# VARIABLE SELECTION AND PREDICTIVE MODELS FOR BIG DATA ENVIRONMENTS

by

Álvaro Méndez Civieta

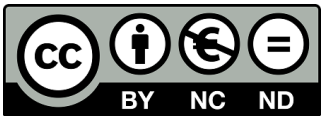
A DISSERTATION
submitted in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
(Mathematical Engineering)

Universidad Carlos III de Madrid

Advisors:
Rosa E. Lillo
M. Carmen Aguilera-Morillo

Tutor:
Rosa E. Lillo

January 2022



*A mi familia*

# Acknowledgements

First of all, I would like to thank my advisors Professors Rosa E. Lillo and M. Carmen Aguilera-Morillo for their permanent support, guidance, advice and encouragement. When I finished my master's degree, I was determined to go into the business field, but I enjoyed so much working with them in my Master thesis that I decided to pursue the Ph.D. studies.

Getting here is by no means a job I have done alone. I am here thanks to the support of my friends and my family, especially my mother and sister, who always encouraged me. But also thanks to all the professors who were involved in my education, especially M. Ángeles Gil and Norberto Corral, who helped me with all their support (and also with all their letters of recommendation). This thesis belongs to all of them.

I am also grateful with the Department of Statistics of Universidad Carlos III de Madrid, for providing the financial support to carry out this thesis.

And I would want to extend my gratitude to professors Jeff Goldsmith and Ying Wei, for their collaboration and support in the last chapter of this thesis. It has been a pleasure working with them.

Finally, I want to acknowledge the financial support received by research grants and projects PIPF UC3M, ECO2015-66593-P (Ministerio de Economía y Competitividad, Spain) and PID2020-113961GB-I00 (Agencia Estatal de Investigaciónm Spain).

# Published and submitted content

The materials from the following sources are included in the thesis. Their inclusion is not indicated by typographical means or references, since they are fully embedded in each chapter indicated below.

- Méndez-Civieta Álvaro, Aguilera-Morillo M. Carmen and Lillo Rosa E. (2021). "Adaptive sparse group LASSO in quantile regression". In: *Advances in Data Analysis and Classification* 15(3), 547–573.
  - https://doi.org/10.1007/s11634-020-00413-8
  - Included in: Chapter 2 (full).
- Méndez-Civieta Álvaro, Aguilera-Morillo M. Carmen and Lillo Rosa E. (2021). "A quantile based dimension reduction technique".
  - http://hdl.handle.net/10016/33469
  - Included in: Chapter 3 (full).
  - Under review in *Chemometrics and intelligent laboratory systems.*
- Méndez-Civieta Álvaro, Aguilera-Morillo M. Carmen and Lillo Rosa E. (2021). "Asgl: A Python Package for Penalized Linear and Quantile Regression". *arXiv preprint arXiv:2111.00472*
  - https://arxiv.org/abs/2111.00472
  - Included in: Chapter 5 (full).

# Other research merits

**Software:**

- Python package **asgl**: adaptive sparse group lasso. Maintainer. https://pypi.org/project/asgl/.

**Research stay:**

- Columbia University Mailman School of Public Health. Research tutor: Jeff Goldsmith.

**Research conference talks:**

Year 2021.

- *Dimension reduction techniques based on quantiles.* Invited talk at 14th International Conference of the ERCIM WG on Computational and Methodological Statistics;
- *fPQR: A quantile based dimension reduction technique for regression.* Invited talk at New Bridges between Mathematics and Data Science;
- *The oracle knows the truth (or a review on penalized regression).* Invited talk at the School of Computer Science and Informatics, Cardiff University;

Year 2020.

- *Quantile regression. An adaptive penalization study.* Contributed talk at fith Congreso de jovenes investigadores de la RSME;

Year 2019

- *A new approach to penalized quantile regression.* Contributed talk at 12th International Conference of the ERCIM WG on Computational and Methodological Statistics;

- *Quantile regression: an adaptive penalized estimation.* Contributed talk at the second Spanish Young Statisticians and Operational Researchers meeting;

- *Adaptive penalization in quantile regression. A genetics case study.* Contributed talk at the XXXVIII Congreso de la Sociedad Española de Estadistica e Investigación Operativa;

Year 2018

- *The sparse group LASSO regularization method in quantile regression models.* Contributed talk at the XXXVII Congreso de la Sociedad Española de Estadistica e Investigación Operativa.

