Optimal Transport and Dynamic Programming in Financial Time Series Analysis

by

Joseph Bunster

of the requirements for the degree of

Masters of Science

Department of Mathematics

New York University

May, 2022

Professor Esteban G. Tabak

Abstract

In this Masters Thesis, we undertake some preliminary work on a way to apply ideas from data-driven optimal transport and dynamical programming to the analysis of time series arising in finance. I'd like to return to these ideas after graduation, maybe as a Ph.D. student. The future application would leverage ideas from Optimal Transport and Dynamic Programming in attempt to find a successful trading strategy. The main component of the application would utilize the solution to the data-driven optimal transport barycenter problem discussed in the research paper "Distributional Barycenter Problem Through Data Driven Flows" written by Esteban G. Tabak, Giulio Trigila, and Wenjun Zhao. This method is chosen as it is an improvement from previous approaches and allows for more generalized cost functions and a single minimization instead of a min max optimization which the problem usually entails and is inherently more difficult to solve. I aim to provide the preliminary work for how it can be applied to financial time series. Below, I describe how this approach could be applied to several numerical examples, investigating: Apple, Tesla, Google, and Amazon (AAPL, TSLA, GOOGL, and AMZN) data sets (collected from Tiingo API) in a Jupyter Notebook using a Euclidean cost function to forecast these time series. Once the conditional probability distribution of tomorrow's stock returns is discovered, dynamic programming can be applied to optimize potential trading profit. The theoretical approach to the Optimal Transport problem has been established since the 1930's, however there are few numerical implementations of this theory to financial time series. This is likely due to the computational cost required to solve this problem coupled with a large time series (years) that the model would

typically be trained on.

Contents

List of Figures

1 Purpose and Significance of the Study

Optimal Transport provides us a rich framework to move from one probability distribution to another. It has been applied to many areas of mathematics, science, engineering, data analysis, and machine learning with great success, but the computational burden that arises from analyzing large data sets limits what can be accomplished. In this Thesis, I perform preliminary work to investigate historical stock information in attempt to forecast future behavior of these time series. The Purpose of this study is to lay the map for how the financial trading application could be used in the future. The goal in time series analysis is to uncover hidden variables in the data and to thereby create accurate forecasts of that time series. As Optimal Transport could be used to predict future stock return behavior, dynamic programming can also be coupled with this approach to maximize trading efficiency. The implementation of this approach is ongoing and is something I wish to investigate as a potential dissertation topic if I pursue my PhD.

Given 2 distinct probability measures ρ, μ, Optimal Transport looks to find a mapping T which minimizes the total cost of transporting probability measures $\rho \rightarrow \mu$ while satisfying the Push Forward Condition, $\mu = T_{\#}\rho$. Usually, the conditions imparted on the problem help dictate an appropriate choice of cost function, but since we are focused on numerical implementations, I would focus on simpler cost functions like the canonical cost function a.k.a. the Euclidean distance.

Definition 1.1. The Euclidean cost function:

$$C(x, y) := \frac{1}{2}(\sum_{j=1}^{n} |x_j - y_j|^2)^{\frac{1}{2}}$$

Methods utilizing Optimal Transport can make powerful density and conditional density estimators and can be used to make forecasts on time series. Since data analysis relies heavily on the properties of the underlying probability distributions, Optimal Transport can provide a beneficial way of viewing your data. Historically, data computation has been limited by technology and therefore this approach was not often used in this field. The significance of this study aims to encourage more community investigation into this application as computational power is exponentially increasing with time and this may produce powerful results if given enough data.

Financial mathematicians may benefit a great deal if this method produces meaningful insights. The ability to uniquely and accurately estimate the true conditional probability density function of the financial time series' return distribution would directly help produce excessive returns or "alpha" for the investor utilizing the approach. The way this is done is by training the conditional kernel density estimation on the training data to find $p(x|z)$. Then it is mapped to the unknown distribution $\mu(y)$ which is the estimation for the true conditional probability distribution we aim to solve for while making sure to minimize the total sum of the transportation cost. Once this is completed, we can use this forecast to decide how to optimally trade in the market.

2 | Preliminaries

The original problem of Optimal Transport dates back to Monge in the 1781. He posed the question what is the best way to move a pile of soil to another location while minimizing the total transportation cost. $c(x, y)$ is the cost for moving each particle from location to a new location. Since we wish to investigate properties of probability distributions, we can normalize each of these piles so that their masses are equal to 1. Let X and Y be Radon spaces (i.e. any probability measure on X or Y is a Radon measure) and ρ, μ are probability measures. That is $\rho \in P(X) \rightarrow \mu \in P(Y)$. Monge formulated the Optimal Transport problem as follows

Definition 2.1. Monge's formulation:

$$inf\{\int_X c(x, T(x))d\mu(x)|T_{\#}(\mu) = \rho\}$$

Here the $T_{\#}(\mu) = \rho$ enforces the push forward condition and taking the infimum of this integral yields our Optimal Transport path. Sometimes the problem is ill posed and therefore having a dual problem is beneficial. Leonid Kantorovich, a soviet mathematician and economist reformulated the problem as finding the probability measure γ on X x Y which attains its infimum.

Definition 2.2. Kantorovich formulation:

$$inf\{\int_{XxY} c(x, y)d\gamma(x, y)|\gamma \in \Gamma(X, Y)\}$$

With an increase in data and rise in computing power over recent years, solutions to problems which seek to estimate some unknown true probability distribution while only observing a sample from that true distribution are improving. The problem of Optimal Transport can be seen below.

Definition 2.3. The Optimal Transport Barycenter problem:
Using the observed probability distribution $\rho(z|x)$, this problem finds the unknown probability distribution $\mu(y)$ by minimizing the transportation cost

$$min_T \int \int c(x, T(x, z))p(x, z)dxdz$$

$$s.t. \ \forall \ zT\, p(\cdot|z) = \mu$$

and the dual problem:

Definition 2.4. Dual sample based Kantorovich formulation of Optimal Transport:

$$min_T max_F L = \int c(x, T(x, z))p(x, z)dxdz + \int (F(y, z) - E_z[F])p_T(y, z)dydz$$

The first part of this function we define as L_C and the second part is defined as L_F. Here, L_F test the enforcement of the Push Forward Condition which is designed to penalize any dependence of $p_T(y|z)$ on z.

where $\Gamma(X, Y)$ is the collection of probability measures on X x Y with marginals μ on X and ρ on Y. Benamou and Brenier proposed an algorithm for studying Optimal Transport on specific Riemmanian Manifolds which uses the densities themselves to make predictions. We however only have samples points of these underlying densities so this will not directly tie over to our problem.

Definition 2.5. Dynamic Programming: A method for mathematical optimization and computer programming. Dynamic programming was developed in the 1950's by Richard Bellman and the purpose is to break larger complex problems down into smaller sub-problems, then solving those sub-problems recursively. The relationship between these smaller sub-problems and the original larger problem are captured in the Bellman equation which has major applications in Optimal Control theory.

Optimal Transport will help predict future stock behavior and the dynamic programming will use the results to optimize the potential trading profits. More details on this will be discusses in the Methodology Section of this Thesis.

3 | Methodology

A method that could be used to investigate the Optimal Transport Problem and related barycenter problem was proposed by Dr. Esteban Tabak and it is based on gradient flows. This approach uses Conditional Kernel Density Estimation to transform data samples into probability density functions and evolve them through a gradient descent optimization process. The advantage of this approach is that is is essentially parameter free, with the only parameter being the choice of bandwidth for the kernel which is a smoothing parameter. Since we are dealing with stock returns, not historical prices, a Gaussian Kernel function is used as stock returns usually follow a somewhat Gaussian distribution even though the tails are usually underestimated.

Definition 3.1. A Gaussian Kernel between 2 data points takes the form

$$K_\alpha(z_k - z_i) = \frac{1}{\sqrt{2\pi}h} e^{\frac{-||z_k - z_i||^2}{2h^2}}$$

where h is our choice of bandwidth which is effectively a smoothing parameter. We will compare Scott's rule of thumb and Silverman's rule of thumb for our bandwidth selection. More information on these bandwidth choices will be discussed later on.

