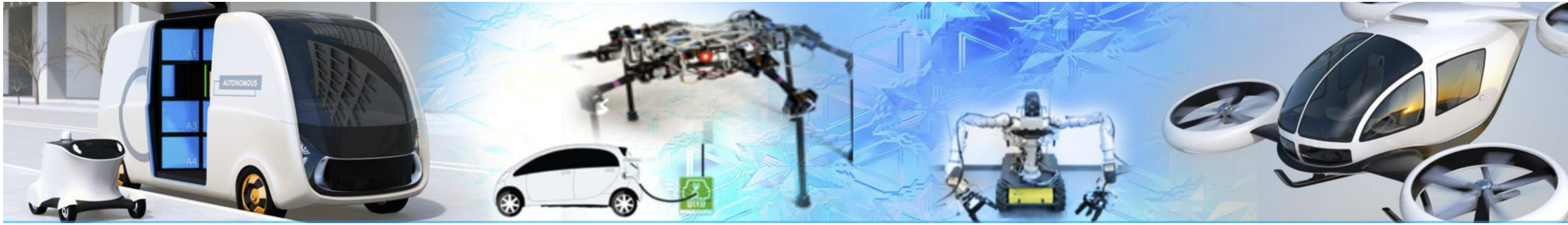


https://www.meng.ucla.edu/autonomous-systems/

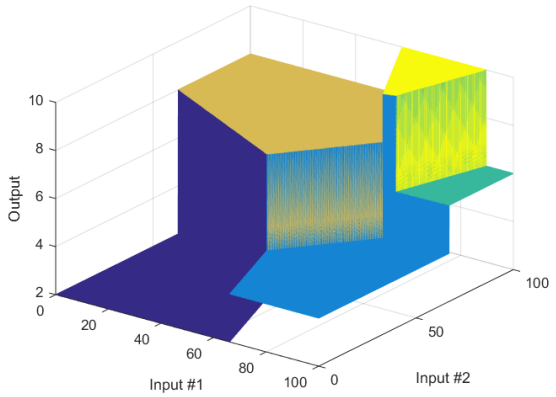
Adaptive Sampling

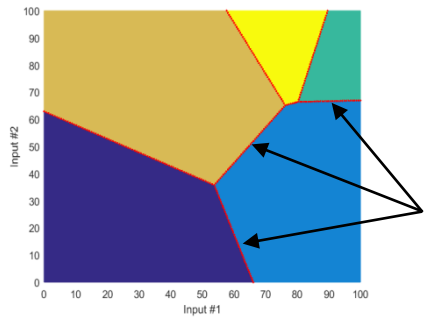
Adaptive Sampling Background and Motivation

- Explosion of autonomous systems development, test, and production
- Testing requires complex simulations with many factors and numerous runs requiring near-real time design augmentation
- Inherently nonlinear and possibly discontinuous as subtle changes in factor levels in the input space lead to large changes in response values
- Desire for new runs to be on or near these **boundary** conditions; not in the design space where performance is consistent or flat
- Best as an iterative process and want new runs to **adapt** to information learned from previous set

working-make-its-autonomous-technology-smarter

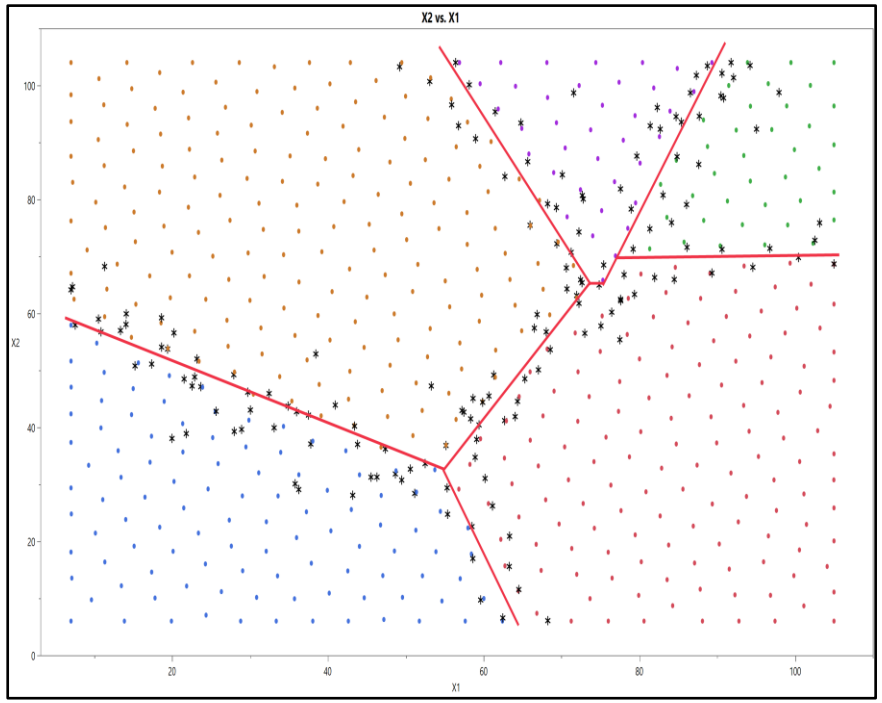
Example Problem





Performance Boundaries

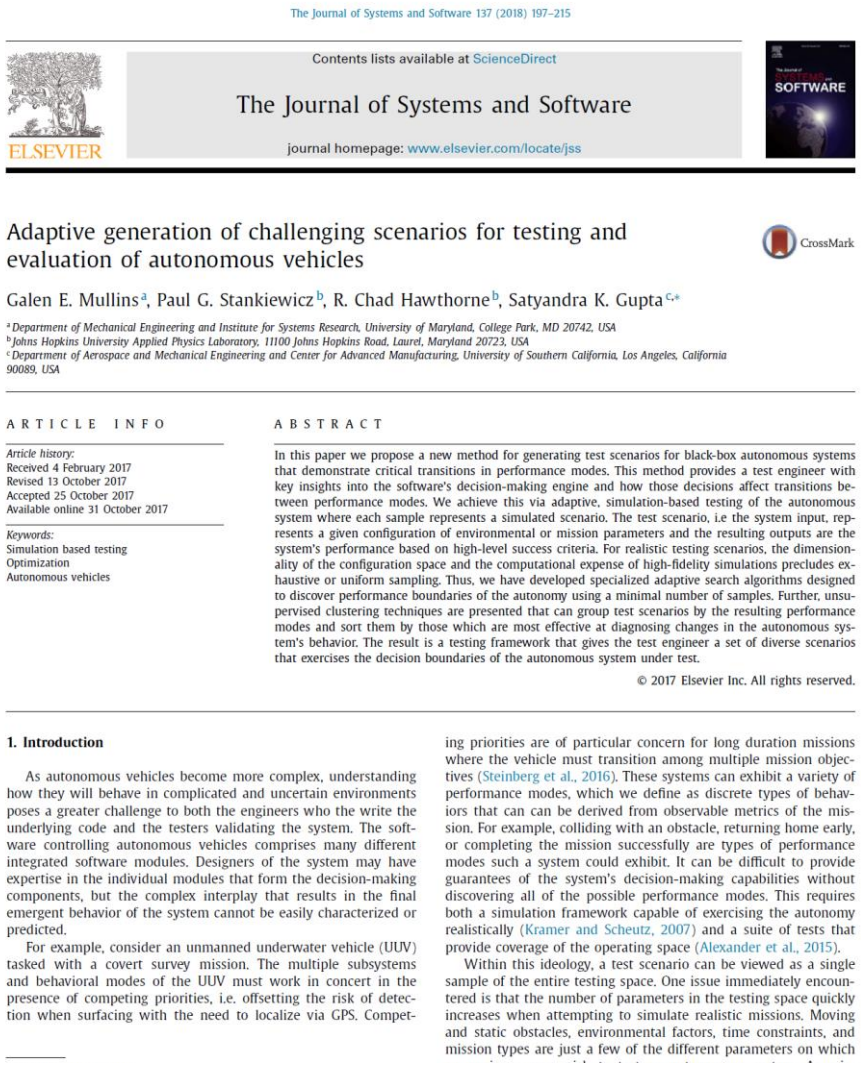
Figure source: *The Journal of Systems and Software* 137 (2018) pp 206



Our solution has new points (asterisks)
Near solid red boundary lines

- Consider Cybertruck's autonomous navigation system executing a lane change
  - Nonlinear "plate" response represents likelihood of collision that varies from 2 to 10 (20-100%)
  - Input factors represent scaled vehicle speed and traffic density
- Augment with new runs on the performance boundary using Adaptive Sampling

Current Options



The Journal of Systems and Software 137 (2018) 197–215

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles

Galen E. Mullins[a], Paul G. Stankiewicz[b], R. Chad Hawthorne[b], Satyandra K. Gupta[c,*]

[a] Department of Mechanical Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA
[b] Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel, Maryland 20723, USA
[c] Department of Aerospace and Mechanical Engineering and Center for Advanced Manufacturing, University of Southern California, Los Angeles, California 90089, USA

ARTICLE INFO

Article history:
Received 4 February 2017
Revised 13 October 2017
Accepted 25 October 2017
Available online 31 October 2017

Keywords:
Simulation based testing
Optimization
Autonomous vehicles

ABSTRACT

In this paper we propose a new method for generating test scenarios for black-box autonomous systems that demonstrate critical transitions in performance modes. This method provides a test engineer with key insights into the software's decision-making engine and how those decisions affect transitions between performance modes. We achieve this via adaptive, simulation-based testing of the autonomous system where each sample represents a simulated scenario. The test scenario, i.e the system input, represents a given configuration of environmental or mission parameters and the resulting outputs are the system's performance based on high-level success criteria. For realistic testing scenarios, the dimensionality of the configuration space and the computational expense of high-fidelity simulations precludes exhaustive or uniform sampling. Thus, we have developed specialized adaptive search algorithms designed to discover performance boundaries of the autonomy using a minimal number of samples. Further, unsupervised clustering techniques are presented that can group test scenarios by the resulting performance modes and sort them by those which are most effective at diagnosing changes in the autonomous system's behavior. The result is a testing framework that gives the test engineer a set of diverse scenarios that exercises the decision boundaries of the autonomous system under test.

