

Partial Moments in QF

Fred Viole

Introduction to Partial Moments

Q. What are partial moments?

A. The elements of variance.

Univariate:

$$LPM(\textit{degree}, \textit{target}, \textit{variable}) = \frac{1}{T} \sum_{t=1}^T [\max(0, \textit{target} - \textit{variable}_t)]^{\textit{degree}}$$

$$UPM(\textit{degree}, \textit{target}, \textit{variable}) = \frac{1}{T} \sum_{t=1}^T [\max(0, \textit{variable}_t - \textit{target})]^{\textit{degree}}$$

- Grounded in probability-weighted integration and rigorous measure theory

$$LPM(n, t, X) = \int_{-\infty}^t (t - x)^n dF(x)$$

- Time-series or cross-sectional

Partial Moment Equivalences

Mean

```
1 library(NNS)
2 set.seed(123); x = rnorm(100); y = rnorm(100)
3
4 mean(x)
```

```
[1] 0.09040591
```

```
1 UPM(degree = 1, target = 0, variable = x) - LPM(degree = 1, target = 0, variable =
```

```
[1] 0.09040591
```

Variance

```
1 # Sample Variance (base R):  
2 var(x)
```

```
[1] 0.8332328
```

```
1 # Sample Variance:  
2 mu_x = mean(x)  
3 (UPM(2, mu_x, x) + LPM(2, mu_x, x)) * (length(x) / (length(x) - 1))
```

```
[1] 0.8332328
```

```
1 # Population Adjustment of Sample Variance (base R):  
2 var(x) * ((length(x) - 1) / length(x))
```

```
[1] 0.8249005
```

```
1 # Population Variance:  
2 UPM(2, mu_x, x) + LPM(2, mu_x, x)
```

```
[1] 0.8249005
```

```
1 # Variance is also the co-variance of itself:  
2 {(Co.LPM(degree_lpm = 1, x = x, y = x, target_x = mu_x, target_y = mu_x)  
3   + Co.UPM(degree_upm = 1, x, x, mu_x, mu_x)  
4   - D.LPM(degree_lpm = 1, degree_upm = 1, x, x, mu_x, mu_x)  
5   - D.UPM(1, 1, x, x, mu_x, mu_x))}
```

```
[1] 0.8249005
```

Skewness

```
1 PerformanceAnalytics::skewness(x)
```

```
[1] 0.06049948
```

```
1 ((UPM(3, mu_x, x) - LPM(3, mu_x, x)) / (UPM(2, mu_x, x) + LPM(2, mu_x, x))^(3/2))
```

```
[1] 0.06049948
```

Kurtosis

```
1 PerformanceAnalytics::kurtosis(x)
```

```
[1] -0.161053
```

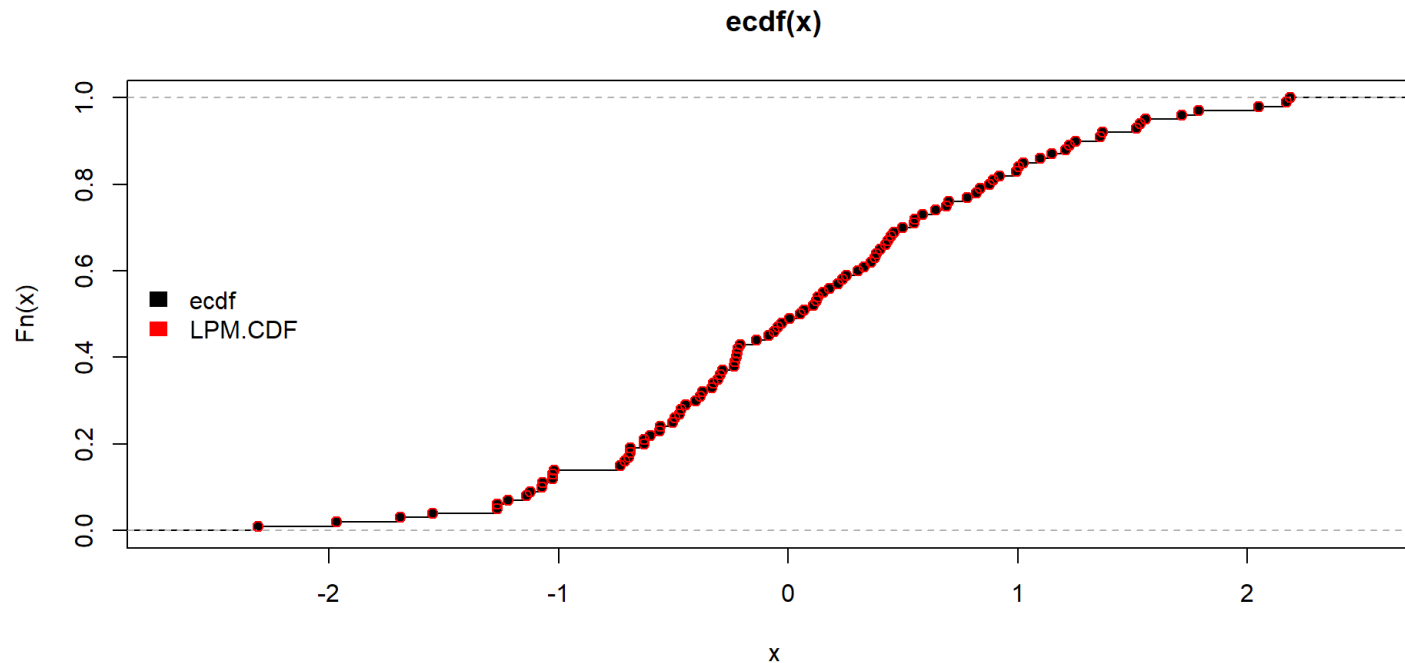
```
1 ((UPM(4, mu_x, x) + LPM(4, mu_x, x)) / (UPM(2, mu_x, x) + LPM(2, mu_x, x))^2) - 3
```

```
[1] -0.161053
```

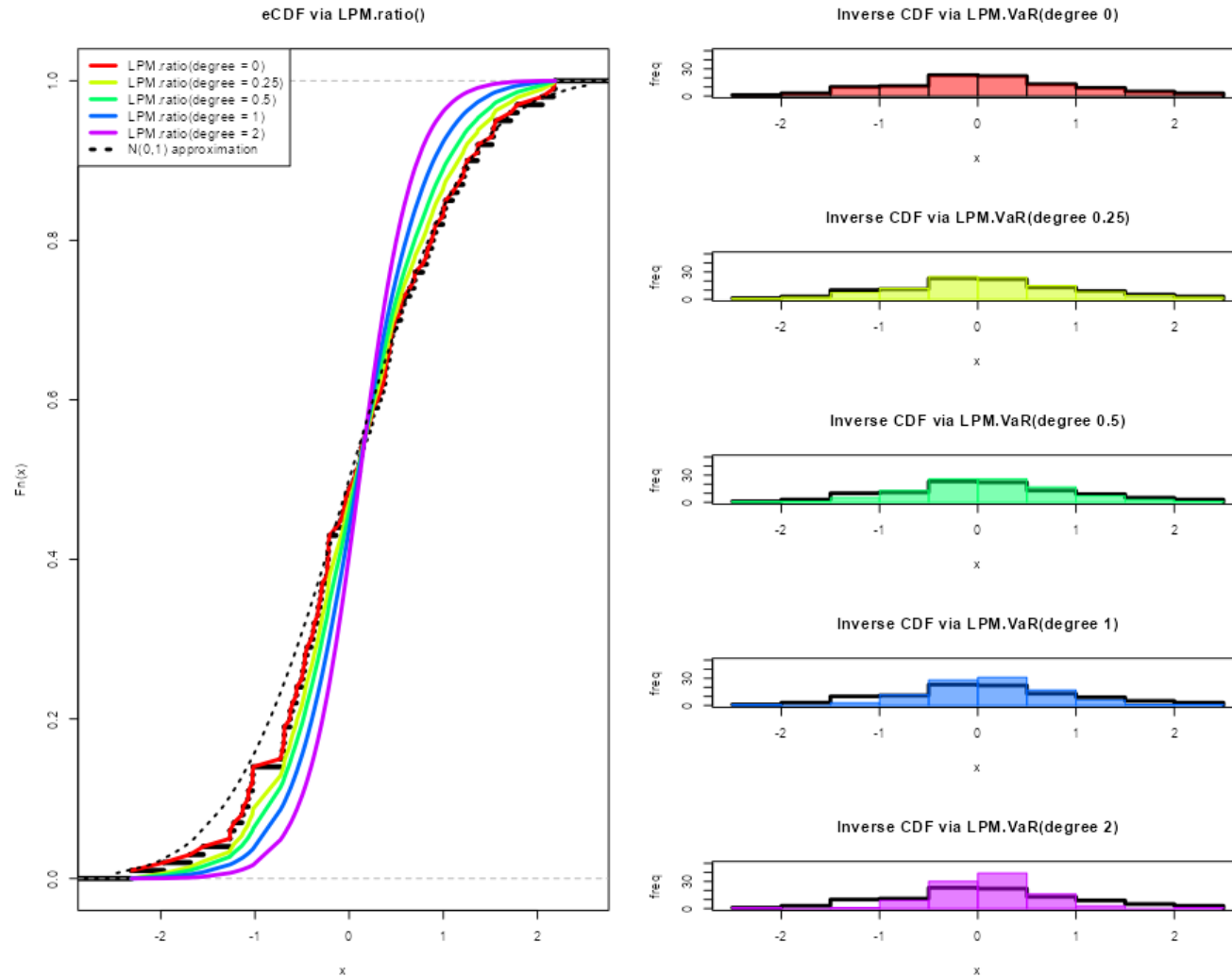
More Equivalences

LPM degree 0 is the eCDF for any distribution

```
1 LPM.CDF = LPM(degree = 0, target = sort(x), variable = x)
2
3 plot(ecdf(x))
4 points(sort(x), LPM.CDF, col='red')
5 legend('left', legend = c('ecdf','LPM.CDF'), fill=c('black','red'), border=NA, bty
```