An example of some data that could be investigated is the data from stocks such as Apple, Google, Tesla, and Amazon. The application would load financial data from Yahoo finance and parses out features of the data including: "open price", "close price", "high", "low", "Adjusted close",

"Volume", "adjusted high", "adjusted low", "adjusted open", "adjusted volume", "dividend cash", and "split factor" then stores it into a Jupyter Notebook. The Jupyter Notebook runs in the following order on the imported data sets where it starts by calculating and plotting returns, histogram of the returns, Kernel Density estimation of the probability density of returns, a conditional kernel density on the returns given yesterdays returns, and more.

Below we see what the beginning of an application like this would look like run with 4 separate stocks (AAPL, TSLA, GOOGL, and AMZN). The first image shows the data set imported using the Tiingo API. This is a snapshot of data that goes for 6 months (120 days) June 2021 - December 2021. The second image is a graph of the stock price over this same time period. The third image is a graph of the daily returns of these stocks. The fourth image shows a histogram of these returns to see how likely returns are to occur. The fifth image shows a Gaussian Kernel Density Estimation of the Probability density function for the same returns.

Apple Stock

The Time Series analyzed here is the Apple stock over the past 6 month period (June 2021 - December 2021). Beneath the table, we can see a graph of the stock price. Additionally, there are graphs of the historical returns, histogram plots, and Kernel Density Estimations of the Probability density functions. It can be seen in the fifth image the kernel density estimation for returns has a positive skew and has large tails. Specifically the negative side for returns has the most prominent tail.

	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFa
2021-06-14T00:00:00+00:00	130.4800	130.5400	127.0700	127.8200	96906490	130.0947	130.1545	126.6947	127.4425	96906490	0.0000	1.0
2021-06-15T00:00:00+00:00	129.6400	130.6000	129.3900	129.9400	62746332	129.2572	130.2143	129.0079	129.5563	62746332	0.0000	1.0
2021-06-16T00:00:00+00:00	130.1500	130.8900	128.4610	130.3700	91339351	129.7657	130.5035	128.0816	129.9850	91339351	0.0000	1.0
2021-06-17T00:00:00+00:00	131.7900	132.5500	129.6500	129.8000	96721669	131.4008	132.1586	129.2671	129.4167	96721669	0.0000	1.0
2021-06-18T00:00:00+00:00	130.4600	131.5100	130.2400	130.7100	108953309	130.0747	131.1216	129.8554	130.3240	108953309	0.0000	1.0
2021-06-21T00:00:00+00:00	132.3000	132.4100	129.2100	130.3000	79663316	131.9093	132.0190	128.8284	129.9152	79663316	0.0000	1.0
2021-06-22T00:00:00+00:00	133.9800	134.0800	131.6200	132.1300	74783618	133.5843	133.6840	131.2313	131.7398	74783618	0.0000	1.0
2021-06-23T00:00:00+00:00	133.7000	134.3200	133.2300	133.7700	60214200	133.3052	133.9233	132.8366	133.3750	60214200	0.0000	1.0