1. Introduction

As autonomous vehicles become more complex, understanding how they will behave in complicated and uncertain environments poses a greater challenge to both the engineers who the write the underlying code and the testers validating the system. The software controlling autonomous vehicles comprises many different integrated software modules. Designers of the system may have expertise in the individual modules that form the decision-making components, but the complex interplay that results in the final emergent behavior of the system cannot be easily characterized or predicted.

For example, consider an unmanned underwater vehicle (UUV) tasked with a covert survey mission. The multiple subsystems and behavioral modes of the UUV must work in concert in the presence of competing priorities, i.e. offsetting the risk of detection when surfacing with the need to localize via GPS. Competing priorities are of particular concern for long duration missions where the vehicle must transition among multiple mission objectives (Steinberg et al., 2016). These systems can exhibit a variety of performance modes, which we define as discrete types of behaviors that can can be derived from observable metrics of the mission. For example, colliding with an obstacle, returning home early, or completing the mission successfully are types of performance modes such a system could exhibit. It can be difficult to provide guarantees of the system's decision-making capabilities without discovering all of the possible performance modes. This requires both a simulation framework capable of exercising the autonomy realistically (Kramer and Scheutz, 2007) and a suite of tests that provide coverage of the operating space (Alexander et al., 2015).

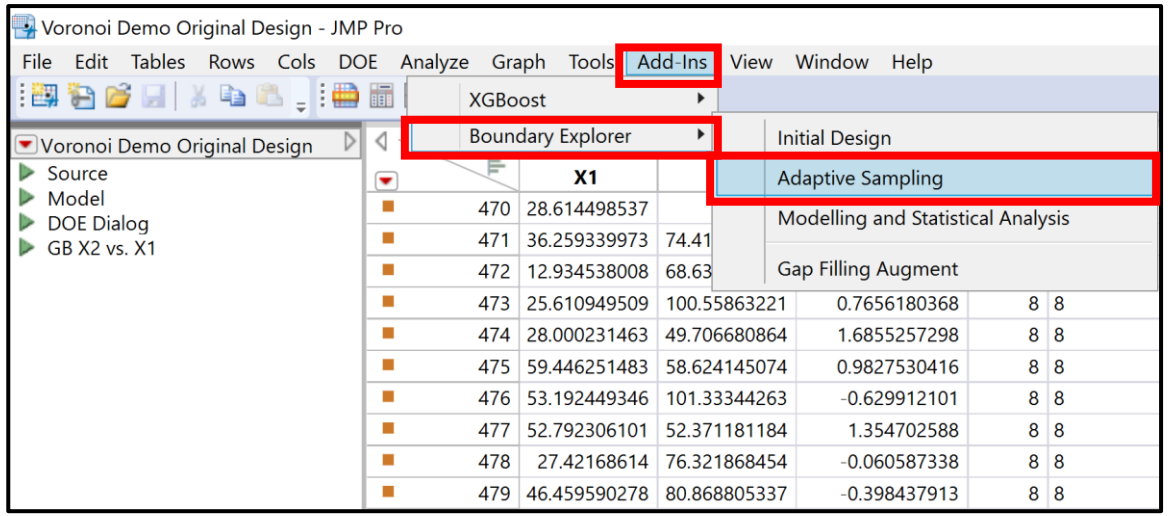
Within this ideology, a test scenario can be viewed as a single sample of the entire testing space. One issue immediately encountered is that the number of parameters in the testing space quickly increases when attempting to simulate realistic missions. Moving and static obstacles, environmental factors, time constraints, and mission types are just a few of the different parameters on which

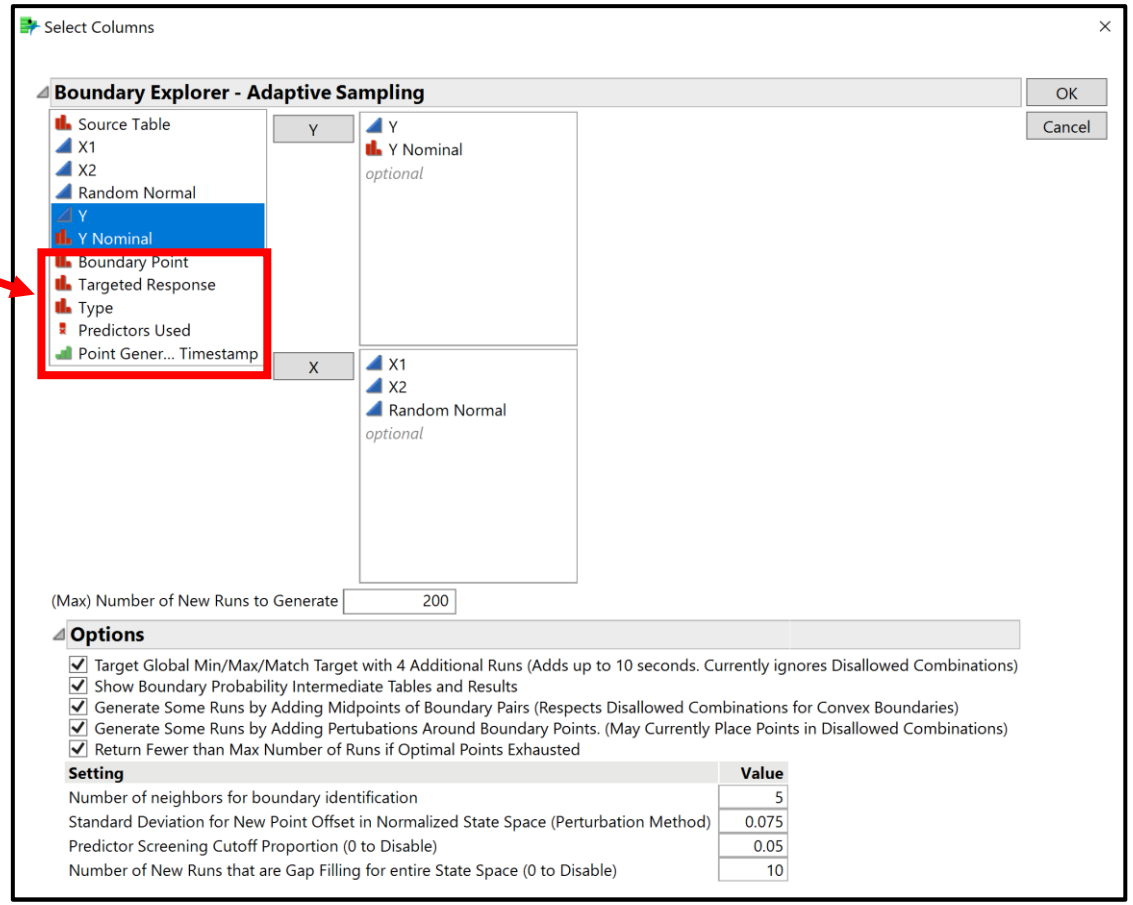
The Journal of Systems and Software 137 (2018) 197–215

- Augment with space filling hoping random points will fall near boundaries
- Manually augment based on response gradients
- Adaptive sampling algorithms from Johns Hopkins University Applied Physics Lab
  - Range Adversarial Planning Tool (RAPT)
  - Advanced capability but difficult accessibility and integration for practitioners
  - Source document for our boundary exploration algorithm development though we had numerous alternative approaches and enhancements

Boundary Exploration

- Augment 500 run design with 200 new points using both the continuous and nominal response
- Create extra factor “Random Normal” that is not statistically significant to show predictor variable screening effectiveness
- New columns automatically added





Boundary Explorer - Methodology

Predictor Screening

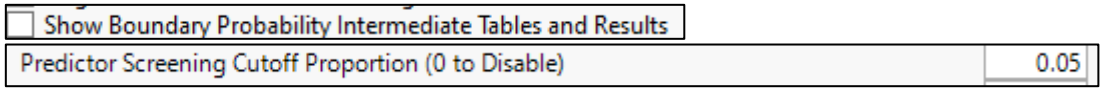
- For each response, the Predictor Screening platform is used to select the factors that are relevant for boundary identification.
- Output hidden by default, can be shown with option to **Show Boundary Probability Intermediate Tables and Results**.
- No user input is required: the factors with proportion greater than or equal to that specified in the initial GUI are retained.

