L3	-0.5	1	-3.055210862	-1.98096285	-2.819043881	-1.831096029	-1.977629107
L1	L2	1	L2	L3	L2	L2	L1	L3	L3	1	-1	-1.732732906	-3.276147113	-2.856975862	-1.666932232	-1.575917551
L3	L1	1	L2	L3	L2	L1	L3	L2	L1	-1	1	-0.774107918	0.4995964144	0.5901216937	-0.381182875	0.6006195426
L2	L3	1	L3	L1	L1	L1	L1	L2	L1	1	-1	1.8140060264	4.0299025766	3.6743935846	3.3738129509	5.4036827958
L2	L3	1	L1	L2	L3	L3	L3	L1	L1	-0.5	1	0.6781423946	3.3647571996	0.8459829241	1.8539304624	1.6032123398
L2	L1	1	L3	L2	L2	L2	L2	L2	L3	0.7	-0.2	-4.357370277	-2.825631914	-6.192788625	-3.05237225	-4.237230415
L1	L2	3	L2	L3	L1	L3	L3	L1	L2	0.6	-0.134544939	-0.041801795	3.2268133807	3.5064522902	4.0524665584	4.0364404545
L2	L3	3	L2	L3	L3	L1	L2	L3	L3	-1	-1	-2.55705354	-3.262155716	-2.371517074	-2.512884742	-0.652068215
L2	L2	1	L3	L3	L3	L1	L1	L1	L2	0.7	-0.2	1.2953038787	1.4323375956	0.1415297661	2.9620076814	-0.741709204
L2	L1	1	L2	L1	L3	L3	L1	L3	L2	-1	1	-1.464511126	0.2894657391	0.9183946638	-2.670824896	-1.470031766
L1	L3	1	L1	L1	L3	L2	L2	L3	L2	-1	1	-0.720256831	0.0645720147	-0.31496915	1.0616488391	-0.018323675
L2	L1	1	L1	L1	L1	L3	L3	L3	L3	1	-1	4.1033749653	5.0453991863	1.9617821707	3.1597588556	3.0215771796
L1	L3	1	L1	L2	L2	L1	L3	L3	L2	-1	-1	-0.763500426	-2.705924109	-2.797855441	-0.569899851	-2.968692184
L3	L3	3	L3	L3	L2	L3	L2	L3	L2	1	-0.5	-0.424650781	-0.331515381	-0.966727468	-0.923679324	0.2210143071

Some Aspects of the Profiler to Test

- Does the profiler display the constrained region correctly?
- Does the profiler only profile allowable regions?
- Do the various profiler options work as expected?
- Does the profiler exhibit a lag in profiling when the user interacts with its controls?
- Does maximizing desirability find the optimum in the constrained space?

Sequential Nature of Testing

- Nightly unit test suite
- Revisit oracles
- Augmentation to improve coverage

Summary

- Challenges of validating statistical software
  - Deriving oracles is difficult
  - Lack of data sets
- Combinatorial testing effective and efficient
- Combined both data set generation and profiler features as factors

Thank you!

STATISTICAL DISCOVERY FROM SAS

jmp.com

References

1. B. Beizer, Software Testing Techniques, Van Nostrand Reinhold, 1983.

2. R. Bryce & C. Colbourn, “Prioritized interaction testing for pair-wise coverage with seeding and constraints,” Information & Software Technology, 48(10), pp. 960 – 970, 2006.

3. D. Cohen, S. Dalal, M. Fredman, & G. Patton, “The AETG System: An approach to testing based on Combinatorial Design,” IEEE TSE, 23(7), 1997, pp. 437-444.

4. M. Cohen, M. Dwyer & J. Shi, “Constructing interaction test suites for highly configurable systems in the presence of constraints: A greedy approach,” IEEE TSE, 34(5), 2008, pp. 633-650.

5. C. Colbourn & V. Syrotiuk, “Coverage, location, detection, and measurement,” IEEE 9th International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2016, pp. 19–25.

6. S. Dalal & C. Mallows, “Factor-covering designs for testing software,” Technometrics, 40(3), 1998, pp. 234-243.

7. G. Demiroz & C. Yilmaz, “Cost-aware combinatorial interaction testing,” Proc. of the International Conference on Advances in System Testing and Validation Lifecycles, 2012, pp. 9–16.

8. I. Dunietz, W. Ehrlich, B. Szablak, C. Mallows, & A. Iannino, “Applying design of experiments to software testing,” Proceedings of the 19th ICSE, New York, 1997, pp. 205-215.

9. L. Ghandehari, Y. Lei, D. Kung, R. Kacker, & R. Kuhn,“Fault localization based on failure inducing combinations,” IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), 2013, pp. 168–177.

10. A. Hartman & L. Raskin, “Problems and algorithms for covering arrays,” Discrete Math, 284(1–3), 2004, pp. 149–156.

11. K. Johnson, & R. Entringer, “Largest induced subgraphs of the n-cube that contain no 4-cycles,” Journal of Combinatorial Theory, Series B, 46(3), 1989, pp. 346-355.

References

12. G. Katona, “Two applications (for search theory and truth functions) of Sperner type theorems,” Periodica Mathematica Hungarica, 3(1-2), 1973, pp. 19-26.

13. D. Kleitman & J. Spencer, “Families of k-independent sets,” Discrete Mathematics, 6(3), 1973, pp. 255-262.

14. R. Lekivetz, & J. Morgan, “On the testing of statistical software,” Journal of Statistical Theory and Practice (2021) (submitted).

15. R. Lekivetz, & J. Morgan, “Fault localization: Analyzing covering arrays given prior information,” 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, 2018.

16. R. Lekivetz & J. Morgan, “Combinatorial Testing: Using blocking to assign test cases for validating complex software systems,” Statistical Theory & Related Fields 5.2 (2021).

17. R. Lekivetz, & J. Morgan, “Covering Arrays: Using Prior Information for Construction, Evaluation and to Facilitate Fault Localization,” Journal of Statistical Theory and Practice 14.1 (2020): 7

18. C. King, J. Morgan, & R. Lekivetz. “Design Fractals: A Graphical Method for Evaluating Binary Covering Arrays,” 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, 2019.

19. J. Morgan, R. Lekivetz, & T. Donnelly. “Covering arrays: Evaluating coverage and diversity in the presence of disallowed combinations,” 2017 IEEE 28th Annual Software Technology Conference (STC). IEEE, 2017.

20. J. Morgan, “Combinatorial Testing: An approach to systems and software testing based on covering arrays,” in Analytic Methods in Systems and Software Testing, eds., F. Ruggeri, R. Kennett, & F. Faltin, Wiley, pp. 131, 2018.

21. J. Morgan, "Combinatorial Testing,” Wiley StatsRef: Statistics Reference Online (2020): pp. 1-10.

22. G. Myers, The Art of Software Testing, Wiley, 1979.