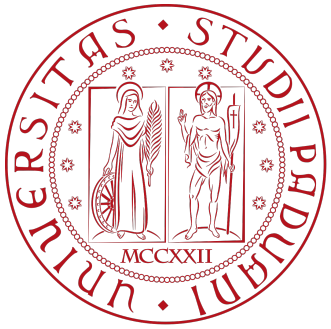
Università degli Studi di Padova

Dipartimento di Scienze Statistiche

Corso di laurea triennale in Statistica per l'Economia e l'Impresa



Tesi di Laurea Triennale

Non–Negative Constrained Penalised Matching Quantiles Estimation

Relatore: **Prof. Mauro Bernardi**
Dipartimento di Scienze Statistiche

Candidato: **Nicolas Bianco**
Matricola: **1101617**

Anno Accademico 2016/2017

Contents

Chapter 1

Introduction

The statistical area that has as main goal to replicate a target distribution, traditionally uses methods such as quantiles regression or density regression; the fist one minimizes a function of quantiles Q_α and the second one considers all the density function $f_y(y)$. Since both are conditional regressions, developping methods that match as well as possible a target unconditional distribution function can represent a new prospective in this research field.
? introduced a method called *Matching Quantiles Estimation* (*MQE*) used to match a target unconditional distribution using a linear combination of some variables; since it represents an important innovation we are going to present it in this introduction.
Let Y be a random variable with density described by $f_Y(y, \theta)$ and $X_{(n\times p)} = (X_1, X_2, ..., X_p)$ a design matrix where X_j for $j = 1, .., p$ is the vector containing the values of the j^{th} variable.
The goal is to find a vector of coefficients $\beta_{(p\times 1)}$ such that the distribution of the linear combination

$$X\beta = X_1\beta_1 + X_2\beta_2 + ... + X_p\beta_p$$

matches the distribution of Y.
Hence, we define β as the value that minimize the difference between the quantiles of the two distribution across all levels $\alpha \in [0, 1]$:

$$\hat{\beta} = argmin_\beta \int_0^1 [Q_Y(\alpha) - Q_{X\beta}(\alpha)]^2 d\alpha \tag{1.1}$$

Where $Q_\eta(\alpha)$ denotes the α^{th} quantile of the random variable η and we can write that

$$P[\eta \leq Q_\eta(\alpha)] = \alpha \text{ , for } \alpha \in [0, 1].$$

Solve the optimization problem defined by the equation (**??**) is not immediate; that's the why we should use *MQE* method that is easier and performs a better fitting especially at the tails of the distribution, that are very important in financial problems.
Let $(Y_1, Y_2, .., Y_n)$ and $(X_{1j}, X_{2j}, .., X_{nj})^T$ be the samples of Y and X_j, for $j = 1, .., p$, respectively; and let's call $(Y_{(1)}, Y_{(2)}, .., Y_{(n)})$ and $(X_{(1)j}, X_{(2)j}, .., X_{(n)j})^T$ their statistic order. Notice that $Y_{(h)}$ is the $\frac{h}{n}^{th}$ sample quantile of the distribution of Y.
Now we can define our matching quantiles estimator across all levels $\alpha \in [0, 1]$ as

$$\hat{\beta} = argmin_\beta \sum_{i=1}^{n} (Y_{(i)} - (X\beta)_{(i)})^2 \tag{1.2}$$

where $((X\beta)_{(1)}, ..., (X\beta)_{(n)})$ is the statistic order of $((X\beta)_1, ..., (X\beta)_n)$ and $(X\beta)_i = X_i\beta$, for $i = 1, .., n$.
Since the equation above does not admit an explicit solution, we have to introduce an iterative algorithm to compute the values of β. Suppose β^k the k^{th} iterated value of β and $X^k_{(i)}$ the statistic order of X_i during the k^{th} iteration.
The first step is to set an initial value β^0 that can be obtained via OLS estimation

$$\hat{\beta}^0 = argmin_\beta \sum_{i=1}^{n} (Y_i - (X\beta)_i)^2 = (X^T X)^{-1} X^T Y \tag{1.3}$$

Then let β^k be estimated by

$$\hat{\beta}^k = argmin_\beta \frac{1}{n} \sum_{i=1}^{n} (Y_{(i)} - (X\beta)^{k-1}_{(i)})^2 \tag{1.4}$$

for each $k \geq 1$.
When $|\hat{\beta}^k - \hat{\beta}^{k-1}| \leq \varepsilon$ with ε constant fixed and small, we stop the iterations and $\hat{\beta} = \hat{\beta}^k$.
Hence *MQE* estimator is obtained by applying OLS estimation repeatedly to the recursively sorted data.
MQE can be used also if we want to match only a part of the target distribution, for example between α_1^{th} and α_2^{th} quantiles such that $0 \leq \alpha_1 \leq \alpha_2 \leq 1$; we can write the *MQE* estimator at the k^{th} iteration as follows

$$\hat{\beta}^k = argmin_\beta \frac{1}{n_2 - n_1} \sum_{i=n_1+1}^{n_2} (Y_{(i)} - (X\beta)^{k-1}_{(i)})^2 \tag{1.5}$$

where $n_i = [n\alpha_i]$ with $i = 1, 2$ and $[x]$ is the integer part of x.
Furthermore, **?**, combine *MQE* method with the concept of shrinkage performing a

penalised regression instead of the classical linear regression (OLS), focusing their attention exclusively on L_1-penalty in order to perform what is called LASSO (**?**). As we know, this particular penalisation provides a sparse coefficient matrix and it does automatic variables selection, but it does not satisfy the oracle property such that $P(\hat{\beta} = \beta) \to 1$ (**?**, **?**), because L_1-penalty usually provides an underfitting, especially for high values of predictors p.
To dealing with this problem, we propose to combine *MQE* with the Elastic-Net (**?**) for its properties such as the capacity to regularize the estimates variance, produce sparsity and provide variables selection. The importance of the shrinkage, the usage of L_q-penalty and the most important penalised regressions are presented in Chapter **??**.
Another important goal of this work is to understand how the introduction of constraints in the penalised regression used in *MQE* can change the performances of the method; the choice of this study comes from the necessity in some real cases to obtain coefficients that satisfy particular characteristics in order to be directly interpretable. In Chapter **??** we present the effects of inequality and equality constraints, focusing our attention on the non-negative constraint.
In order to confirm the theory, we support the work with two empirical applications to index tracking in Chapter **??**.

Chapter 2

Penalised matching quantile estimation

There are some situations where OLS estimator is not the best choice, for example when the number of parameters is large or when columns of X are highly correlated, it can provide high variance in the estimates.
One way of dealing with this problem is to introduce a penalty in the optimization problem in order to reduce the variance, although introducing bias; hence, instead of minimizing the mean squared error, we minimize $\sum_{i=1}^{n} l_Y(y_i; X_i\beta) + \psi(\lambda, q, \beta)$ where $l_Y(y; X\beta)$ is the log-likelihood for Y and $\psi(\cdot)$ is a penalty function with parameters λ, that controls the trade-off between bias and variance and found via cross validation, and q that is the degree of the L_q-penalty we want to use, where

$$L_q = ||x||_q = (\sum_{i=1}^{p} |x_i|^q)^{1/q}, \qquad \forall q > 0 \tag{2.1}$$

In the following parts we consider the model

$$Y = X\beta + \epsilon \quad , \qquad \epsilon \sim N(0, \sigma^2)$$

In order to define the notation, coefficients vector β, in a sparse model, can be written as $\beta = (\beta_1, \beta_2)^T$ where $\beta_1 = (\beta_1, .., \beta_k)^T$ contains the non-zeros and $\beta_2 = (0, .., 0)^T$ contains the coefficients that are set to zero. If the non-zeros number is k, the zeros number is $z = p - k$.
$Y_{(n\times 1)}$ is the response vector and $X_{(n\times p)}$ is the design matrix that can be written as $X = (X_1, X_2)$ according to the partition of β.
The penalised optimization problem consists in find β such that minimizes $\sum_{i=1}^{n}(y_i - X_i\beta)^2 + \psi(\lambda, q, \beta)$. Notice that combining MQE with a L_q penalisation we obtain

the estimator at the k^{th} iteration $\hat{\beta}^k$ as

$$\hat{\beta}^k = argmin_\beta \frac{1}{n}\sum_{i=1}^{n}(Y_{(i)} - (X\beta)_{(i)}^{k-1})^2 + \psi(\lambda_k, q, \beta^k) \tag{2.2}$$

where λ_k is the shrinkage parameter at the k^{th} iteration found by cross validation; infact λ assumes different values at each step: although values in Y, X and their equivalents sorted are the same, $Y_{(i)}$ remains fixed while $X_{(i)}$ changes according to $\hat{Y}$ at each iteration.
? define a penalty function as a good one if it provides estimator with these three properties:

- Unbiasedness: The estimator $\hat{\beta}$ is defined such that $E(\hat{\beta}) \to \beta$.
- Sparsity: The estimator $\hat{\beta}$ is a thresholding rule, which sets some coefficients to zero providing a automatic variables selection.
- Continuity: The estimator $\hat{\beta}$ is continuous in data.

Furthermore, they also present the Oracle property that is advisable for a penalty function in order to dealing with the biasedness problem; before explain briefly it, let's define the regularity conditions that guarantee asymptotic normality of maximum likelihood estimates.