Predictor Screening

Copy Selected

	Y1			
Predictor	Contribution	Portion		Rank
X1	171.200	0.6640		1
X2	81.776	0.3172		2
X3_Noise	4.837	0.0188		3

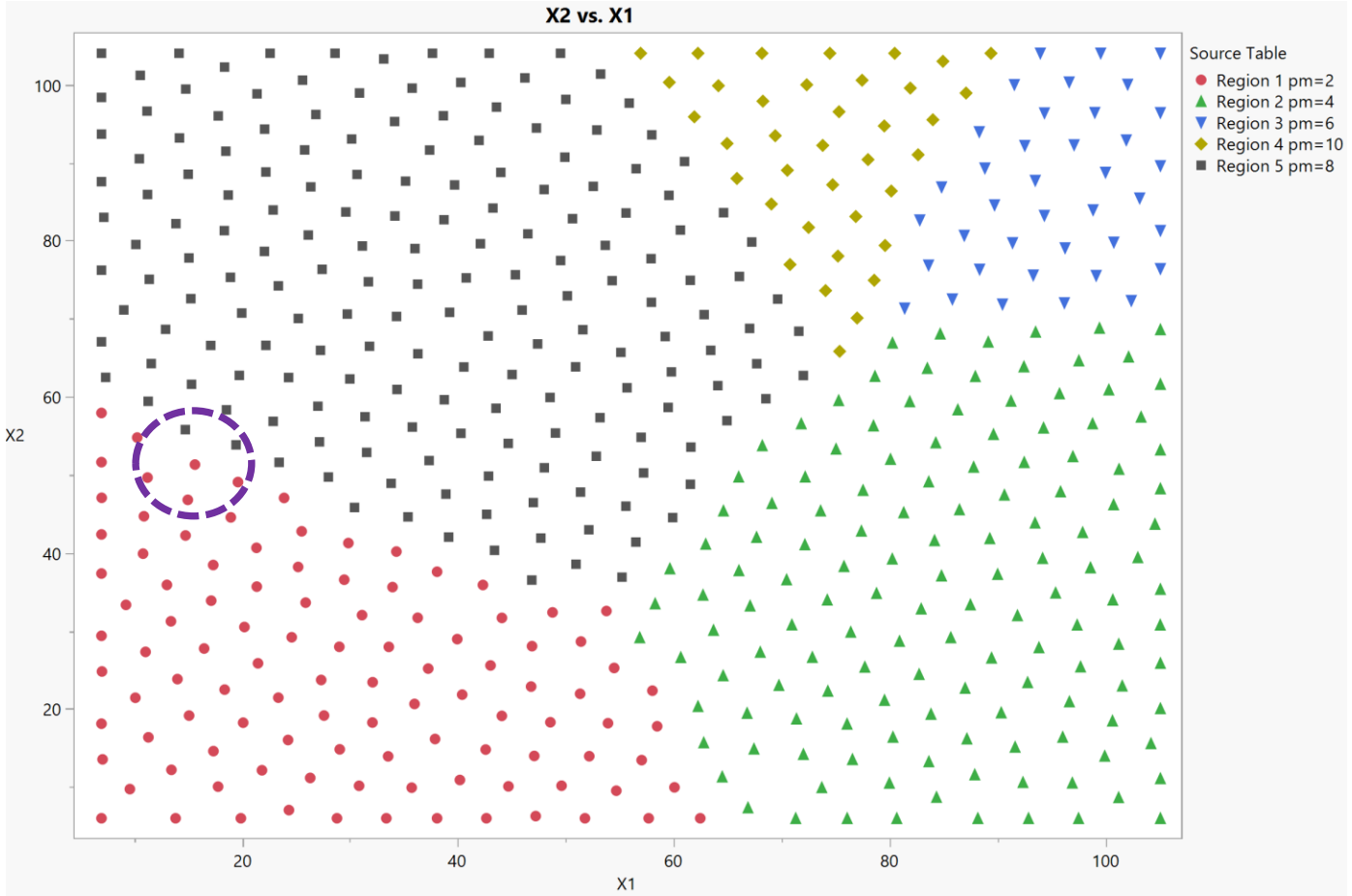
Relevant GUI Options



Boundary Explorer - Methodology

K-Nearest Neighbor Using Screened Predictors

- Using the factors retained by Predictor Screening, a 5-Nearest Neighbors model is fit to the response (nominal/continuous).



Relevant GUI Options

Setting	Value
Number of neighbors for boundary identification	5

Boundary Explorer - Methodology

Continuous Response (Y)

Find K Nearest Neighbors and Distances

- For each row (existing design point), the nearest neighbors and their distances are saved from the KNN platform to an intermediate data table.
- This information is used to detect the boundary pairs. A different approach is required for nominal vs. continuous responses.

Score Boundary Probabilities

- For each point, calculate the standard deviation of the response (Y) across the *k* nearest neighbors (5 in this example).
- Also calculate the mean distance of the 5 nearest neighbors from the parent point.
- Calculate the **_Information** metric as the product of the standard deviation and the mean distance.
- Points with relatively large values of **_Information** are either in an area with a steep gradient or in an area farther than typical away from other points (or some combination thereof).
  - How do we decide what is a large/medium/small value of **_information?**

Intermediate Output

																Information
Boundary Point	**l**	**Y**	**Y Nominal**	**RowNear 1**	**RowNear 2**	**RowNear 3**	**RowNear 4**	**RowNear 5**	**Y Predictor**	**Near 1 Y**	**Near 2 Y**	**Near 3 Y**	**Near 4 Y**	**Near 5 Y**	**_Std Dev**	**_Mean_Distance**	**_StdDev*Distance**
Yes	4	2	2	35	415	25	32	334	4.1958238369	2	8	2	2	8	3.286335345	0.15828503	0.52017769
Yes	4	2	2	13	332	230	244	5	3.2988267089	2	8	4	4	2	2.4494897...	0.15894877	0.38934338
Yes	9	2	2	33	334	437	101	48	4.5002520539	2	8	8	2	2	3.286335345	0.15665077	0.51480696
No	4	2	2	1	43	72	37	67	2.8549975919	2	2	2	2	2	0	0.16114945	0
No	2	2	2	59	67	40	43	36	3.0616647098	2	2	2	2	2	0	0.15779289	0

Boundary Explorer - Methodology

Continuous Response (Y)

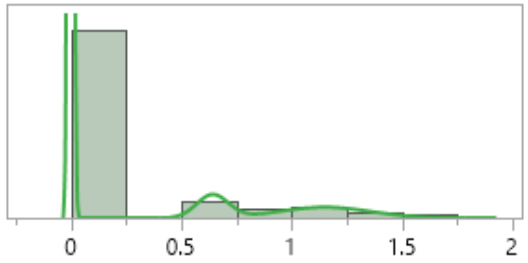
Score Boundary Probabilities

- Fit a Normal 3-Mixture to **_Information**.
- Use the largest of the three distribution means as the lower cutoff for high probability boundary points.
- Use the intersection of the first and second largest normal densities as the lower cutoff for medium probability boundary points.

Intermediate Output

Distributions

_Information



Compare Distributions

Show	Distribution	AICc ^	BIC	-2*LogLikelihood
☑	Normal 3 Mixture	-1017.508	-991.8754	-1034.262

Fitted Normal 3 Mixture Distribution

Parameter		Estimate	Lower 95%	Upper 95%
Location	μ1	9.913e-6	-0.001474	0.0014939
Location	μ2	0.641225	0.6046217	0.6778284
Location	μ3	1.1484517	1.0574633	1.2394401
Dispersion	σ1	0.0097256	.	.
Dispersion	σ2	0.0717909	0.0455074	0.1132549
Dispersion	σ3	0.2087651	0.1444668	0.3016809
Probability	π1	0.825	0.6634473	0.918527
Probability	π2	0.0738863	0.0439815	0.1215395
Probability	π3	0.1011137	0.0647804	0.1544596

After sorting so that mu1>mu2>mu3:

```
high.prob.cutoff = mu1;
med.prob.cutoff = mu1 * var2 / (var1 + var2) + mu2 * var1 / (var1 + var2);
```

Boundary Explorer - Methodology

Continuous Response (Y)

Score Boundary Probabilities

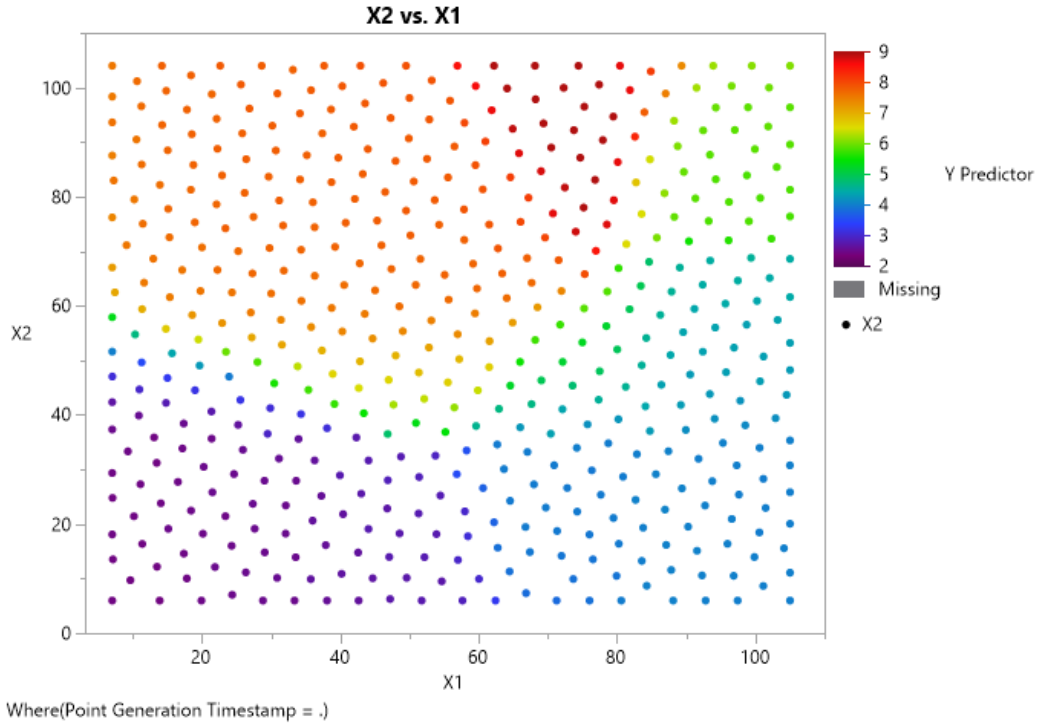
- For each medium and high probability point, record the neighbor with response (Y) that is most different from the point under consideration as the **Rowused.** The point and the **Rowused** are determined to be a boundary pair with corresponding **Boundary Prob**.