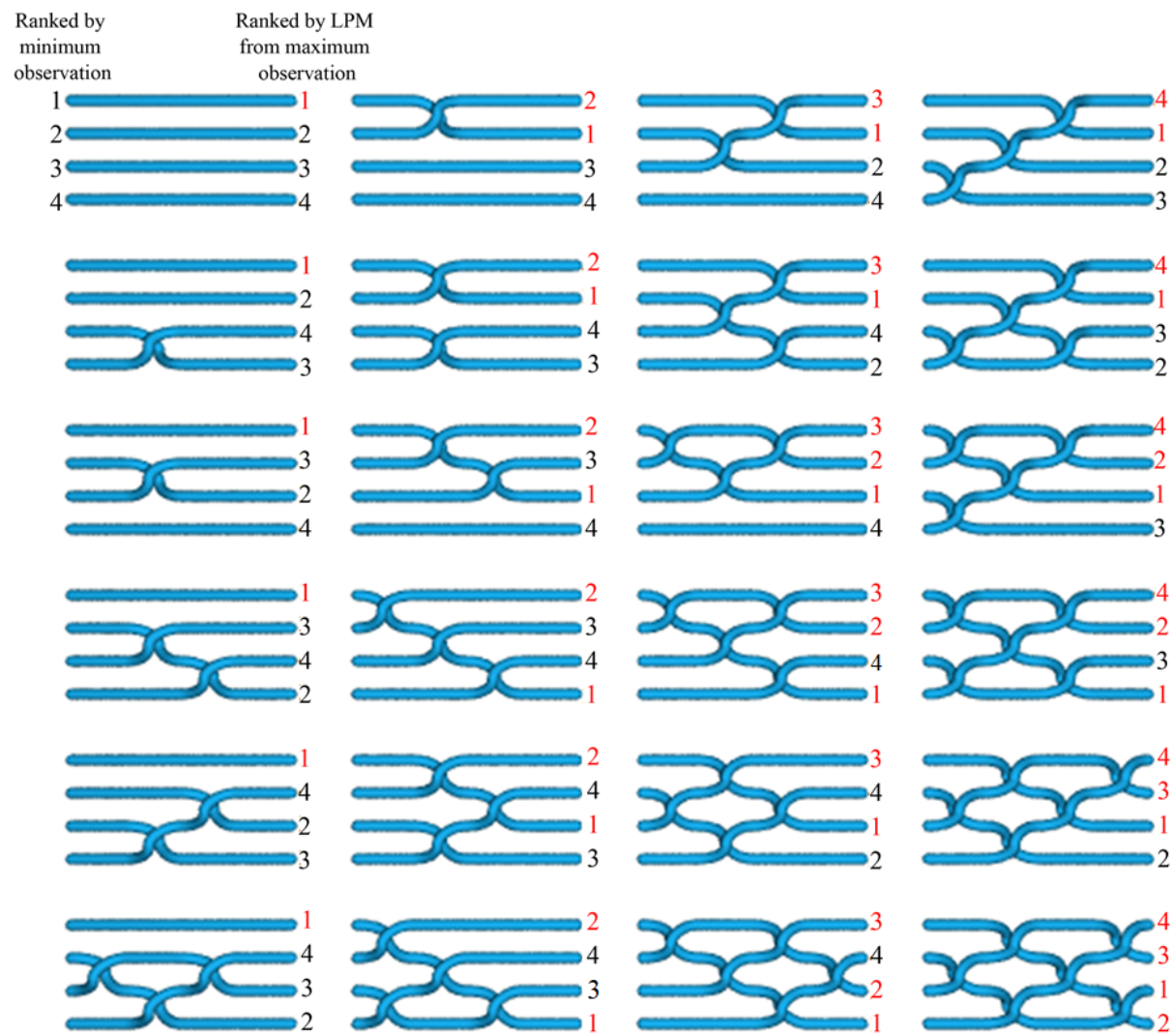


More CDFs and PDFs



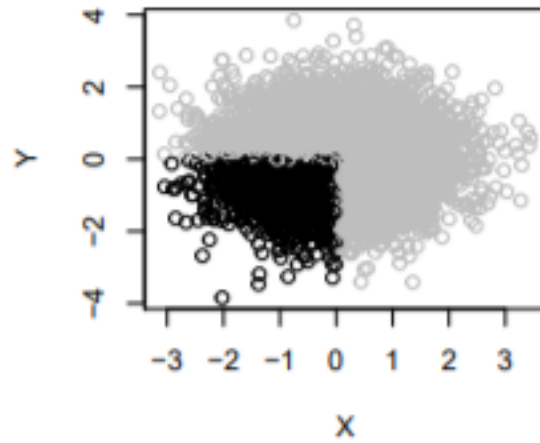
Still empirical CDFs, all we've done is increase the *degree*

Comparing CDFs Led to Stochastic Dominant Efficient Sets

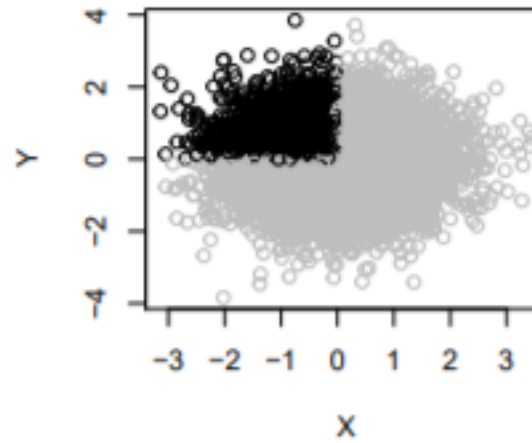


Multivariate

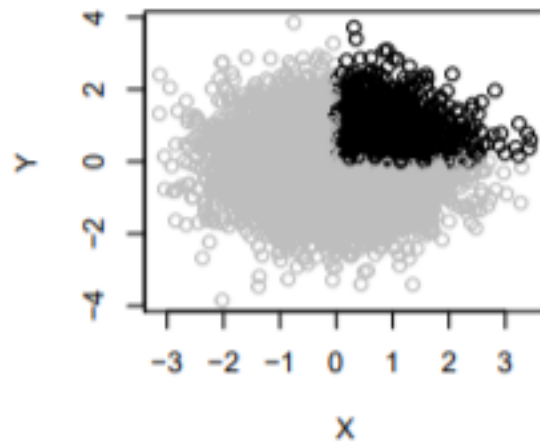
CLPM Quadrant



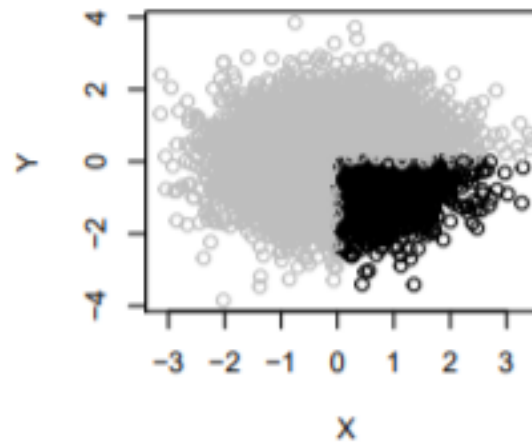
DUPM Quadrant



CUPM Quadrant



DLPM Quadrant



$$CLPM(degree, target, x, y) = \frac{1}{T} \sum_{t=1}^T [\max(0, target - x_t)]^{degree} [\max(0, target - y_t)]^{degree}$$

$$CUPM(degree, target, x, y) = \frac{1}{T} \sum_{t=1}^T [\max(0, x_t - target)]^{degree} [\max(0, y_t - target)]^{degree}$$

$$DLPM(degree, target, x, y) = \frac{1}{T} \sum_{t=1}^T [\max(0, x_t - target)]^{degree} [\max(0, target - y_t)]^{degree}$$

$$DUPM(degree, target, x, y) = \frac{1}{T} \sum_{t=1}^T [\max(0, target - x_t)]^{degree} [\max(0, y_t - target)]^{degree}$$

Covariance Equivalence

Covariance Elements and Covariance Matrix

The covariance matrix (Σ) is equal to the sum of the co-partial moments matrices less the divergent partial moments matrices.

$$\Sigma = CLPM + CUPM - DLPM - DUPM$$

```
> cov.mtx = PM.matrix(LPM_degree = 1, UPM_degree = 1, target = 'mean', variable = cbind(x,y), pop_adj = TRUE)
> cov.mtx
$cupm
      x      y
x 0.4299250 0.1033601
y 0.1033601 0.5411626

$dupm
      x      y
x 0.0000000 0.1469182
y 0.1560924 0.0000000

$dldm
      x      y
x 0.0000000 0.1560924
y 0.1469182 0.0000000

$clpm
      x      y
x 0.4033078 0.1559295
y 0.1559295 0.3939005

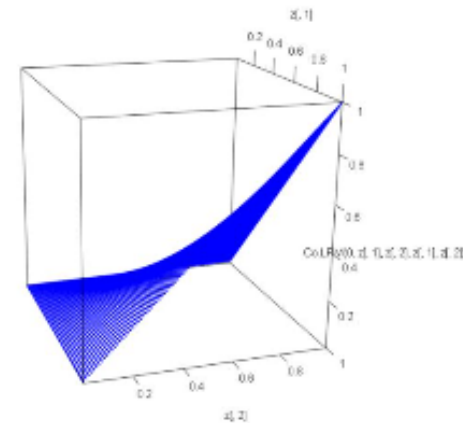
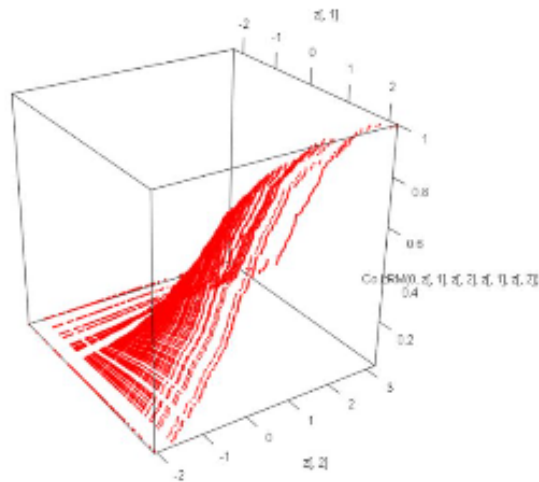
$cov.matrix
      x      y
x 0.83323283 -0.04372107
y -0.04372107 0.93506310

# Reassembled Covariance Matrix
> cov.mtx$cupm + cov.mtx$clpm - cov.mtx$dupm - cov.mtx$dldm
      x      y
x 0.83323283 -0.04372107
y -0.04372107 0.93506310

# Standard Covariance Matrix
> cov(cbind(x,y))
      x      y
x 0.83323283 -0.04372107
y -0.04372107 0.93506310
```

Copulas

```
1 # Data
2 set.seed(123); x = rnorm(100); y = rnorm(100); z = expand.grid(x, y)
3
4 # Plot
5 rgl::plot3d(z[,1], z[,2], Co.LPM(0, z[,1], z[,2], z[,1], z[,2]), col = "red")
6
7 # Uniform values
8 u_x = LPM.ratio(0, x, x); u_y = LPM.ratio(0, y, y); z = expand.grid(u_x, u_y)
9
10 # Plot
11 rgl::plot3d(z[,1], z[,2], Co.LPM(0, z[,1], z[,2], z[,1], z[,2]), col = "blue")
```



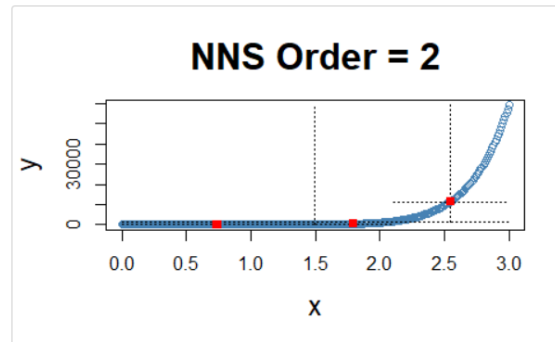
Partitioning Led to:

Nonlinear Correlation & Dependence

Nonlinear Relationship

Note the fact that all observations occupy the co-partial moment quadrants.

```
x = seq(0, 3, .01) ; y = x ^ 10
```



```
cor(x, y)
```

```
## [1] 0.6610183
```