Remark 1. *Let $Y = (Y_1, .., Y_N)$ i.i.d. with density $f_{Y_i}(y_i, \beta)$; the first and second logarithmic derivatives of $f_{Y_i}(y_i, \beta)$ satisfy:*

- $E_\beta(\frac{\partial log(f_{Y_i}(y_i,\beta))}{\partial \beta_j}) = 0 \quad for \quad j = 1, .., p$
- $I_{F_{jk}}(\beta) = -E_\beta(\frac{\partial^2 log(f_{Y_i}(y_i,\beta))}{\partial \beta_j \partial \beta_k}) \quad for \quad j = 1, .., p \quad and \quad k = 1, .., p$
- *Information matrix conditioned if $\hat{\beta} = \beta$: $I_F(\beta) \mid_{\hat{\beta}=\beta}$ is finite and definite positive*
- *There is a region $\omega \subset \Theta$, where Θ is the parametric space, that contains the real value β and such that for almost all Y_i exist $\frac{\partial^3 log(f_{Y_i}(y_i,\beta))}{\partial \beta_j \partial \beta_k \partial \beta_h}$ and $|\frac{\partial^3 log(f_{Y_i}(y_i,\beta))}{\partial \beta_j \partial \beta_k \partial \beta_h}| \leq \phi(y)$ for all $\beta \in \omega$.*

Theorem 2. *Suppose* $Y = (Y_1, .., Y_N)$ *i.i.d. with density described by* $f_{Y_i}(y_i, \beta)$ *and such that satisfies the conditions in Remark* **??**, *assume a penalty function* $\psi(\lambda, q, \beta)$*: if* $\lambda \to 0$, $n \to +\infty$ *and* $\lambda\sqrt{n} \to +\infty$, *the estimator* $\hat{\beta}$ *satisfies*

1. *Sparsity :* $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2)$ *such that* $\hat{\beta}_2 = (0, .., 0)$
2. *Asymptotic normality :* $\sqrt{n}(\hat{\beta}_1 - \beta_1) \to N(0, I_F^{-1}(\beta_1))$

Hence the penalty function $\psi(\lambda, q, \beta)$ *holds the Oracle property.*

2.1 L2-Penalty

If we fix $q = 2$ in (**??**) in order to obtain a L_2-penalty, we perform the RIDGE regression that provides an optimization problem such that

$$\hat{\beta} = argmin_\beta (Y - X\beta)^2 + \lambda ||x||_2 \tag{2.3}$$

and the solution of (**??**) is

$$\hat{\beta} = (X^T X + \lambda I_p)^{-1} X^T Y \tag{2.4}$$

Hence we can easily see that when

$$\begin{array}{ll} \lambda \to 0, & \hat{\beta}_{RIDGE} \to \hat{\beta}_{OLS} \\ \lambda \to +\infty, & \hat{\beta}_{RIDGE} \to 0 \end{array}$$

Hence, the penalty that has to be added at the k^{th} iteration in (**??**) while we are performing *MQE* is given by

$$\psi(\lambda_k, 2, \beta^k) = \lambda_k ||\beta^k||_2 = \lambda_k \sum_{j=1}^{p} \beta_j^{2^k}$$

OLS estimates do not always exist; infact if X is not full rank, $X^T X$ is not invertible and there is no unique solution for β. This problem does not occur with RIDGE regression because for any design matrix X, the quantity $(X^T X + \lambda I_p)$ is invertible and there is always a unique solution for β.
This penalised regression is used in order to reduce the variance of the estimates, but can not be used to do subset selection; infact, unimportant coefficients may be shrunken towards zero, but they are not exactly zero, so they are still in the model. If we want to semplify the interpretation, we need a penalty that shrinks some coefficients exactly to zero in order to obtain a sparse model.

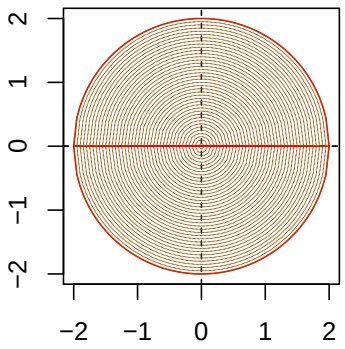


Figure 2.1: Parametric space $\Theta = \{\beta_j \in \Theta \mid \sum_{j=1}^{p} \beta_j^2 \leq t\}$ for $\beta = (\beta_1, \beta_2)$ defined by L_2 penalisation with $t = 2$ and $p = 2$.

2.2 L1-Penalty

If, instead of $q = 2$, we set $q = 1$, we include in the regression a L_1-penalty and perform what is called LASSO (**?**)

$$\hat{\beta} = argmin_{\beta}(Y - X\beta)^2 + \lambda||x||_1 \tag{2.5}$$

that provides an optimization problem with a convex objective function and can be solved efficiently for large problems.

Let's remark what a convex function is and why LASSO objective function is convex.

Theorem 3. *A region Δ is convex if for any $x_1, x_2 \in \Delta$*

$$x = \alpha x_1 + (1 - \alpha)x_2 \in \Delta, \qquad \forall \alpha \text{ in } (0, 1)$$

A function $f(x)$ is convex if its domain Δ is convex and $f(x) = f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2)$.

Theorem 4. *A matrix $H_{(p\times p)}$ is positive semidefinite (PSD) if $z^T H z \geq 0$, $\forall$ $z_{(p\times 1)}$*

Theorem 5. *Suppose $x_{p\times 1}$ vector and $f(x)$ is a function of p variables with continuous second order derivatives defined on a convex domain Δ.*
If its hessian $\bigtriangledown^2 f(x)$ is PSD $\forall$ $x \in \Delta$, then $f(\cdot)$ is convex.

Theorem 6. *If $f(x)$ and $g(x)$ are convex functions defined on a convex domain Δ, $h(x) = f(x) + g(x)$ is also convex on Δ.*

Theorem 7. *The LASSO object function in (**??**) is convex.*

Proof. Let $h(\beta)$ be the LASSO objective function. We can write it as $h(\beta) = f(\beta) + g(\beta)$ where

$$f(\beta) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X\beta_i)^2 = ||y - X\beta||_2^2$$
$$g(\beta) = \lambda\sum_{j=1}^{p}|\beta_j| = \lambda||\beta||_1$$

Notice that both $f(\beta)$ and $g(\beta)$ are defined on R^p that is convex.
$\bigtriangledown^2 f(\beta) = X^T X$ and $\forall\ z_{p\times 1}$ we have $z^T \bigtriangledown^2 f(\beta) z = z^T X^T X z = ||Xz||_2^2 \geq 0$. Hence the hessian of $f(\beta) = \bigtriangledown^2 f(\beta)$ is PSD. Hence, for the Theorem **??**, $f(x)$ is convex.
For any β_1,β_2 and any $\alpha \in (0, 1)$, let $\beta = \alpha\beta_1 + (1-\alpha)\beta_2$. Now we can write $g(\beta)$ as

$$\begin{aligned} g(\beta) &= \lambda||\alpha\beta_1 + (1-\alpha)\beta_2||_1 \\ &\leq \lambda||\alpha\beta_1||_1 + \lambda||(1-\alpha)\beta_2||_1 \\ &= \lambda\alpha||\beta_1||_1 + \lambda(1-\alpha)||\beta_2||_1 \\ &= \alpha g(\beta_1) + (1-\alpha)g(\beta_2) \end{aligned}$$

Hence $g(\beta)$ is convex, by Theorem **??**.
Since both $f(\beta)$ and $g(\beta)$ are convex and since $h(\beta) = f(\beta) + g(\beta)$, applying the Theorem **??**, $h(\beta)$, the objective function of LASSO regression, is also convex. □

If we introduce a L_1-penalty in the k^{th} iterated regression in (**??**) in order to perform *MQE* combined with LASSO, the penalty function is described by

$$\psi(\lambda_k, 1, \beta^k) = \lambda_k||\beta^k||_1 = \lambda_k\sum_{j=1}^{p}|\beta_j^k|$$

LASSO regression solves the problem of the interpretability of the model shrinking some coefficients exactly to zero, but its main problems are that it does not provide a regularization of variance and it does not possess the oracle property (Theorem **??**) since its resulting estimator is strongly biased, especially for cases in which the number of regressors p is higher than n because it can select at most n non-zero coefficients.

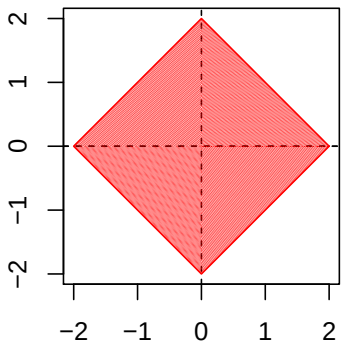


Figure 2.2: Parametric space $\Theta = \{\beta_j \in \Theta \mid \sum_{j=1}^{p} |\beta_j| \leq t\}$ for $\beta = (\beta_1, \beta_2)$ defined by L_1 penalisation with $t = 2$ and $p = 2$.