Intermediate Output

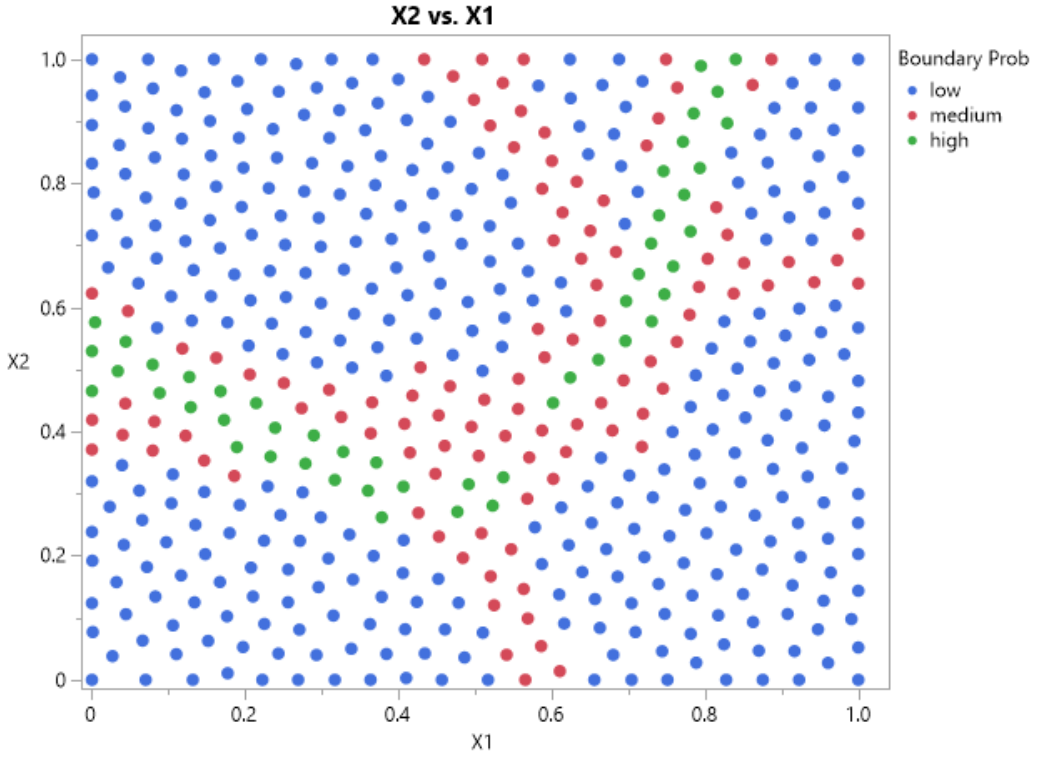
X1	X2	X3_Noise	Y2	_Information	Boundary Prob	Rowused
0.2663316583	0.3417085427	0.3044905877	-5	1.37929737	high	148

Predicted Boundary Probabilities for Initial Space Filling Points

True Boundaries and Sections



Predicted Boundary Probabilities



Boundary Explorer - Methodology

Nominal Response (Y)

Score Boundary Probabilities

- **Concat neighbors** concatenates the response for each row with the responses of the k nearest points (6 in this example)
- The **Boundary Prob** is set to
  - low if all of the neighbors are the same.
  - medium if only one element of **concat neighbor**s is different from others
  - high if there are two or more differences in **concat neighbors**.
- **Rowused** records the row number of the closest neighbor that has a different response. This is assumed to be the pairing point that spans the closest boundary.
- Could also use the more granular **Unalikeability** developed by Agresti (2011) (not used at the moment).

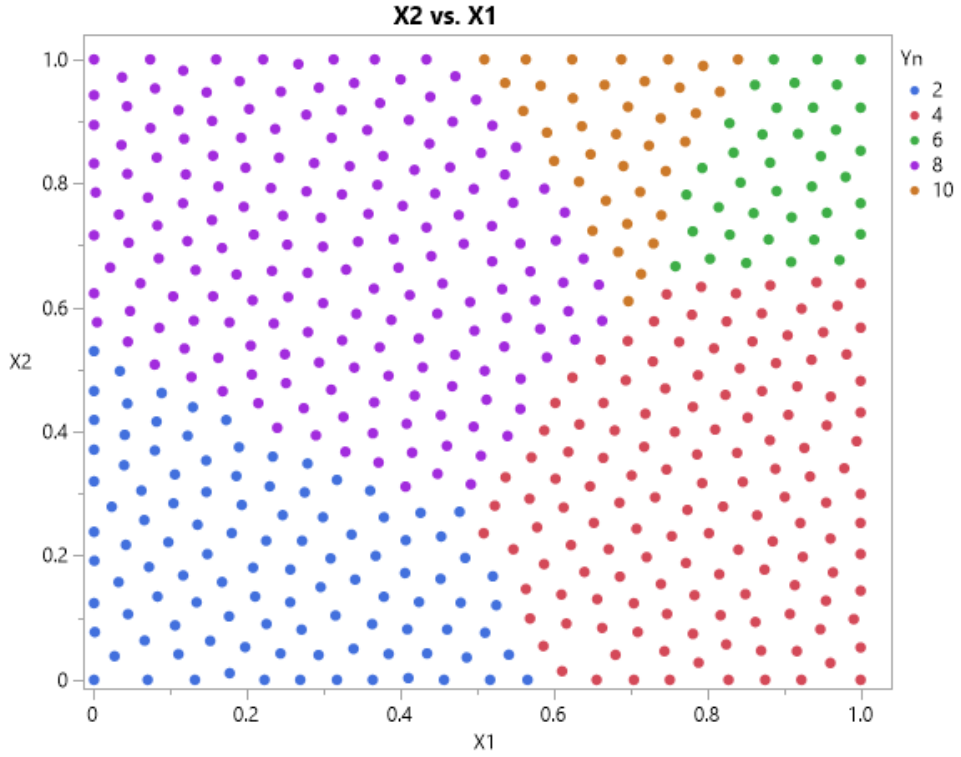
Intermediate Output

concat neighbors	Boundary Prob	Unalikeability	Rowused
C C C C C C	low	0	.
A A A A B A	medium	0.33333333	184
A A A C C C	high	0.6	258

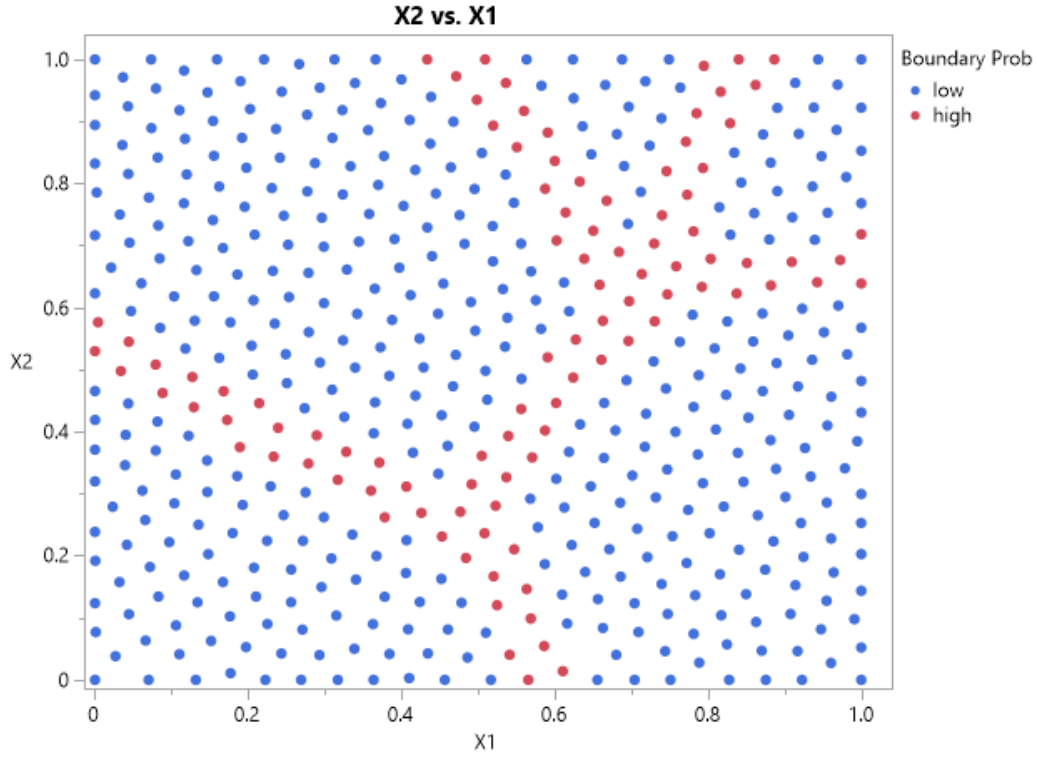
Agresti, A., Kateri, M., 2011. Categorical Data Analysis. Springer.