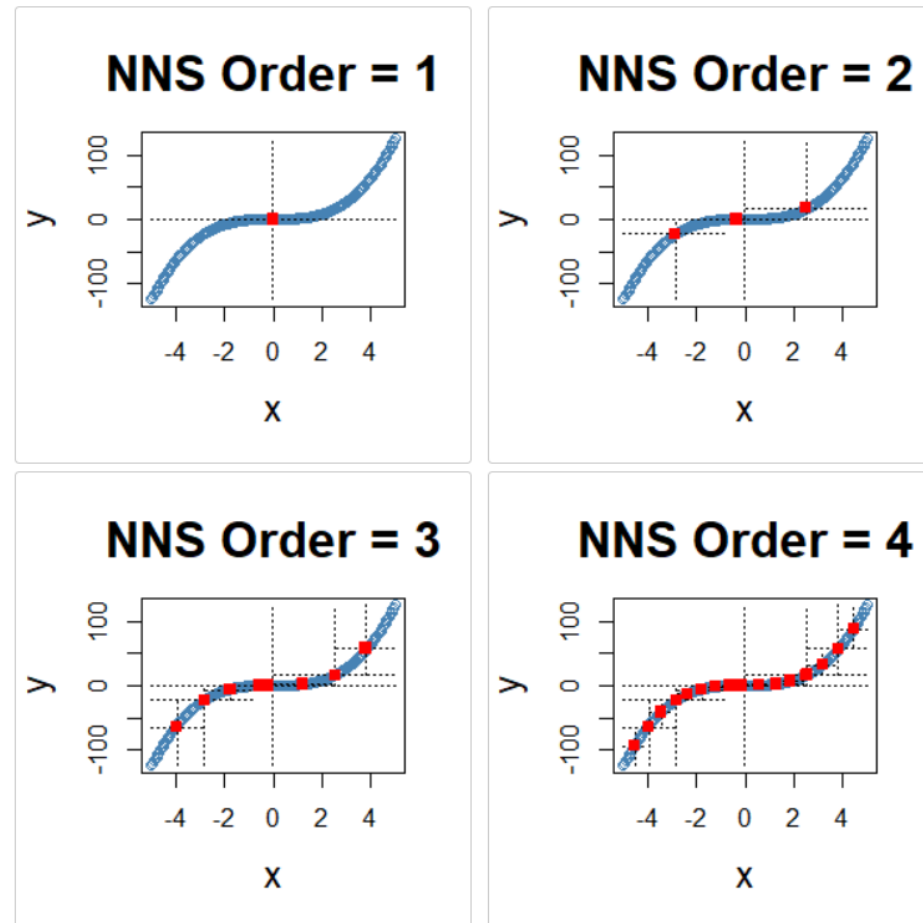
```
NNS.dep(x, y)
```

```
## $Correlation  
## [1] 0.9325653  
##  
## $Dependence  
## [1] 0.9325653
```

Compared to Mutual Information, Distance Correlation,
Chatterjee's ξ ...

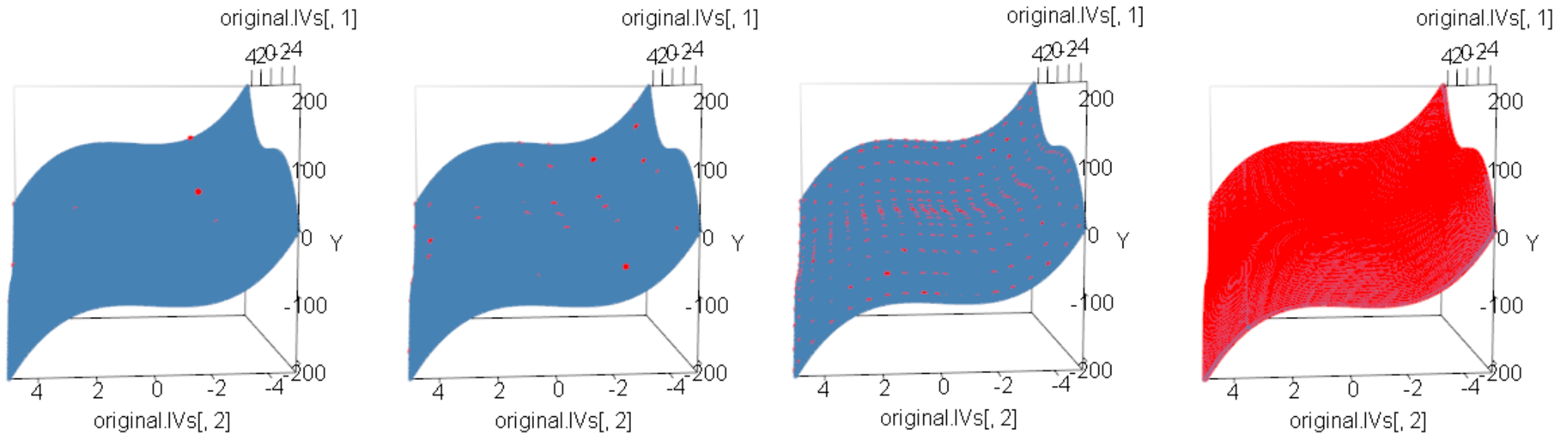
Partitioning Also Led to:

Nonlinear Regression



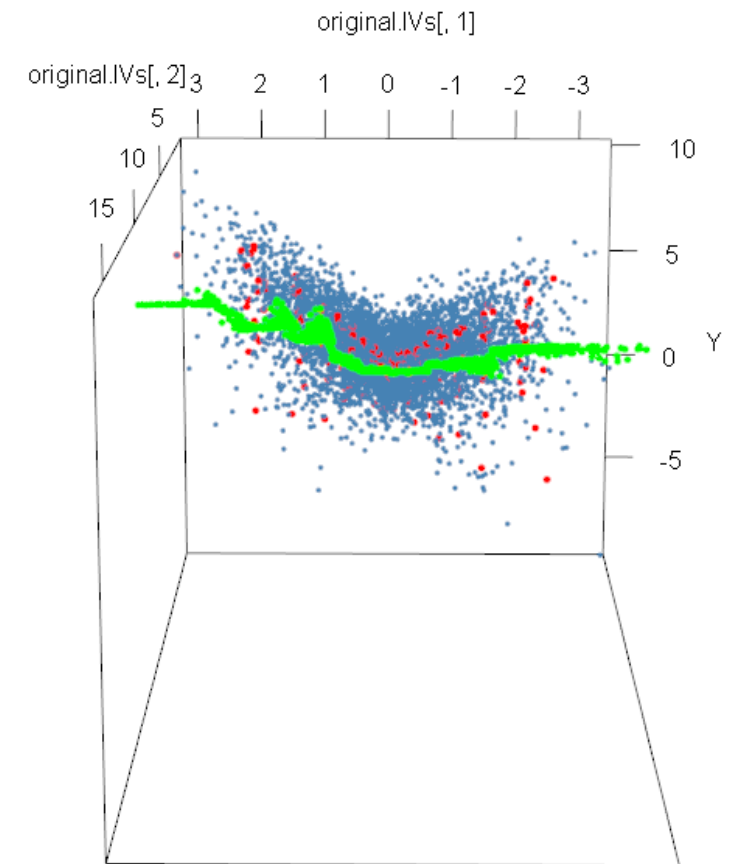
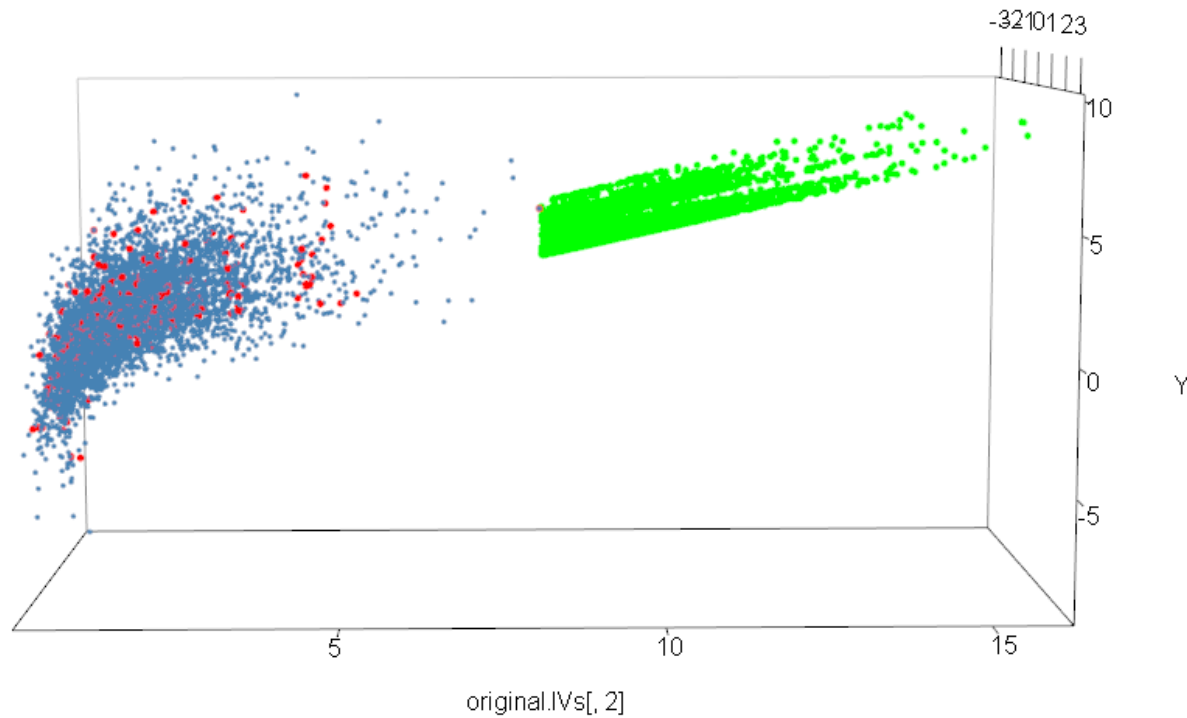
Offers dynamic bandwidth solution