2.3 Elastic-Net Penalty

Elastic-Net penalty is given by a combination of L_1 and L_2 penalties, and that simultaneously does automatic variable selection, shrinks the coefficients and can select groups of correlated variables, while LASSO usually tends to select only one variable from these groups; hence it seems that Elastic-Net performs better than LASSO in terms of prediction accuracy. All these properties are discussed and proved with the support of simulation studies in **?**.
The optimization problem to solve in this types of regression is

$$\hat{\beta} = argmin_{\beta} \frac{1}{n} \sum_{i=1}^{n} (Y_i - X\beta_i)^2 + \lambda[\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2] \tag{2.6}$$

where α controls the influence of L_1-penalty and L_2-penalty and λ is the tuning parameter. Notice that even in Elastic-Net regression we are working with a convex objective function.
Moreover, we can write the optimization problem defined in (**??**) in its dual form as

$$\hat{\beta} = argmin_{\beta} \frac{1}{n} \textstyle\sum_{i=1}^{n} (Y_i - X\beta_i)^2 \qquad \text{subject to} \quad \alpha \textstyle\sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \leq t$$

Hence if $\alpha = 1$ we perform the LASSO regression, if $\alpha = 0$ the RIDGE regression and if $\alpha \in (0, 1)$ the Elastic-Net regression. The penalty function $\psi(\cdot)$ in (**??**) can

be written as

$$\psi(\lambda_k, 1, \beta^k) = \lambda_{1_k}||\beta^k||_1 + \lambda_{2_k}||\beta^k||_2 = \lambda_{1_k}\sum_{j=1}^{p}|\beta_j^k| + \lambda_{2_k}\sum_{j=1}^{p}\beta_j^{2^k}$$

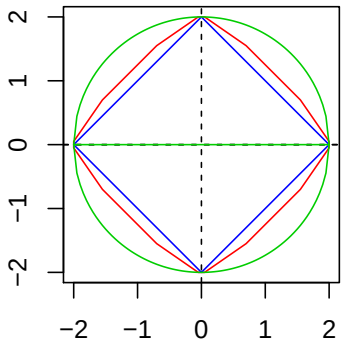


Figure 2.3: The area under the red curve represents the parametric space $\Theta = \{\beta_j \in \Theta \mid \sum_{j=1}^{p}(1-\alpha)\beta_j^2 + \alpha|\beta_j| \leq t\}$ for $\beta = (\beta_1, \beta_2)$ defined by Elastic-Net penalisation with $\alpha = 0.7$, $t = 2$ and $p = 2$. This space is compared to those defined by L_2 (under green curve) and L_1 (under blue curve) penalties.

2.4 Example on real data

Recalling *MQE* method, we can combine it with a penalised regression in order to obtain a sparse model that can be very useful if we are working with large matrices and we want to consider only the variables that exhibit the strongest effects and obtain a model easily interpretable.
Let's see now the results of an application of *MQE* combined with the penalised regressions described in the previous parts.
Applying the algorithm on the dataset *DJIA30*, which contains the historical returns of the *Dow Jones Industrial Average* and its constituentes from 20/03/2008 to 10/03/2017, we obtained a vector β that provides a linear combination $X\beta$ such that $F_Y(y) = F_{X\beta}(x\hat{\beta})$, where Y is the random variable describing the distribution of the DJIA's returns.
In order to get the results, *'glmnet'* library, created by **?**, has been used in R, which

contains *cv.glmnet(x,y)*, function for cross validation, and *glmnet(x,y,...)*, function for fitting a sparse model.
As λ we used the optimal value found via cross validation, that changes at each iteration; as α we considered the cases $\alpha = 1, 0.7, 0.5, 0.2, 0$. Furthermore, we estimated a model without intercept and considering $Y \sim N(\mu, \sigma^2)$.
Focusing on how *Penalised MQE* matches a target distribution, we obtained the following results.

Method	Alpha	Mean	Standard Deviation	Min	q(0.25)	q(0.5)	q(0.75)	Max	RMSE	KS p-value
Index DJIA		0.00042	0.00980	-0.05706	-0.00355	0.00034	0.00509	0.06611	-	-
MQE-LASSO	1	0.00039	0.00972	-0.05548	-0.00347	0.00023	0.00496	0.06710	0.00154	0.99570
MQE-ElasticNet	0.7	0.00040	0.00977	-0.05590	-0.00346	0.00025	0.00495	0.06741	0.00132	0.99861
MQE-ElasticNet	0.5	0.00040	0.00980	-0.05618	-0.00349	0.00026	0.00502	0.06719	0.00117	0.99747
MQE-ElasticNet	0.2	0.00041	0.00986	-0.05665	-0.00348	0.00026	0.00505	0.06834	0.00086	0.99987
MQE-RIDGE	0	0.00041	0.00990	-0.05693	-0.00351	0.00030	0.00504	0.06859	0.00081	0.99999

Table 2.1: Mean, Standard Deviation, Minimum, Quartiles, Maximum, Root-Mean square error and p-value of a Kolmogorov-Smirnov two samples test are reported in order to confront several different penalised matching quantiles estimation for $n = 200$.

From the Table **??** we can see that each method perform a good replication of the index. A remark that must be done is that the quantiles matching is almost perfect across all levels, but especially in the tails of the distribution that play a relevant role in finance because they contain the strongest negative and positive values.

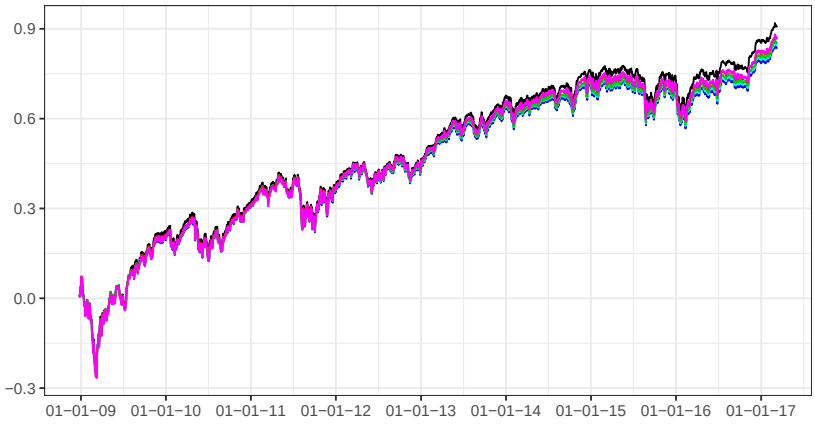


Figure 2.4: Cumulative sums for *MQE* combined with LASSO (blue), Elastic-Net $\alpha = 0.7$ (light blue), Elastic-Net $\alpha = 0.5$ (green), Elastic-Net $\alpha = 0.2$ (red) and RIDGE (violet). The black line represents the real sum of returns of the Index.

Considering the Table **??** and Fig. **??**, we can see how until a certain point each method replies almost perfectly the returns of the index, but then all of them perform worse than the index, and in a different way between themselves.
RIDGE regression provides the closest cumulative sum, it has the highest mean (0.41), but also the highest standard deviation (0.0099); at the contrary, *MQE* combined with LASSO provides returns with a lower cumulative sum and mean (0.39), but even a lower standard deviation (0.0097), which means less risk.
Hence, there's no methods to exclude, because there's no one that gives us losses, but neither to accept immediatly, because it depends on the trade off the investor requires between risk and return.

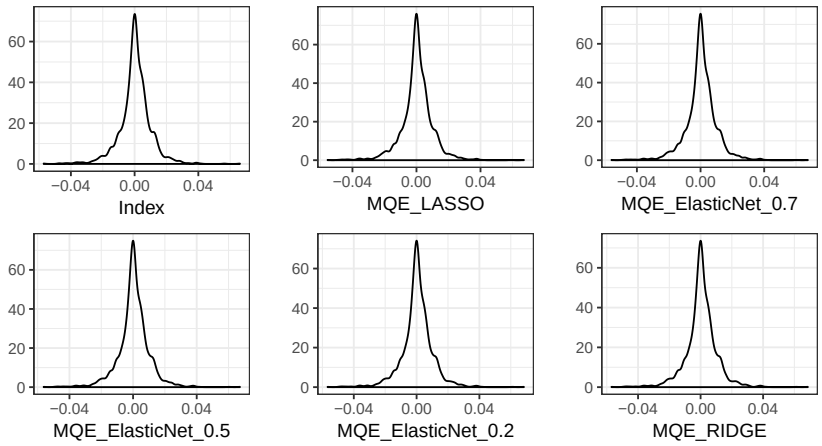


Figure 2.5: Densities produced by each method.

Looking at the values in the Table **??** and according to Fig. **??** and Fig. **??** we can suppose that the historical series of returns come from the same distribution. Infact, performing the Kolmogorov-Smirnov test on two samples in order to verify the null hypothesis

$$H_0 : F_Y(y) = F_{X\beta}(x\hat{\beta})$$

we obtained always $p - value > 0.05$. Hence, with a level of 95% of confidence, we accept all the null hypotesis for each case we implemented. For each method performed we found linear combination $X\beta$ such that matches the distribution of Y (index returns) using *Penalised Matching Quantiles Estimation*. Notice that the best performance we obtain in terms of Kolmogorov-Smirnov test is when we combine *MQE* with RIDGE penalty; this might be because with this method there's no coefficients set to zero, but we consider all the 30 constituents of the index.

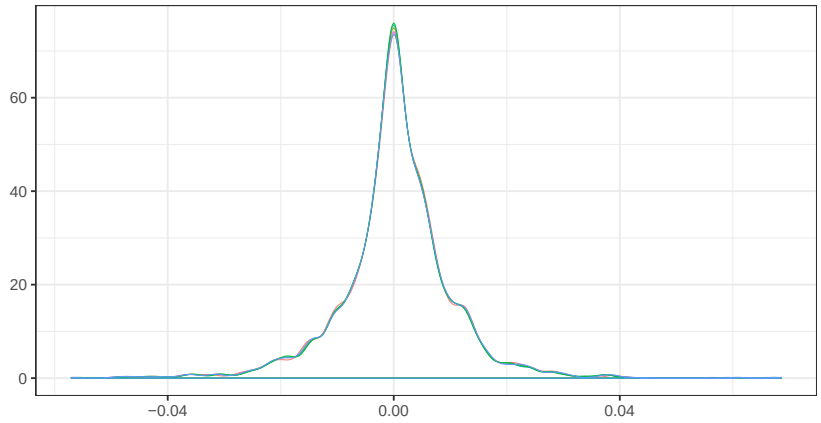


Figure 2.6: Densities overlapped of the returns given by the application of several different regressions such as LASSO (blue), Elastic-Net $\alpha = 0.7$ (light blue), Elastic-Net $\alpha = 0.5$ (green), Elastic-Net $\alpha = 0.2$ (red) and RIDGE (violet). The black line represents the real sum of returns of the Index.