Predicted Boundary Probabilities for Initial Space Filling Points

True Boundaries and Sections



Predicted Boundary Probabilities

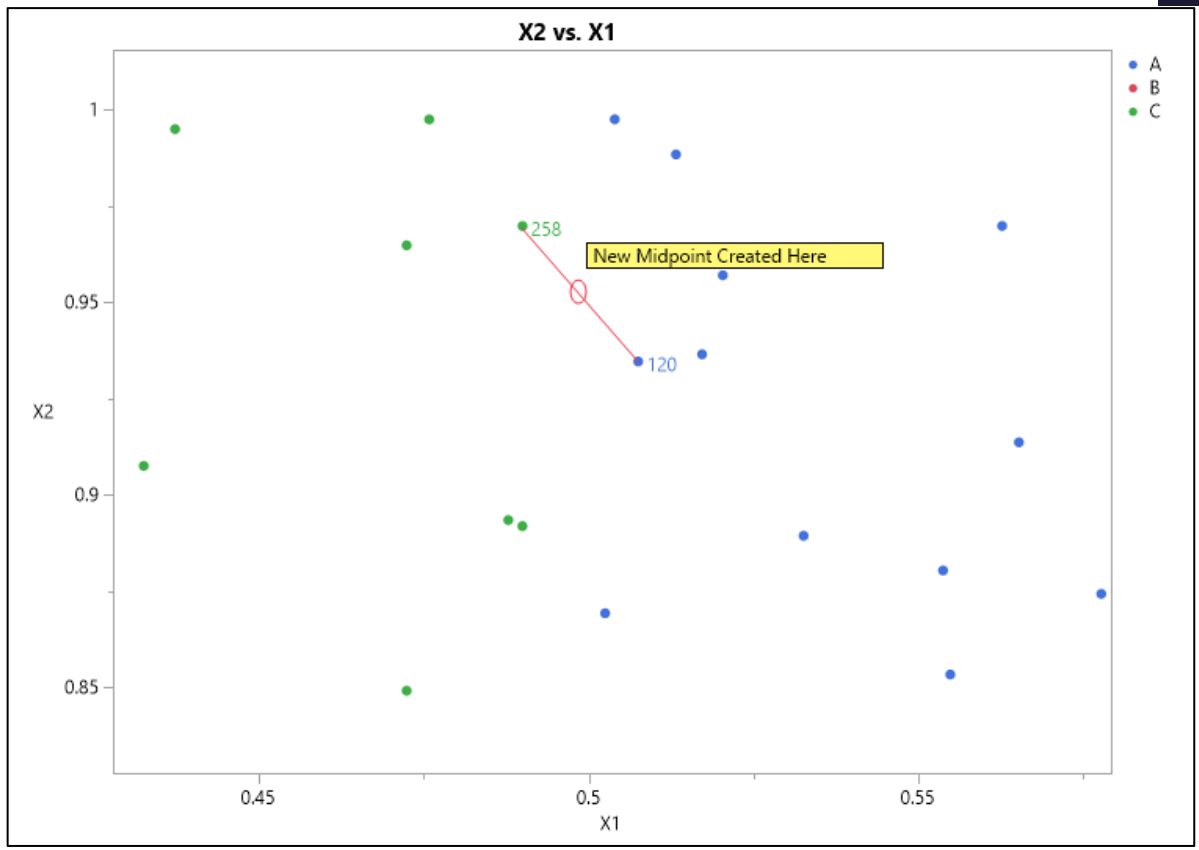


Boundary Explorer - Methodology

Continuous or Nominal Response (Y)

Create new points with Midpoint Method

- For each continuous factor (X)
  - The mean of the coordinates of the two boundary points is taken to create the setting for the new point in this factor
- For each nominal factor (X)
  - The setting for the new point is chosen at random from the levels recorded for each point of the boundary pair. This means that if a boundary pair shares a common setting for a nominal factor, their midpoint will also share that common setting for the same nominal factor.



concat neighbors	Boundary Prob	Unalikeability	Rowused
C C C C C C	low	0	•
A A A A B A	medium	0.33333333	184
A A A C C C	high	0.6	258

This is Row 120

Relevant GUI Options

☑ Generate Some Runs by Adding Midpoints of Boundary Pairs (Respects Disallowed Combinations for Convex Boundaries)

Boundary Explorer - Methodology

Continuous or Nominal Response (Y)

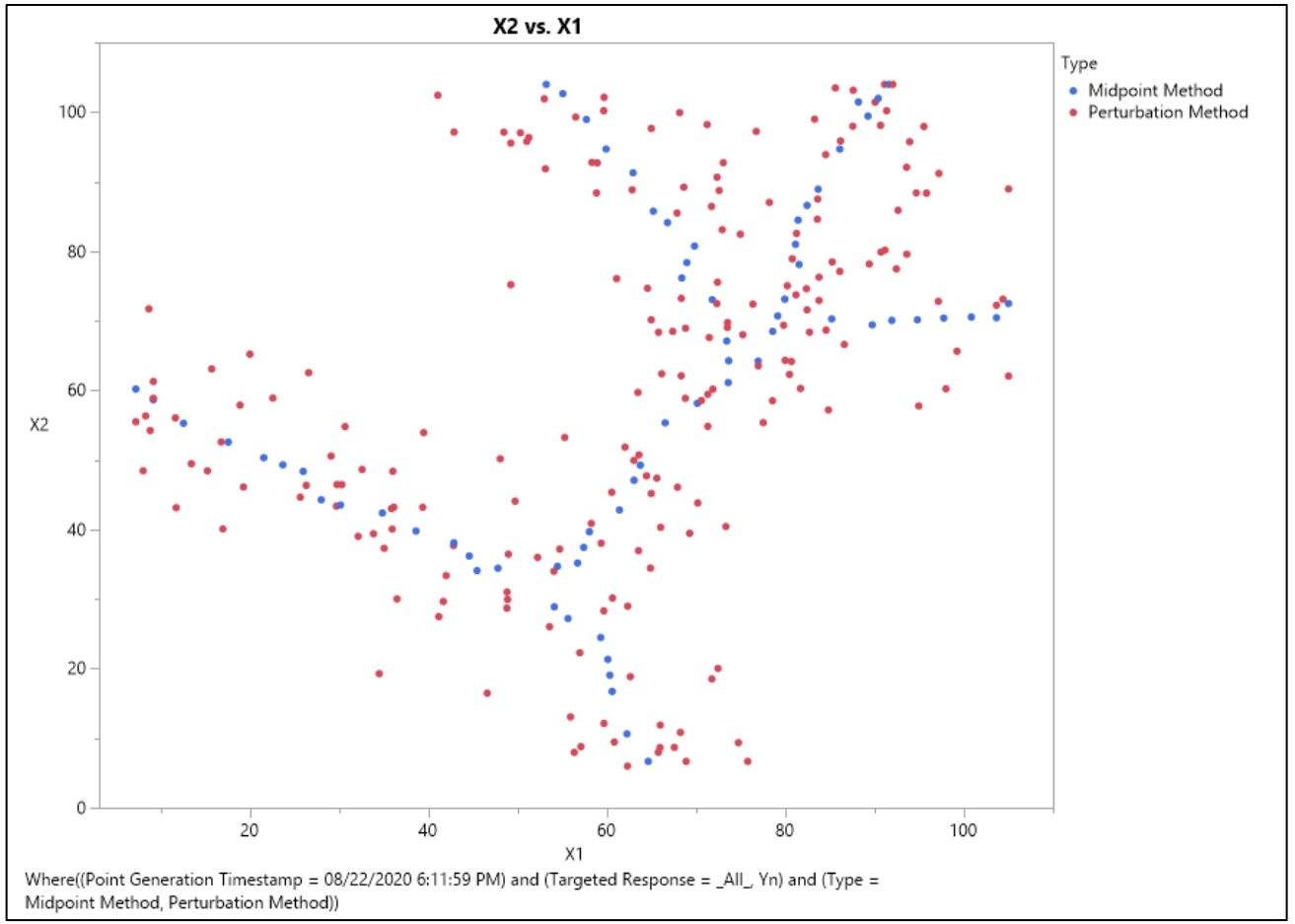
Create new points with the Perturbation Method

- **Boundary Prob** == high points receive two new perturbation points
- **Boundary Prob** == medium points receive one new perturbation point
- These spread wider than the midpoints and are useful for finding yet-undiscovered boundaries in future iterations.

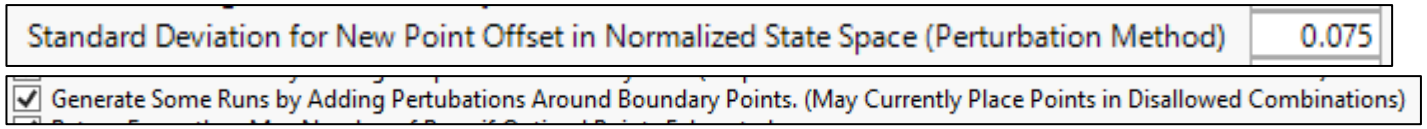
For each continuous factor (X)

- The coordinate of the new perturbation point is taken to be that of the original point plus a random normal (mean=0, standard deviation specified in settings with a default of 0.075) perturbation. Requires boundary protection.

- For each nominal factor (X)
  - The level is set at random from the possible levels of the factor. For high probability boundary points, one of the two newly created points is restricted to have the same level of the nominal factor as the generating point.



