TS-ParamScape3D

User Manual (v2.0) - ParamScape3D with Single Line Collapse and Heatmap with Water Level

Plotter - From millions of runs to one clear map!

This manual was designed for the earlier version of TS-ParamScape3D, but the new version is just a clean up version and allows the reading of optimization text files with greater than 8000 combinations.

What "Single Line Collapse" means

When your optimization includes **more than two parameters**, multiple runs can land on the same **(X, Y)** address.
ParamScape3D compresses ("collapses") those hidden dimensions into a single value per address.

In this version, the collapse rule is:

- **Mean** of the selected metric at each address (the surface height)
- **Standard deviation** at each address (variability)
- Optional display as **Mean/Std** (signal-to-noise)

The new **2D Line (1 Param)** mode applies the same idea with one parameter:
for each **X** value, it plots the **mean** (line) and **±1σ** (band), and the tooltip reports **Mean / Std / N**.

###

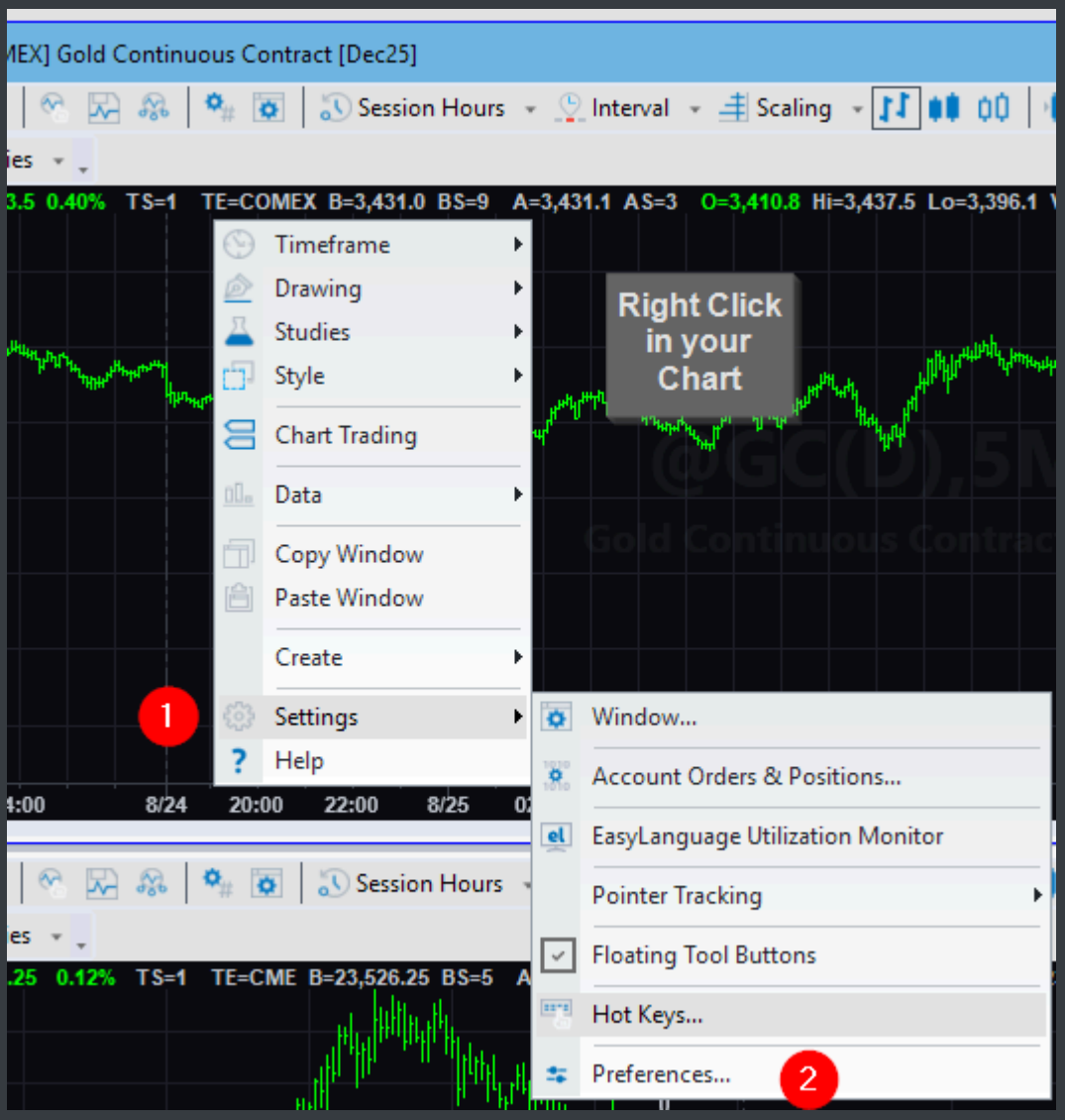
Link to TS-ParamScape3D Video Manual

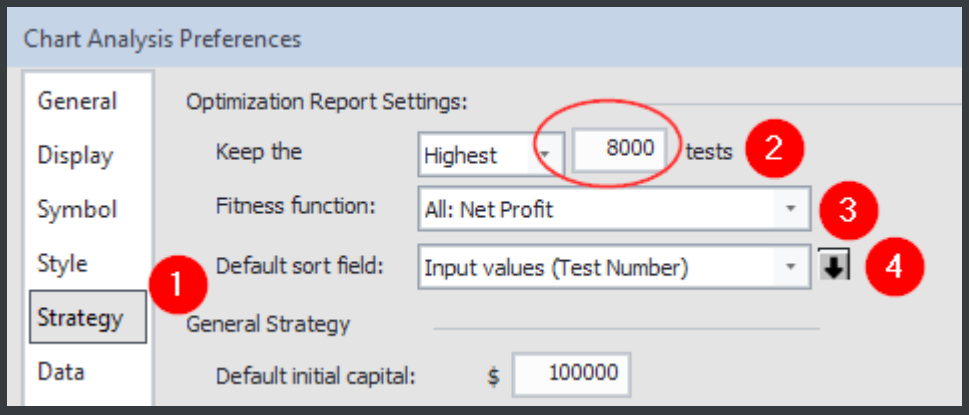
https://vimeo.com/1114322067?share=copy

Get the data out of TradeStation or MultiCharts

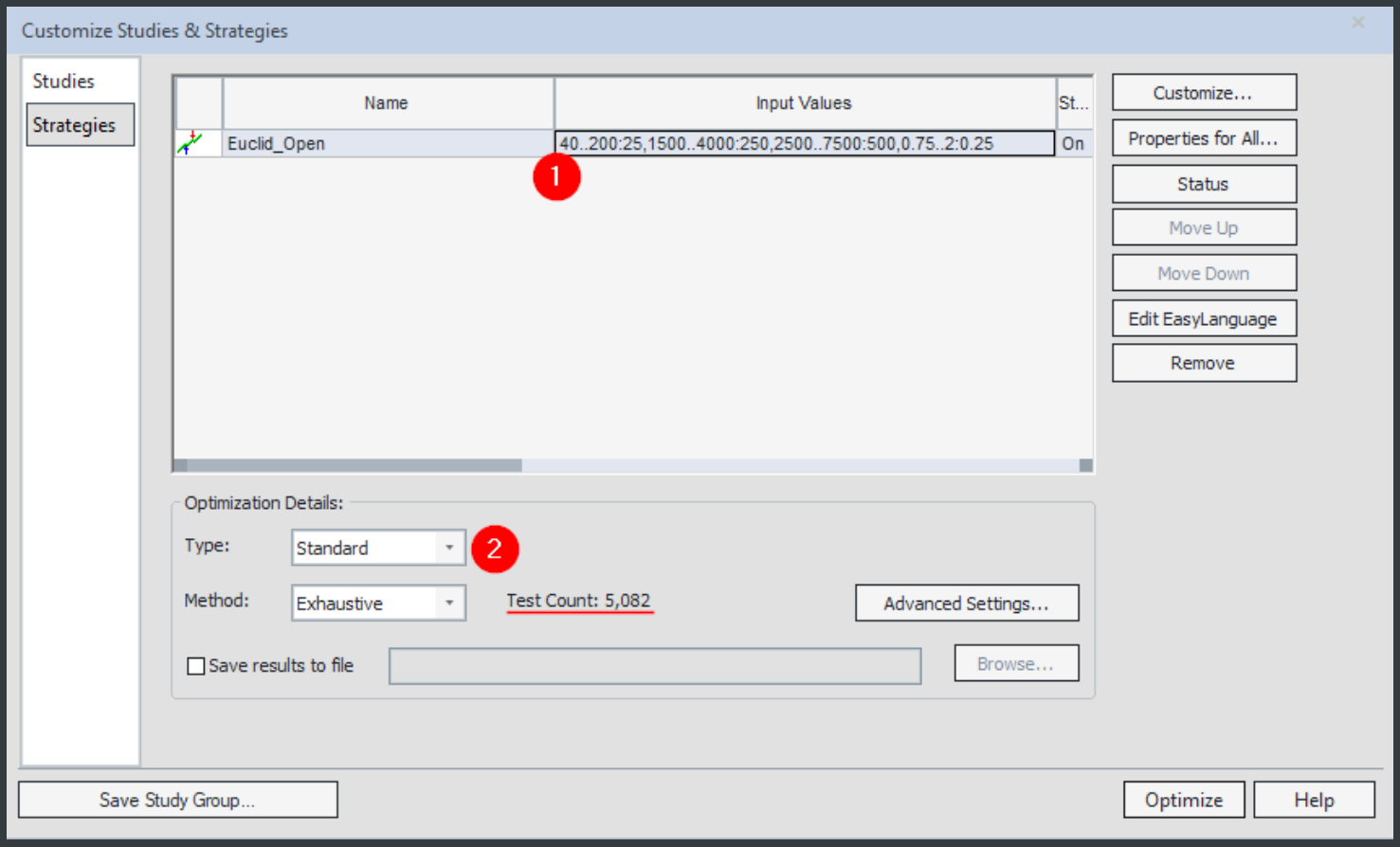
Before you can do anything with these tools you must get the data out of TradeStation. When you run an optimization across more than 3 parameters it is easy to get 10 of thousands of results. TradeStation can only keep track of the top 8000 permutations of the parameters (if you need to track more just email me and I can provide a utility). Once you create an optimization and open the report you can save it as a tab delimited text file. This text file is what TS-ParamScape-3D uses to create its plots.

Modify you settings to store 8ooo optimization runs





Make sure you set up your TradeStation or MultiChart Optimization Reports properly.



Run limit (and avoid holes) - note: new version can read many more combinations using the .csv output module I now include. Instructions follow below.

TradeStation stores max ~8,000 optimization results.

Your total runs = *(levels of Param1)* × *(levels of Param2)* × … × *(levels of ParamK)*.

If you exceed 8,000, TS-ParamScape-3D will have sparse coverage and **surface plots may show holes** (missing X–Y cells).

Rules of thumb

- **Prioritize X & Y density.** Keep a full grid for X and Y so every cell has data.
- **Trim the others.** Reduce the number of levels on Z/Size/Slice parameters first.
- **Coarse→refine.** Run a coarse scan to find a good region, then narrow ranges and rerun inside it.
- **Math check:** multiply your chosen levels; keep the product ≤ **8,000**.