Anyway the goal would be matching as better as possible the index returns distribution, but using the smallest number of constituents which exhibit the strongest effects; hence, to define which method has to be choosen as the best one, we need to set a measure of trade off between the number of coefficients introduced in the model and how much variance we want to explain, that are values directly proportional. Choosing properly a bound for this measure, we can identify the best combination MQE-penalty to use in our application.
The reduction of the number of coefficients used become very important when we are working with large matrices where we have $p > n$.

Chapter 3

Constrained penalised matching quantiles estimation

Starting from a generalized penalised regression

$$\hat{\beta} = argmin_{\beta} \sum_{i=1}^{n} (Y - X\beta)^2 + \psi(\lambda, q, \beta) \tag{3.1}$$

where $\psi(\lambda, q, \beta)$ is a penalty factor depending on q, we can add inequality and equality constraints on the coefficients β in order to obtain a constrained penalised regression

$$\hat{\beta} = argmin_{\beta} \sum_{i=1}^{n} (Y - X\beta)^2 + \psi(\lambda, q, \beta) \qquad s.t \quad A\beta = b \quad and \quad C\beta \geq d \tag{3.2}$$

Usually this kind of regressions are implemented in algorithms in order to be applicable to the reality that sometimes requires particular conditions that must be respected.
Inequality constraints define a area Ω and can modify the parametric space Θ for β. In particular it can be

$$\begin{cases} \Omega & if \quad \Omega \subset \Theta \\ \Delta = \{\beta \in R^p \quad | \quad \beta \in \Omega \cap \Theta\} & if \quad \Omega \cap \Theta \neq \varnothing \\ \Theta & if \quad \Omega \supset \Theta \end{cases}$$

Also equality constraints define a region, Φ, in the geometrical space such that Ω can be modified as follows

$$\begin{cases} \Phi & if \quad \Phi \subset \Theta \\ \Delta = \{\beta \in R^p \quad | \quad \beta \in \Phi \cap \Theta\} & if \quad \Phi \cap \Theta \neq \varnothing \\ \varnothing & if \quad \Phi \cap \Theta = \varnothing \end{cases}$$

Another important remark that must be done about the two type of constraints is that equality constraint is stronger than the inquality one.
Furthermore, we can introduce in the *Penalised MQE*, presented in the Chapter. **??**, some constraints in order to make the method applicable to real cases that require some constriction on the parameters.

3.1 Non-negative constraints

One of the most used inequality constraints consists in imposing the positivity of the coefficients: $\beta_j \geq 0$, $\forall j = 1, .., p$. This constraint can be written in matricial form as follows

$$\begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ ... \\ \beta_p \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ ... \\ 0 \end{bmatrix} \tag{3.3}$$

$$I_p \qquad \beta_{p\times 1} \geq 0_{p\times 1}$$

As the parametric space changes, also the optimization problem (**??**) when we include a L_1-penalty does

$$\hat{\beta} = argmin_\beta (Y - X\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j \tag{3.4}$$

infact

$$\beta_j \geq 0 \quad \Rightarrow \quad |\beta_j| = \beta_j, \quad \forall j = 1, .., p \tag{3.5}$$

(**??**) can be also written in its dual form as follows

$$\hat{\beta} = argmin_\beta (Y - X\beta)^2 \qquad s.t \quad \sum_{j=1}^{p} \beta_j \leq t \tag{3.6}$$

This regression is a particular case of LASSO, called Non-Negative LASSO for its parameters properties.

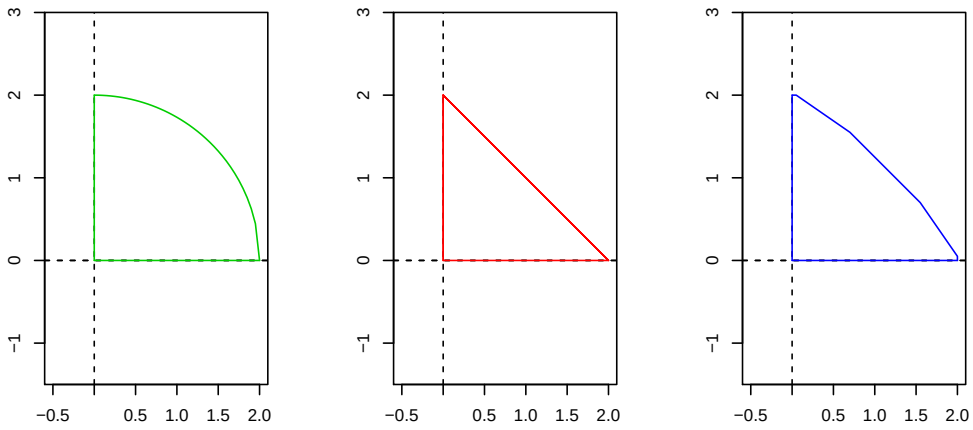


Figure 3.1: Parametric space Θ in case of L_2 (green), L_1 (red) and *ElasticNet* (blue) non negative costrained penalisation for $t = 2$, $\beta_{(2\times1)} = (\beta_1, \beta 2)$ and $\alpha = 0.7$.

Notice that since L_2-penalty considers $\lambda||\beta||_2^2 = \lambda \sum_{j=1}^{p} \beta_j^2$, where each β_j is considered squared, the form of a L_2-penalised regression does not change.

3.2 Equality constraints

A penalised regression subject to equality constraints follows the form

$$\hat{\beta} = argmin_\beta \sum_{i=1}^{n} (Y - X\beta)^2 + \psi(\lambda, q, \beta) \qquad s.t \quad A\beta = b \tag{3.7}$$

with $A_{(k\times p)}$, k represents the number of costraints and $b_{(k\times1)}$.
Suppose $f(\beta)$ a function continuous and derivable in β to be minimized constrained to $B(\beta) = b$.
Writing the constraint as $g(\beta) = B(\beta) - b$ we can define the lagrangian form of the constrained optimization problem

$$L(\beta, \mu) = f(\beta) + \mu g(\beta) \tag{3.8}$$

In order to find the minimum of $f(\beta)$ subject to $B(\beta) = b$ we need to solve the

following system of equations

$$\begin{cases} \dfrac{\partial L}{\partial \beta} = \dfrac{\partial f(\beta)}{\partial \beta} + \mu \dfrac{\partial g(\beta)}{\partial \beta} = 0 \\ \dfrac{\partial L}{\partial \mu} = g(\beta) = 0 \end{cases} \tag{3.9}$$

If $f(\beta)$ is convex (its hessian H is PSD), the solution $(\hat{\beta}, \hat{\mu})$ represents the minimum for $f(\beta)$ constrained to $B(\beta) = b$.

In our case we consider a function $f(\beta) = \sum_{i=1}^{n}(Y - X\beta)^2 + \psi(\lambda, q, \beta)$ where according to the form of $\psi(\lambda, q, \beta)$ we can define the LASSO problem or the Elastic-Net one. Since these penalised regressions provide a convex objective function, the solution of (**??**) in these cases represents the unique constrained optimal solution for the minimization problem in (**??**).

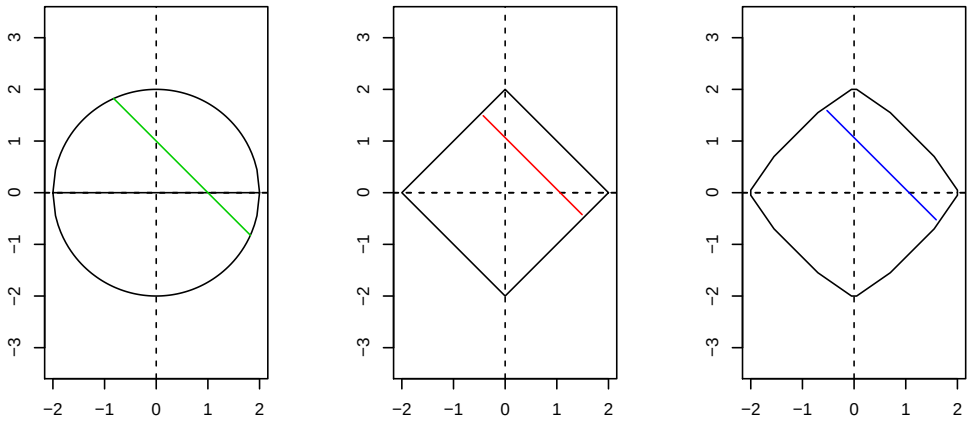


Figure 3.2: Parametric space Θ in case of L_2 (green), L_1 (red) and *ElasticNet* (blue) costrained penalisation with equality constraint $\sum_{j=1}^{2} \beta_j = 1$, for $t = 2$, $\beta_{(2\times 1)} = (\beta_1, \beta 2)$ and $\alpha = 0.7$. Notice that if $t < 1$ there would not be any solution for the optimization problems.