Figure 3.1: Apple Description

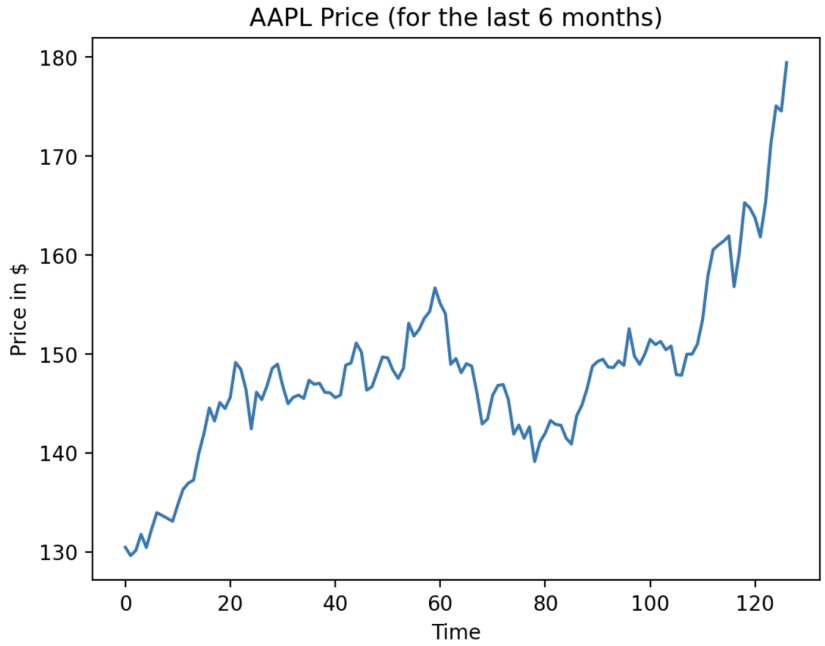

Figure 3.2: Apple Historical Price

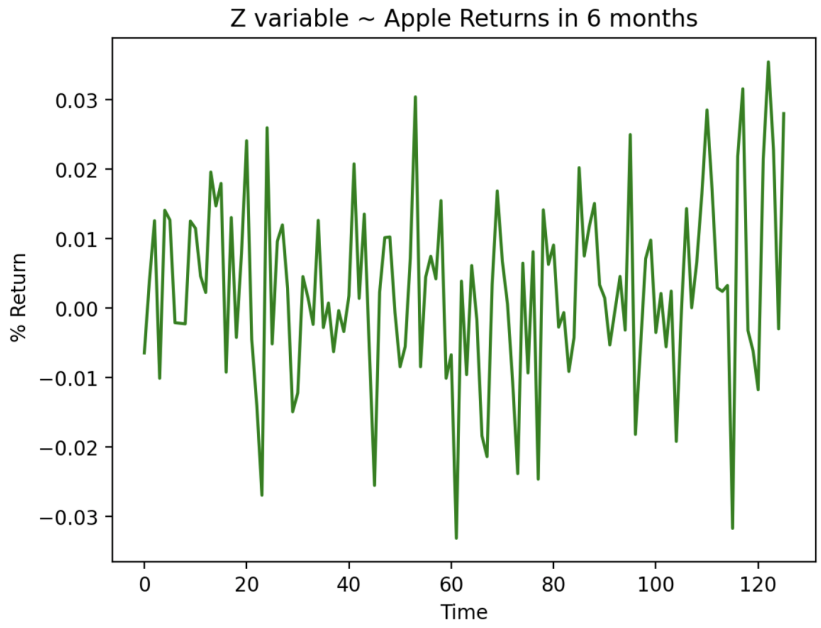

Figure 3.3: Z Variable: Apple Historical Price lagged

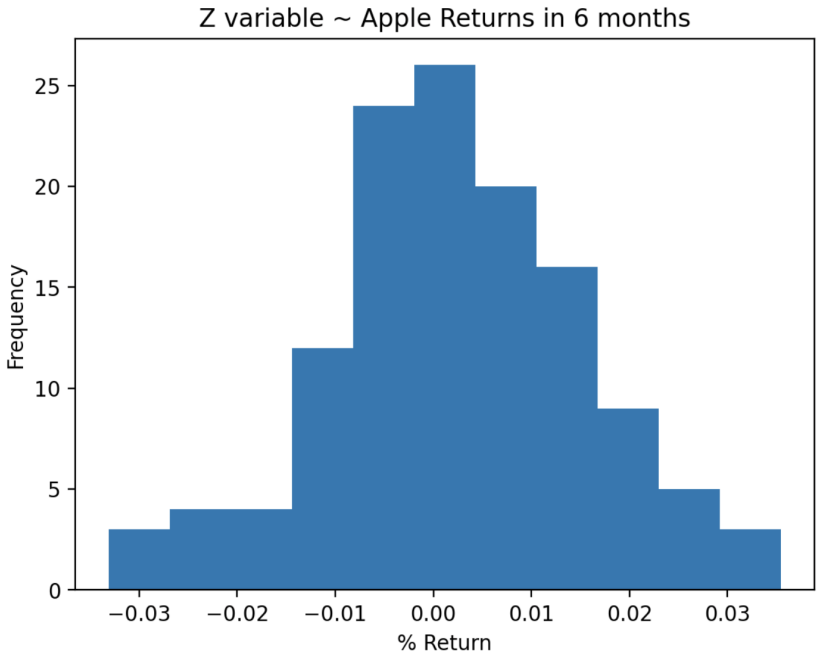

Figure 3.4: Z Variable Histogram

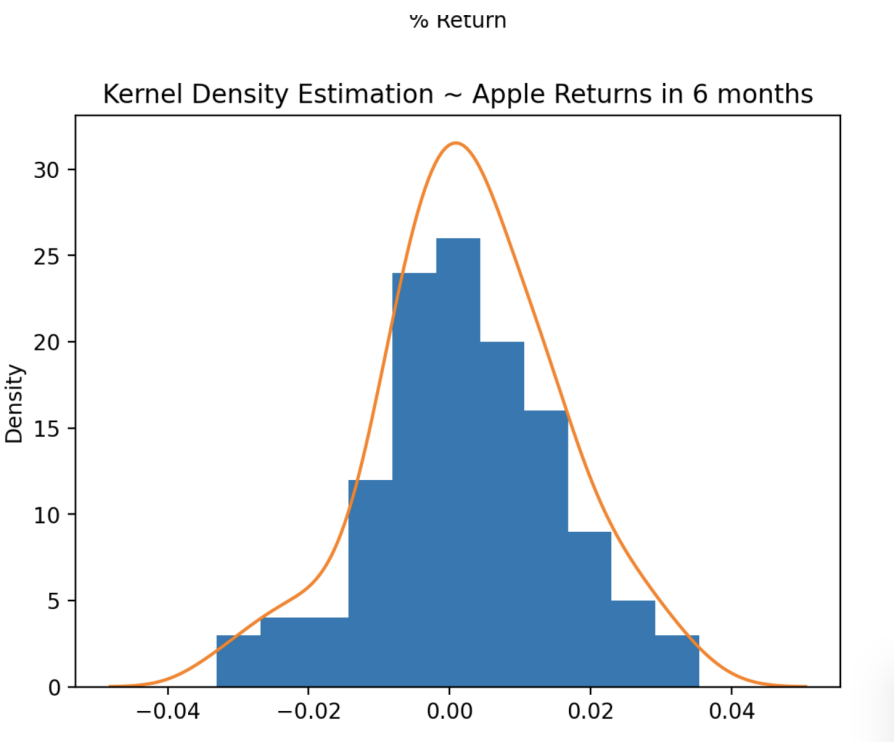

Figure 3.5: Z Variable Kernel Density Estimation

TSLA Stock

The Time Series analyzed here is the Tesla stock over the past 6 month period (June 2021 - December 2021). Beneath the table, we can see a graph of the stock price. Additionally, there are graphs of the historical returns, histogram plots, and Kernel Density Estimations of the Probability density functions. It can be seen in the fifth image the kernel density estimation for returns has a positive skew, appears symmetric and has large tails.

	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash
2021-06-14T00:00:00+00:00	617.6900	625.4900	609.1800	612.2300	20423983	617.6900	625.4900	609.1800	612.2300	20423983	0.0000
2021-06-15T00:00:00+00:00	599.3600	616.7900	598.2300	616.6900	17764145	599.3600	616.7900	598.2300	616.6900	17764145	0.0000
2021-06-16T00:00:00+00:00	604.8700	608.5000	593.5000	597.5350	21845739	604.8700	608.5000	593.5000	597.5350	21845739	0.0000
2021-06-17T00:00:00+00:00	616.6000	621.4700	601.3400	601.8878	22701350	616.6000	621.4700	601.3400	601.8878	22701350	0.0000
2021-06-18T00:00:00+00:00	623.3100	628.3500	611.8000	613.3700	24560905	623.3100	628.3500	611.8000	613.3700	24560905	0.0000
2021-06-21T00:00:00+00:00	620.8300	631.3900	608.8800	624.4800	24812741	620.8300	631.3900	608.8800	624.4800	24812741	0.0000
2021-06-22T00:00:00+00:00	623.7100	628.5693	615.5000	618.2500	19158892	623.7100	628.5693	615.5000	618.2500	19158892	0.0000