Equal-level guidance (max levels per parameter)

- **3 params:** up to **20×20×20 = 8,000**
- **4 params:** up to **9×9×9×9 = 6,561** (10×10×10×10 = 10,000 ❌)
- **5 params:** up to **6×6×6×6×6 = 7,776** (7^5 = 16,807 ❌)
- **6 params:** up to **4×4×4×4×4×4 = 4,096** (5^6 = 15,625 ❌)

Handy exact combos (hit 8,000 exactly)

- **4 params: 10×10×10×8 = 8,000** or **16×10×10×5 = 8,000**
- **5 params: 10×10×10×4×2 = 8,000**
- **6 params: 5×5×5×4×4×4 = 8,000**

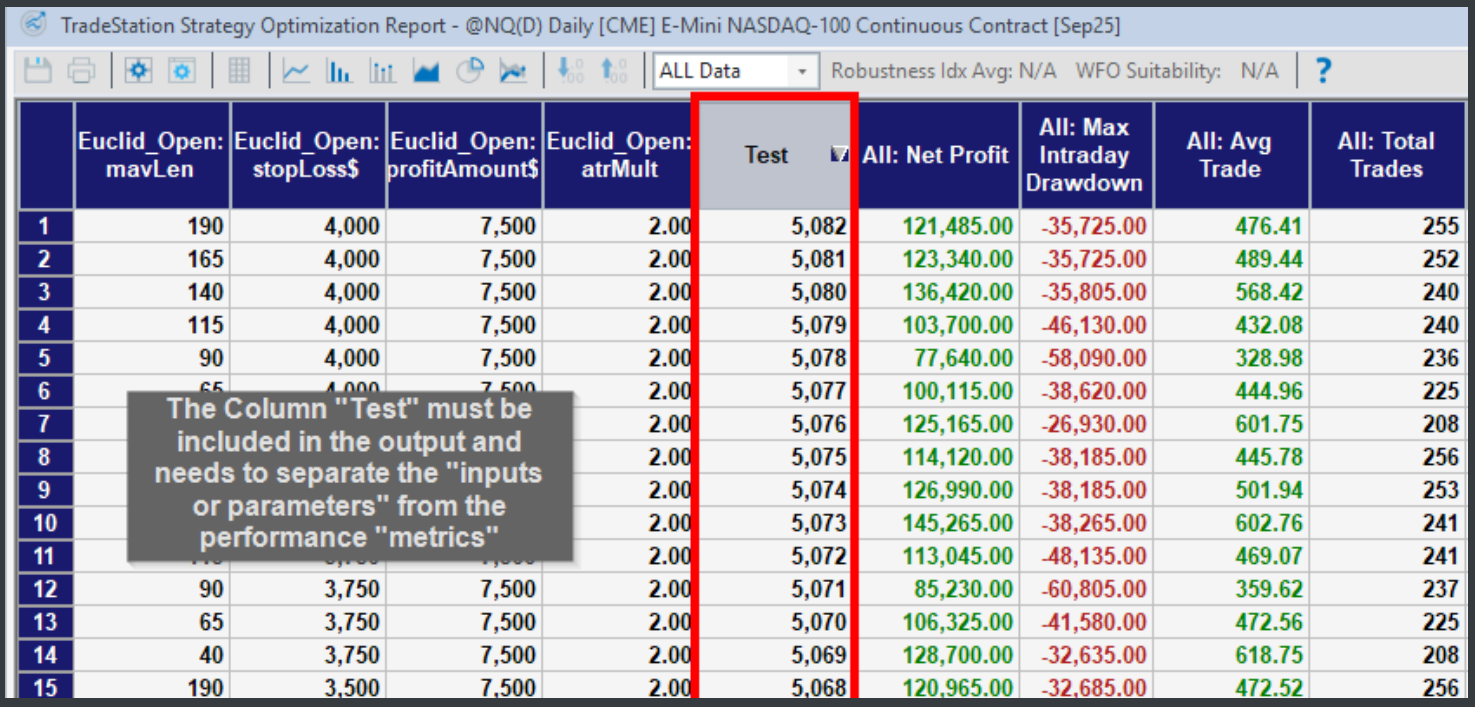
Practical presets

- **If optimizing 4 params:** aim for ~**9–10** levels on the first three, and **8 or fewer** on the fourth.
- **If optimizing 5–6 params:** target **6 levels** each (5–6 mix), or reduce one/two params to **4 levels** and keep X/Y higher.

> **Tip to avoid holes:** Ensure every **(X,Y)** pair appears at least once. If you must cut levels, cut them on parameters **other than X and Y**.

Modify you TradeStation Optimization

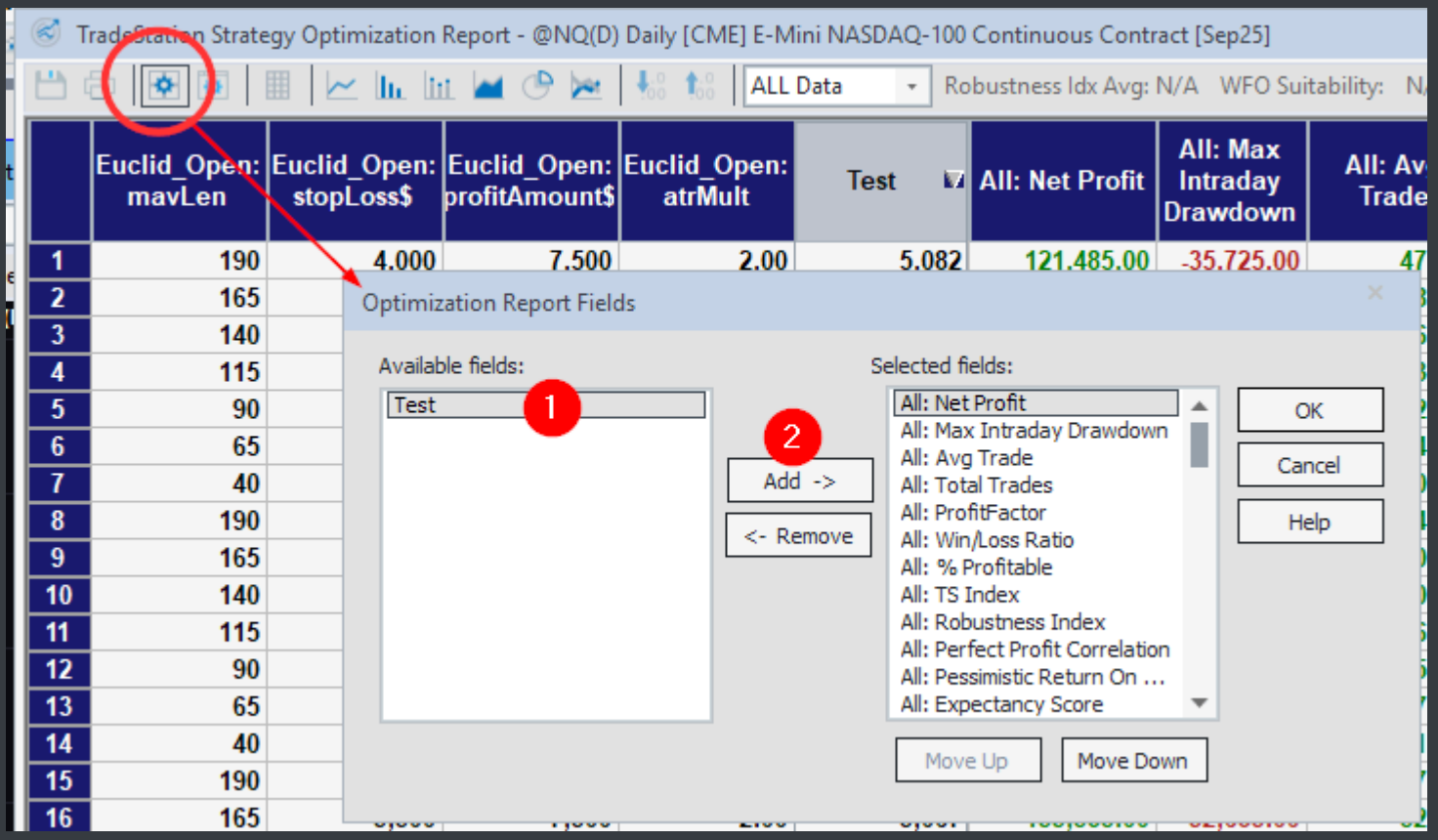
When you save the TradeStation optimization report the output is saved in a file that is Tab delimited (TSV). TS-ParamScape3D was specifically designed to read this format. In most tools, the software will output Comma delimited format. TS-ParamScape3D automatically scans the headers in the output and finds the column with the heading "Test". This column is used to separate the inputs or parameters from the performance metrics. It is imperative that is is located in the proper location. Just like the following.



	Euclid_Open: mavLen	Euclid_Open: stopLoss$	Euclid_Open: profitAmount$	Euclid_Open: atrMult	Test	All: Net Profit	All: Max Intraday Drawdown	All: Avg Trade	All: Total Trades
1	190	4,000	7,500	2.00	5,082	121,485.00	-35,725.00	476.41	255
2	165	4,000	7,500	2.00	5,081	123,340.00	-35,725.00	489.44	252
3	140	4,000	7,500	2.00	5,080	136,420.00	-35,805.00	568.42	240
4	115	4,000	7,500	2.00	5,079	103,700.00	-46,130.00	432.08	240
5	90	4,000	7,500	2.00	5,078	77,640.00	-58,090.00	328.98	236
6				2.00	5,077	100,115.00	-38,620.00	444.96	225
7				2.00	5,076	125,165.00	-26,930.00	601.75	208
8				2.00	5,075	114,120.00	-38,185.00	445.78	256
9				2.00	5,074	126,990.00	-38,185.00	501.94	253
10				2.00	5,073	145,265.00	-38,265.00	602.76	241
11				2.00	5,072	113,045.00	-48,135.00	469.07	241
12	90	3,750	7,500	2.00	5,071	85,230.00	-60,805.00	359.62	237
13	65	3,750	7,500	2.00	5,070	106,325.00	-41,580.00	472.56	225
14	40	3,750	7,500	2.00	5,069	128,700.00	-32,635.00	618.75	208
15	190	3,500	7,500	2.00	5,068	120,965.00	-32,685.00	472.52	256