Chapter 4

Application to index Tracking

Tracking an index means that we are trying to replicate its growth across the time; by doing it we want to use just a subset of its constituents, the ones with the strongest effects, and not all of them.
Instead of what has been proposed by **?**, we want to replicate the distribution of the index returns using a *Non-Negative Constrained Penalised MQE* including different penalties in order to compare them. Since financial assets are high correlated between them, we expect better performances from Elastic-Net because of its capacity to select entirely groups of correlated variables if they are relevant, property that LASSO doesn't have.
First of all, in the financial framework, we are interested in the interpretation of the coefficients as weights; hence, if we want to read β_j as the percentage of our capital that we want to invest in the j^{th} constituent, we require that the sum of betas is equal to one.

$$\sum_{j=1}^{p} \beta_j = 1 \tag{4.1}$$

Furthermore, with the constriction in equation (**??**), we suppose we invest all of our capital in each period.
Another characteristic we require for our betas is the positivity, so we suppose that short sales are not allowed.

$$\beta_j \geq 0 \qquad \forall j = 1, .., p \tag{4.2}$$

If we want to use *MQE* to solve an index tracking problem, we need to introduce the constraints in (**??**) and (**??**) in the regression we use to perform the method.
Suppose we want to perform a *Penalised MQE* in order to shrink some coefficients to zero and include in our portfolio only the constituents that exhibit the strongest

effects on index performances; the optimization problem to solve will be

$$\hat{\beta} = argmin_{\beta} \frac{1}{n} \sum_{i=1}^{n} (Y_{(i)} - X\beta_{(i)})^2 + j(\beta) \qquad s.t. \quad \sum_{j=1}^{p} \beta_j = 1 \quad and \quad \beta_j \geq 0 \quad (4.3)$$

where

$$j(\beta) = \lambda \sum_{j=1}^{p} |\beta_j| \quad \text{if } MQE\text{-LASSO is performed}$$

and

$$j(\beta) = \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \quad \text{if } MQE\text{-Elastic-Net is performed}$$

Notice that, introducing the constraints in the optimization problem for the penalised regression we are fixing a specific form of sparsity by imposing the λ value that controls the trade off between bias and variance.
Let's consider the L_1 penalised regression in order to understand better what happens; writing the LASSO constriction using its dual form, we can write the optimization problem as

$$\hat{\beta} = argmin_{\beta} \frac{1}{n} \sum_{i=1}^{n} (Y_i - X\beta_i)^2 \qquad s.t. \quad \sum_{j=1}^{p} |\beta_j| \leq t, \quad \sum_{j=1}^{p} \beta_j = 1 \quad and \quad \beta_j \geq 0 \quad (4.4)$$

Let's now consider the following system of equations given by

$$\begin{cases} \sum_{j=1}^{p} |\beta_j| \leq t \\ \sum_{j=1}^{p} \beta_j = 1 \\ \beta_j \geq 0 \end{cases} \quad (4.5)$$

$\beta_j \geq 0$ means that $|\beta_j| = \beta_j$, $\forall j = 1, .., p$.
This constriction provides a Non-Negative LASSO regression because of the constraint $\beta_j \geq 0$ that sets all the coefficients equal or higher than zero.
The objective function for this particular regression is given by (**??**).
Let's see what happens to the region in Fig. **??** if we add also the constraint $\sum_{j=1}^{p} \beta_j = 1$.
Since the equality constraint is stronger than the inequality one, we can consider only

$$\begin{cases} \sum_{j=1}^{p} \beta_j = 1 \\ \beta_j \geq 0 \end{cases} \quad (4.6)$$

as constraints of the optimization problem (**??**).

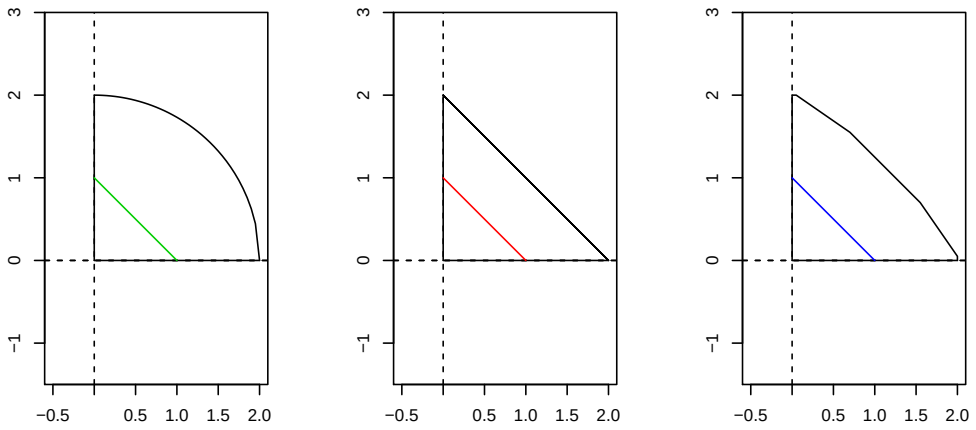


Figure 4.1: Parametric space Parametric space Θ in case of L_2 (green), L_1 (red) and *ElasticNet* (blue)in case of L_2 (green), L_1 (red) and *ElasticNet* (blue) non-negative costrained penalisation with equality constraint $\sum_{j=1}^{2} \beta_j = 1$, for $t = 2$, $\beta_{(2\times 1)} = (\beta_1, \beta 2)$ and $\alpha = 0.7$.

Hence, when we introduce the positivity of the coefficients and their sum to one, we are imposing a fixed value $t = 1$, that corresponds to a fixed value λ, for the penalty factor and at the same time we define a parametric space Θ that is the same for each L_q-penalty we decide to introduce in the regression, as we can see in Fig. **??**. Since that, the difference between the different penalisations consists in the first unconstrained estimation $\hat{\beta}^0$ required by *MQE* method in order to set the initial value for $\hat{\beta}$ and the models will be evaluated only for the capacity to select the real subset of coefficients to set to zero; infact variables selection is very important mostly in the applications where $p > n$.
Althought imposing a λ value is statistically incorrect, it represents the solution when we want to build a model applicable to the reality.

4.1 Dow Jones Industrial Average 30

Let's see now what happens if we perform the *MQE* method on a real dataset in order to describe a realistic case of index tracking. We consider the returns of *Dow Jones Industrial Average 30* from 20/03/2008 to 10/03/2017 and we try to replicate it using all or a part of its constituents. In this application we use a forecasting estimation instead of a contemporary one because in the real cases we don't invest

day by day, but we would like to invest now for some days ahead; hence, we perform *Constrained Penalised MQE* for different lengths of rolling window, $n = 50$ and $n = 100$, and for $h = 1$ and $h = 5$ days ahead. Furthermore, we suppose that short sales are not allowed and in a second moment that we are also using all the capital.

h = 1

$n = 20$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00022	0.01201	0.50467			
MQE-LASSO	1	0.00004	0.01013	0.09826	0.01606	0.74352	5
MQE-ElasticNet	0.7	0.00009	0.01264	0.20931	0.01823	0.79519	8
MQE-ElasticNet	0.5	0.00008	0.01272	0.19036	0.01827	0.80826	9
MQE-ElasticNet	0.2	0.00013	0.01291	0.30703	0.01842	0.82500	10

$n = 50$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00022	0.01205	0.50667			
MQE-LASSO	1	0.00031	0.01212	0.71723	0.01772	0.48668	10
MQE-ElasticNet	0.7	0.00029	0.01220	0.65696	0.01777	0.51650	11
MQE-ElasticNet	0.5	0.00032	0.01215	0.72512	0.01773	0.49428	11
MQE-ElasticNet	0.2	0.00031	0.01225	0.69988	0.01784	0.50603	12

$n = 100$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00026	0.01202	0.58406			
MQE-LASSO	1	0.00013	0.01204	0.29314	0.01768	0.32907	10
MQE-ElasticNet	0.7	0.00012	0.01208	0.26774	0.01766	0.32990	11
MQE-ElasticNet	0.5	0.00012	0.01211	0.26510	0.01772	0.33405	11
MQE-ElasticNet	0.2	0.00013	0.01217	0.29717	0.01776	0.33031	12

h = 5

$n = 20$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00021	0.01201	0.49670			
MQE-LASSO	1	0.00005	0.01139	0.11471	0.01679	0.78452	4
MQE-ElasticNet	0.7	0.00027	0.01404	0.61488	0.01895	0.88880	7
MQE-ElasticNet	0.5	0.00017	0.01452	0.39560	0.01925	0.90648	8
MQE-ElasticNet	0.2	0.00035	0.01514	0.81272	0.01980	0.92854	9

$n = 50$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00023	0.01206	0.51982			
MQE-LASSO	1	0.00015	0.01288	0.35180	0.01816	0.51854	10
MQE-ElasticNet	0.7	0.00018	0.01295	0.40465	0.01821	0.48668	10
MQE-ElasticNet	0.5	0.00018	0.01300	0.41906	0.01823	0.52429	11
MQE-ElasticNet	0.2	0.00018	0.01307	0.41264	0.01830	0.52820	11

$n = 100$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00026	0.01201	0.58309			
MQE-LASSO	1	0.00026	0.01245	0.57214	0.01766	0.31464	10
MQE-ElasticNet	0.7	0.00025	0.01247	0.55121	0.01768	0.31696	11
MQE-ElasticNet	0.5	0.00024	0.01254	0.53919	0.01772	0.32012	12
MQE-ElasticNet	0.2	0.00026	0.01261	0.57354	0.01778	0.32748	13

Table 4.1: *Penalised MQE* and constrained to the positivity of coefficients.
Mean, Standard Deviation, Cumulative returns, root-Mean Square Error, Turnover and non-zero coefficients for several models found applying *MQE* in order to replicate *Dow Jones Industrial Average 30*

h = 1

$n = 20$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00022	0.01201	0.50467			
MQE-LASSO	1	0.00007	0.01268	0.17353	0.01830	1.18623	10
MQE-ElasticNet	0.7	0.00026	0.01253	0.61464	0.01820	0.45595	16
MQE-ElasticNet	0.5	0.00027	0.01253	0.61721	0.01820	0.45425	16
MQE-ElasticNet	0.2	0.00028	0.01249	0.64275	0.01817	0.45643	16