Figure 3.6: Tesla Info

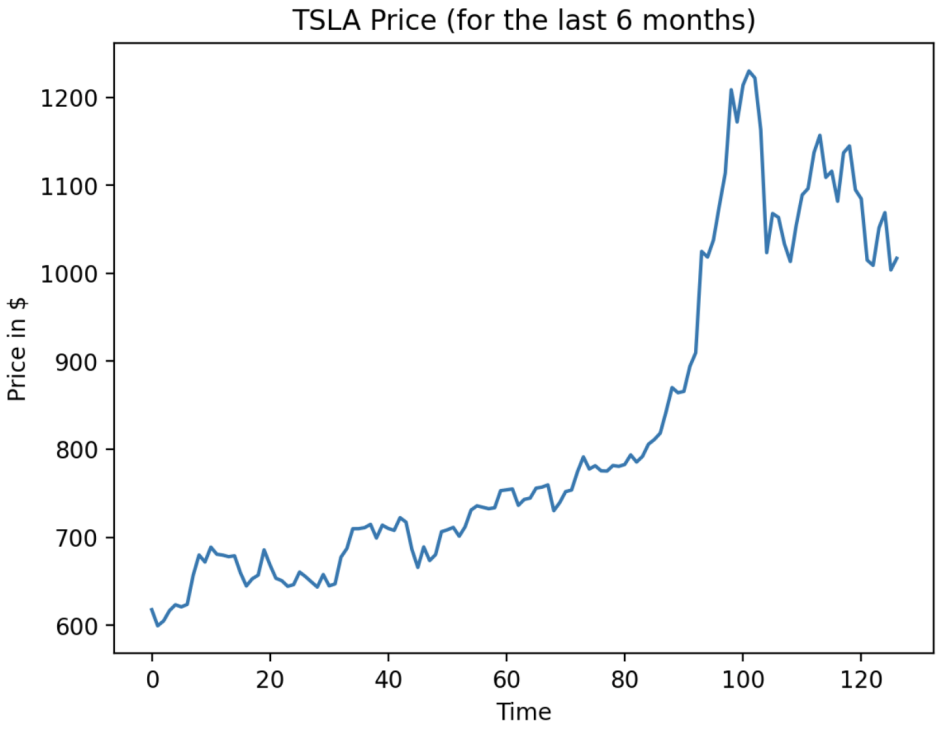

Figure 3.7: Tesla Historical Price

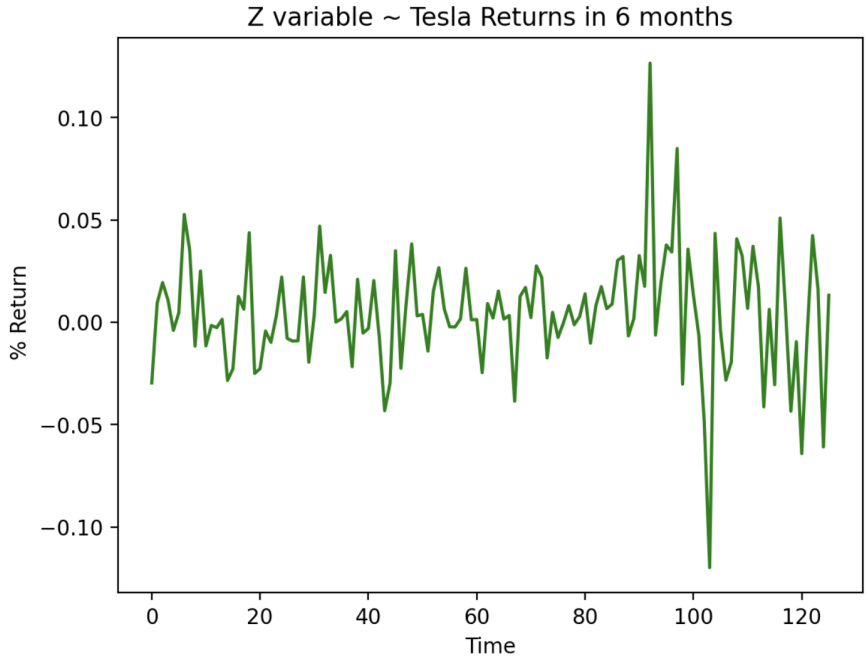

Figure 3.8: Tesla Historical Returns

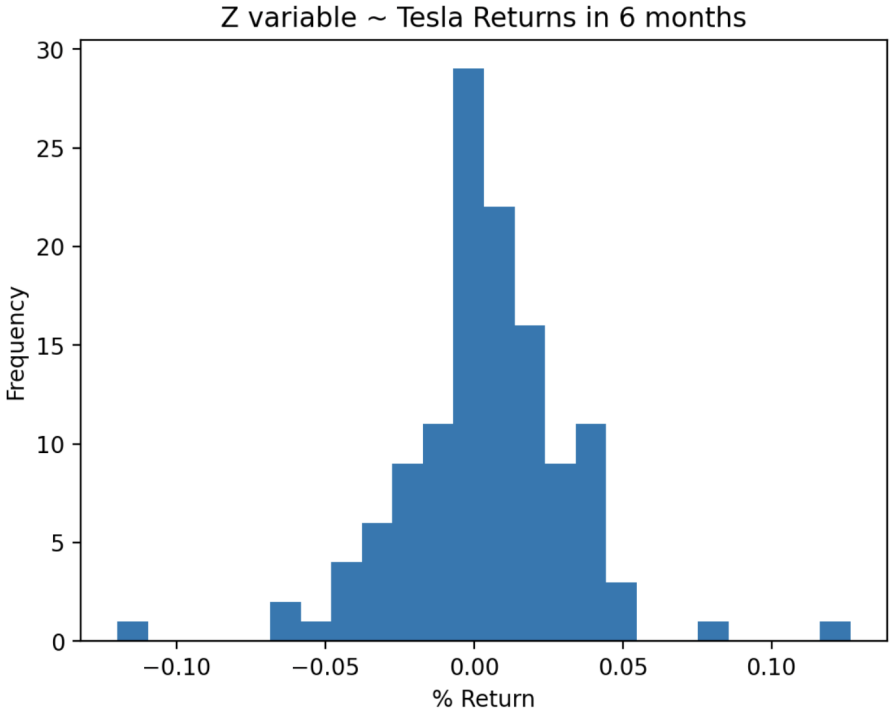

Figure 3.9: Z Variable Returns Histogram

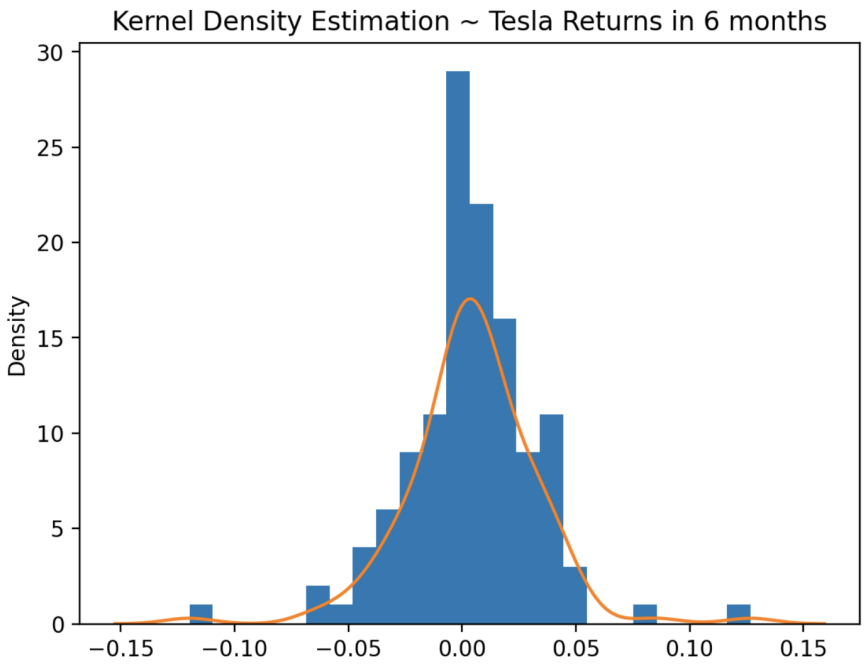

Figure 3.10: Z Variable Kernel Density Estimation

GOOGL Stock

The Time Series analyzed here is the Google stock over the past 6 month period (June 2021 - December 2021). Beneath the table, we can see a graph of the stock price. Additionally, there are graphs of the historical returns, histogram plots, and Kernel Density Estimations of the Probability density functions. It can be seen in the fifth image the kernel density estimation for returns has a positive skew and has large tails. Specifically the positive side for returns has the most prominent tail.

	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash
2021-06-14T00:00:00+00:00	2,527.0400	2,528.2300	2,500.9400	2,513.3900	1127522	2,527.0400	2,528.2300	2,500.9400	2,513.3900	1127522	0.0000
2021-06-15T00:00:00+00:00	2,520.6600	2,537.2400	2,512.9700	2,530.4400	1109130	2,520.6600	2,537.2400	2,512.9700	2,530.4400	1109130	0.0000
2021-06-16T00:00:00+00:00	2,513.9300	2,530.4700	2,482.9994	2,524.9500	1307168	2,513.9300	2,530.4700	2,482.9994	2,524.9500	1307168	0.0000
2021-06-17T00:00:00+00:00	2,527.4200	2,543.9300	2,510.3000	2,510.4600	1287791	2,527.4200	2,543.9300	2,510.3000	2,510.4600	1287791	0.0000
2021-06-18T00:00:00+00:00	2,511.3500	2,527.7800	2,492.0600	2,514.1100	2665310	2,511.3500	2,527.7800	2,492.0600	2,514.1100	2665310	0.0000