Making Sure Test is in the output and is in the correct location

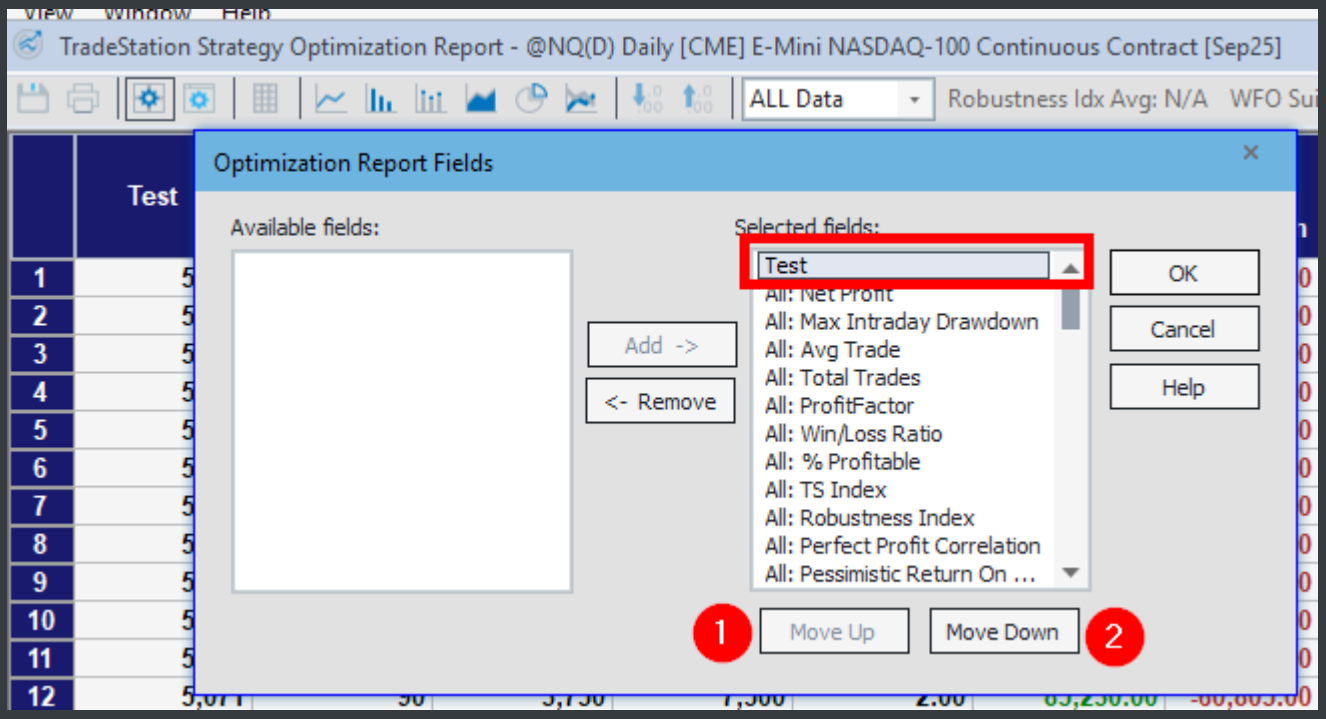
If you don't see **Test**, then click on this (Customize Fields) Icon in the window header.



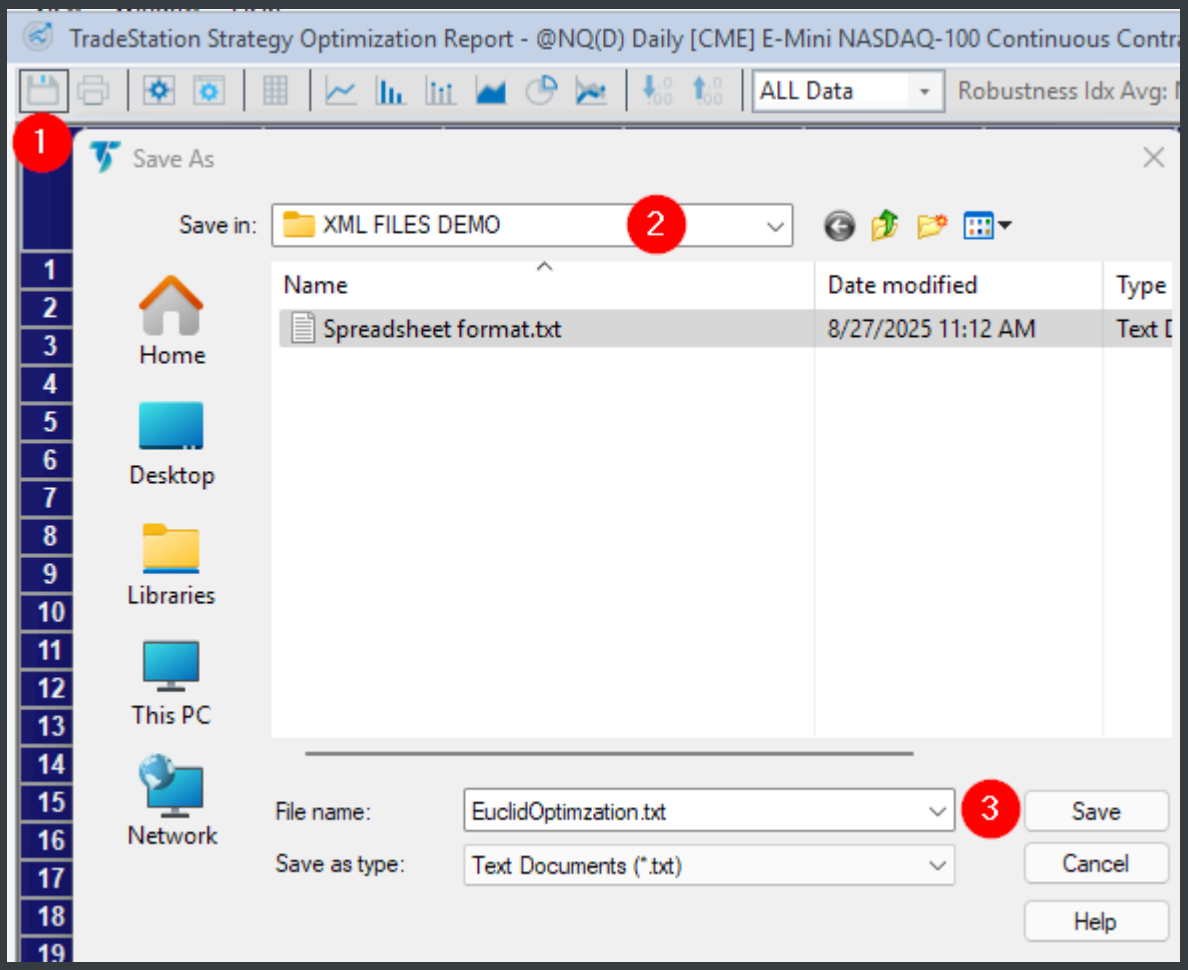
If you see Test, but it is not in the right location you can either drag it into the right location - right after the last Input name.

	Test	Euclid_Open: mavLen	Euclid_Open: stopLoss$	Euclid_Open: profitAmount$	Euclid_Open: atrMult	All: Net Profit	All: Max Intraday Drawdown	All: Avg Trade	All: Total Trades	All: ProfitFactor	All: Win/Loss Ratio	All: % Profitable	All: TS Index
1						121,485.00	-35,725.00	476.41	255	1.32	1.16	53.33	462.48
2	5,081	165	4,000	7,500	2.00	123,340.00	-35,725.00	489.44	252	1.33	1.13	53.97	469.54
3	5,080	140	4,000	7,500	2.00	136,420.00	-35,805.00	568.42	240	1.39	1.15	54.58	499.12
4	5,079	115	4,000	7,500	2.00	103,700.00	-46,130.00	432.08	240	1.29	1.18	52.08	281.00

Or you can move it up or down with the Customize Fields button. Just make sure it is at the top of the Selected Fields list.



Save the report to a convenient location



The TradeStation 8,000-Row Issue

TradeStation standard optimization reports are useful, but they may not represent the full search space when the optimization is large.

The practical problem is this: a standard report can be limited to the top results. That means the exported file may be biased toward the winners rather than showing the entire optimization landscape.

This matters because ParamScapeRank is strongest when it can see the whole search space. If the import only contains the top rows, the app can still rank and suggest ranges, but the analysis is based on a filtered view of the optimization.

When the standard report is fine

Use the standard tab-delimited report when:

- the full optimization is under the export limit,

- you are running a focused second pass,
- or you only need a quick read on a smaller optimization.

When the full CSV export is preferred

Use the full ParamScape CSV export when:

- the optimization exceeds the standard report limit,
- you need every test row,
- you want a less biased view of the search space,
- or you plan to use ParamScapeRank to reduce a very large optimization into a focused follow-up pass.

ParamScape Full CSV Export

The ParamScape full-export CSV is a custom export format designed for large optimizations. If you want to report on more than 8000 combinations you will have to put a small hook into your strategy code. You can put this near the top of the code to help you name parameters. Before inserting this code, make sure you have created a folder to hold the exported data. In this example I have created a folder named "C:\TS-OptimizationTextFiles" and a file named "VWMSystem4.txt." You will want to use an appropriate file name.

The example shows the existing strategy inputs. Somewhere after this you should insert this snippet of code. Here I am storing the inputs into array elements. Don't worry if you don't understand this, but do make sure you align the proper parameters[ ] with the corresponding input names. Also you will need to manually delete the file)("VWMSystem4.txt") each time you run a fresh optimization or it will simply be appended.

```
//Existing Strategy Inputs:
Inputs: Price(Close), MomLen(5), AvgLen(20), ATRLen(5), ATRPcnt(.25), SetupLen(5);

arrays: parameters[10](-9999.00);
vars: lineOutput("");
parameters[1] = MomLen;
parameters[2] = AvgLen;
parameters[3] = ATRLen;
parameters[4] = ATRPcnt;
parameters[5] = SetupLen;
condition1 = CalcPerformanceMetrics(parameters,"C:\TS-
OptimizationTextFiles\VWMSystem4.txt",100000);
```

The **CalcPerformanceMetrics** requires just three paramters:

1. the parameters array after you have populated it with the corresponding input names
2. the directory and filename of the file that will contain the data incased in " "
3. a dollar value such as $100,000 to help facilitate the sharpe and other ratios

It is created from an EasyLanguage function called:

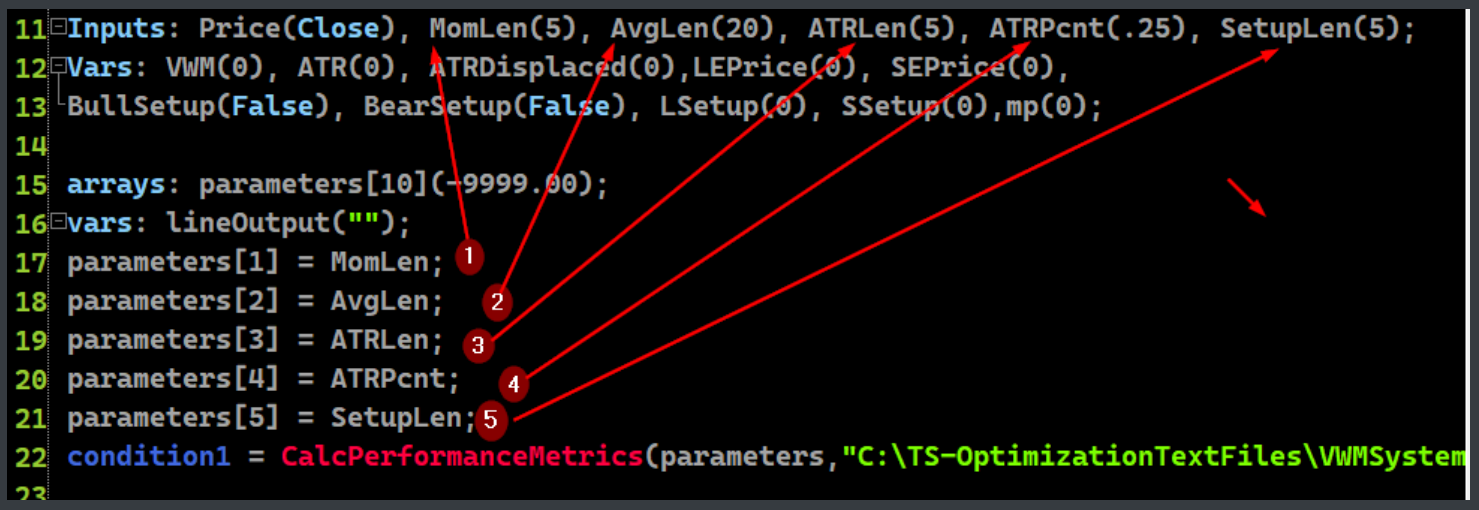
```
CalcPerformanceMetrics
```

This format is:

- comma-delimited,
- headerless,
- numeric,
- and designed to write every optimization row.