Relevant GUI Options

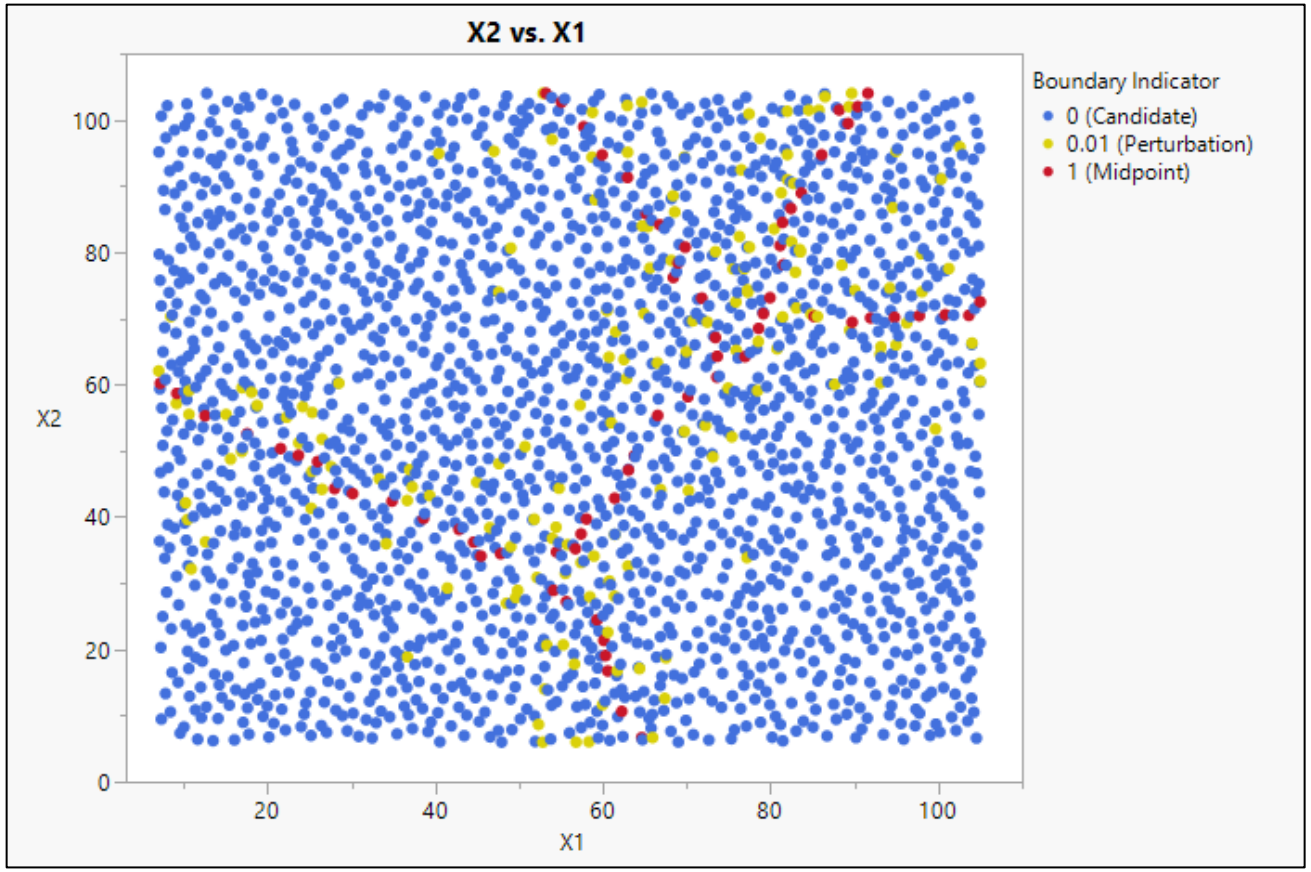


Boundary Explorer - Methodology

Continuous or Nominal Response (Y)

Create new points with a Random Forest

- After midpoint and perturbation runs have been added independently for each response, the midpoint and perturbation runs are collected (from this and all previous iterations).
- These are overlain with a set of candidate points generated by a fast flexible filling design over the entire space (blue). These candidate points receive a weight of 0 in a continuous indicator column.
- This is similar to the approach used for the Gap Filling design.
- **Midpoint** runs (red) receive a weight of 1 in the indicator column
- **Perturbation** runs (yellow) receive a weight of 0.01
- The continuous indicator column is fit with a random forest over all of the factors. The predictions for the candidate points are saved.

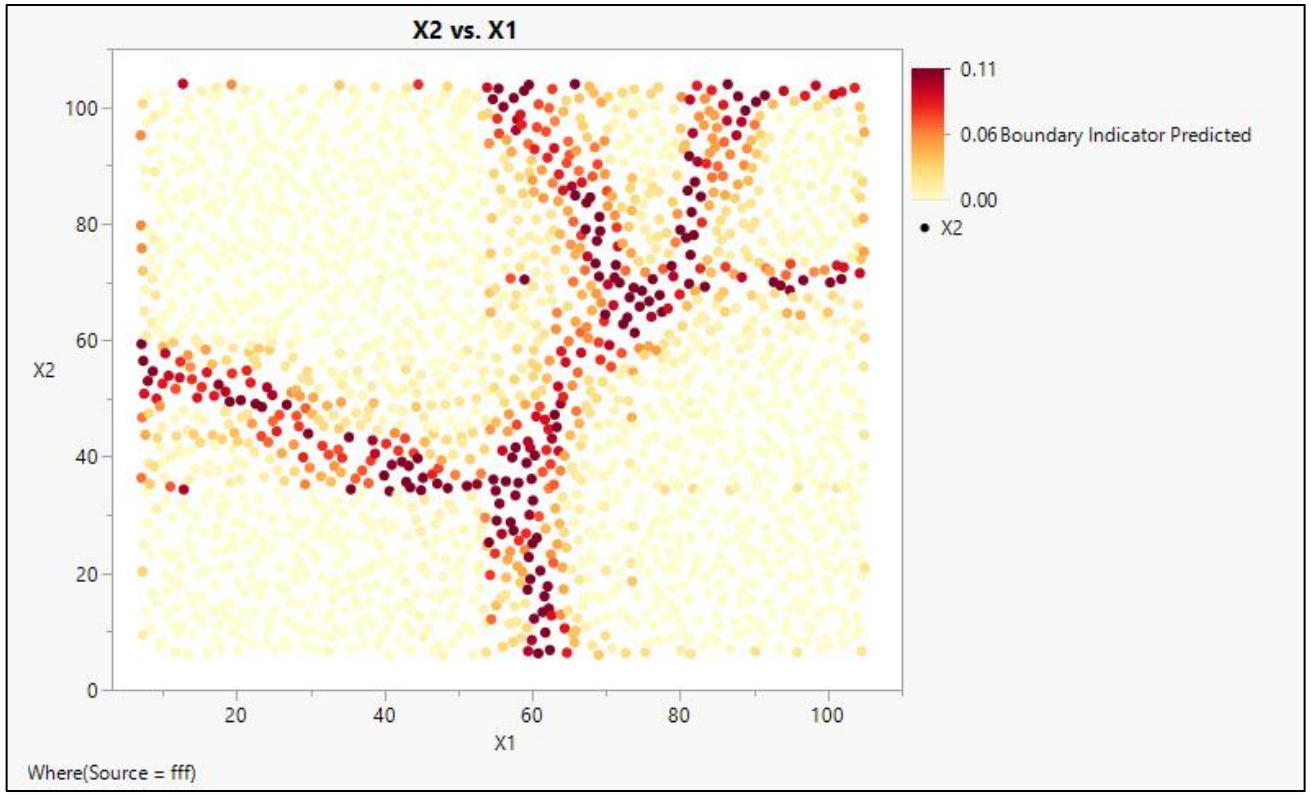


Boundary Explorer - Methodology

Continuous or Nominal Response (Y)

Create new points with a Random Forest

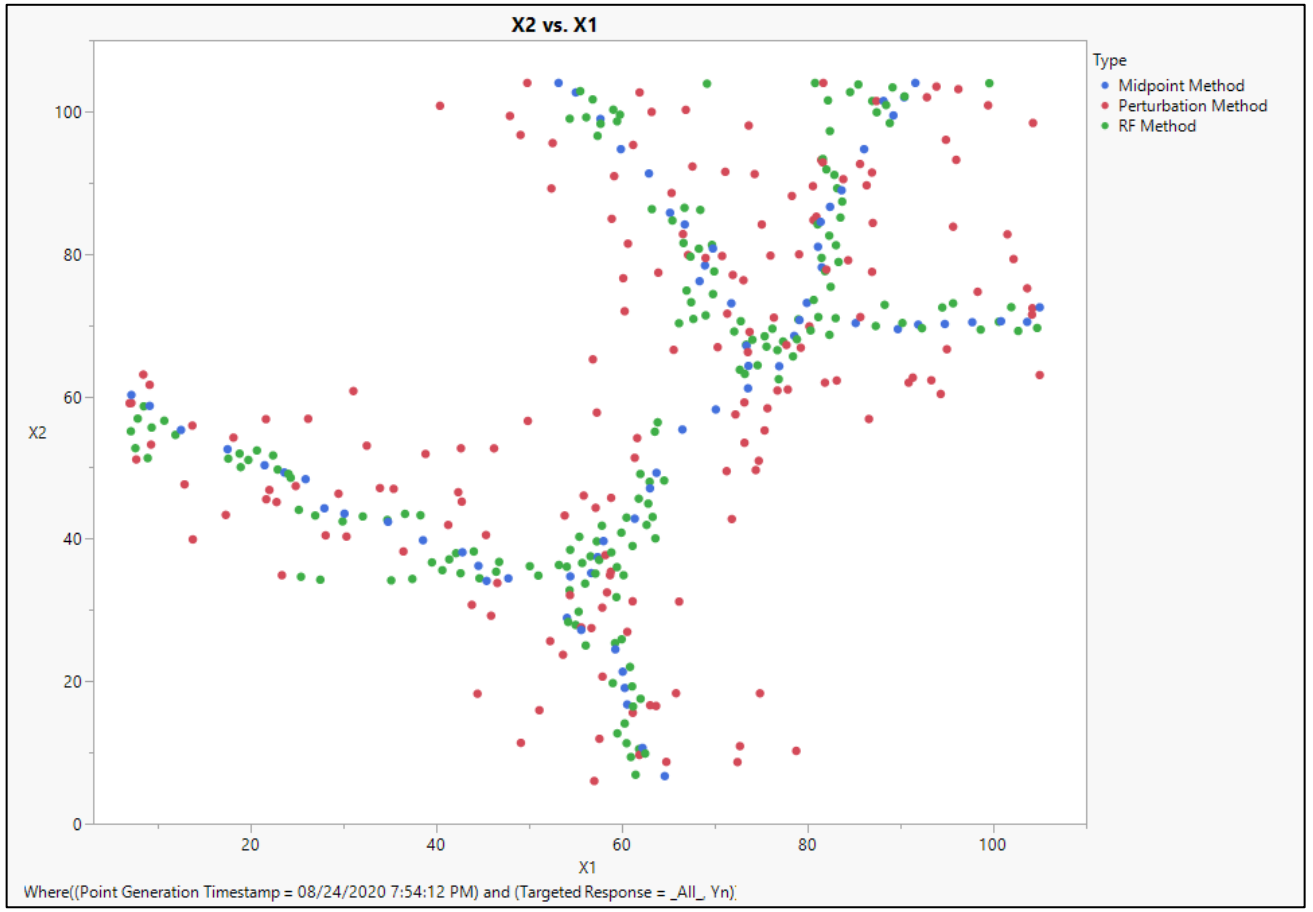
- The candidate runs with the largest predicted values for the indicator column are those that are closest to known boundary points and are most likely to also represent boundary points. These are sorted in decreasing order and the top runs are taken as **RF Method** runs.
- While the **midpoint** and **perturbation** approaches produce a fixed number of runs, the **RF** method can be used to generate an arbitrary number of points.
- The **RF** method considers all responses at once, so any areas that represent boundaries for multiple responses will receive a larger share of additional runs.