# Abstract

In recent years, the advances in data collection technologies have presented a difficult challenge by extracting increasingly complex and larger datasets. Traditionally, statistics methodologies treated with datasets where the number of variables did not exceed the number of observations, however, dealing with problems where the number of variables is larger than the number of observations has become more and more common, and can be seen in areas like economics, genetics, climate data, computer vision etc. This problem has required the development of new methodologies suitable for a high dimensional framework.

Most of the statistical methodologies are limited to the study of averages. Least squares regression, principal component analysis, partial least squares... All these techniques provide mean based estimations, and are built around the key idea that the data is normally distributed. But this is an assumption that is usually unverified in real datasets, where skewness and outliers can easily be found. The estimation of other metrics like the quantiles can help providing a more complete image of the data distribution.

This thesis is built around these two core ideas. The development of more robust, quantile based methodologies suitable for high dimensional problems. The thesis is structured as a compendium of articles, divided into four chapters where each chapter has independent content and structure but is nevertheless encompassed within the main objective of the thesis.

First, Chapter 1 introduces basic concepts and results, assumed to be known or referenced in the rest of the thesis. A possible solution when dealing with high dimensional problems in the field of regression is the usage of variable selection techniques. In this regard, sparse group lasso (SGL) has proven to be a very effective alternative. However, the mathematical formulation of this estimator introduces some bias in the model, which means that it is possible that the variables selected

by the model are not the truly significant ones. Chapter 2 studies the formulation of an *adaptive sparse group lasso* for quantile regression, a more flexible formulation that makes use of the adaptive idea, this is, the usage of adaptive weights in the penalization to help correcting the bias, improving this way variable selection and prediction accuracy. An alternative solution to the high dimensional problem is the usage of a dimension reduction technique like partial least squares. Partial least squares (PLS) is a methodology initially proposed in the field of chemometrics as an alternative to traditional least squares regression when the data is high dimensional or faces colinearity. It works by projecting the independent data matrix into a subspace of uncorrelated variables that maximize the covariance with the response matrix. However, being an iterative process based on least squares makes this methodology extremely sensitive to the presence of outliers or heteroscedasticity.

Chapter 3 defines the *fast partial quantile regression*, a technique that performs a projection into a subspace where a quantile covariance metric is maximized, effectively extending partial least squares to the quantile regression framework. Another field where it is common to find high dimensional data is in functional data analysis, where the observations are functions measured along time, instead of scalars. A key technique in this field is functional principal component analysis (FPCA), a methodology that provides an orthogonal set of basis functions that best explains the variability in the data. However, FPCA fails capturing shifts in the scale of the data affecting the quantiles.

Chapter 4 introduces the *functional quantile factor model*. A methodology that extends the concept of FPCA to quantile regression, obtaining a model that can explain the quantiles of the data conditional on a set of common functions.

In Chapter 5, **asgl**, a Python package that solves penalized least squares and quantile regression models in low and high dimensional is introduced frameworks is introduced, filling a gap in the currently available implementations of these models. Finally, Chapter 6 presents the final conclusions of this thesis, including possible lines of research and future work.

# Resumen

En los últimos años, los avances en las tecnologías de recopilación de datos han planteado un difícil reto al extraer conjuntos de datos cada vez más complejos y de mayor tamaño. Tradicionalmente, las metodologías estadísticas trataban con conjuntos de datos en los que el número de variables no superaba el número de observaciones, sin embargo, enfrentarse a problemas en los que el número de variables es mayor que el número de observaciones se ha convertido en algo cada vez más común, y puede verse en áreas como la economía, la genética, los datos relacionados con el clima, la visión por ordenador, etc. Este problema ha exigido el desarrollo de nuevas metodologías adecuadas para un marco de alta dimensión.