Because the file has no headers, TS-ParamScapeRank opens a mapping dialog after detecting the active parameter slots.

Example EasyLanguage setup



A typical strategy might define a parameter array like this:

```
arrays: parameters[10](-9999.00);

parameters[1] = lookBack;
parameters[2] = mult;
parameters[3] = maxAtrRisk;
parameters[4] = atrLenCal;

condition1 = CalcPerformanceMetrics(parameters, "C:\OptimizationResults\Example.txt",
100000);
```

In this example:

- `parameters[1]` contains `lookBack`,
- `parameters[2]` contains `mult`,
- `parameters[3]` contains `maxAtrRisk`,
- `parameters[4]` contains `atrLenCal`,
- unused slots remain at `-9999.00`.

The `-9999.00` value tells TS-ParamScapeRank that the slot was not used.

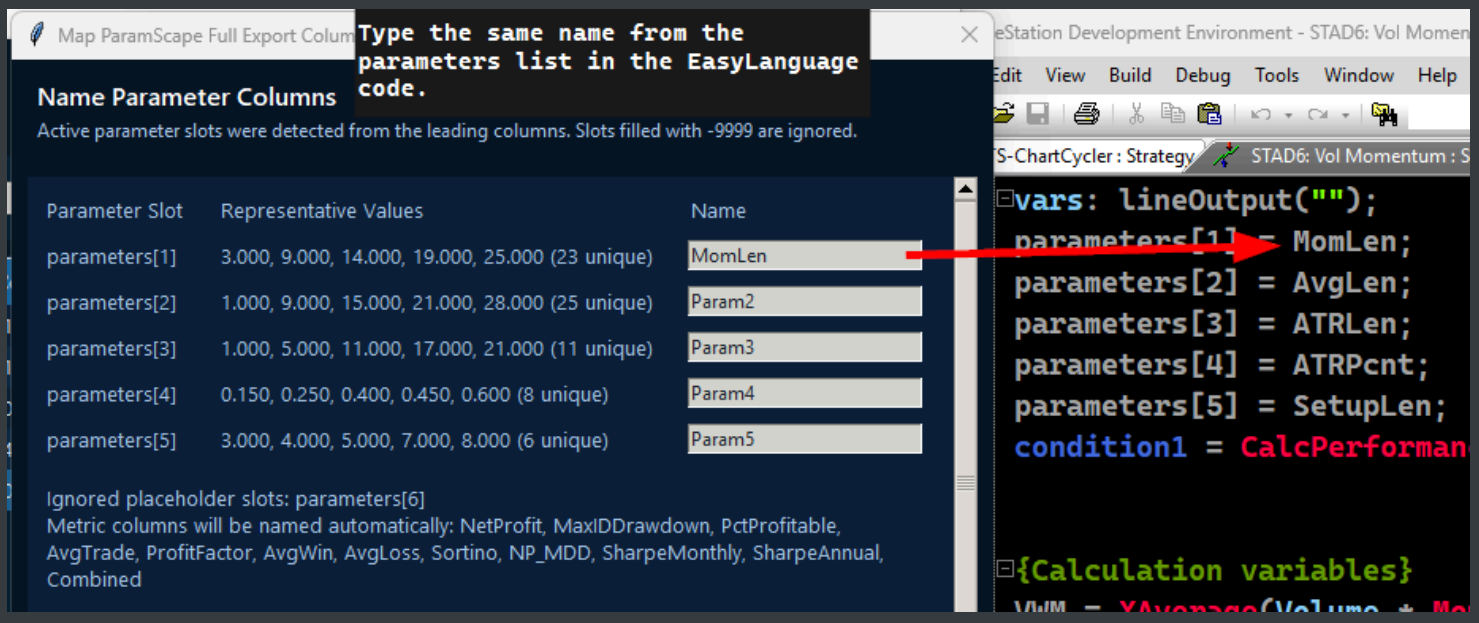
Mapping a Headerless Full CSV File

When TS-ParamScapeRank detects a headerless full-export CSV, it attempts to identify the active parameter slots automatically.

The app:

1. reads the numeric comma-delimited file,
2. detects leading parameter slots,
3. ignores slots filled only with `-9999.00`,
4. separates active parameter columns from metric columns,
5. opens a parameter-name mapping dialog,
6. and automatically names the performance metric columns.

Representative values



The mapping dialog shows representative values from across the full file, not just the first few rows. This helps you identify which slot corresponds to which strategy input.

For example, a slot might show:

```
20, 40, 60, 80, 100 (5 unique)
```

That might help you recognize it as `lookBack` or `mavLen`.

If a column is constant, the dialog marks it as constant. Constant columns may still be active, but they should be interpreted carefully because they did not vary during the optimization.

Naming parameters

Use the exact names of the parameters at at least use clear names such as:

```
lookBack
mult
maxAtrRisk
atrLenCal
```

Avoid duplicate names. Also avoid using metric names such as `NetProfit` or `ProfitFactor` as parameter names.

Automatically named metric columns

The full-export metric columns are automatically named:

```
NetProfit
MaxIDDrawdown
PctProfitable
AvgTrade
ProfitFactor
AvgWin
AvgLoss
Sortino
NP_MDD
SharpeMonthly
SharpeAnnual
Combined
```

If the full-export file contains more metric columns than the standard list, extra columns may be named generically as `Metric1`, `Metric2`, and so on.

TS-ParamScape3D [PLOTTER]

Using Plotly Plots

The Plotly tools now include three interactive views:

- **Plot with Plotly (Surface)** - interactive 3D contour/surface chart.
- **Plotly Heatmap** - a flattened 2D version of the same selected X/Y/Z relationship.
- **Volumetric Scatter** - full point-cloud view of the loaded optimization data.

The surface and heatmap views also include a **Water level** slider. This lets you raise or lower a threshold so the strongest regions stand out visually.

When you select **“Plot with Plotly (Surface)”** or **“Volumetric Scatter”**, TS-ParamScape3D will generate an **interactive chart** using the Plotly library.

Important: Browser Window Launch

- **What happens:** As soon as the plot is created, your **default web browser** (Chrome, Edge, Firefox, etc.) will automatically open a new tab or window.
- **Why:** Plotly runs in the browser so you can zoom, rotate, pan, and hover over data points for detailed inspection.
- **What to expect:** If your browser was already open, you’ll see a new tab appear. If no browser was open, one will launch automatically.

What You Can Do in the Browser

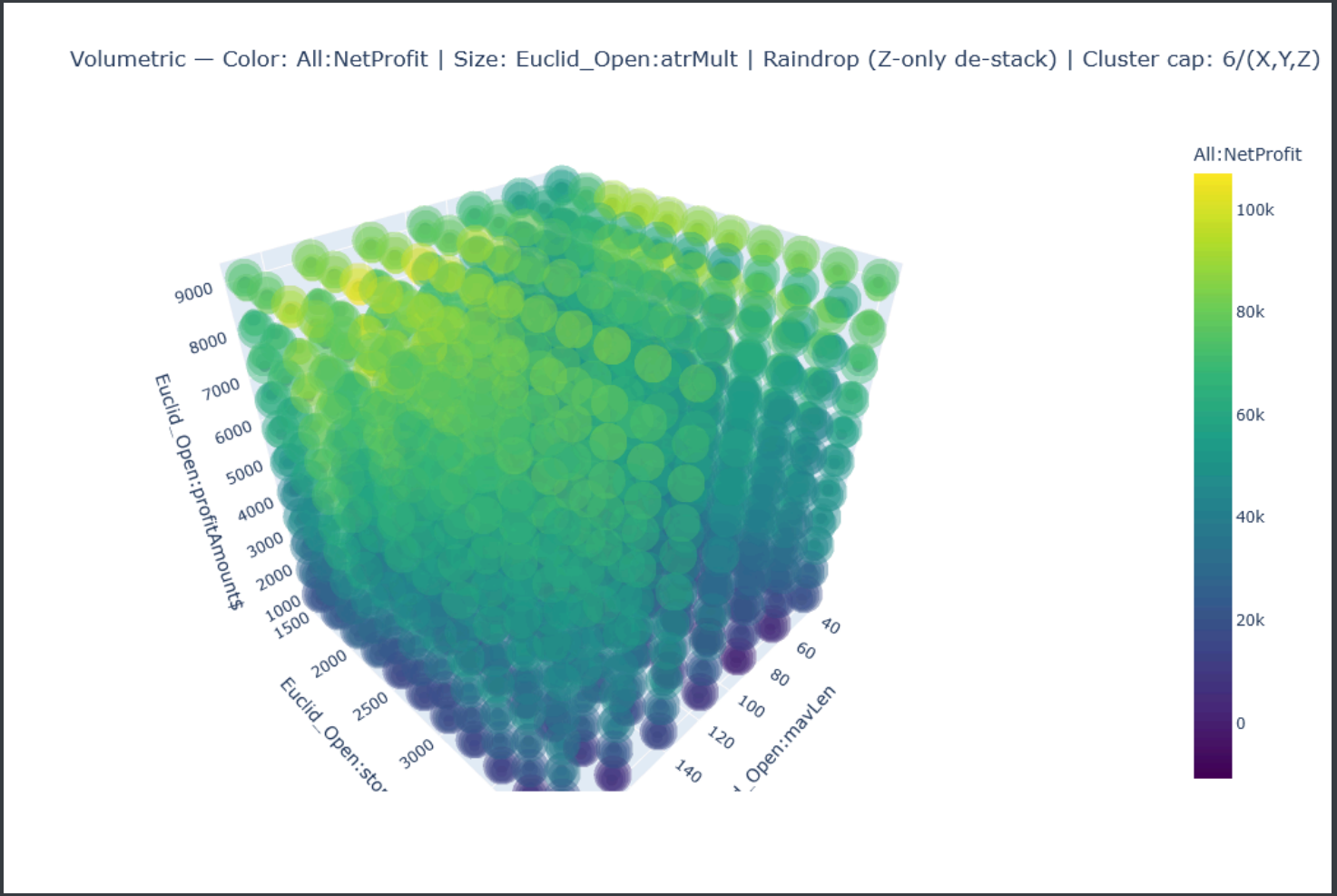
- Rotate the 3D chart with your mouse.
- Zoom in/out with your scroll wheel.
- Hover to see tooltips with exact parameter/metric values.
- Save or screenshot directly from the browser if needed.

Tips & Warnings

- Be prepared for your browser to “pop up” — it’s normal behavior, not an error.
- Some antivirus or firewall software may briefly flag the automatic browser launch; just allow it, as

no data is sent online. The file is rendered **locally** on your machine.
- If nothing happens, check that your system has a default browser set.