Figure 3.11: Google Info

Figure 3.12: Google Historical Price

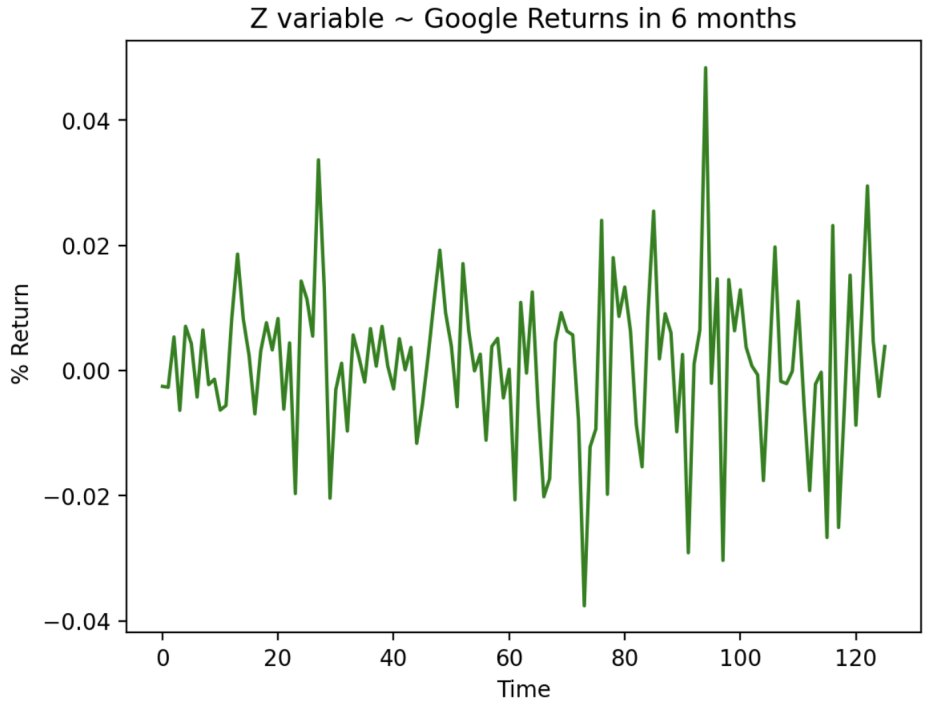

Figure 3.13: Google Returns

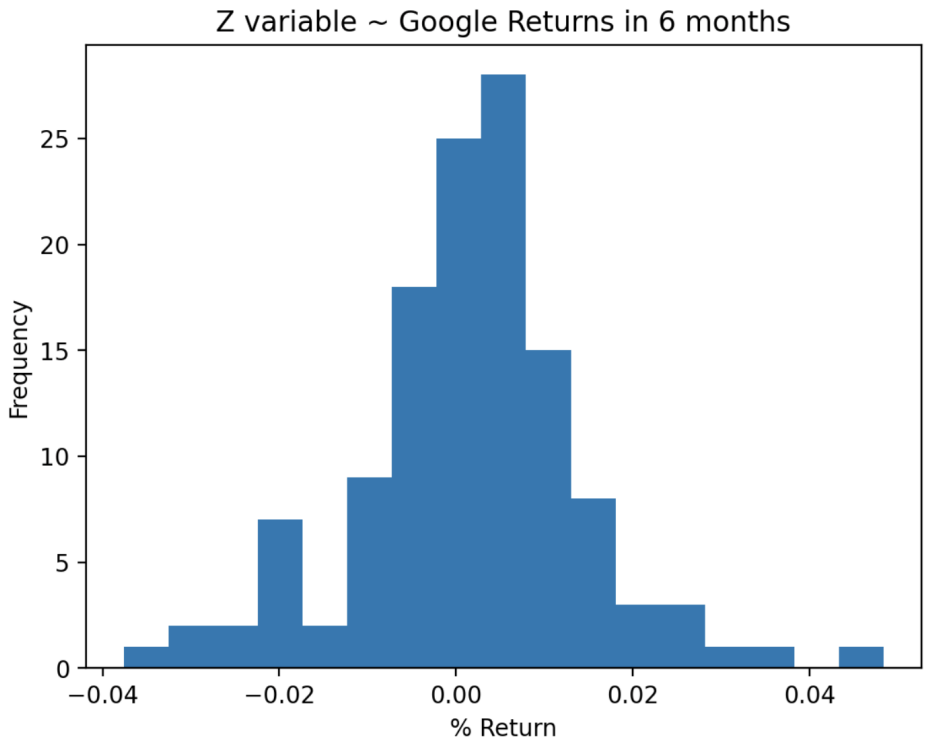

Figure 3.14: Google Histogram

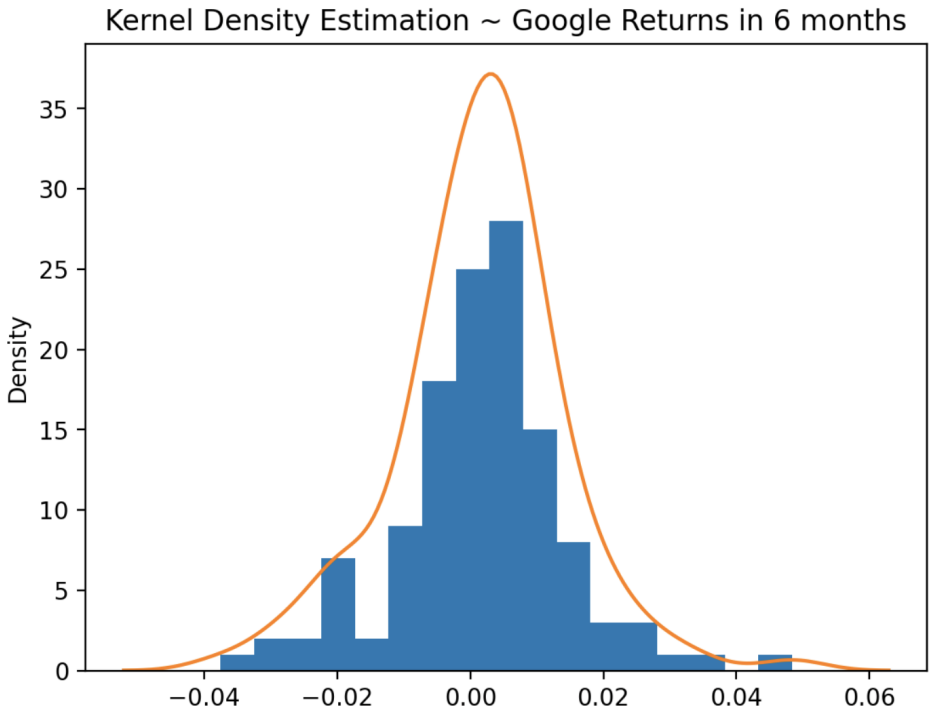

Figure 3.15: Z Variable Kernel Density Estimation

AMAZN Stock

The Time Series analyzed here is the Amazon stock over the past 6 month period (June 2021 - December 2021). Beneath the table, we can see a graph of the stock price. Additionally, there are graphs of the historical returns, histogram plots, and Kernel Density Estimations of the Probability density functions. It can be seen in the fifth image the kernel density estimation for returns has a positive skew and has large left tail.

	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash
2021-06-14T00:00:00+00:00	3,383.8700	3,385.0000	3,335.5000	3,346.8300	2569655	3,383.8700	3,385.0000	3,335.5000	3,346.8300	2569655	0.0000
2021-06-15T00:00:00+00:00	3,383.1300	3,396.9866	3,363.1101	3,384.0000	2426200	3,383.1300	3,396.9866	3,363.1101	3,384.0000	2426200	0.0000
2021-06-16T00:00:00+00:00	3,415.2500	3,426.3500	3,360.5300	3,392.0000	4152189	3,415.2500	3,426.3500	3,360.5300	3,392.0000	4152189	0.0000
2021-06-17T00:00:00+00:00	3,489.2400	3,497.2000	3,401.0000	3,403.1800	5136531	3,489.2400	3,497.2000	3,401.0000	3,403.1800	5136531	0.0000
2021-06-18T00:00:00+00:00	3,486.9000	3,507.0000	3,473.7100	3,479.9900	5247737	3,486.9000	3,507.0000	3,473.7100	3,479.9900	5247737	0.0000
2021-06-21T00:00:00+00:00	3,453.9600	3,482.0000	3,434.0000	3,476.4200	3277130	3,453.9600	3,482.0000	3,434.0000	3,476.4200	3277130	0.0000
2021-06-22T00:00:00+00:00	3,505.4400	3,523.7800	3,456.0900	3,458.0600	3345082	3,505.4400	3,523.7800	3,456.0900	3,458.0600	3345082	0.0000