$n = 50$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00022	0.01205	0.50667			
MQE-LASSO	1	0.00023	0.01241	0.51907	0.01830	0.80431	15
MQE-ElasticNet	0.7	0.00022	0.01220	0.49651	0.01802	0.11614	27
MQE-ElasticNet	0.5	0.00021	0.01220	0.49242	0.01802	0.11568	27
MQE-ElasticNet	0.2	0.00022	0.01220	0.50817	0.01802	0.11541	27

$n = 100$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00026	0.01202	0.58406			
MQE-LASSO	1	0.00025	0.01193	0.56306	0.01778	0.53957	16
MQE-ElasticNet	0.7	0.00025	0.01214	0.56933	0.01795	0.04592	28
MQE-ElasticNet	0.5	0.00025	0.01214	0.56542	0.01795	0.04581	28
MQE-ElasticNet	0.2	0.00025	0.01214	0.56202	0.01795	0.04520	28

h = 5

$n = 20$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00021	0.01201	0.49670			
MQE-LASSO	1	0.00007	0.01346	0.17146	0.01843	1.18290	8
MQE-ElasticNet	0.7	0.00027	0.01254	0.62631	0.01774	0.45566	16
MQE-ElasticNet	0.5	0.00027	0.01254	0.63128	0.01774	0.45420	16
MQE-ElasticNet	0.2	0.00028	0.01250	0.64069	0.01773	0.45739	16

$n = 50$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00023	0.01206	0.51982			
MQE-LASSO	1	0.00018	0.01245	0.40037	0.01771	0.81509	13
MQE-ElasticNet	0.7	0.00022	0.01221	0.50850	0.01753	0.11634	27
MQE-ElasticNet	0.5	0.00022	0.01220	0.50815	0.01753	0.11549	27
MQE-ElasticNet	0.2	0.00022	0.01218	0.51319	0.01752	0.11602	27

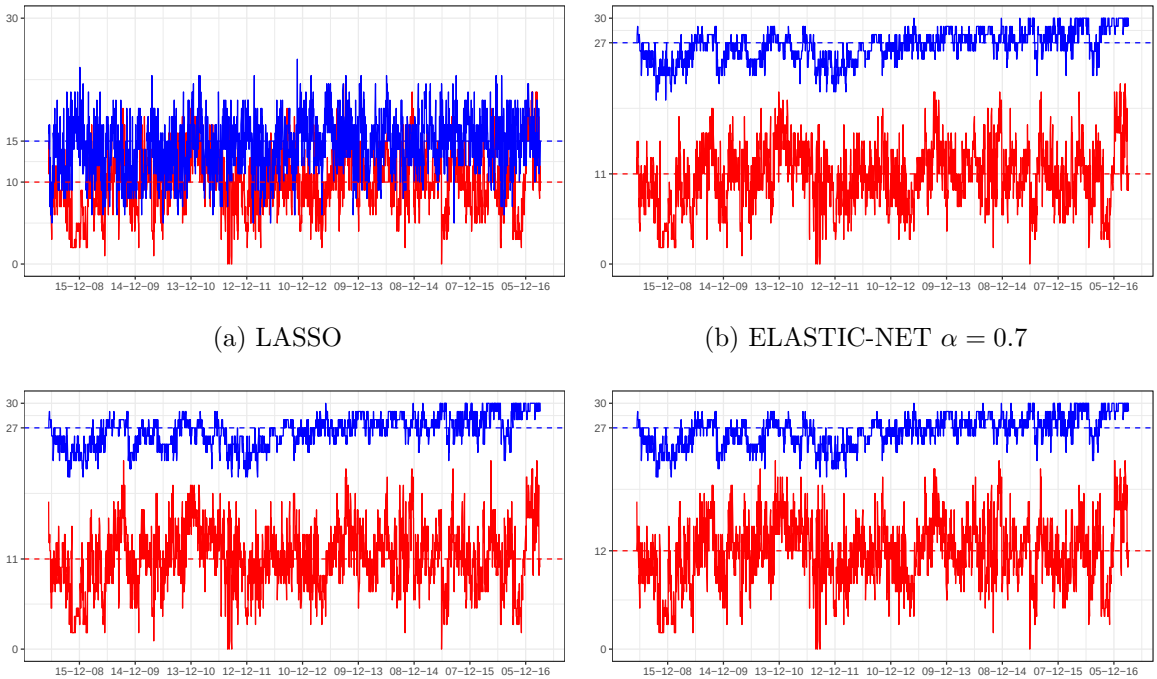
$n = 100$

Method	Alpha	Mean	SD	CumSum	RMSE	Turnover	Non-zero
Index		0.00026	0.01201	0.58309			
MQE-LASSO	1	0.00032	0.01218	0.71695	0.01756	0.53942	16
MQE-ElasticNet	0.7	0.00026	0.01215	0.57586	0.01746	0.04617	28
MQE-ElasticNet	0.5	0.00026	0.01212	0.57563	0.01746	0.04632	28
MQE-ElasticNet	0.2	0.00026	0.01211	0.57342	0.01747	0.04526	28

Table 4.2: *Penalised MQE* and constrained to the positivity of coefficients and their sum equal to one.
Mean, Standard Deviation, Cumulative returns, root-Mean Square Error, Turnover and non-zero coefficients for several models found applying *MQE* in order to replicate *Dow Jones Industrial Average 30*

Considering the results of the two penalised regression with different constraints, we notice that adding the constraint $\sum_{j=1}^{p} = 1$, the number of non-zero coefficients for each type of regression increases, hence we obtain a matrix less sparse, as we can see in Fig. **??**, and, as a consequence, the value of each β_j decreases (Fig. **??**). This happens because we are imposing a $t = 1$ in the L_1-penalty written with its

dual form, $\sum_{j=1}^{p} \leq t$, that corresponds to a fixed value for λ in $\lambda||\beta||_1$: so we are fixing a sparsity degree.



(a) LASSO

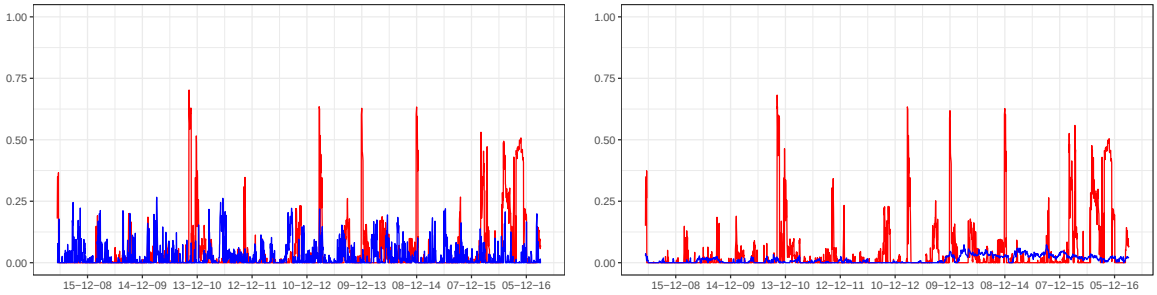
(b) ELASTIC-NET $\alpha = 0.7$

(c) ELASTIC-NET $\alpha = 0.5$

(d) ELASTIC-NET $\alpha = 0.2$

Figure 4.2: How the number of non-zero changes while we add the equality constraint. Values for $n = 50$ and $h = 1$, where the red line represents the case with only the non negative constraints and the blue one the case with both the constraints. The dashed line overlapped represents the mean of non-zero coefficients and it is the value considered in Table **??** and Table **??**

The estimates in the models with both constraints provide a better matching for μ and σ; and another value that changes a lot is the turnover, infact, in Table **??** turnovers are almost the same for each method and on the contrary in Table **??** *MQE*-LASSO ones are always the highest even if the number of coefficients in the model is the lowest. Furthermore, LASSO has the minor number of non-zero coefficients for every n and every h and its estimates are more variables also because L_1-penalty does not provide a regularization of the variance. At the contrary, Elastic-Net estimates give us always a good matching with the target distribution; moreover, for $n < p$ we obtain cumulative returns higher than the index for each value $\alpha = 0.2, 0.5, 0.7, 1$, where α controls the influence of L_1 and L_2 penalties in the Elastic-Net regression.



(a) LASSO

(b) ELASTIC-NET $\alpha = 0.7$

(c) ELASTIC-NET $\alpha = 0.5$

(d) ELASTIC-NET $\alpha = 0.2$

Figure 4.3: How the value of the coefficient linked to *NIKE* changes while we add the equality constraint. Values for each type of *Penalised MQE* for $n = 50$ and $h = 1$, where the red line represents the case with only the non negative constraints and the blue one the case with both the constraints.

Hence, looking at these results, we can conclude that combine *MQE* with Elastic-Net provides sometimes the same goodness of matching than combining it with LASSO and sometimes a better one, but never a worse one; this happens especially in the regression with coefficients constrained to be positive and to sum to one, that represents the more realistic case.