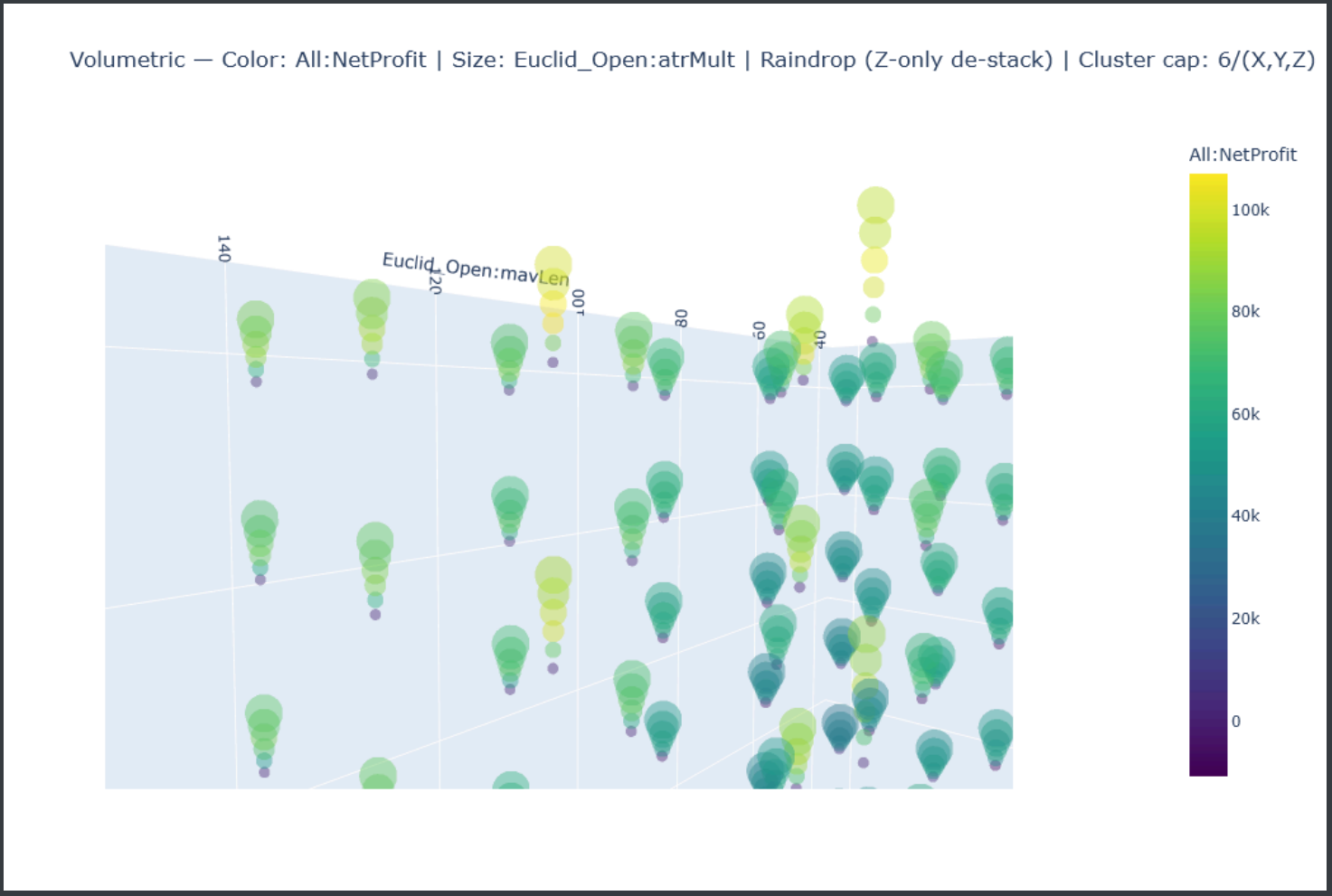
This guide walks you through **TS-ParamScape-3D's controls and workflow** so you can load data, choose views, and interpret the charts with confidence. We'll start with the **Volumetric view (Plotly only)**, which plots every optimization result on a 3D grid and reveals stacks when multiple parameter combinations land on the same (X, Y). Then we'll cover the **Surface view**, which **aggregates all results at each (X, Y)**—ideal when more than two parameters are optimized and you want one value per location.



How four parameters are encoded in one 3D view

- **X-axis — Moving Average Length (days):** left→right = shorter → longer averages.
- **Y-axis — Stop Loss ($):** front→back = tighter → looser stops.
- **Z-axis — Profit Objective ($):** bottom→top = smaller → larger profit targets.
- **Encoding (Size) — ATR Multiplier:** marker **size** scales with the Average True Range multiplier (bigger dot = larger ATR multiple).

Zoom in to see that each point is in fact a cluster of different values.

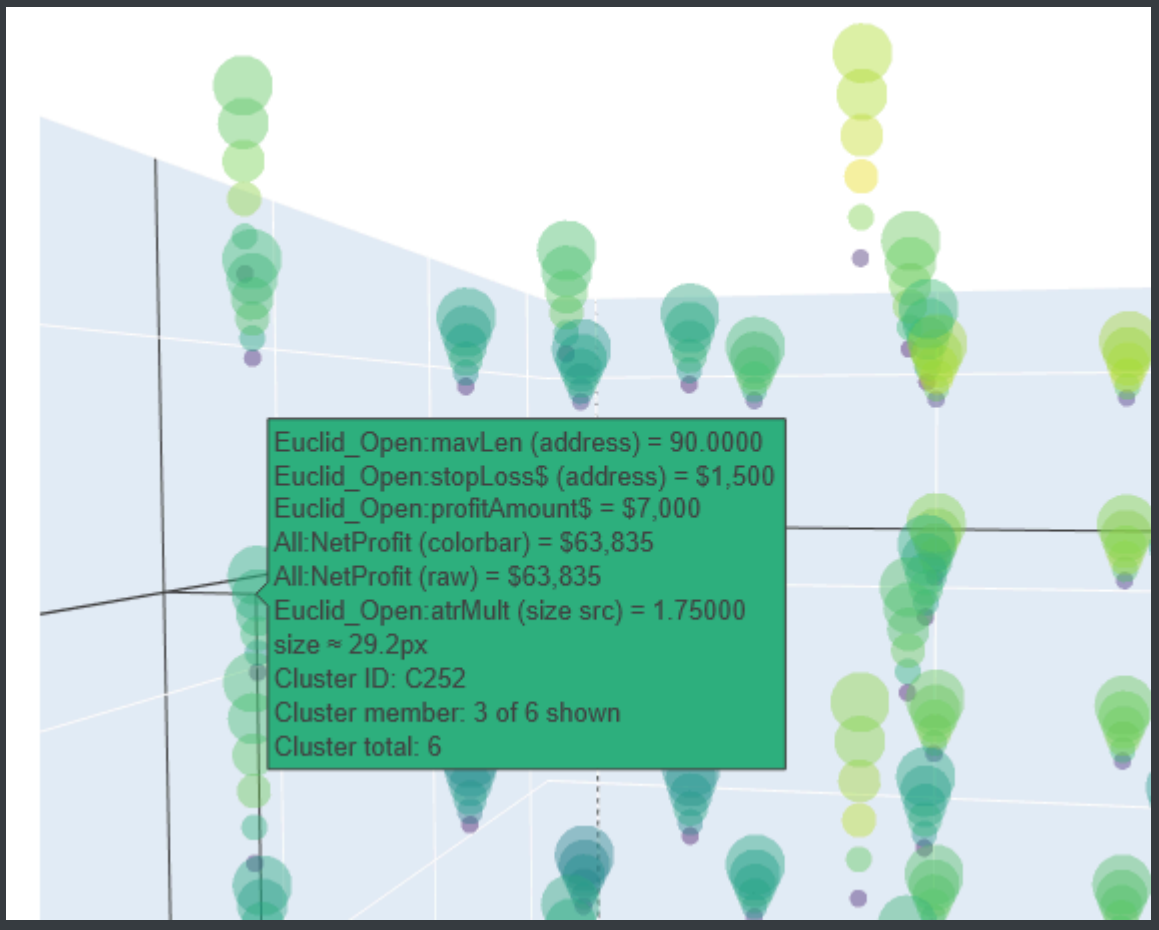


How to read the Volumetric Plot

- Each dot is one backtest run at `(MA length, Stop loss, Profit objective, ATR multiple)`.
- Turn on **De-stack (Z-only "raindrop")** to separate overlapping results that share the same X and Y—clusters become vertical "inverted raindrops." These multiple values are "jittered" by small offsets of the Z axis.
- **Hover** any dot to see the exact (non-jittered) values.

What does the volumetric de-stacked chart show you?

You can zoom into the cloud and find the values that show the highest **NetProfit** and examine those values across the fourth parameter - in this case the average true range multiplier.



Selected cluster: six runs at the same (X, Y, Z) address.

X = *mavLen* (days), Y = *stopLoss $*, Z = *profitAmount $*. Color encodes **Net Profit**; marker **size** encodes **atrMult**. Points that share the same (X, Y, Z) are separated with a tiny **Z-only** offset; within this constant-Z stack the offsets follow **atrMult** (small → large), creating a mini "raindrop/pyramid"—a chart within the chart.

This cluster lets me know that at a moving average of length 90, a stop loss of $1,500 and profit objective of $7,000 and an average true range multiplier of 1.75 yields over $63K in profit. In almost all instances, a smaller average true range multiplier relates to a smaller net profit? This leads me to believe that an optimization over a larger set of multipliers might provide better results. Right now, other than the bottom plot of the stack, the values along the other plots in the stack are similarly colored. This means that the parameters at this particular X and Y and Z address demonstrates a higher level of robustness. You have to play with this to really get a feel for how it works.

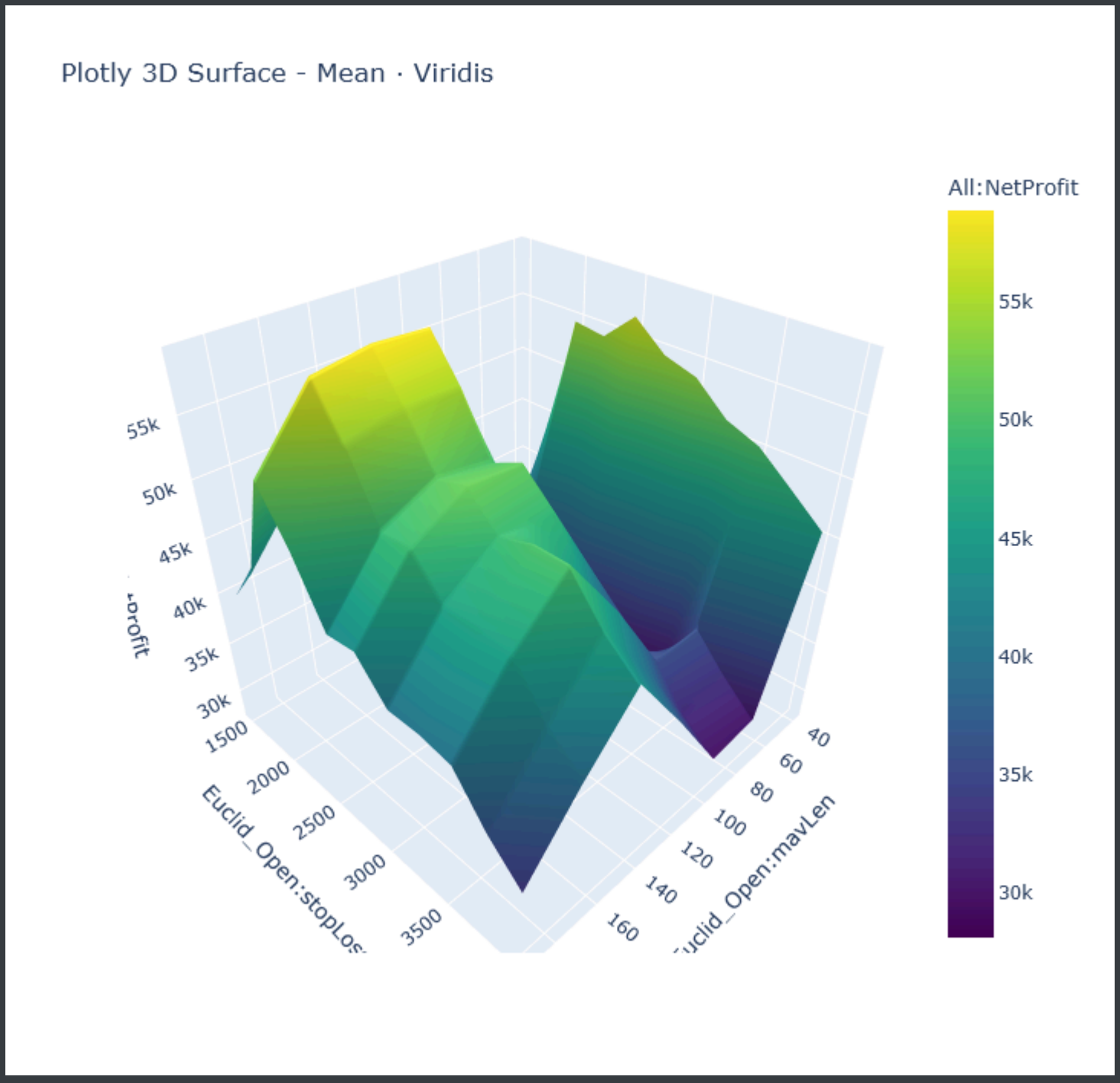
Creating 3D Surface plots with more than two parameters.