Figure 3.16: Amazon Info

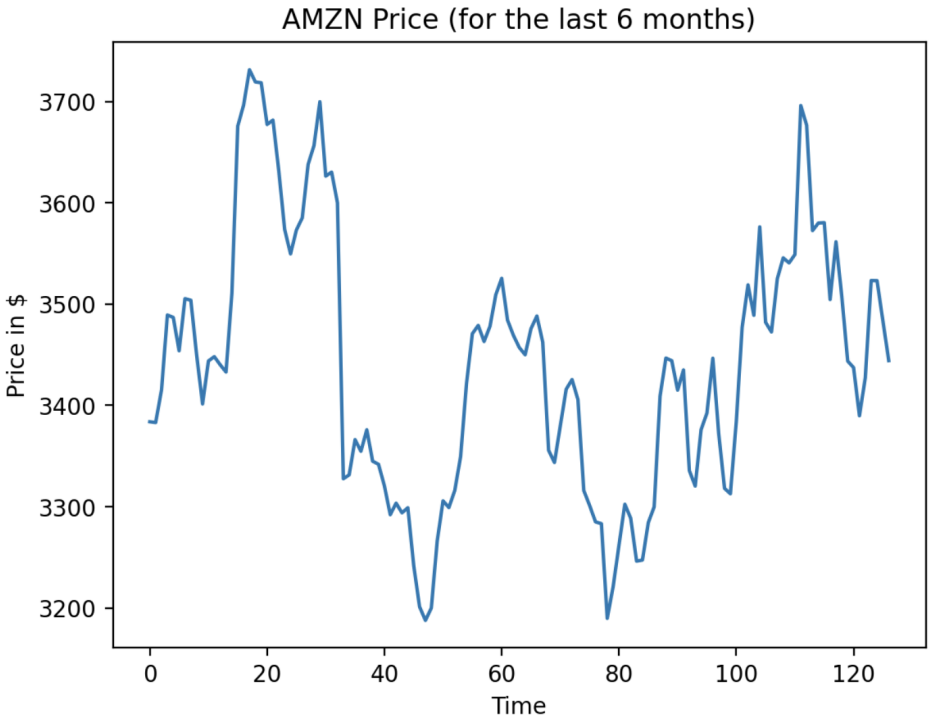

Figure 3.17: Amazon Historical Price

Figure 3.18: Amazon Returns

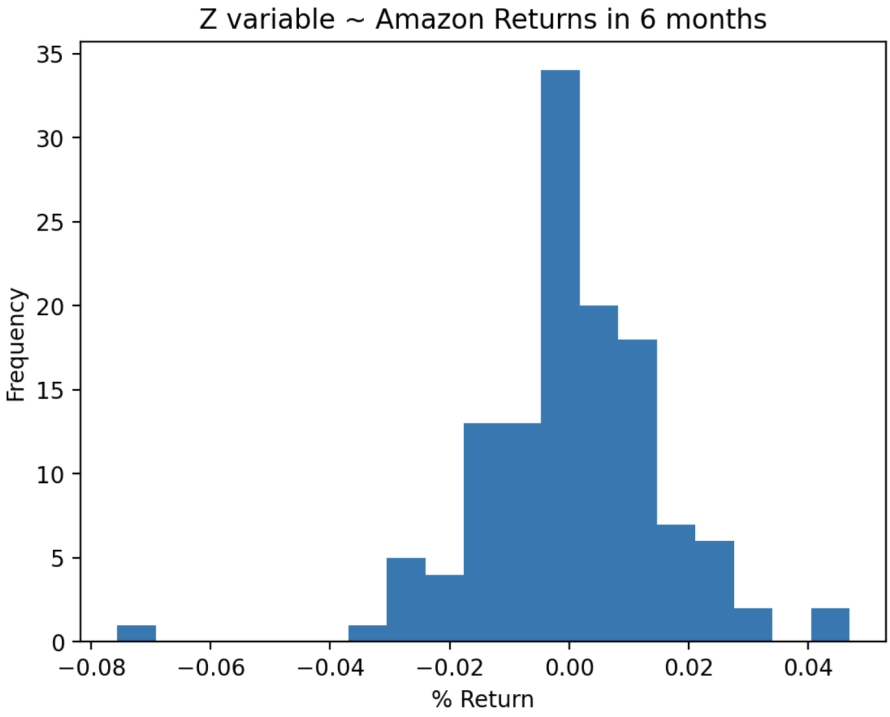

Figure 3.19: Amazon Histogram

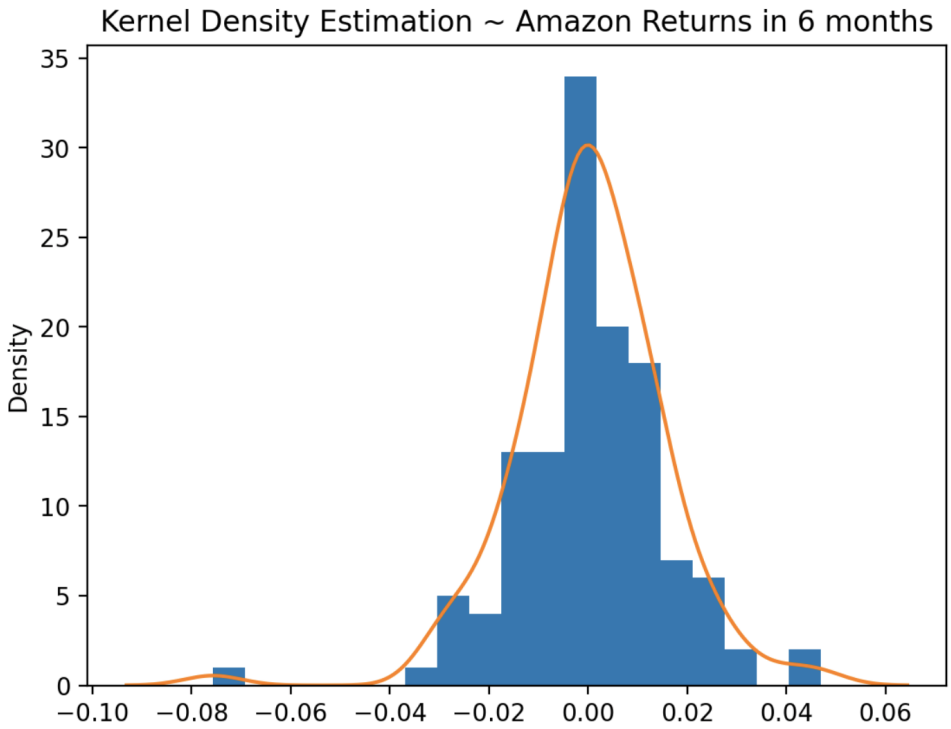

Figure 3.20: Z Variable Kernel Density Estimation

As mentioned earlier, the data samples discussed are historical returns of 4 different stocks (Apple, Tesla, Google, Amazon). X is labeled as the historical returns, and z is the lagged returns. After computing the lagged returns, we want to make a Matrix K such that the entry $K_{ik} = K_b(z_k - z_i)$

This matrix is symmetric and positive definite as each entry must be positive and the distance between $d(z_i - z_k) = d(z_k - z_i)$. Since we are concerned with probabilities, the next set is to force this symmetric positive definite matrix to be Bi-stochastic.

Definition 3.2. Symmetric Matrix:

$$\forall(i,k); K_{i,k} = K_{k,i}$$

Definition 3.3. Positive Definite Matrix:

$$\forall z \in R^n; z^T K z \geq 0$$

The goal of these steps so far is to compute the test function L_F which enforces the very important Push Forward Condition. But this is only half of the objective, as we also need to compute the L_C which is the cost associated with the Optimal Transport map. A cost function is a function that measures the performance of a Machine Learning model for given data. Cost function quantifies the error between predicted values and expected values and presents it in the form of a single real number. This procedure to solve the optimal transport problem relies on gradient descent to optimize its parameters and therefore the cost function needs to be differentiable. Depending on the problem Cost Function can be formed in many different ways. The purpose of Cost Function is to be either:

1.) Minimized - then returned value is usually called cost, loss or error. The goal is to find the values of model parameters for which Cost Function return as small number as possible.

2.)Maximized - then the value it yields is named a reward. The goal is to find values of model parameters for which returned number is as large as possible.

2 measures of model performance in machine learning are MAE and MSE. MAE is Mean Absolute Error, it measures how close predictions are to the outcomes and it is calculated as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

MSE is Mean Squared Error, it measures the squared error loss of the predictions and measures the quality of an estimator. MSE is also the second moment of the error. It is calculated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

These scores can be used to compare actual data against forecasted data and come up with a score of how well the forecasts fits the actuals. We can use these numbers to compare against other forecasting algorithms and measure their performances against one another. This is an important step when choosing a model.

According to Dr. Tabak, "A natural way to estimate $F(y,z) = \rho_T(y|z)$ from these samples is through a conditional kernel density estimation (CKDE) in the Nadaraya- Watson form:"

Definition 3.4. Test function enforcing the push forward condition $T_{\#}(\rho(x)) = \mu(y)$:

$$L_F = F(y,z_k) = p_T(y,z_k) \approx \frac{\sum_i K_\alpha(y,y_i)K_\beta(z_k,z_i)}{\sum_j K_\beta(z_k,z_j)} = \sum_i^N K_\alpha(y,y_i)Z_{ik}$$

For computational efficiency, $C_{il} = Z_{il} - \frac{1}{N}\sum_k Z_{ik}$ should be pre-computed and stored.

Bandwidth selection strongly influences the estimate obtained from the KDE (much more so than the actual shape of the kernel). Bandwidth selection can be done by a “rule of thumb”, by cross-validation, by “plug-in methods” or by other means. The Gaussian KDE uses a rule of thumb, the default is Scott’s Rule.

Definition 3.5. Scott's Rule of Thumb for Bandwidth Selection:

$$Scott'sBandwidth = n^{\frac{-1}{d+4}}$$

n: Number of Data points

d: Dimension of the data

Another option for choosing a bandwidth is to use Silverman's Rule of Thumb.