Multivariate Regression

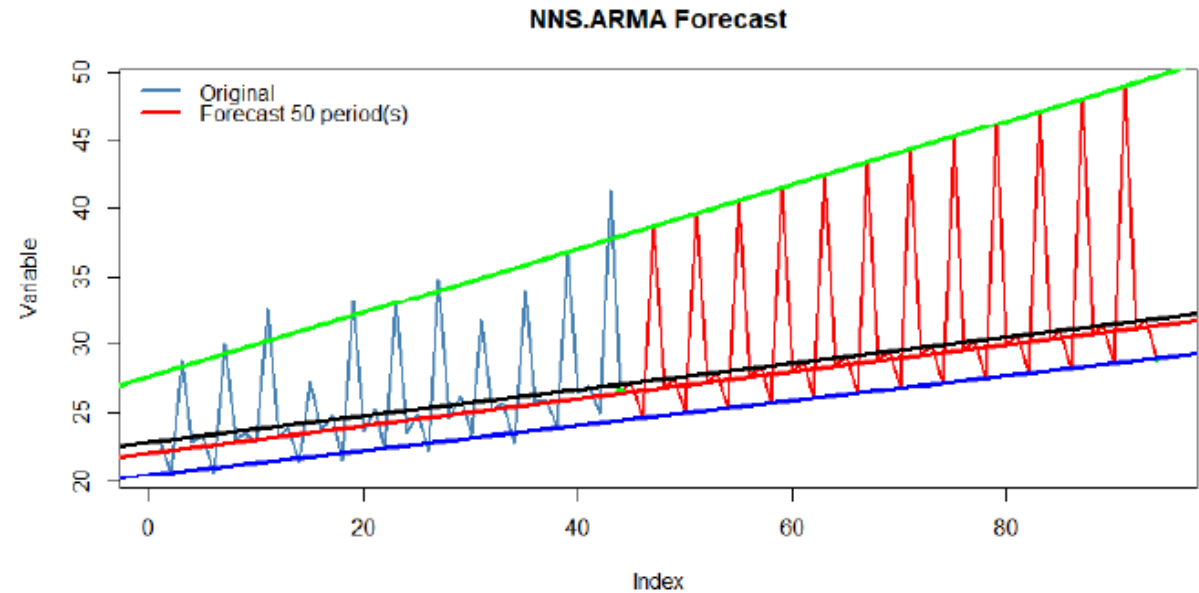
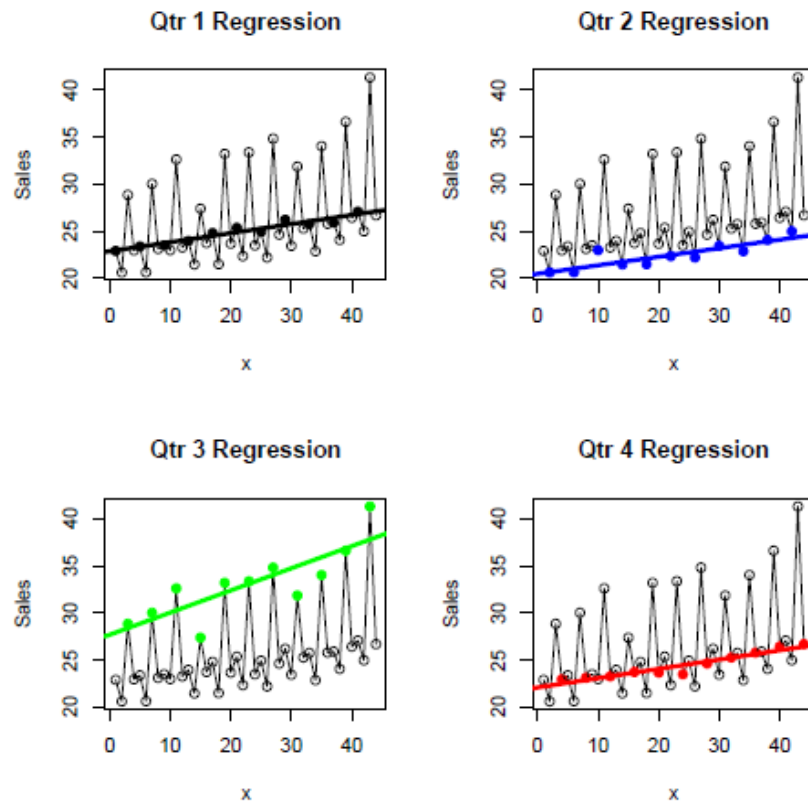


- Similar to kNN regression... cluster **centroids** instead of observations
- k -fold cross-validation for number of **centroids**

Very good at extrapolation



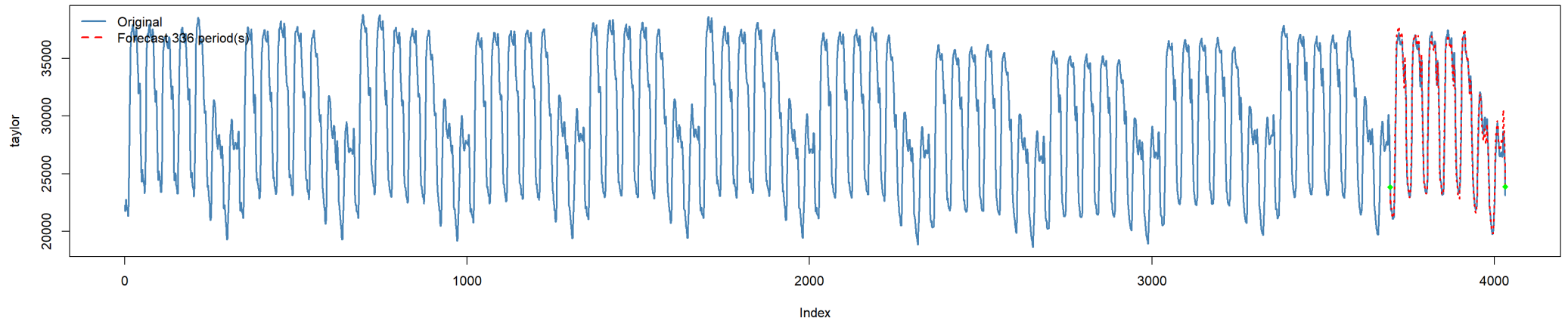
Nonlinear Regression Led to Time-Series Forecasting:



Time-Series Forecasting with Nonlinear Regression

```
1 require(forecast); data(taylor)
2
3 NNS.ARMA(taylor, h = 336, training.set = length(taylor)-336,
4           seasonal.factor = 336, plot = TRUE, method = 'nonlin')
```

NNS.ARMA Forecast



and a multivariate extension via a nonparametric vector autoregression!

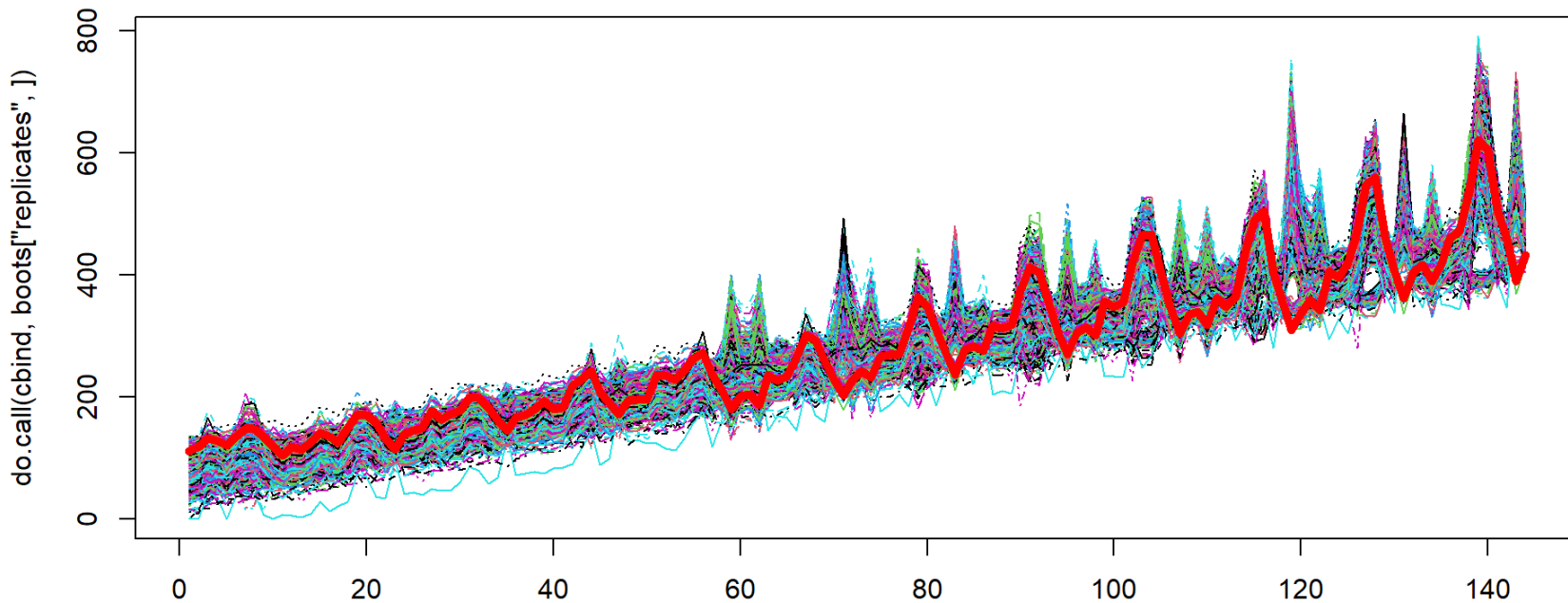
Generative and Synthetic Data

Maximum Entropy Bootstrap for Time-Series

- Specify a `rho` or `drift` for any replicate... ensures $\rho \in [-1, 1]$ correlation spectrum is covered.
- Standard IID MC or block bootstrap cannot accomplish this.

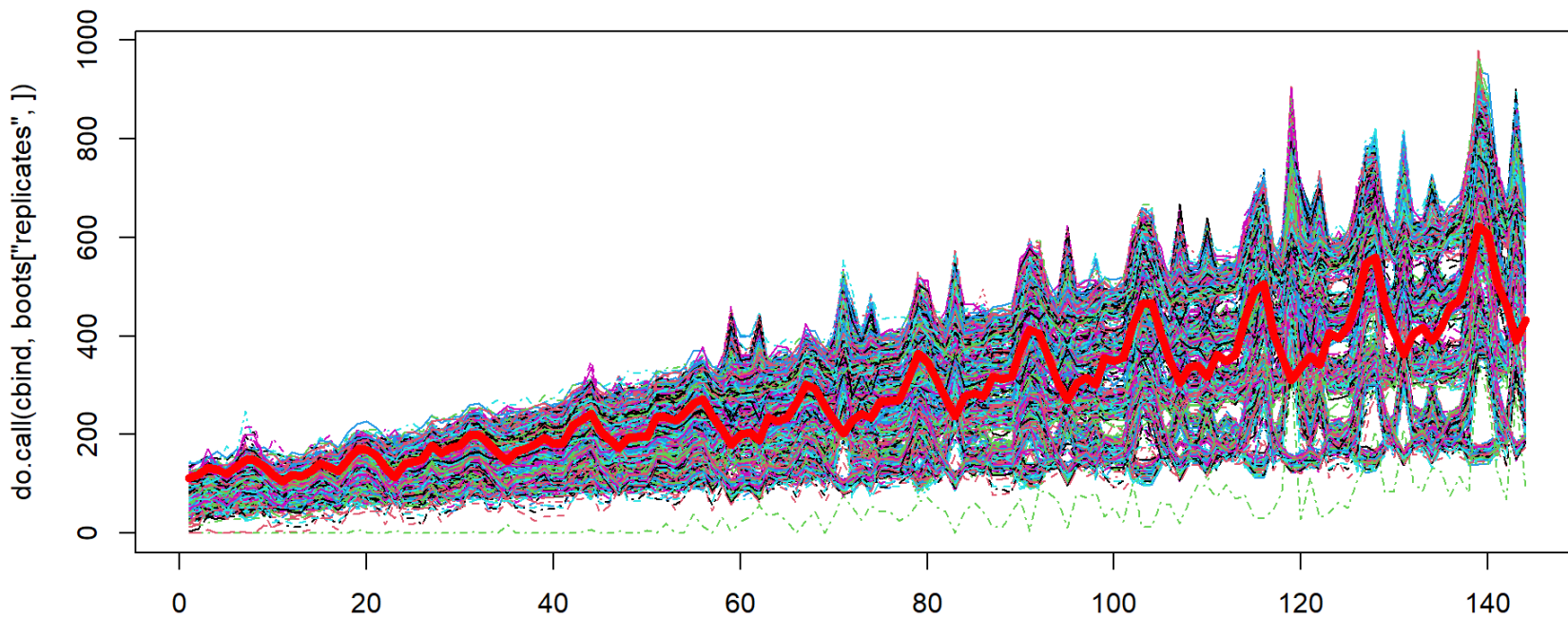
Static Drift

```
1 boots = NNS.meboot(AirPassengers, reps = 100, rho = seq(-1, 1, .25),  
2                   xmin = 0, target_drift_scale = 1)  
3  
4 matplot(do.call(cbind, boots["replicates", ]), type = "l")  
5 lines(1:length(AirPassengers), AirPassengers, lwd = 5, col = "red")
```



Specify a Drift for Replicates (RfR perhaps)

```
1 boots = NNS.meboot(AirPassengers, reps = 100, rho = seq(-1, 1, .25),  
2               xmin = 0, target_drift = 1:4)  
3  
4 matplot(do.call(cbind, boots["replicates", ]), type = "l")  
5 lines(1:length(AirPassengers), AirPassengers, lwd = 5, col = "red")
```



Ultimately Led to This General Statistical Toolkit:



The screenshot displays the NNS software interface. At the top left is the NNS logo, a white hexagon with a blue border containing the text "NNS" in red and "NONLINEAR NONPARAMETRIC STATISTICS" in smaller red text below it. To the right of the logo is a square QR code. Below the logo and QR code is a horizontal bar with two sections: "NNS version 11.1" and "licence GPL-3". Below this bar, the text "NNS offers:" is followed by a bulleted list of statistical methods.