La mayoría de las metodologías estadísticas se limitan al estudio de la media. Regresión por mínimos cuadrados, análisis de componentes principales, mínimos cuadrados parciales... Todas estas técnicas proporcionan estimaciones basadas en la media, y están construidas en torno a la idea clave de que los datos se distribuyen normalmente. Pero esta es una suposición que no suele verificarse en los conjuntos de datos reales, en los que es fácil encontrar asimetrías y valores atípicos. La estimación de otras métricas como los cuantiles puede ayudar a proporcionar una imagen más completa de la distribución de los datos.

Esta tesis se basa en estas dos ideas fundamentales. El desarrollo de metodologías más robustas, basadas en cuantiles, adecuadas para problemas de alta dimensión. La tesis está estructurada como un compendio de artículos, divididos en cuatro capítulos en los que cada uno de ellos tiene un contenido y una estructura independientes pero que, sin embargo, se engloban dentro del objetivo principal de la tesis.

En primer lugar, el Capítulo 1 introduce conceptos y resultados básicos, que se suponen conocidos o a los que se hace referencia en el resto de la tesis. Una posible solución cuando se trata con problemas de alta dimensión en el campo de la regresión es el uso de técnicas de selección de variables. En este sentido, el sparse

group lasso (SGL) ha demostrado ser una alternativa muy eficaz. Sin embargo, la formulación matemática de este estimador introduce cierto sesgo en el modelo, lo que significa que es posible que las variables seleccionadas por el modelo no sean las verdaderamente significativas. El Capítulo 2 estudia la formulación de un *adaptive sparse group lasso* para la regresión cuantílica, una formulación más flexible que hace uso de la idea adaptive, es decir, el uso de pesos adaptativos en la penalización para ayudar a corregir el sesgo, mejorando así la selección de variables y la precisión de las predicciones. Una solución alternativa al problema de la alta dimensionalidad es el uso de una técnica de reducción de dimensión como los mínimos cuadrados parciales. Los mínimos cuadrados parciales (PLS por sus siglas en inglés) es una metodología definida inicialmente en el campo de la quimiometría como una alternativa a la regresión tradicional por mínimos cuadrados cuando los datos son de alta dimensión o tienen problemas de colinearidad. Funciona proyectando la matriz de datos independiente en un subespacio de variables no correlacionadas que maximiza la covarianza con la matriz de respuesta. Sin embargo, al ser un proceso iterativo basado en mínimos cuadrados, esta metodología es extremadamente sensible a la presencia de valores atípicos o heteroscedasticidad.

El Capítulo 3 define el *fast partial quantile regression*, una técnica que realiza una proyección en un subespacio en el que se maximiza una métrica de covarianza cuantílica, extendiendo de forma efectiva los mínimos cuadrados parciales al marco de la regresión cuantílica. Otro campo en el que es habitual encontrar datos de alta dimensión es el del análisis de datos funcionales, en el que las observaciones son funciones medidas a lo largo del tiempo, en lugar de escalares. Una técnica clave en este campo es el análisis de componentes principales funcionales (FPCA por sus siglas en inglés), una metodología que proporciona una base ortogonal de funciones que explica la mayor cantidad posible de variabilidad en los datos. Sin embargo, el FPCA no capta los cambios de escala de los datos que afectan a los cuantiles.

El Capítulo 4 presenta el *functional quantile factor model*. Una metodología que extiende el concepto de FPCA a la regresión cuantílica, obteniendo un modelo que puede explicar los cuantiles de los datos condicionados a un conjunto de funciones comunes.

En el capítulo 5 **asgl**, un paquete para Python que resuelve modelos de mínimos cuadrados y regresión cuantílica penalizados en entornos de baja y alta dimensión es presentado, llenando un vacío en las implementaciones actualmente disponibles de estos modelos. Por último, el Capitulo 6 presenta las conclusiones finales de esta tesis, incluyendo posibles líneas de investigación y trabajo futuro.