Definition 3.6.

$$Silverman'sBandwidth = (n * \frac{(d+2)}{4})^{\frac{-1}{d+4}}$$

n: Number of Data points

d: Dimension of the data

In a future application, this Conditional Kernel Density Estimation will be applied to each stock. Then one could create an image that compares the smoothness of the resulting Conditional Kernel Density Estimation using Scott's rule of thumb, Silverman's rule of thumb, and a constant $\frac{Silverman's}{3}$. The last choice is rougher as a smaller bandwidth yields bumpier curves. Please refer to the graphs directly beneath to see how different bandwidths change the results for the Kernel Density Estimation of each stock. Note the Conditional Kernel Densities for each stock have not been computed as this is still ongoing work for the future.

Apple

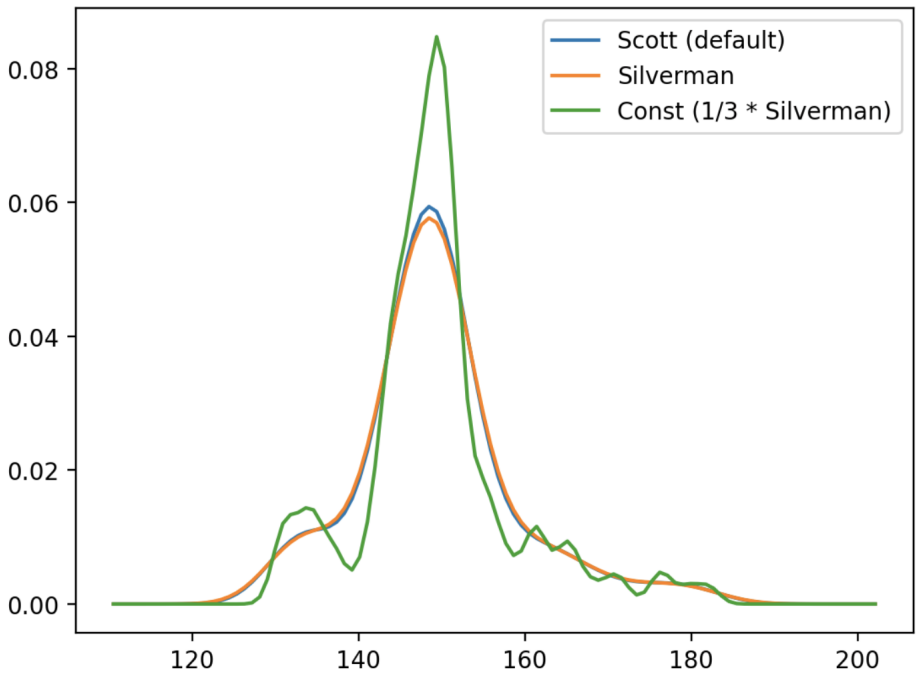

Figure 3.21: Smoothed Kernel Density Estimation: Apple

Tesla

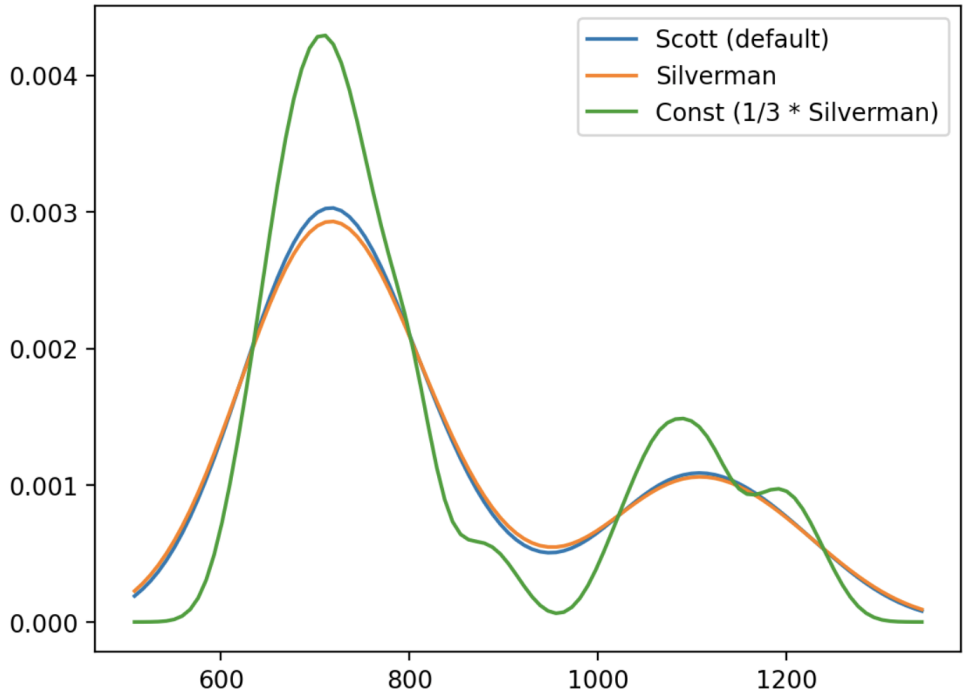

Figure 3.22: Smoothed Kernel Density Estimation: Tesla

Google

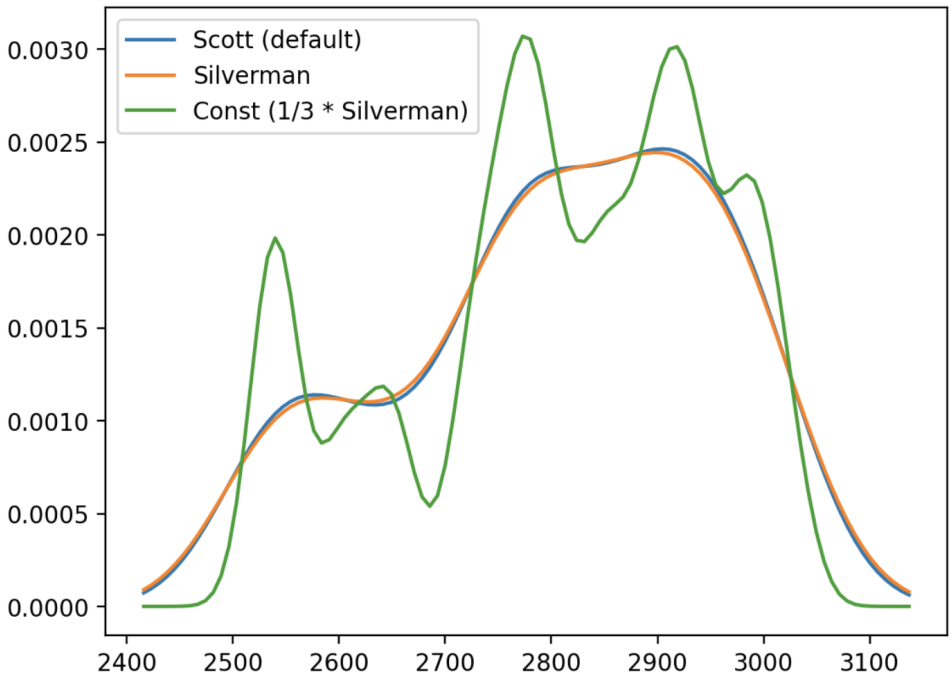

Figure 3.23: Smoothed Kernel Density Estimation: Google

Amazon

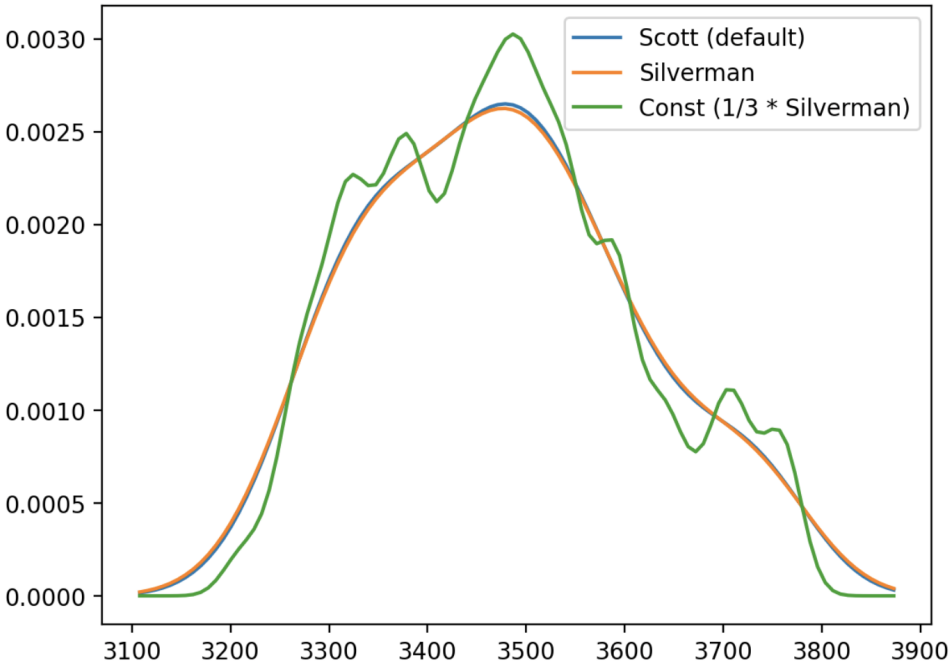

Figure 3.24: Smoothed Kernel Density Estimation: Amazon

Bi-stochastic Matrix with Sinkhorn

An alternative conditional density estimator

Given that the following matrix $Z \in R^{N*N}$ is a normalized version of similar kernels in z-space, $Z_{i,k} = \frac{K_b(z_k,z_i)}{\sum_{j=1}^{N} K_b(z_k,z_j)}$ is not the only option for a robust conditional estimator. The main requirements for Z is that the entries $Z_{i,k}$ must be non-negative and add to zero row wise to guarantee the estimated $\rho_T(y|z_k)$ is positive and integrates to 1. Therefore, a symmetric matrix Z with non-negative entries who's rows add to 1is necessarily bi-stochastic, a natural candidate is the unique bi-stochastic matrix Z that derives from the symmetric and positive Kernel matrix $K_{i,k} = K_b(z_k, z_i)$ through Sinkhorn's factorization: $Z = DKD$, where D is a diagonal matrix with positive diagonal entries.

Definition 3.7. Bi-stochastic matrix: $\forall i, k$;

$$\sum_i K_{i,k} = 1$$

and

$$\sum_k K_{i,k} = 1$$

To force the matrix $K_{i,k}$ to be bi-stochastic, we can use an adaption of Sinkhorn's Algorithm which finds a diagonal matrix D such that matrix Z is Bi-stochastic.

Definition 3.8. Sinkhorn's algorithm: $Z = DKD$

The data driven problem takes the form

$$min_y max_\lambda \sum_i c(x_i, y_i) + \lambda \sum_{i,l} K_\alpha(y_l, y_i) C_{i,l}$$

Here the Test function which forces the conditional distributions to be mapped to the same barycenter is embedded in the optimization problem. This gives us a single variable

optimization problem instead of a saddle point optimization problem which can me much harder to solve.

The goal is to solve for y, for which we need to use gradient descent. More sophisticated implicit methods have been developed but here one could simplify the process for a test case by explicitly plugging in parameters $\lambda = \frac{1}{2}$, where λ represents the constraint on the objective function, $\eta = \frac{1}{2}$ where η represents the learning rate of the algorithm, $\alpha = 0.1$ where α represents the bandwidth of the kernel.

Gradient Descent Procedure:

Initialize by setting $y_i = x_i$,

Then update the $y's$ as follows:

$$y^{n+1} = y^n - \eta \nabla_y L(y, \lambda)|_{y=y_i^n}$$

Here the Objective function is:

$$L = \sum c(x_i, T(x_i)) + \lambda \sum_{i,k} K_\alpha(y_i, y_k) C_{ik}$$

and the Gradient of the Objective function with respect to y is:

$$\nabla_y L|_{y=y_i^n} = \frac{\partial c(x_i, y)}{\partial y} + \lambda \sum_k \frac{\partial K_\alpha(y, y_i^n)}{\partial y} C_{ik}$$

The Gradient of the Kernel function with respect to y:

$$\frac{\partial K_\alpha(y)}{\partial y} = -\frac{2y}{\sigma^2} e^{\frac{-y^2}{\sigma^2}}$$

Then store the results in variable $y_{history}$

This is the hidden signal that we want to use to make prediction on the behavior of the $x's$ in the data.