NNS version 11.1 licence GPL-3

NNS offers:

- Numerical Integration & Numerical Differentiation
- Partitional & Hierarchical Clustering
- Nonlinear Correlation & Dependence
- Causal Analysis
- Nonlinear Regression & Classification
- ANOVA
- Seasonality & Autoregressive Modeling
- Normalization
- Stochastic Dominance
- Advanced Monte Carlo Sampling

Further Questions? Ask our GPT!



[https://chatgpt.com/g/
g-67b5de85c6e081919570830a6656a387-ovvo-labs-gpt](https://chatgpt.com/g/g-67b5de85c6e081919570830a6656a387-ovvo-labs-gpt)

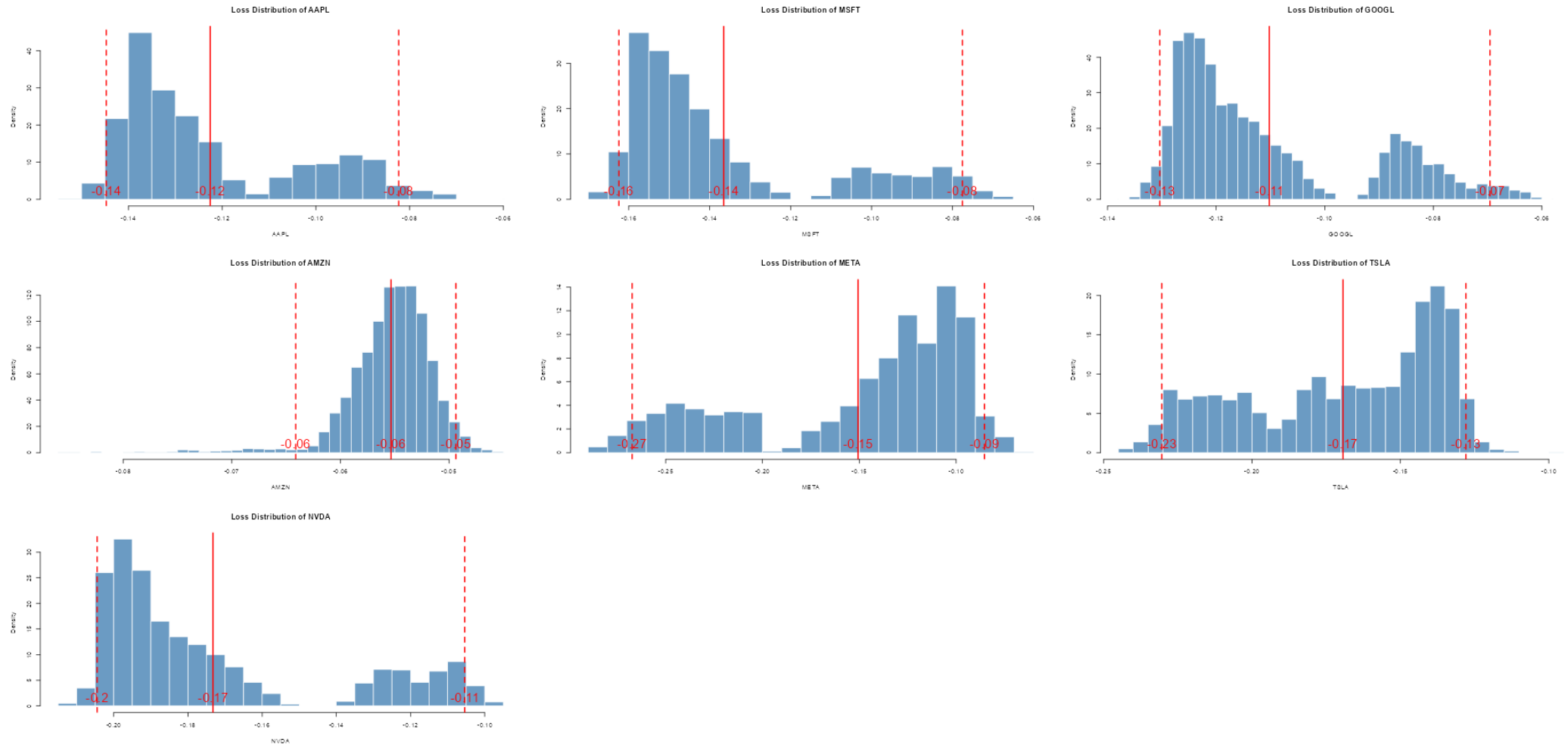
Finance Applications?
You bet!

Stress Testing

- Restrict observations to *CLPM* and use `NNS.reg()`, maintains dependence structure
- Reverse the procedure... instead of what happens to portfolio if S&P 500 drops 10%, what portfolio returns would lead to a 10% S&P 500 drop? Distribution of scenarios.

https://github.com/OVVO-Financial/Finance/blob/main/stress_test.md

Distribution of Losses



Utility Theory

- Markowitz dedicated a quarter of his 1959 book to utility theory. Was it a waste of time? Absolutely not!
- Each of us has unique risk profiles, costs of capital, and preferences. Parsing variance to reflect these subjective interpretations is critical for representing individual preferences.
- Partial moments are the *perfect* method to achieve this. Summary statistics just don't cut it!

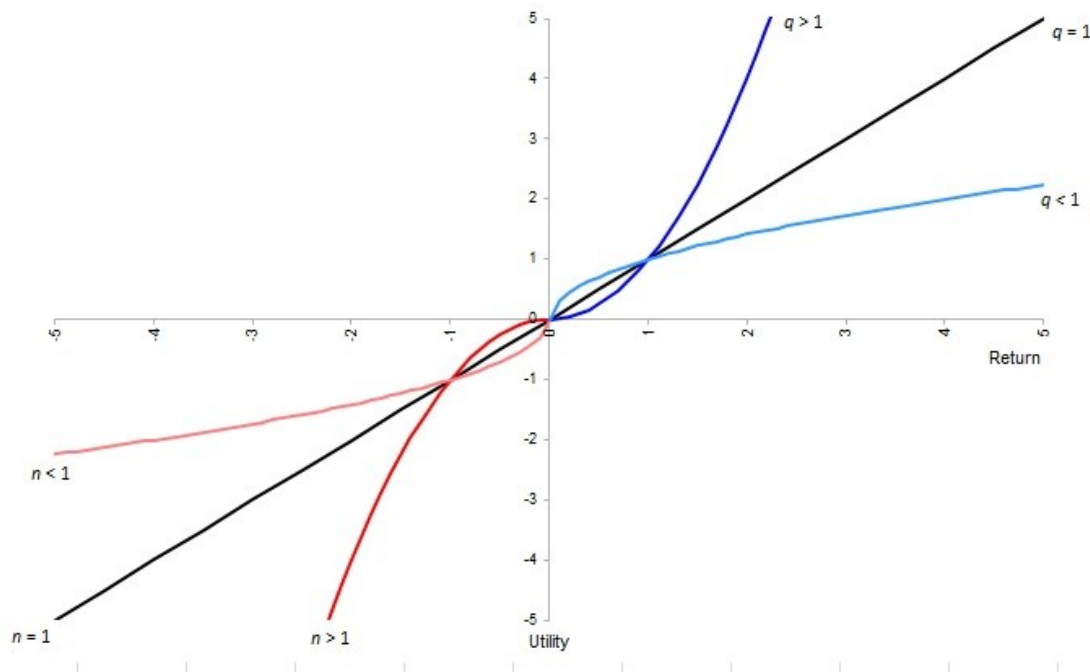
Let's Revisit Degrees

$$LPM(\textcolor{red}{n}, \textcolor{blue}{target}, variable) = \frac{1}{T} \sum_{t=1}^T [\max(0, \textcolor{blue}{target} - variable_t)]^{\textcolor{red}{n}}$$

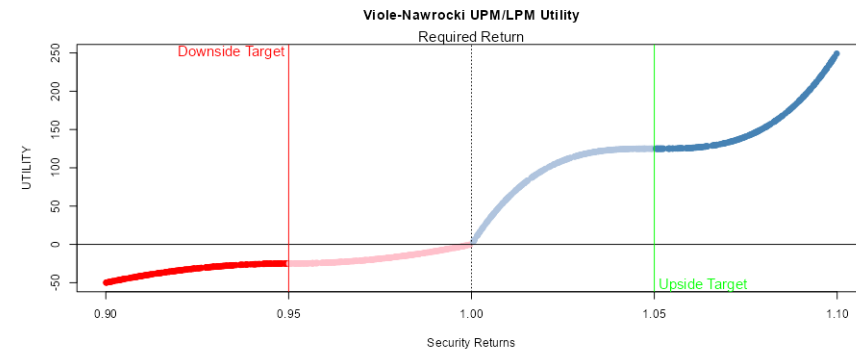
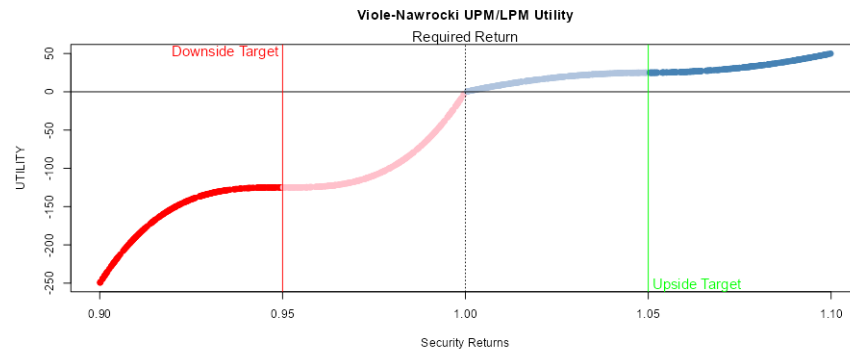
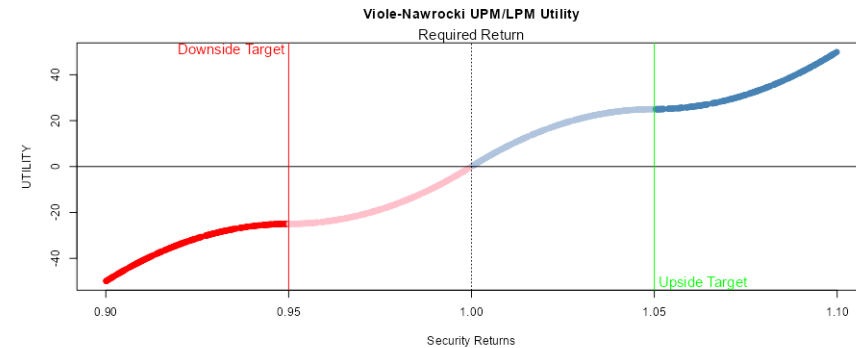
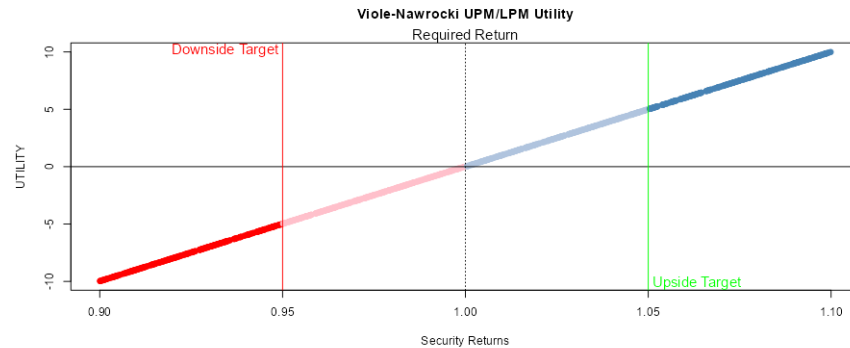
$$UPM(\textcolor{blue}{q}, \textcolor{blue}{target}, variable) = \frac{1}{T} \sum_{t=1}^T [\max(0, variable_t - \textcolor{blue}{target})]^{\textcolor{blue}{q}}$$