Plotting just the results of just two parameters is relatively straight forward. However, plotting optimzations of three or more is quite a bit more complicated. With this tool, we can plot two of N parameters against several objective functions. Below is a chart of moving average length versus stop loss dollars. The Z-Axis this time is the mean or average of all the addresses the contain the X and Y parameters. Take a look at this table.

Test	Mav Len	StopLoss$	Profit $	ATRMult	Net PnL	Max DD.	
1	40	1500	2500	0.75	6325	-4840	
2	40	1500	3000	0.75	5790	-6210	
3	40	1500	3500	0.75	7790	-6210	
4	40	1500	4000	0.75	4290	-6210	
5	40	1500	4500	0.75	5790	-6210	
6	40	1500	5000	0.75	7290	-6210	
7	40	1500	5500	0.75	5835	-6210	
8	40	1500	6000	0.75	6835	-6210	
9	40	1500	6500	0.75	7835	-6210	
…							
…							
62	40	1500	5500	2	38555	-8720	
63	40	1500	6000	2	43555	-8720	
64	40	1500	6500	2	47340	-8720	
65	40	1500	7000	2	51265	-8720	Average
66	40	1500	7500	2	55265	-8720	**34608.8**

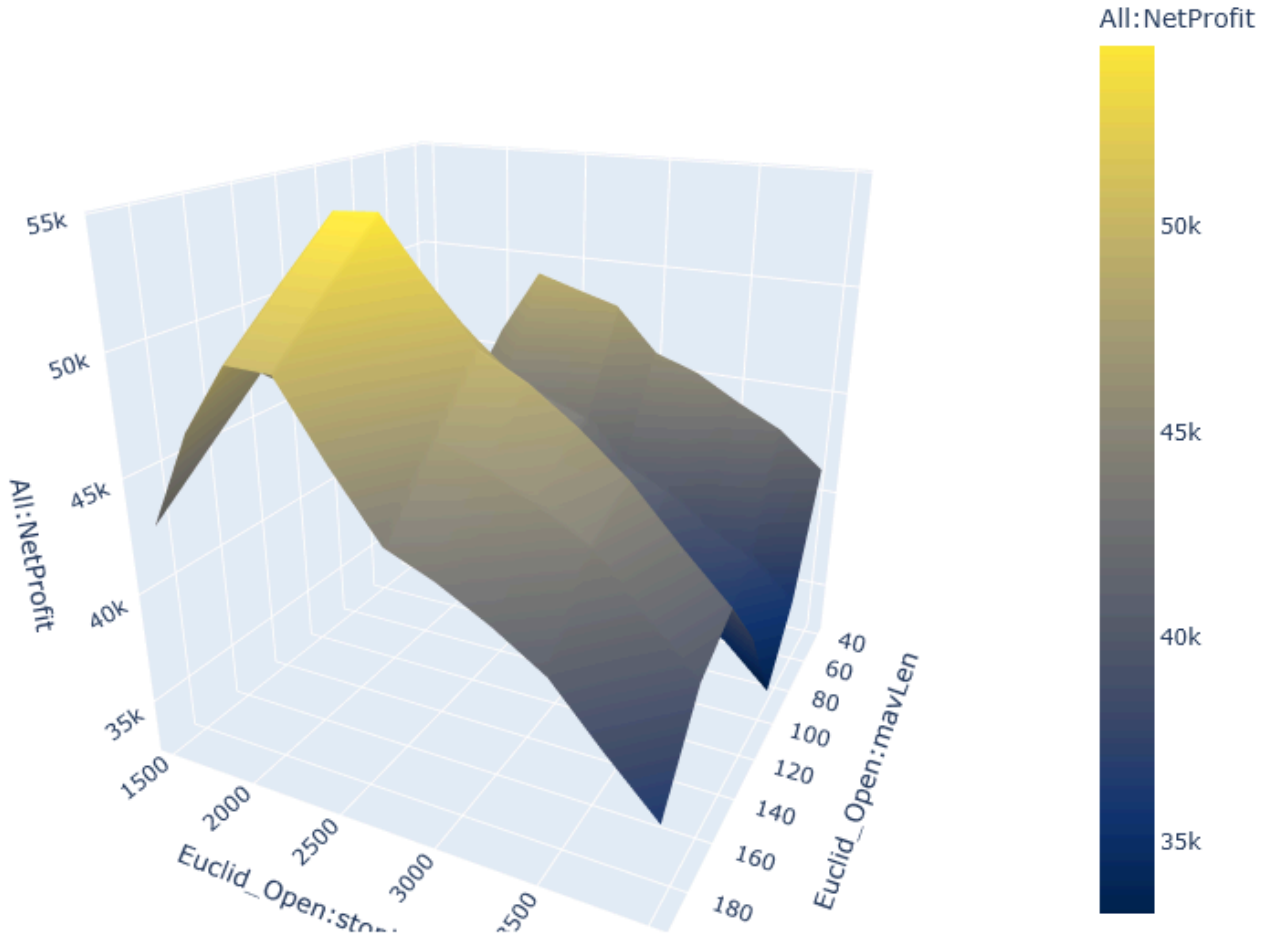
Across **66 parameter combinations** with **Mav Len fixed at 40** and **StopLoss$ fixed at 1500**, the average result represents the overall performance of this setting. This average reflects how the strategy behaves under different values of **Profit$** and **ATRMult**, giving a broader sense of robustness. If you locate the point X = 40 and Y = 1500 on the figure you will see the 34608 **mean** value. If a point

on the surface is high and surrounded by similar neighbors, then you know you may have found a robust combination set for X and Y.

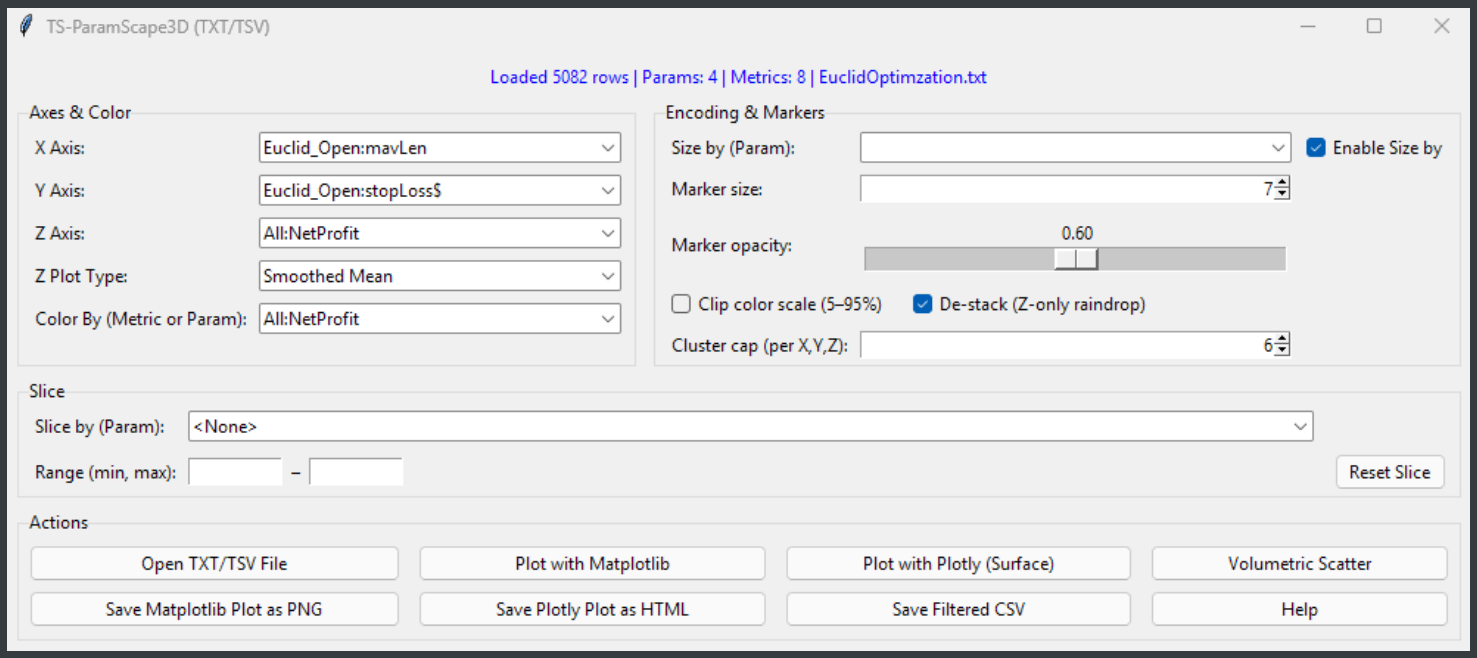


You can further smooth the surface by averaging each point with its **3 × 3 neighborhood**. This reduces noise and produces a smoother contour. Sharp or brittle spikes in the mean surface will still stand out, while broad high regions will appear even more clearly as plateaus.