4.2 Standards' and Poor 500

The second application regards *Standards' and Poor 500* data from 04/01/1984 to 26/04/2016 containing the value of the index and the original constituents present at each time in the sampling period; hence, working with this dataset means work in the same conditions as an analist did in the reality. In particular it is composed by 8147 returns of SP500 and its constituents are in total 1299, but every day they

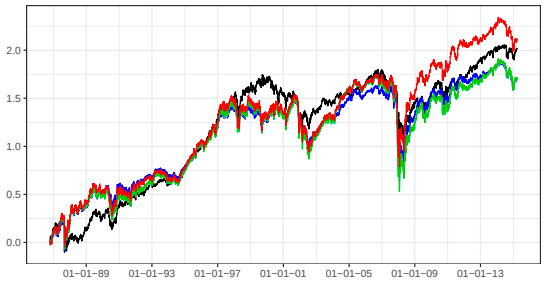
are at most 500; so, for each rolling window, we are going to consider as variables the biggest subset of constituents that are in the index for all the duration of that period that usually are $p \approx 400$.
Furthermore, we perform a *Penalised MQE* costrained to the positivity of coefficients and fixing $n = 728, 364, 184$, which means 2,1 and 1/2 years of daily returns; the estimation is contemporary, hence $h = 0$, and the penalised regressions used are the LASSO and Elastic-Net with $\alpha = 0.2, 0.5, 0.7$. The tuning parameter λ has been selected via cross validation as the optimum lambda value.

		$n = 184$					
Method	Alpha	Mean	SD	CumSum	RMSE	KS P-value	Non-zero
Index		0.00027	0.01146	2.01700			
MQE-LASSO	1	0.00023	0.01033	1.70393	0.01411	0.00525	24
MQE-ElasticNet	0.7	0.00023	0.01160	1.70190	0.01494	0.00294	57
MQE-ElasticNet	0.5	0.00023	0.01163	1.70693	0.01494	0.00294	60
MQE-ElasticNet	0.2	0.00028	0.01172	2.11167	0.01500	0.01190	63
		$n = 364$					
Method	Alpha	Mean	SD	CumSum	RMSE	KS P-value	Non-zero
Index		0.00029	0.01132	2.26889			
MQE-LASSO	1	0.00021	0.01030	1.66712	0.01377	0.00095	26
MQE-ElasticNet	0.7	0.00022	0.01131	1.71183	0.01431	0.00224	53
MQE-ElasticNet	0.5	0.00022	0.01133	1.71478	0.01431	0.00336	56
MQE-ElasticNet	0.2	0.00028	0.01156	2.16524	0.01452	0.00446	60
		$n = 728$					
Method	Alpha	Mean	SD	CumSum	RMSE	KS P-value	Non-zero
Index		0.00030	0.01124	2.41109			
MQE-LASSO	1	0.00017	0.01028	1.38682	0.01338	0.00078	23
MQE-ElasticNet	0.7	0.00021	0.01138	1.70674	0.01370	0.08363	51
MQE-ElasticNet	0.5	0.00021	0.01141	1.67775	0.01372	0.09054	54
MQE-ElasticNet	0.2	0.00024	0.01155	1.92492	0.01390	0.12310	59

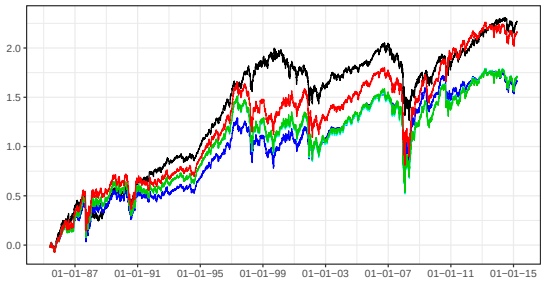
Table 4.3: Mean, Standard Deviation, Cumulative returns, root-Mean Square Error, p-value of Kolmogorov-Smirnov two samples test and non-zero coefficients for several models found applying *Non-negative Penalised MQE* in order to replicate *SP500*.

As we can see in Table **??** the penalties we combined with *MQE*, for different α values, provide different results; infact, it seems that, althought there's few cases in which we accept the null hypothesis of Kolmogorov-Smirnov test, its p-value increase while L_2-penalty influence is increasing. Looking at the values for the mean, the standard deviation and cumulative sum we can easily see how *MQE* combined with Elastic Net penalty for $\alpha = 0.2$ provides the closest values to the index ones, even if the RMSE is lower for a bigger value of α.
As we know from previous parts, elastic net shrinks less coefficients to zero than LASSO; that's probably the why *MQE* with a combination of L_1 and L_2 penalties performs a better distribution matching than the L_1-penalty alone can do. For this reason, in order to have an almost perfect matching, we can think about perform *MQE* combined exclusively with a RIDGE regression, but as explained in Chapter **??** it does not shrinks any coefficient exactly to zero, that is a property we require if we want to have an easier interpretable model and do variables selection;

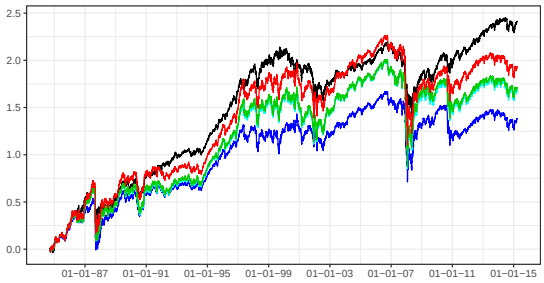
hence we can suppose that our best choice is to combine *MQE* with a elastic net penalisation for a very small value α. Let's see more in detail the particularity of the different models, focusing our attention on the best one and the worst one in terms of capacity to match a target distribution: *MQE*-ElasticNet $\alpha = 0.2$ and *MQE*-LASSO, respectively (Fig. **??**).



(a) n = 728



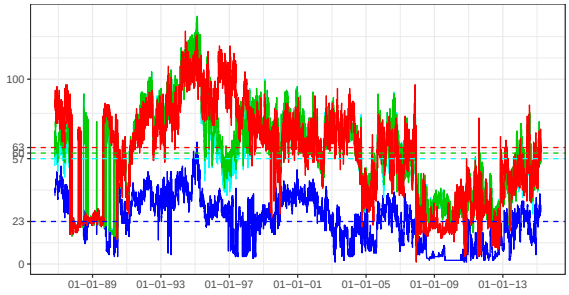
(b) n = 364



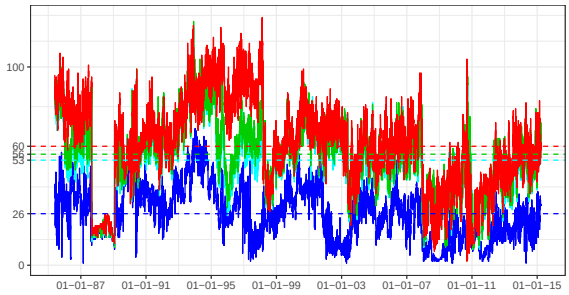
(c) n = 184

Figure 4.4: Cumulative sums for estimated returns of models found via *Non-negative Penalised MQE* compared with *SP500* index returns (black line) using a contemporary estimation. The penalisation used are LASSO (blue), Elastic-Net $\alpha = 0.7$ (light blue), Elastic-Net $\alpha = 0.5$ (green), Elastic-Net $\alpha = 0.2$ (red).

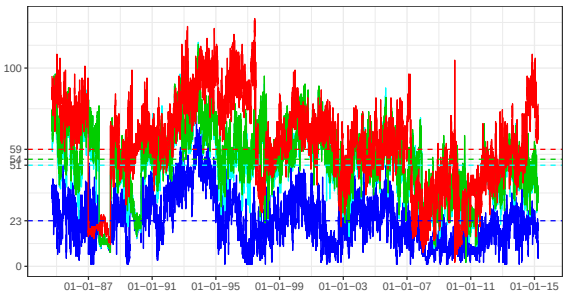
In order to conclude our analysis we consider how the number of non-zero coefficients (Fig. **??**) and the adjusted R-squared (Fig. **??**) change according to the penalisation introduced in the regression; we expect to notice a direct relationship between the two values (Fig. **??**).



(a) n = 728

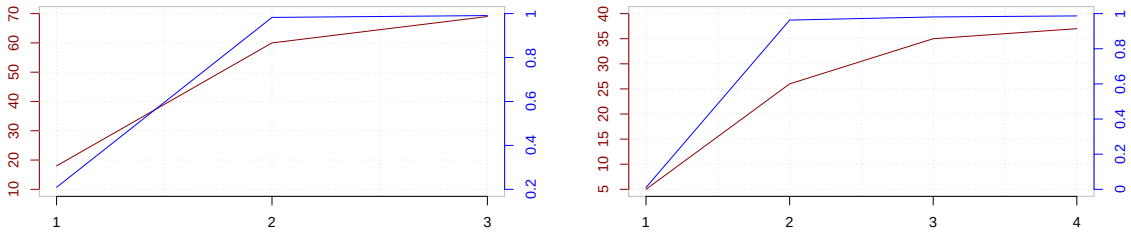


(b) n = 364



(c) n = 184

Figure 4.5: Non-zero coefficients at each rolling window while a contemporary estimation is performed; the dashed line overlapped represents the mean of non-zero coefficients and it is the value considered in Table **??**. The penalisation used are LASSO (blue), Elastic-Net $\alpha = 0.7$ (light blue), Elastic-Net $\alpha = 0.5$ (green), Elastic-Net $\alpha = 0.2$ (red).