loss-aversion: $\textcolor{red}{n}$

gain-seeking: $\textcolor{blue}{q}$



Utility Functions



NOBODY is linear ($n = 1$) wrt losses!

Risk-aversion ($n : q$) and loss-aversion (n) are two distinct preferences

Utility Embedded Into Covariance Matrix

Covariance Elements and Covariance Matrix

The covariance matrix (Σ) is equal to the sum of the co-partial moments matrices less the divergent partial moments matrices.

$$\Sigma = CLPM + CUPM - DLPM - DUPM$$

```
> cov.mtx = PM.matrix(LPM_degree = 1, UPM_degree = 1, target = 'mean', variable = cbind(x,y), pop_adj = TRUE)
> cov.mtx
$cupm
      x      y
x 0.4299250 0.1033601
y 0.1033601 0.5411626

$dupm
      x      y
x 0.0000000 0.1469182
y 0.1560924 0.0000000

$dipm
      x      y
x 0.0000000 0.1560924
y 0.1469182 0.0000000

$clpm
      x      y
x 0.4033078 0.1559295
y 0.1559295 0.3939005

$cov.matrix
      x      y
x 0.83323283 -0.04372107
y -0.04372107 0.93506310

# Reassembled Covariance Matrix
> cov.mtx$cupm + cov.mtx$clpm - cov.mtx$dupm - cov.mtx$dipm
      x      y
x 0.83323283 -0.04372107
y -0.04372107 0.93506310

# Standard Covariance Matrix
> cov(cbind(x,y))
      x      y
x 0.83323283 -0.04372107
y -0.04372107 0.93506310
```

Matrix Properties

Positive Semi-Definite

```
> dim>Returns)
[1] 1429  501
> pm_matrices = NNS::PM.matrix(1, 1, "mean", Returns, pop_adj = T)
> matrixcalc::is.positive.semi.definite(pm_matrices$clpm)
[1] TRUE
> matrixcalc::is.positive.semi.definite(pm_matrices$cupm)
[1] TRUE
```


Transposes of One Another

```
> pm_matrices = NNS::PM.matrix(1, 1, "mean", mag_7_returns, pop_adj = T)
> pm_matrices
$cupm
      AAPL      AMZN      GOOGL      META      MSFT      NVDA      TSLA
AAPL 0.0001845879 0.0001257108 0.0001172715 0.0001499652 0.0001290853 0.0001904245 0.0001842340
AMZN 0.0001257108 0.0002194009 0.0001282581 0.0001768822 0.0001294801 0.0001975487 0.0001893952
GOOGL 0.0001172715 0.0001282581 0.0001814092 0.0001648536 0.0001303299 0.0001885374 0.0001629186
META 0.0001499652 0.0001768822 0.0001648536 0.0003509361 0.0001520305 0.0002415405 0.0002034251
MSFT 0.0001290853 0.0001294801 0.0001303299 0.0001520305 0.0001633589 0.0002029688 0.0001696033
NVDA 0.0001904245 0.0001975487 0.0001885374 0.0002415405 0.0002029688 0.0005310263 0.0003034602
TSLA 0.0001842340 0.0001893952 0.0001629186 0.0002034251 0.0001696033 0.0003034602 0.0007570225

$dupm
      AAPL      AMZN      GOOGL      META      MSFT      NVDA      TSLA
AAPL 0.000000e+00 1.584127e-05 1.027990e-05 1.529635e-05 7.045037e-06 1.889708e-05 2.855015e-05
AMZN 1.241220e-05 0.000000e+00 1.186534e-05 1.222438e-05 9.328401e-06 2.026998e-05 3.713813e-05
GOOGL 1.049113e-05 8.067339e-06 0.000000e+00 1.055072e-05 7.663992e-06 1.751569e-05 3.895092e-05
META 1.366089e-05 1.565359e-05 1.123151e-05 0.000000e+00 1.383689e-05 2.654724e-05 5.462058e-05
MSFT 6.947570e-06 7.269397e-06 6.539639e-06 1.220053e-05 0.000000e+00 1.139308e-05 3.088400e-05
NVDA 1.607725e-05 2.083073e-05 1.837323e-05 2.652328e-05 1.397346e-05 0.000000e+00 4.768053e-05
TSLA 3.725337e-05 4.413379e-05 4.755441e-05 6.066748e-05 4.622815e-05 6.768202e-05 0.000000e+00

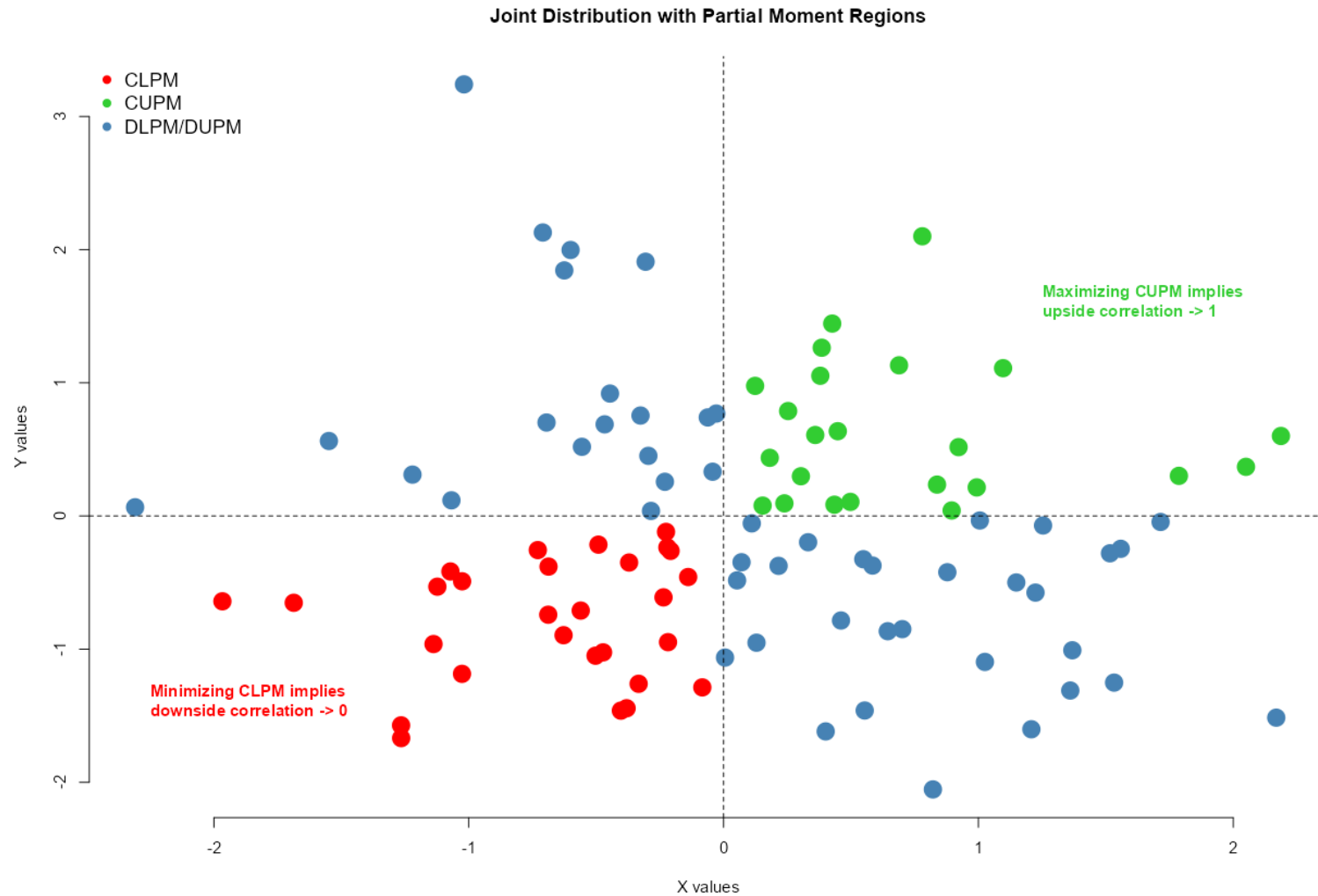
$d1pm
      AAPL      AMZN      GOOGL      META      MSFT      NVDA      TSLA
AAPL 0.000000e+00 1.241220e-05 1.049113e-05 1.366089e-05 6.947570e-06 1.607725e-05 3.725337e-05
AMZN 1.584127e-05 0.000000e+00 8.067339e-06 1.565359e-05 7.269397e-06 2.083073e-05 4.413379e-05
GOOGL 1.027990e-05 1.186534e-05 0.000000e+00 1.123151e-05 6.539639e-06 1.837323e-05 4.755441e-05
META 1.529635e-05 1.222438e-05 1.055072e-05 0.000000e+00 1.220053e-05 2.652328e-05 6.066748e-05
MSFT 7.045037e-06 9.328401e-06 7.663992e-06 1.383689e-05 0.000000e+00 1.397346e-05 4.622815e-05
NVDA 1.889708e-05 2.026998e-05 1.751569e-05 2.654724e-05 1.139308e-05 0.000000e+00 6.768202e-05
TSLA 2.855015e-05 3.713813e-05 3.895092e-05 5.462058e-05 3.088400e-05 4.768053e-05 0.000000e+00

$s1pm
      AAPL      AMZN      GOOGL      META      MSFT      NVDA      TSLA
AAPL 0.0001761168 0.0001402162 0.0001400667 0.0001632621 0.0001391880 0.0002161162 0.0002228611
AMZN 0.0001402162 0.0002132808 0.0001500598 0.0001929205 0.0001455512 0.0002330055 0.0002182102
GOOGL 0.0001400667 0.0001500598 0.0001857633 0.0001826160 0.0001418376 0.0002097453 0.0001989730
META 0.0001632621 0.0001929205 0.0001826160 0.0003562793 0.0001681899 0.0002635573 0.0002452277
MSFT 0.0001391880 0.0001455512 0.0001418376 0.0001681899 0.0001637322 0.0002146819 0.0002047791
NVDA 0.0002161162 0.0002330055 0.0002097453 0.0002635573 0.0002146819 0.0004638918 0.0003588282
TSLA 0.0002228611 0.0002182102 0.0001989730 0.0002452277 0.0002047791 0.0003588282 0.0006856359

$cov.matrix
      AAPL      AMZN      GOOGL      META      MSFT      NVDA      TSLA
AAPL 0.0003607047 0.0002376736 0.0002365672 0.0002842700 0.0002542807 0.0003715663 0.0003412916
AMZN 0.0002376736 0.0004326817 0.0002583852 0.0003419247 0.0002584335 0.0003894535 0.0003263335
GOOGL 0.0002365672 0.0002583852 0.0003671725 0.0003256874 0.0002579639 0.0003623938 0.0002753862
META 0.0002842700 0.0003419247 0.0003256874 0.0007072154 0.0002941830 0.0004520273 0.0003333648
MSFT 0.0002542807 0.0002584335 0.0002579639 0.0002941830 0.0003270911 0.0003922841 0.0002972703
NVDA 0.0003715663 0.0003894535 0.0003623938 0.0004520273 0.0003922841 0.0009949181 0.0005469258
TSLA 0.0003412916 0.0003263335 0.0002753862 0.0003333648 0.0002972703 0.0005469258 0.0014426584

> identical(pm_matrices$d1pm, t(pm_matrices$dupm))
[1] TRUE
```