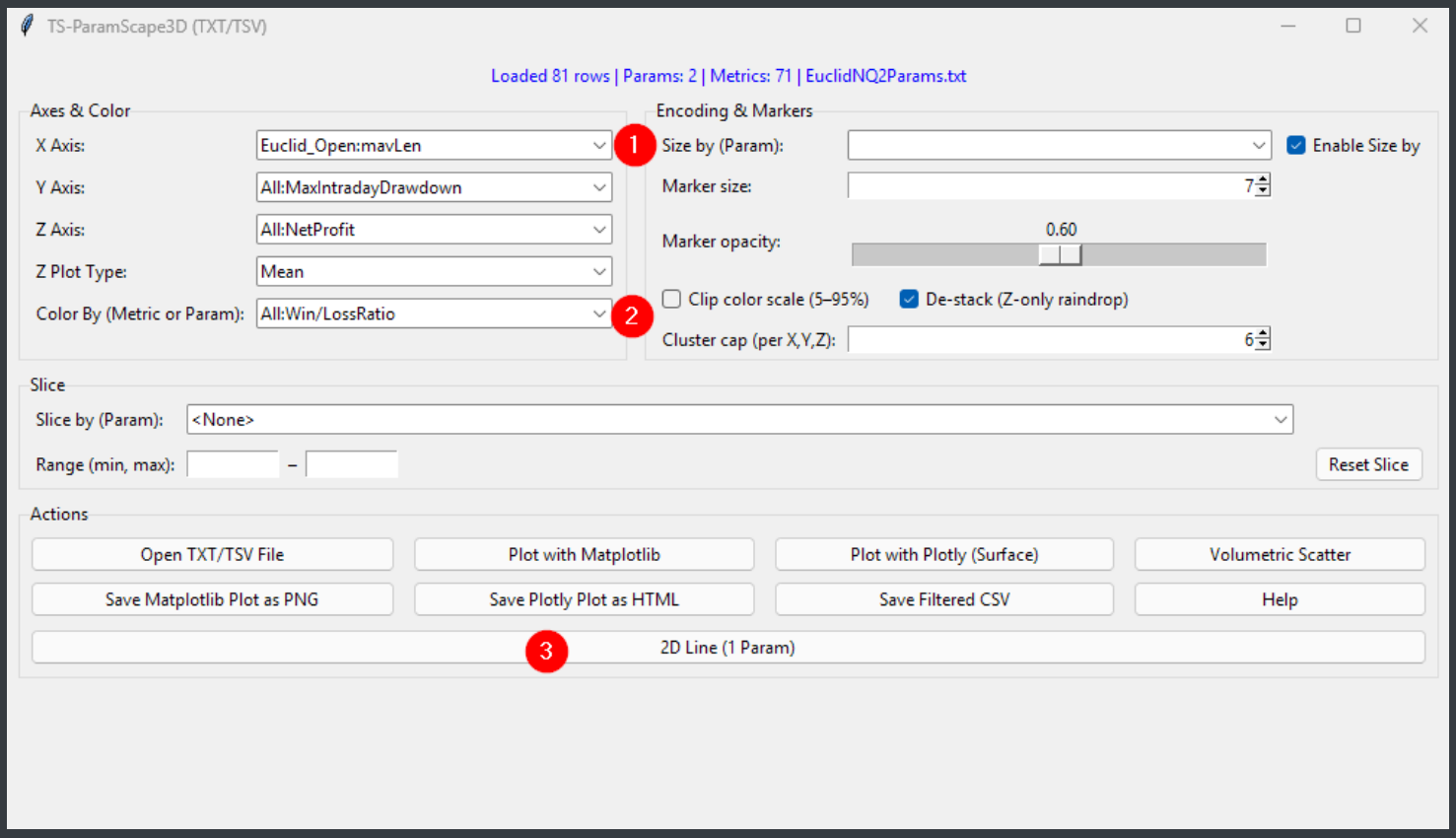
Plotly 3D Surface - Smoothed Mean · Cividis



User Inteface - original - without Single Line Collapse



With Single Line Collapse - 2D Line Chart - Highlighting one parameter



##

!(assets/TS-ParamScape3D%20interface2.png)

Simple 2D Mode and Advanced 3D Mode

- ### 2D Line quick start (Single Line Collapse)

  1. **Open** your TSV file.
  2. Set **X Axis** to the parameter you want to sweep (example: `stopLoss$`).
  3. Set **Color By** to the metric you want on the Y axis (example: `NetProfit`).
  4. (Optional) Use **Slice** to lock a second parameter to a range.
  5. Click **2D Line (1 Param)**.

  Hover behavior: Move your mouse near a point on the line to see a tooltip with:

  - **Mean** at that X
  - **Std Dev** at that X
  - **N** = number of optimization runs contributing to that X bucket

  > Tip: If the std dev feels “too large,” you’re probably averaging over many different combinations of the other parameters.
  > Use **Slice** to narrow the population.
  >
  > **Note:** Y Axis and Z Axis are not used in 2D Line mode.

- ### 1. Axes & Color

  This section defines how your 3D plot is constructed.

  - **X Axis / Y Axis / Z Axis** – Choose which parameters or metrics from the dataset are mapped to each axis.
  - **Z Plot Type** – Select how Z-axis values are aggregated. Options include:
    - *Mean* – Average values at each (X, Y) coordinate.
    - *Smoothed Mean* – Mean averaged with its neighbors (e.g., 3×3 grid) to produce a smoother surface.
    - *Standard Deviation* – Shows variability at each coordinate.
    - *Mean/Std* – Ratio of mean to variability, useful for robustness checks.
  - **Color By (Metric or Param)** – Determines how marker color is mapped (e.g., by NetProfit, Sharpe, or another parameter).

---

2. Encoding & Markers

Controls how data points are displayed.

- **Size by (Param)** – Scale marker size according to a chosen parameter/metric.
- **Marker size** – Adjust base size of all markers (slider from small to large).
- **Marker opacity** – Controls transparency; lower values reveal density/overlap.
- **Clip color scale (5–95%)** – Removes extreme outliers so color shading emphasizes the bulk of the data.
- **De-stack (Z-only raindrop)** – Separates markers at the same (X,Y) so they "rain down" along Z, making dense clusters more visible.
- **Cluster cap (per X,Y,Z)** – Limits the maximum number of markers drawn per coordinate to avoid overplotting.

---

3. Slice

Focuses analysis on a subset of data.

- **Slice by (Param)** – Choose a parameter to filter/slice by (e.g., fix ATRMult = 2.0).
- **Range (min, max)** – Numeric bounds for the slice.
- **Reset Slice** – Clears slicing and restores the full dataset.

---

4. Actions

Launch plots, save images, or export data.

Current file loading supports CSV, TXT, and TSV optimization files. The Plotly tools include both the 3D surface view and the 2D heatmap view.

- **Open TXT/TSV File** – Load optimization results.
- **Plot with Matplotlib** – Quick 3D surface visualization using Matplotlib.
- **Save Matplotlib Plot as PNG** – Save static image output.
- **Plot with Plotly (Surface)** – Interactive 3D surface plot using Plotly.
- **Save Plotly Plot as HTML** – Save interactive Plotly chart for sharing.
- **Volumetric Scatter** – 3D scatter plot showing all data points in full detail.
- **Save Filtered CSV** – Export current subset (after slicing/filters).

- **Help** – Opens documentation.

---

5. File / Status Bar

At the top you see:

- **Loaded rows / Params / Metrics** – Confirms how many rows were loaded and how many parameters/metrics are available.
- **File name** – Displays the currently loaded dataset..

Plot types (what you see & why it matters)

0) 2D Line (1 Param) — Mean ± 1σ (Single Line Collapse)

- **What it is:** A 2D line chart where **X** is a single parameter and **Y** is the selected **Color By** field.
- **What you see:**
  - Line = **mean(Y)** at each X
  - Band = **mean(Y) ± 1σ(Y)** at each X
  - Hover tooltip = **X, mean, std dev, and N (count of runs)**
- **Why:** Perfect for "one knob" analysis when you want to see the **overall trend** of a metric while other parameters vary.
- **Use when:** You only care about one parameter sweep, or you're using **Slice** to lock down a second parameter and want a clean curve.

1) Surface – Mean

- **What it is:** A 3D surface where Z is the average value at each (X,Y) address.
- **Why:** Gives the cleanest *overall* shape of the response surface—great for spotting broad ridges, basins, and knees.
- **Use when:** You want a quick read on which regions of parameter space tend to work best on average.

2) Surface – Smoothed Mean (3×3 neighborhood)

- **What it is:** Mean surface after local averaging of each address with its neighbors.

- **Why:** Suppresses jagged noise so macro-structure is easier to see; highlights stable plateaus and de-emphasizes one-off spikes.
- **Use when:** You care about robustness and want to see sustained “good neighborhoods,” not single-point flukes.

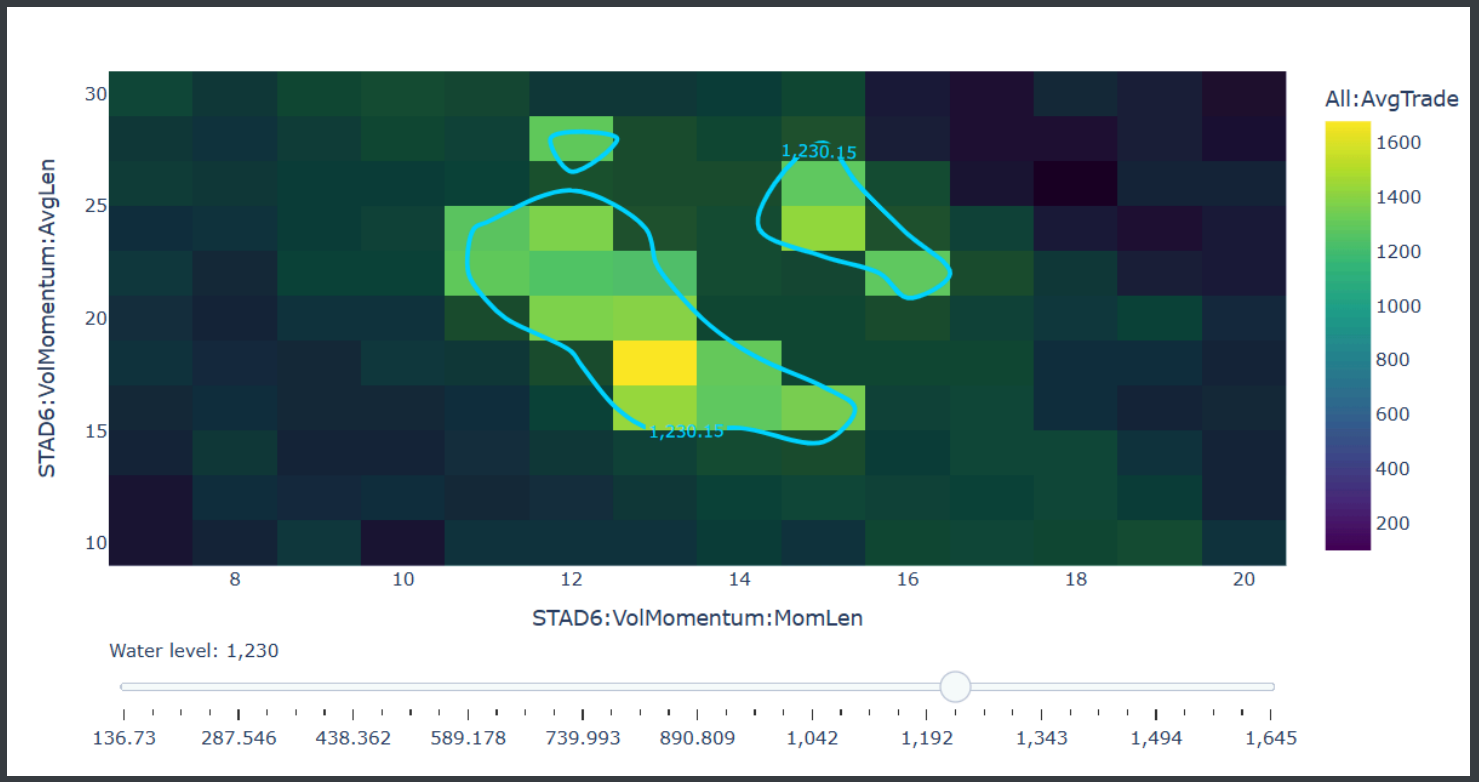
3) Surface – Standard Deviation

- **What it is:** A surface of variability at each (X,Y) address.
- **Why:** Shows how *reliable* results are. Low std = consistent; high std = sensitive/erratic.
- **Use when:** Comparing two promising regions—pick the one with lower volatility if performance is similar.