# Contents

# List of Figures

# List of Tables

# CHAPTER 1

# Introduction

This thesis is centered on the research of two key ideas: the usage of quantile based methodologies as a robust alternative to traditional mean based models, and the analysis of high dimensional problems. The resulting work is a compendium of four independent chapters that study different aspects of high dimensional problems from a quantile perspective. This first chapter introduces basic concepts and results, assumed to be known or referenced in the rest of the thesis. It is organized as follows. Section 1.1 introduces a general perspective of least squares linear regression models and quantile regression models, showing the main benefits of the usage of quantile regression. In Section 1.2, the concept of penalized regression is introduced, discussing some of the most known penalizations in the literature. This section sets the basic concepts required for Chapter 2. In Section 1.3 the usage of dimension reduction techniques based on projections is introduced, emphasizing principal component analysis and partial least squares, required for a better understanding of Chapter 3. Section 1.4 introduces a series of basic concept from the functional data analysis framework that will be used in Chapter 4. Finally, Section 1.5 describes the structure and main contributions of this thesis, providing a complete perspective of this work.

## 1.1 Linear regression

Regression analysis is a statistical field that studies relations between variables. Consider a sample of $N$ observations taken from a random vector $X^t = (X_1, \ldots, X_p)$ and stored into a data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times p}$. The rows of $\boldsymbol{X}$ are denoted as $\boldsymbol{x}_i^t$ and the columns

are samples drawn from the random variables in $\boldsymbol{X}$. This data matrix, usually called the independent data matrix, is assumed to be numeric with the columns coming from possibly different sources including numeric variables, transformations of numeric variables, dummy encodings of qualitative variables etc. Additionally, consider another sample of $N$ observations taken from a random vector $Y = (Y_1, \ldots, Y_l)$ and stored into a matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times l}$ called the response matrix. Usually $l = 1$ and then the response matrix is a univariate response vector denoted as $\boldsymbol{y}$. Chapters 2 and 5 of this thesis assume that the response is univariate, while in chapter 3 a multivariate response is studied.

The objective in regression models is to use the known information about the variables to try to estimate the relation between $\boldsymbol{X}$ and $\boldsymbol{y}$. Probably the most widespread formulation of a regression model is the linear regression model, where such relation is assumed to be linear and can be expressed as,

$$Y = X^t\boldsymbol{\beta} + \varepsilon, \tag{1.1}$$

where $\varepsilon$ is a random variable called the error and $\boldsymbol{\beta} \in \mathbb{R}^p$ is a coefficient vector such that $X^t\beta$ best approximates $Y$. Regression models can be posed as optimization problems where the coefficient vector minimizes the value of a risk function conditional on the available information $R(\beta|\boldsymbol{X}, \boldsymbol{y})$,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\max}\ R(\beta|\boldsymbol{X}, \boldsymbol{y}). \tag{1.2}$$

The global minimum in this problem can be obtained as long as $R(\cdot)$ is convex, and different risk functions define different types of regression models.

### 1.1.1 Least squares models

Least squares assumes an underlying relation of the form,

$$\mathbb{E}(Y \mid X) = X^t\boldsymbol{\beta}. \tag{1.3}$$

For this reason, it is often said that least squares regression provides the conditional mean of the response variable given a set of independent variables. Least squares seek to obtain the value of $\boldsymbol{\beta}$ such that it minimizes the sum of the squared error $\boldsymbol{\varepsilon}_i$, leading to the following risk function,

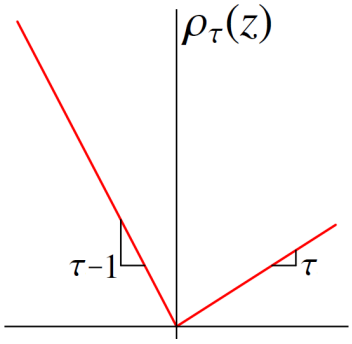
$$R(\boldsymbol{\beta}) = \frac{1}{2N}\|\boldsymbol{y} - \boldsymbol{X\beta}\|_2^2. \tag{1.4}$$