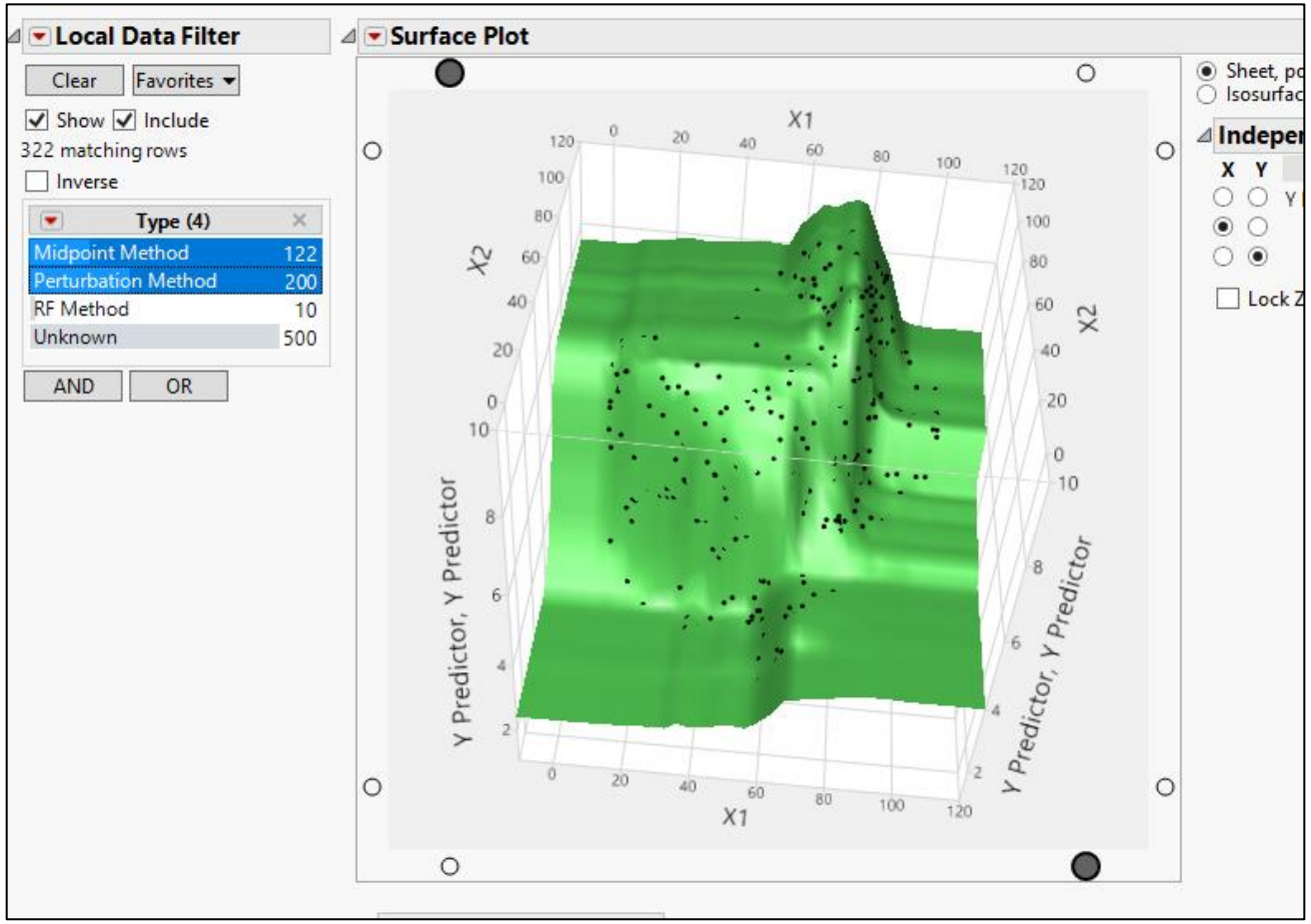
Boundary Explorer - Methodology

Final Set of New Runs

- **Blue:** Midpoint
- **Red**: Perturbation
- **Green**: Random Forest
- This represents the new runs added after one iteration of this tool. It is designed to work in multiple iterations.
- After collecting the response for these new runs, the tool is run again (using both the original space filling runs and the results from the first iteration).
- The wider spread of the perturbation points makes them useful for exploring along previously neglected boundaries.

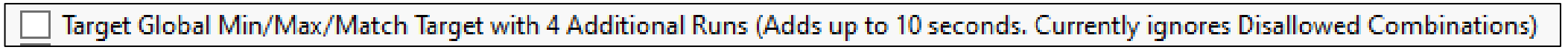


Steepest Gradients are Targeted with New Runs

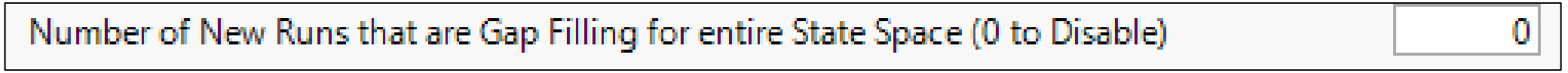


Other Options

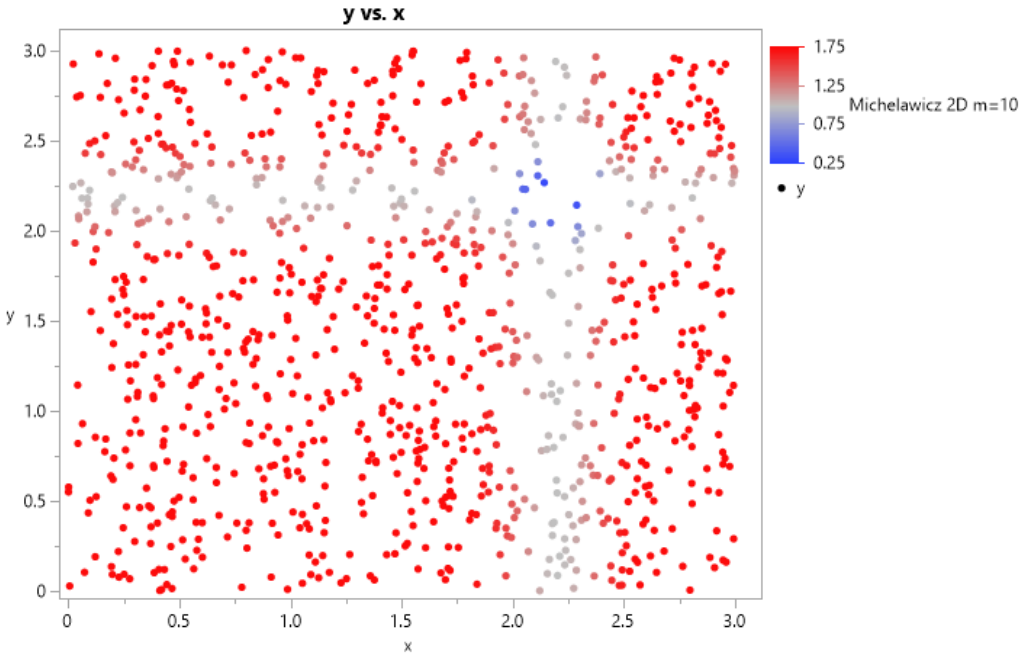
- Also an option to add runs near the minimum/maximum/match target (using column property) of continuous responses (fit with a random forest).