Logical Objective Functions



Consistent objective for all risk preferences!

Expected Partial Moments

Longer post and paper link on conditioning returns via entropy proxies



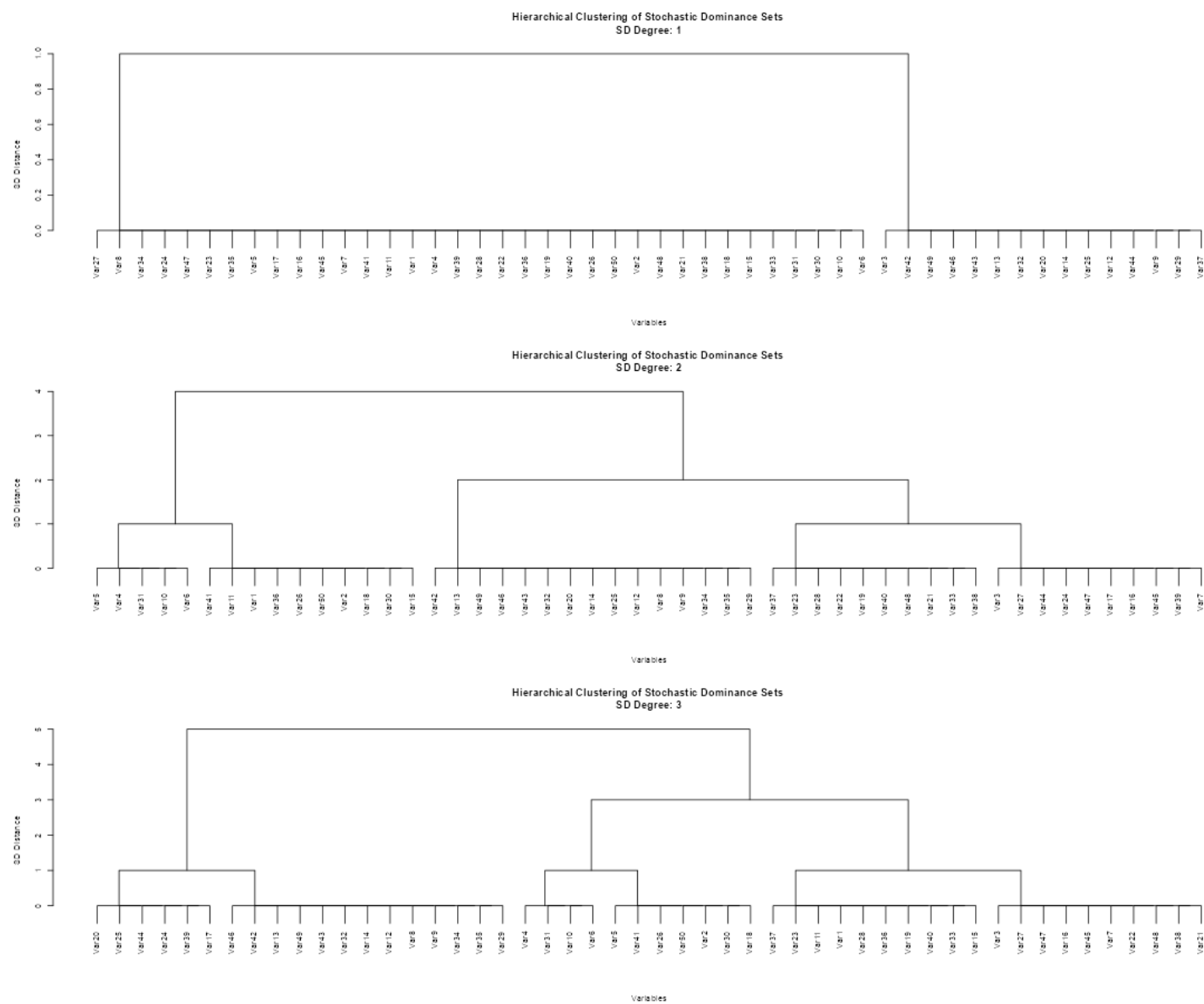
<https://www.linkedin.com/pulse/expected-partial-moments-fred-viole/>



Other Portfolio “Theories”

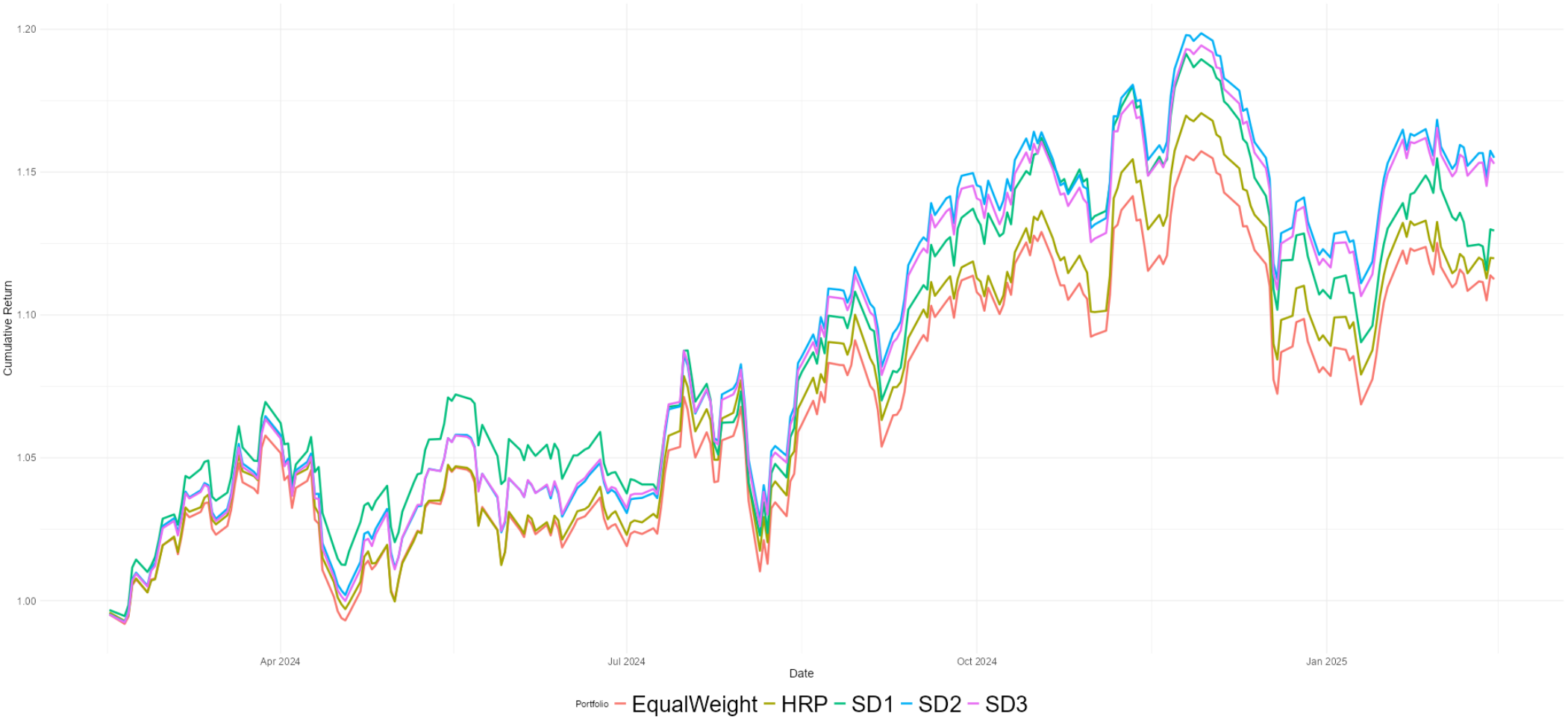
- Diversification for its own sake, with no identified utility function or preference being satisfied.
- Hierarchical Risk Parity (HRP) aims to create portfolios that are more robust to input variations, avoiding extreme mean-variance allocations.
- Stochastic Portfolio Theory (SPT) values growth rates but ignores investor preferences — *how* you achieve final wealth matters!