This function is convex and can be minimized by means of some gradient descent based algorithm, although it also has an analytical closed form solution that emerges naturally taking partial derivatives with respect to $\boldsymbol{\beta}$ and equaling the result to $0$,

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^t\boldsymbol{X})^{-1}\boldsymbol{X}^t\boldsymbol{y}. \tag{1.5}$$

If the error term is independent, homoscedastic and normally distributed, requirement usually summarized as $\varepsilon \sim \mathrm{N}(0, \sigma\boldsymbol{I})$, then the maximum likelihood estimation of $\boldsymbol{\beta}$ produces the same estimation as equation (1.5).

Figure 1.1: Quantile regression loss check function $\rho_\tau(\cdot)$.




### 1.1.2 Quantile regression models

The estimation provided by least squares regression is based on the mean, and this makes it very sensitive to the presence of outliers. Additionally, when dealing with skewed or heteroscedastic data, it can be useful to obtain estimations based on the quantiles, as it can provide a more complete picture of the distribution of the response than just the mean estimate. Consider a real valued random variable $Z$ and define its distribution function as,

$$F(Z) = P(Z \leq z), \tag{1.6}$$

then given $\tau \in (0, 1)$ the $\tau$th quantile of $Z$ is defined as,

$$F^{-1}(\tau) = \inf\{z : F(z) \geq \tau\}. \tag{1.7}$$

A quantile regression model assumes the existence of an underlying relation between $X$ and the quantiles of $Y$,

$$Q_\tau(Y \mid X) = X^t\boldsymbol{\beta}, \tag{1.8}$$

where $Q_\tau(Y)$ denotes the $\tau$th quantile of the variable $Y$. This way, just as least squares regression provides estimates of the conditional mean of $Y$, quantile regression can be understood as a model that provides estimates of the conditional quantiles. To show how it works, let us define the quantile regression loss check function,

$$\rho_\tau(z) = z(\tau - I(z < 0)). \tag{1.9}$$

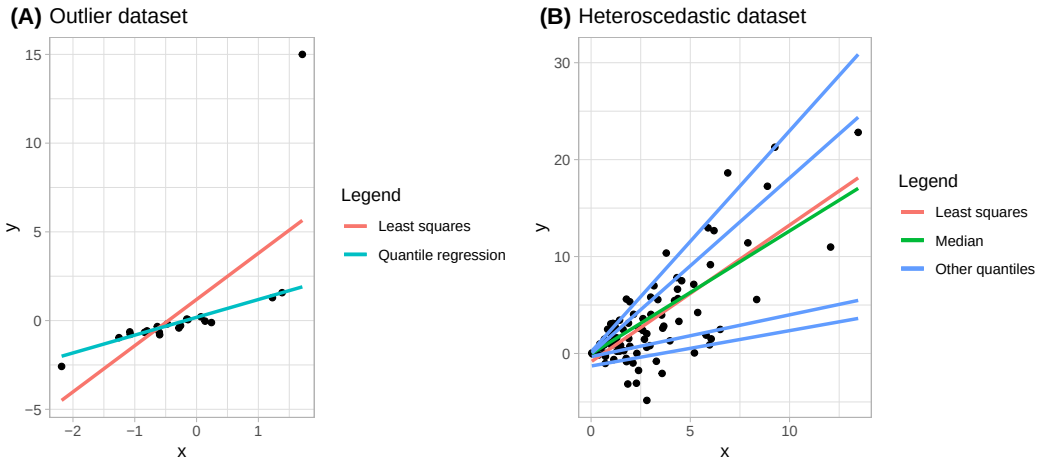
Figure 1.1 shows the graphical representation of function $\rho_\tau(\cdot)$. Then the $\tau$th quantile of the random variable $Z$ can be obtained as a result of the following optimization problem,

$$\hat{\alpha} = \arg\min \mathbb{E}\, \rho_\tau(Z - \alpha). \tag{1.10}$$

Based on this idea, the quantile regression risk function is defined as,

$$R(\boldsymbol{\beta}) = \frac{1}{2N} \sum_{i=1}^{N} \rho_\tau(y_i - \boldsymbol{x}_i^t\boldsymbol{\beta}). \tag{1.11}$$

Figure 1.2: Comparison of least squares and quantile regression. (A) comparison in the presence of outliers. (B) comparison in the presence of heteroscedastic data.