(a) T = 100 (b) T = 6500

Figure 4.6: How the number of non-zero (dark red on left y-axe) and the adjusted R-squared (blue on right y-axe) change at each iteration of *MQE*. Values for two generic rolling windows T while *MQE* combined with Elastic-Net is performed with $\alpha = 0.2$ and $n = 184$

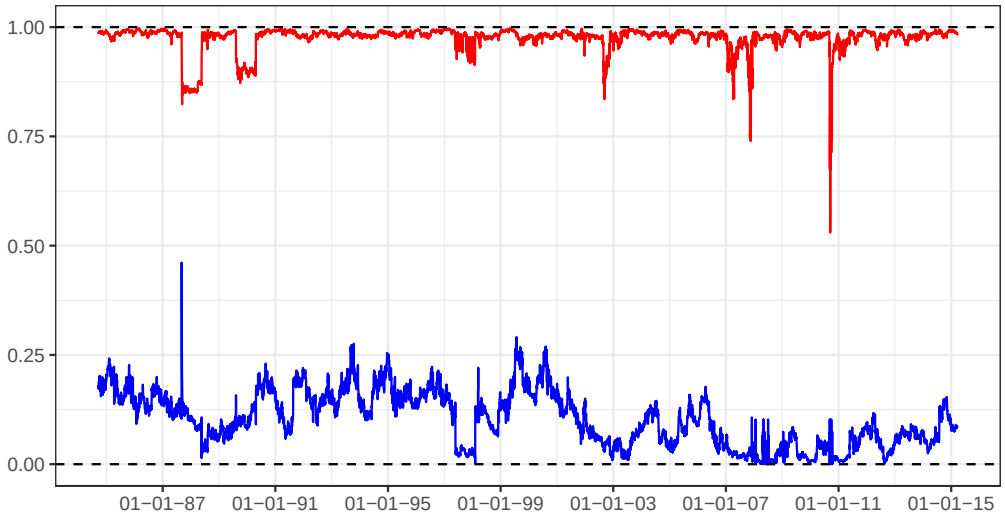


Figure 4.7: How R-squared changes from first estimation (blue) to second one (red) during *MQE* iterative algorithm. Values for *MQE* combined with Elastic-Net $\alpha = 0.2$ when $n = 184$

Chapter 5

Conclusion

Supported by empirical results from the applications, we can affirm that *Matching Quantiles Estimation* method performs very well for small datasets where $n > p$, but can be improved when the number of regressors p is higher than the observations n; since the L_1-penalty is linked to the problems we presented in Chapter **??**, we combined *MQE* with the Elastic-Net penalisation in order to exploit the main features of this penalty. Infact, we noticed that while the influence of the L_2-penalty increases, the results seems to be better; anyway we can not consider exclusively this penalty in our regression because we require to set some coefficients exactly to zero.
The introduction of constraints in the regression causes changes in the parametric space; hence number of non zero coefficients and their value are the most sensitive measures (Fig. **??** and Fig. **??**). Sometimes these constraints are mandatories if we are working in a particular workspace, even if their introduction is incorrect in a statistical point of view (as presented in Chapter **??** while we submit the coefficients to be positive and their sum equal to one) because they might fix a particular model of sparsity and the choice of the tuning parameter, that is an interesting part in penalised regressions, become not relevant.
Usually, when we are working with penalties and constraints, obtain a good fitting in terms of R^2 or $\overline{R}^2$ is very difficult and from the applications we noticed that the most of the times we can not reach neither 50%; but applying *MQE* method that presents an iterative algorithm to find the optimal value for β and that requires to sort the data at each iteration, the empirical analysis shows that immediatly after the first sorting and the estimation the R^2 and $\overline{R}^2$ values increase and the will continue to do this at each subsequent iteration (Fig. **??**). Hence, even if we consider a constrained penalised regression, *MQE* allows us to obtain an almost perfect fitting, we couldn't ever reach with a classic regression.

Chapter 6

Appendix A.
R-code for applications

Regression coefficients

Upload dataset

```
> sp500 = read.table("sp500.txt", header=TRUE)
```

Upload libraries

```
> library(glmnet)
```

Define response vector and design matrix

```
> mX.full = sp500[,2:1300]
> vY.full = sp500$RENDIMENTI
```

```
> h = 1  # h-steps ahead in the estimation
> p = ncol(mX.full)  # total number of predictors
> N = length(vY.full)  # total number of observations
> n = 250  # dimension of rolling window
> T = N-n+1  # number of rolling windows
> a = 0.7  # alpha value in the penalised regression
```

```
> mRegP = matrix(c(rep(0, T*p)), T, p)  # matrix in which
# i'll save betas
> nzero = rep(NA,T)  # number of non-zero coefficients
# vector
> varY = rep(NA,T)  # total variances vector
> varRes = rep(NA,T)  # residual variances vector
> Rs = rep(NA,T)  # R-squared vector
> ARs = rep(NA,T)  # Adjusted R-squared vector > lambdas =
rep(NA,T)  # lambda values selected
```

For each rolling window

```
> for (t in 1:T) {
# define start and end
> start = t
> end = t+n-1
# define design matrix with only the consituentes that are
# in the index during all the duration of the rolling
# window.
# h controls if we perform a contemporary estimation or
# a prevision estimation.
> vIndi=setIndiLen(mX.full,start,end)
> mX = as.matrix(mX.full[start:(end-h),vIndi])
# define response vector
> vY = vY.full[(start+h):end]
# First penalised estimation
# lambda via cross-validation
> l = cv.glmnet(mX,vY)$lambda.min
> fit = glmnet(mX,vY, alpha=a, lambda = l, intercept =
FALSE, lower.limits = 0)
> vRegP = as.vector(fit$beta)
> vY.hat0 = mX %*% vRegP
# sort data as required in MQE
> vY_ = sort(vY)
> mX_ = ordina.mX(vY.hat0, mX)
# define values for the convergence of the algorithm
> dTolX = sqrt(as.vector(vRegP)%*%as.vector(vRegP))
> res0 = resid(vY, vY.hat0, n)
```

```
> dTolFun = res0
> minTolX = 0.05
> minTolFun = 1e-06
> RsLim = 0.5
# main part of MQE-method.  Iterative estimates until
# convergence.
> while ((dTolX > minTolX) || (dTolFun > minTolFun) ||
(RsLim > Rs_) {

# lambda via cross-validation
> l = cv.glmnet(mX_,vY_)$lambda.min
> fit_ = glmnet(mX_,vY_, alpha=a, lambda = l, intercept =
FALSE, lower.limits = 0)
> vRegP_ = fit_$beta
> vY.hat = mX_%*%vRegP_
> dTolX = sqrt((as.vector(vRegP_)-as.vector(vRegP))%*%
(as.vector(vRegP_)-as.vector(vRegP)))
> res.new = resid(vY_, vY.hat, n)
> dTolFun = abs(res.new-res0)
> res0 = res.new
> mu = mean(vY_)
> varY_ = sum((vY_ - mu)^2)
> varRes_ = sum((vY_ - vY.hat)^2)
> Rs_ = 1-varRes_ / varY_
# sort design matrix
> mX_ = ordina.mX(vY.hat, mX_)
# update betas vector
> vRegP = vRegP_

}
# save the values defined above for the rolling window
> mu = mean(vY_)
> nzero[t] = length(which(vRegP!=0))
> varY[t] = varY_
> varRes[t] = varRes_
> Rs[t] = Rs_
> ARs[t] = 1 - ((1-Rs[t])*(length(vY_)-1)/(length(vY_)-
nzero[t]-1))
```

```
> lambdas[t] = l
# save betas vector in the matrix defined at the beginning
> mRegP[t,vIndi]=vRegP
}
```

Save results in a .txt file

```
> tab = cbind(nzero,varY,varRes,Rs,ARs)
> write.table(tab, file="stats__elnet_h_n.txt", sep=" ",
col.names = FALSE, row.names=FALSE)
> write.table(mRegP.las, file="results_elnet_h_n.txt",
sep=" ", col.names = FALSE, row.names=FALSE)
```

Estimation

Let's estimate the predicted values $\hat{Y}$, given by $X\hat{\beta}$

Upload the matrix containing betas for each window

```
> coeff.full = read.table("results_elnet_h_n.txt")
```

Set the response vector and the design matrix

```
> vY = sp500$RENDIMENTI
> mX.full = sp500[,2:1300]
```

Initialize the predicted values vector

```
> vY.hat=c(rep(NA, T))
```

For each window

```
> for (i in 1:T) {
> vIndi=setIndiLen(mX.full,i,199+i)
> mX = mX.full[199+i, vIndi]
> coeff = coeff.full[i, vIndi]
> vY.hat[i] = as.matrix(mX)%*%t(as.matrix(coeff))
}
```

Save the $\hat{Y}$ vector paired with the response one, Y

```
> r = cbind(c(vY[(200+h):8147],rep(NA,h)), vY.hat)
> write.table(r, file="rendimenti_elnet_h_n.txt", sep=" ",
col.names = FALSE, row.names=FALSE)
```

Functions used

Function to select the subset of constituents present in the index during all the window

```
> setIndiLen = function(mX.full,start,end) {
# design matrix during the window
> mX=mX.full[start:end,]
> vIndi=c(1:ncol(mX))     # vector with the positions
# if for some column of mX, its length without NA values
# is different from the length with them, we exclude that
# constituent from the set of regressors
> for (i in ncol(mX):1) {
> v.con=mX[,i]
> v.senza=na.omit(v.con)
> if (length(v.senza)!=length(v.con)) {
> vIndi=vIndi[-i]
}
}
> return(vIndi)
}
```

Function to sort mX matrix at each step of the algorithm

```
> ordina.mX = function(vY.hat,mX) {
> Val.Ind = sort(as.vector(vY.hat),index.return=TRUE)
# ordered vector and position of predicted values
> vInd = Val.Ind$ix    # vector of position index
> mX.ordinata = mX[vInd,]
}
```

Function to obtain the new value given by the residuals for the convergence of the algorithm

```
> resid = function(vY,vY.hat,n) {
> dFun = (vY - as.vector(vY.hat))%*%(vY -
as.vector(vY.hat))/n
> return(dFun)
}
```