4) Surface – Mean / Std (signal-to-noise)

- **What it is:** Mean divided by standard deviation.
- **Why:** Balances level vs. stability in one map; favors high returns that are also consistent.
- **Use when:** Screening for robust hotspots to seed walk-forward or deeper testing.

5) Plotly Heatmap (2D surface view)

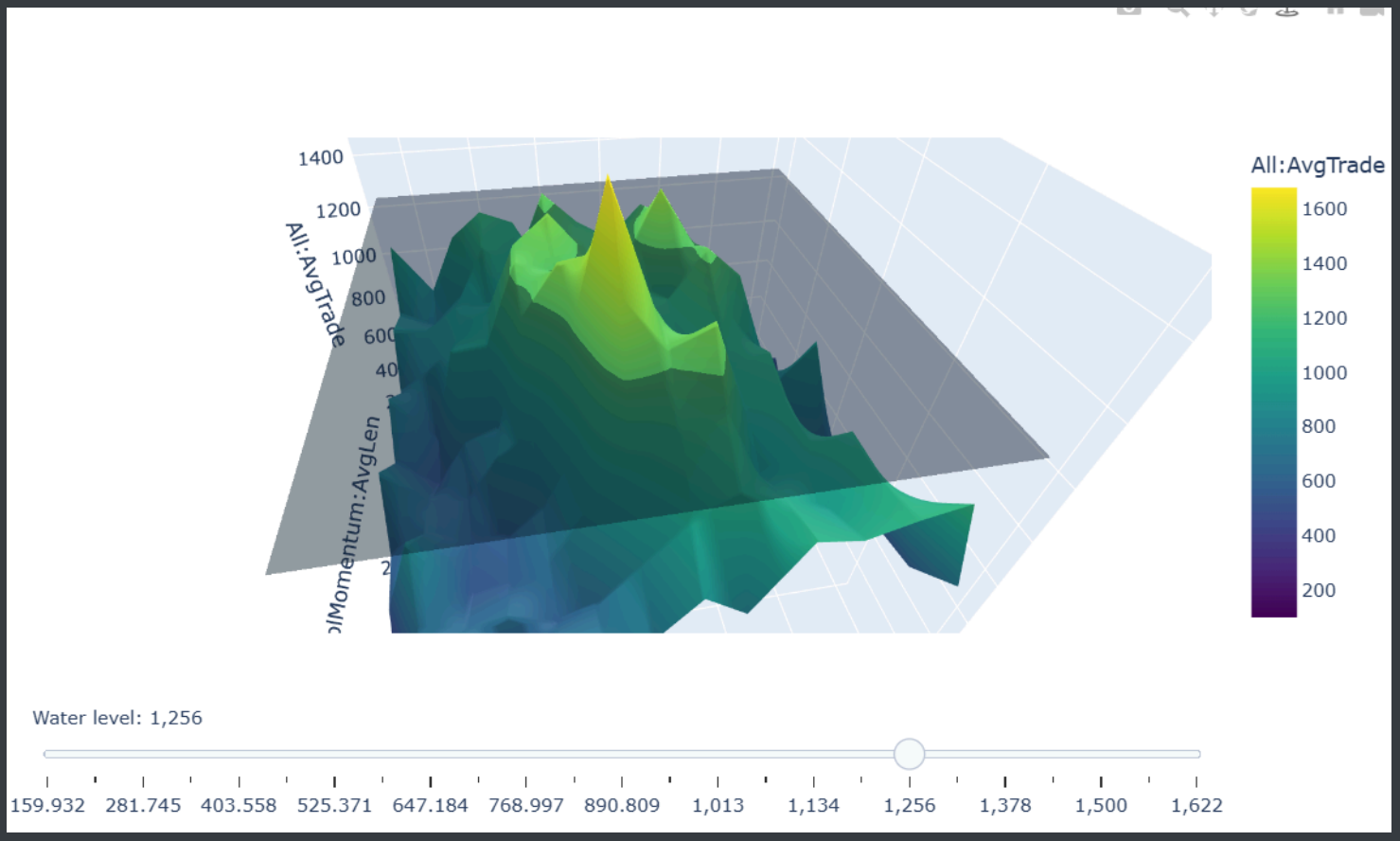


- **What it is:** A flattened 2D view of the selected X and Y parameters, colored by the selected result

metric.

- **Why:** It is easier to read exact regions of strength and weakness when you do not need to rotate a 3D surface.
- **Use when:** You want to compare objective functions quickly, explain a region to another trader, or print a compact visual summary.
- **How to read it:** Each heatmap cell represents one X/Y address after the selected collapse method has been applied. Brighter or stronger colors indicate better values according to the active color scale.

Water level on Plotly charts



The Plotly surface and Plotly heatmap include a **Water level** slider. This slider sets a threshold value for the selected result metric.

- On the **3D surface**, the water level appears as a dark horizontal plane. As the level rises, only the peaks above the threshold stand clearly above the water.
- On the **heatmap**, cells below the threshold are darkened. Cells at or above the threshold remain visible.
- The numeric water level is shown beside the slider so you can report the exact threshold used.

- The cyan outline on the heatmap is an interpolated contour guide. It helps show the approximate boundary where the selected metric crosses the water level.

Important interpretation note: each heatmap cell still contains one exact value. The cyan line does not mean different parts of the same cell have different measured values. It is a visual interpolation drawn between neighboring cells so the threshold boundary is easier to see. When cells are large or the data changes sharply, the line can cross through part of a cell. The darkened cells are the strict discrete test; the cyan line is the visual guide.

Separate cyan "lakes" are normal. They mean there are separate regions where the selected metric rises above the water level. If two strong regions are separated by cells below the threshold, the heatmap will show them as separate islands.

6) Volumetric Scatter (3D point cloud)

- **What it is:** Every run as a dot in 3D; color/size can encode another metric/param. “De-stack” helps separate same-address points along Z.
- **Why:** Full-fidelity view—no aggregation. You can see multi-modal clusters, outliers, and how dense each area really is.
- **Use when:** You suspect hidden structure (e.g., two distinct regimes at same address) or want to verify the surface isn’t hiding bimodality.

Why both Matplotlib and Plotly?

Matplotlib (Surface / PNG):

- **Speed & reproducibility:** Fast to render, deterministic output—ideal for batch runs and reports.
- **Publication-ready images:** Crisp PNGs that drop straight into papers, PDFs, or emails.
- **Lightweight & offline:** Minimal overhead; good on slower machines or very large grids.

Plotly (Interactive Surface / HTML):

- **Exploration:** Rotate, zoom, and hover to inspect exact values without re-plotting.
- **Sharing:** Saves to a self-contained HTML—perfect for sending an interactive chart to a colleague/client.
- **Detail on demand:** Tooltips + dynamic perspective help confirm whether a “ridge” is real or an

artifact.

Rule of thumb:

- Use **Matplotlib** to *document and publish*.
- Use **Plotly** to *discover and explain*—especially when walking through results live or probing a tricky region.

Quick recipe (choosing the right view)

- **Threshold check:** Use **Plotly Heatmap** with the **Water level** slider to show which X/Y regions remain above a chosen objective value.
- **First pass:** Surface → **Smoothed Mean** to see stable plateaus.
- **Risk check:** Surface → **Std Dev**, then **Mean/Std** to favor robust zones.
- **Truthing: Volumetric Scatter** with "De-stack" on—look for clusters vs. stray spikes.
- **Deliverables:** Save **Matplotlib PNG** for the report; save **Plotly HTML** for interactive review.

That's it! You now know what each control does and how to build views that reveal structure and robustness in your optimization results. If you'd like a one-page quick-start with screenshots, we can add it next

FAQ & Troubleshooting

❓ What kind of file can I load?

- You can load **TXT** or **TSV** files that contain parameter optimization results.
- The file should have column headers, with **parameters** first and **metrics/objectives** following.
- Example: `mavLen, stopLoss$, atrMult, profit$, NetProfit, Drawdown, Sharpe…`

❓ Why are there multiple values at the same (X, Y) point?

- If you keep some parameters free (e.g., Profit$ or ATRMult), you will get multiple results at the same (mavLen, stopLoss$).
- The surface options (*Mean, Smoothed Mean, Std, Mean/Std*) exist to summarize those clusters.
- For raw inspection, use **Volumetric Scatter** with "De-stack" to see all points.

❓ What's the difference between *Mean* and *Smoothed Mean*?

- **Mean:** Direct average of results at each (X,Y).
- **Smoothed Mean:** Each address is averaged with its 3×3 neighborhood, reducing noise and highlighting broader patterns.
- *Tip:* Use Smoothed Mean when surfaces look "spiky" or brittle.

❓ When should I use Matplotlib vs. Plotly?

- **Matplotlib:** Fast, lightweight, reproducible plots for reports and PNG exports.
- **Plotly:** Interactive exploration—rotate, zoom, hover, and share HTML.
- *Tip:* Many users first explore in Plotly, then export polished Matplotlib PNGs for documentation.

❓ Why does the color scale sometimes look strange?

- Extreme outliers can "saturate" the color map.
- Enable **Clip color scale (5–95%)** to exclude extreme 5% tails and rescale the shading.

❓ The scatter plot looks stacked—points hide behind each other!

- Check **De-stack (Z-only raindrop)**.
- This offsets identical (X,Y) points along Z so clusters are visible.

❓ My plot looks too dense—how do I make it clearer?

- Lower the **Marker size** or **opacity**.
- Use **Cluster cap** to limit the number of points per (X,Y,Z).
- Or switch to a **surface plot** instead of scatter.

❓ Slicing doesn't seem to work—why?

- Make sure you selected a parameter in **Slice by (Param)** first.
- Then enter a numeric min/max range.
- Use **Reset Slice** to clear and restore the full dataset.

❓ The program says "No NetProfit found"

- Check your column headers: they must match exactly (case-sensitive, no spaces).
- If your objective column is named differently (e.g., `All:NetProfit`), select it manually in the drop-down.

Troubleshooting

⚠️ Plot won't render / blank chart

- Ensure you've loaded a file with valid numeric data.
- Check that X, Y, and Z axes are all assigned to valid parameters/metrics.

⚠️ Performance is slow with big files

- Try reducing marker size or opacity.
- Use **Cluster cap** to limit the number of points drawn.
- Slice the data to a smaller range.
- For huge files, Matplotlib may be faster than Plotly.

⚠️ Exported PNG/HTML missing labels

- Ensure you've set axis labels via X/Y/Z selections before exporting.
- If using Plotly, hover tooltips replace some labels—use mouseover to see values.

⚠️ Colors don't match expectations

- Remember that color is determined by the **Color By** dropdown, not the Z axis.
- Check that you haven't accidentally selected a different parameter for coloring.

TS-ParamScape-3D — Third-Party Notices

This product bundles third-party components licensed under open-source terms.
Each license applies only to its respective component(s).

Plotly.py

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

plotly.js

Kaleido (static image export engine for Plotly)

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Other bundled libraries (if any)

If your installer bundles additional Python packages (e.g., via PyInstaller), include their licenses here. Common examples:

- NumPy — BSD 3-Clause
- Pandas — BSD 3-Clause
- SciPy — BSD 3-Clause
- scikit-learn — BSD 3-Clause
- joblib — BSD 3-Clause

Paste the exact license texts from each project’s source distribution or wheel into this section.