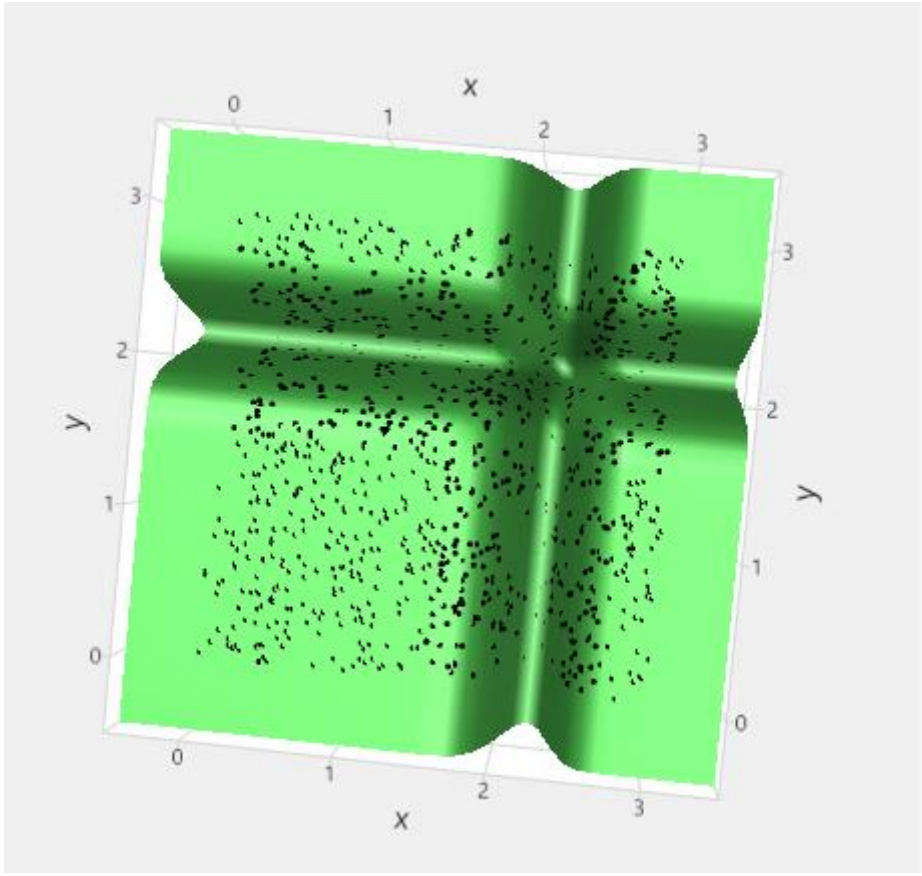


- Built in option to add Gap Filling runs

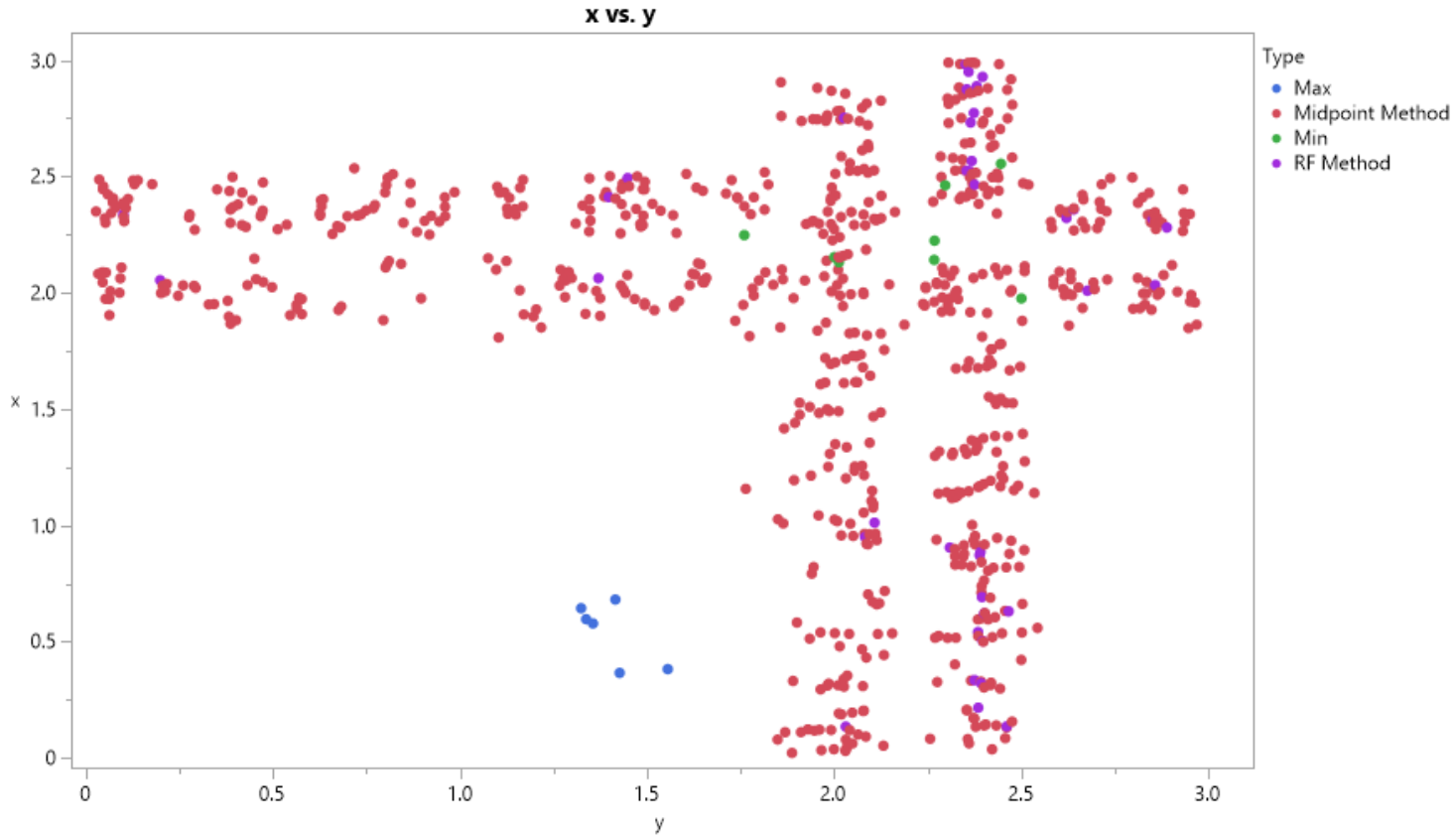


Michelawicz 2D: Initial Space Filling Runs



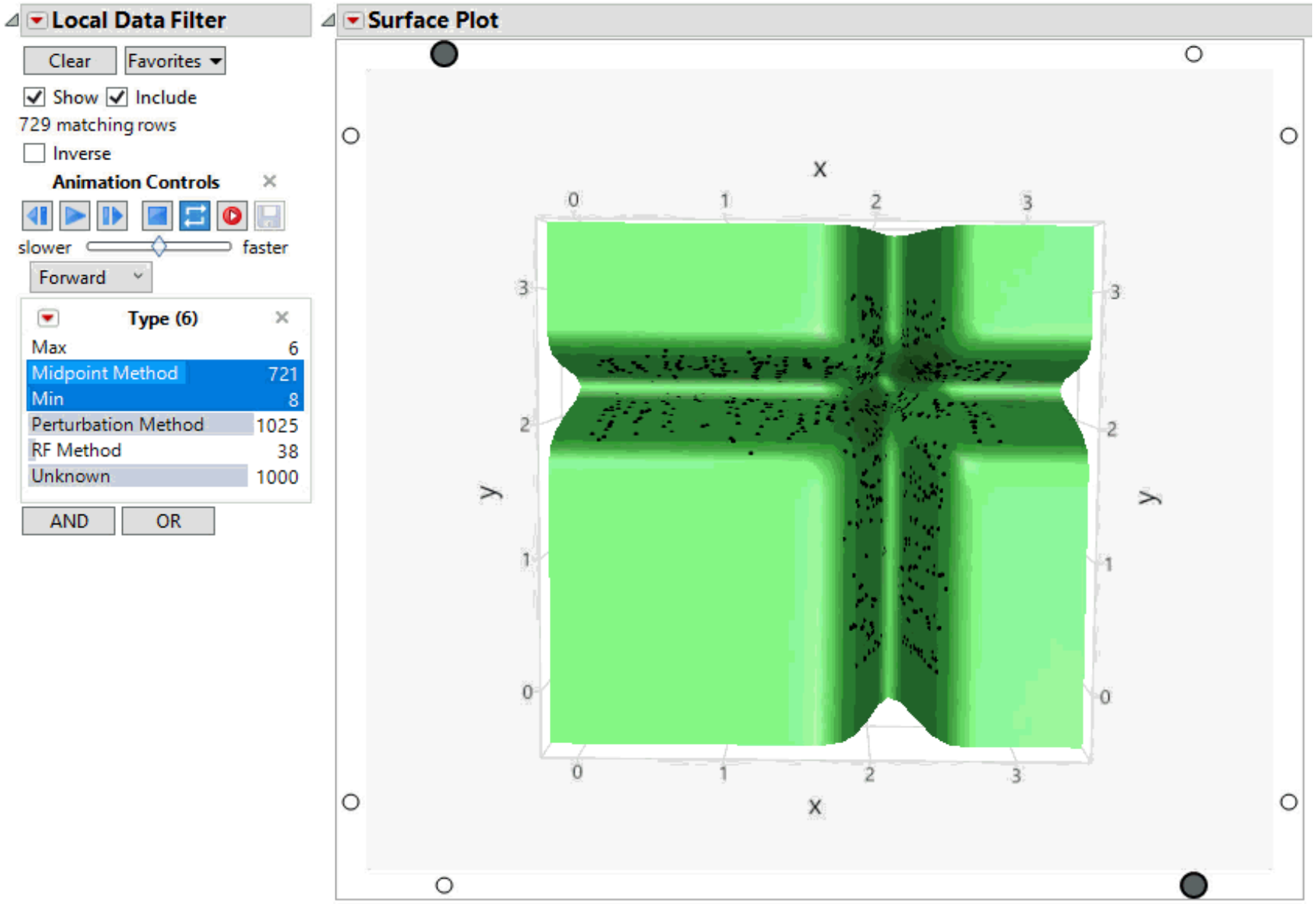


After Two Iterations



Where((Point Generation Timestamp = 08/27/2020 12:44:53 AM, 08/27/2020 12:46:56 AM) and (Targeted Response = _All_, Michelawicz 2D m=10) and (Type = Max, Midpoint Method, Min, RF Method))

After Two Iterations



Summary

- Two new design augmentation tools now available to practitioners
- Gap filling offers significant improvement over space filling to target sparse areas of design space
- Adaptive sampling critical in large scale simulation models and requires judicious choice of new runs
  - Highly recommend iteratively generating points
  - Though motivation came from complex simulation models with near-real time augmentation needed through automation, methods scale well across many design choices, factors, runs and applications.
- Key points
  - DOE is not just for physical systems
  - Design should account for expected surface complexity to support analyses
  - Think sequential experimentation
  - Augmented tests should be in areas of most interest
  - AI/ML is the enabler

Questions?