Next the application would calculate the Bi-stochastic matrix using Sinkhorn's algorithm and produce a probability distribution of tomorrow's returns given where the stock is today. We can use this information to set up a dynamic programming problem that attempts to maximize potential trading profits. Since the Optimal transport portion of this algorithm only predicts a distribution of stock returns for tomorrow, the problem of maximizing profit can be broken down into day by day choices that follow the same pattern and can be iterated for any amount of time.

From the initial day the application is run Day_0, the Optimal Transport segment would forecast a distribution of likely returns for tomorrow Day_1. Then one can integrate and find the area under the conditional Kernel density estimation for positive returns (PR) and compare them with the area of negative returns (NR). We use the formula below to decide when to make a profitable trade. I have also included a safety factor SF in the model to be sure we only trade when there is a significant difference between the Upside and Downside as to be more conservative in the approach. This is something that would be set by the user of the application depending on their risk tolerances.

Case 1:

If $PR_{Day_1} > NR_{Day_1}$ and

$$PR_{Day_1} - NR_{Day_1} - SF > 0$$

then the application suggests we should purchase the stock at Day_0 and rerun the Optimal Transport prediction again tomorrow at Day_1. If $PR_{Day_2} > NR_{Day_2}$ and

$$PR_{Day_2} - NR_{Day_2} - SF > 0$$

then we should hold off selling on Day_2 or consider buying more stock and rerun the Optimal Transport prediction again for Day_3

But if $PR_{Day_2} < NR_{Day_2}$, and

$$NR_{Day_2} - PR_{Day_2} - SF > 0$$

then sell at Day_1.

Case 2:

If $PR_{Day_1} < NR_{Day_1}$, and

$$NR_{Day_1} - PR_{Day_1} - SF > 0$$

Short the stock by selling it on Day_0 and rerun the Optimal Transport forecast again at Day_1 to see when to buy again. This is essentially the same as Case 1 but here we start off shorting the stock.

Case 3:

If neither the Case 1 nor Case 2 are met, do not buy or sell the stock and wait until more stock movement is predicted.

This can be applied to many stock at once if computational expense was not relevant, in which case trade whichever stocks are expected to have the biggest price swings as these can make for larger profits.

4 Conclusion

This Thesis discusses preliminary work for the future development of a trading application that leverages ideas from the distributional barycenter problem, an extension of the optimal transport barycenter problem for predicting future stock returns and dynamic programming for optimal trading execution. The application would avoid the difficulties that would arise in adversarial approaches by reformulating the problem as a single parameter optimization problem. This would result in a simpler approach that looks for a minimum rather than a saddle point of the objective function. This application may provide significant insight into time series behavior in finance, and ultimately may be used to trade profitably in the financial markets. With the rise in computing power, the problem of computational complexity for this model will be a smaller barrier to overcome. Typically when training models in finance, it is important to take into account both benign and stressed periods in the market to build a well rounded model. This allows a model to be trained on better data which can ultimately make it more reliable. In the future, I plan to build this application, and ultimately investigate how this model performs using a large set of training data.

Bibliography

[1] Esteban G. Tabak, Giulio Trigila, and Wenjun Zhao, *Distributional Barycenter problem through Data-Driven Flows*, Tabak-2021.

[2] Max Kuang and Esteban G. Tabak, *Sample Based Optimal Transport and Barycenter Problems,* Kuang, Tabak, 2017.

[3] D.P. Bourne, *A Brief Introduction to Optimal Transport Theory,* Bourne, 2018.

[4] Soheil Kolouri,*Optimal Transport and Wasserstein Distance,* Kolouri 2018.

[5] Cedric Vilanni, *Topics in Optimal Transport, No. 58. American Mathematical Soc.*, Vilanni 2003.