Stochastic Dominant Clusters



SD Clusters vs. HRP OOS

HRP vs. SD-Based (Clusters Preserved) vs. 1/N Portfolio Performance (Out-of-Sample)



Partial Moments in the Wild

To showcase the effectiveness of NNS in quantitative finance, I've designed the following applications:

- MacroNow nowcasting

https://ovvo.shinyapps.io/macronow_intro

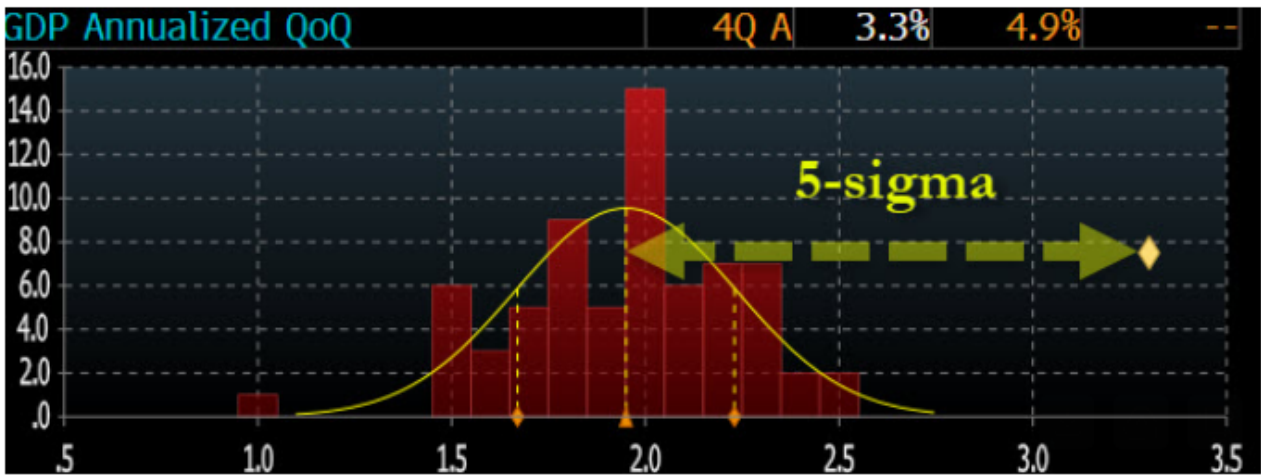
- Options

https://ovvo.shinyapps.io/options_intro

- Portfolio

https://ovvo.shinyapps.io/portfolio_intro

Macro Forecasts



REPORTED THURSDAY, JANUARY 25 AT 8:30 AM

0.1

YOUR DELTA TO ACTUAL

+37

POINTS

1 / 21

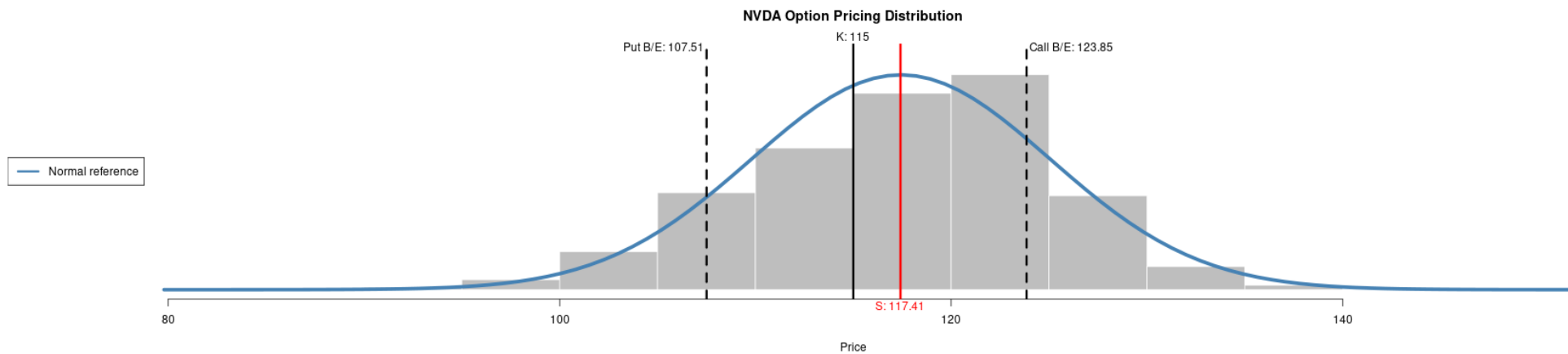
YOUR RANK

>

	Value (%)
Estimize Consensus 21 estimates	1.8
› Your Estimate	3.4
› Reported Values	3.3

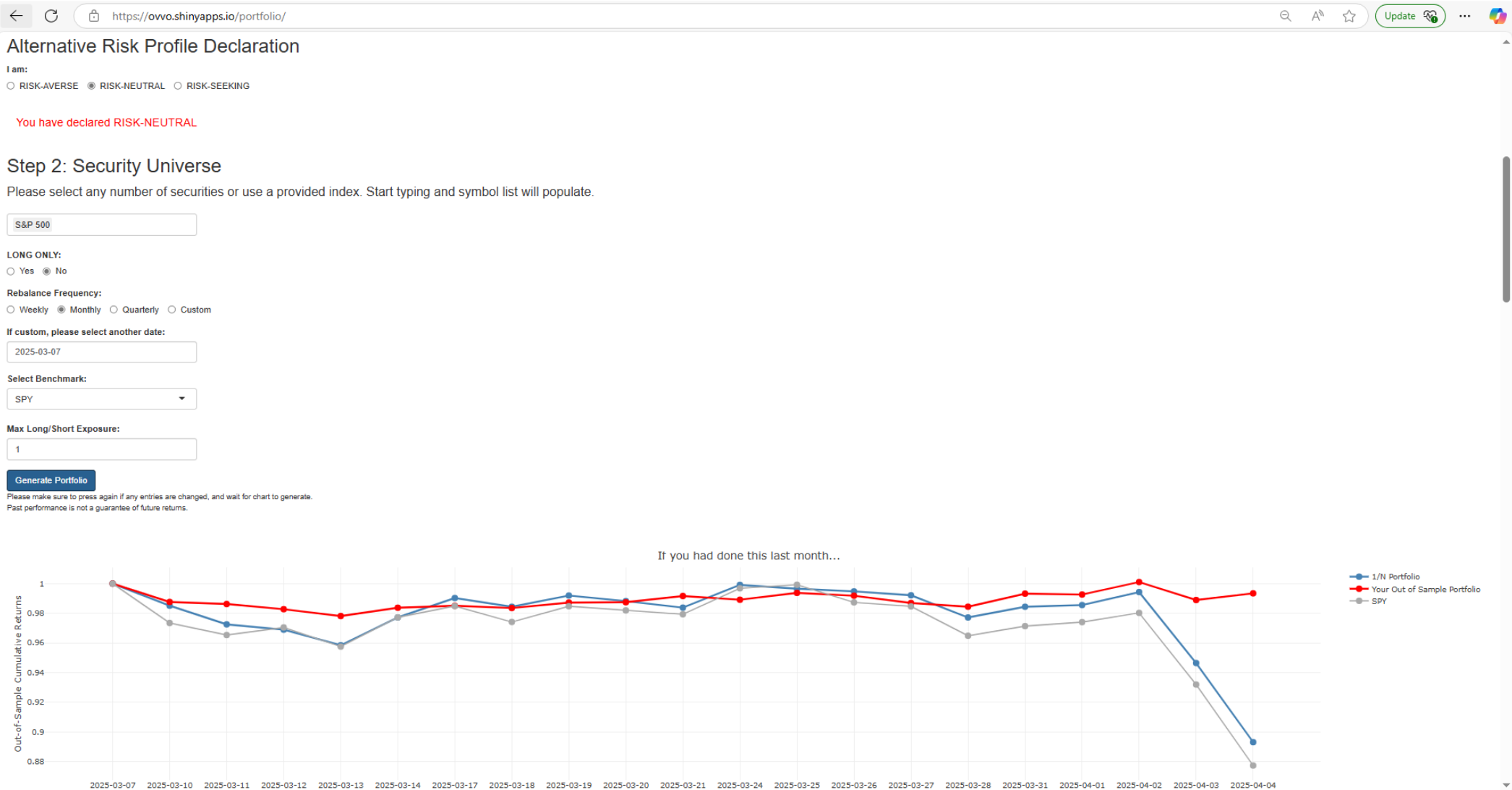
Options Pricing

An obvious extension is to use **UPM** for calls and **LPM** for puts



- Ditches $N(d_1)$ and $N(d_2)$ for nonparametric simplicity
- Integrates real-world ($\mathbb{P}$) shape, adjusts to risk-neutral ($\mathbb{Q}$) directly

Portfolio Tool



More Informed Dispersion Measures

$$RD_t = \sqrt{\sum_{i=1}^N w_i (R_{i,t} - R_{I,t})^2}$$

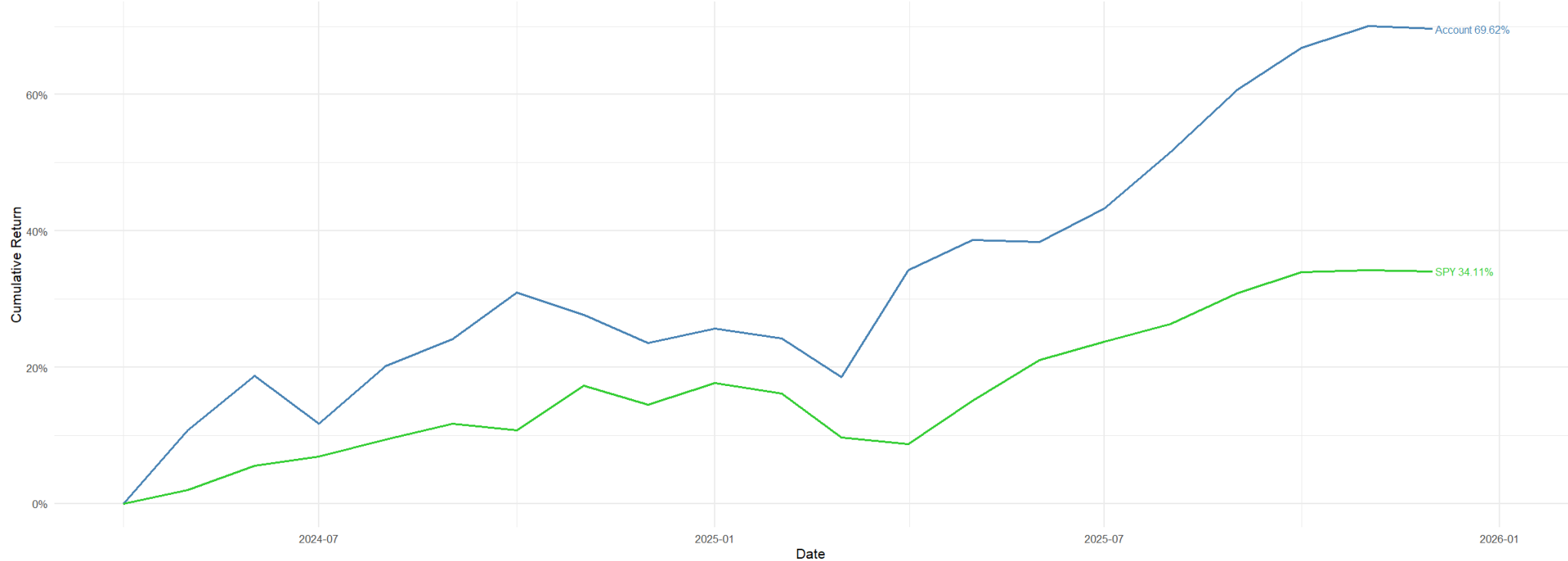
where $R_{I,t} = \sum_{i=1}^N w_i R_{i,t}$ for N index members

$$RD_t^{LPM} = \sqrt{\sum_{i=1}^N w_i [\max(0, R_{I,t} - R_{i,t})]^2}$$

$$RD_t^{UPM} = \sqrt{\sum_{i=1}^N w_i [\max(0, R_{i,t} - R_{I,t})]^2}$$

Better Stats to Arb

Cumulative Monthly Returns - SPY vs. Account U*****
Alpha (ann.): 0.27 | Beta: 0.37 | Corr: 0.21 | Sharpe: 1.85
SPY Max DD: 7.57% | Account Max DD: 9.51%
Account Annualized Returns: 37.31%
YTD 2025-12-19 - Account: 37.23% | SPY: 17.07%



Follow on LinkedIn



<https://www.linkedin.com/company/ovvo-labs/>