Quantile regression, unlike least squares regression, does not require any assumptions on the error term $\varepsilon$, and being based on the quantiles, provides estimates that are robust against the presence of outliers and heteroscedastic data. A complete review on quantile regression can be seen in [Koenker, 2005]. This thesis makes important contributions to the quantile regression framework, including high dimensional regression models, and quantile based dimension reduction techniques.

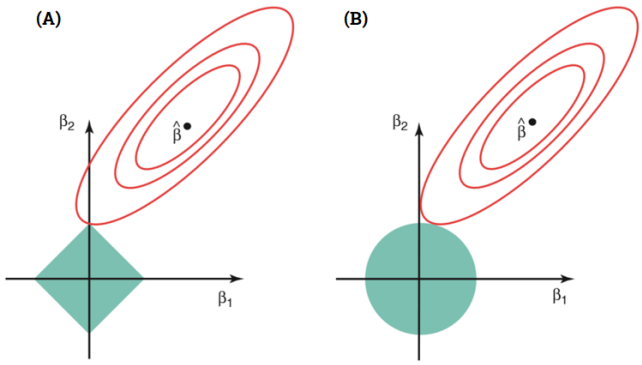
**Example:** *This example shows the benefit of quantile regression in the presence of outliers. Consider the simple linear model* $Y = X\beta + \varepsilon$ *with* $X \sim N(0,1)$ *and* $\varepsilon \sim N(0, 0.2)$. *Following this formulation* 20 *observations are drawn, and an outlier is introduced by manually changing the value of* $y_5$ *from* 1.8 *to* 15. *Figure 1.2 (A) shows a comparison of the estimations provided by the least squares regression and the median regression. It is clear that the least squares estimation is influenced by the presence of the outlier while the median regression provides a better estimation.*

**Example:** *This example shows the benefit of quantile regression in heteroscedastic data. Consider the simple linear model* $Y = X\beta + \varepsilon$ *with* $X \sim \chi(3)$ *and* $\varepsilon_i \sim N(0, x_i)$. *Following this formulation* 100 *observations are drawn. Figure 1.2 (B) shows a comparison of the estimations provided by the least squares regression and the quantile regression for quantiles* 10%, 25%, 50%, 75% *and* 90%. *The usage of quantile regression here allows to recover a complete picture of the distribution of the response variable.*

## 1.2 Variable selection techniques

The models introduced in Section 1.1 work very well as long as the independent data matrix $\boldsymbol{X}$ has full column rank, but start to fail when this does not hold. Usually,

Figure 1.3: Penalized least squares. (A) lasso penalty. (B) ridge penalty. The red ellipsoids are the contour lines of the residual sums of squares in a least squares model.




$\boldsymbol{X}$ will fail to have full column rank under two circumstances:

- Two or more variables from $\boldsymbol{X}$ are linearly dependent. This problem is called colinearity,
- The number of variables $p$ is larger than the number of observations $N$. This is called high dimensional data.

One of the most popular solutions to face both problems is the usage of a ridge penalty, defined as,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_2^2 \right\}, \tag{1.12}$$

where $R(\boldsymbol{\beta})$ is a risk function like the ones from Section 1.1 and $\lambda$ is a hyper parameter controlling the penalization. The ridge penalty introduces some bias in the model, but in return reduces the variability. However, it does not perform variable selection because it bounds the euclidean distance of the parameters to the origin, and thus, all the parameters are proportionally penalized.

Probably the most known variable selection penalization is lasso [Tibshirani, 1996], where the $L_2$ norm from ridge is exchanged for an $L_1$ norm,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1 \right\}, \tag{1.13}$$

Figure 1.3 shows a two dimensional example based on least squares regression of both lasso penalty (A) and ridge penalty (B). The red ellipsoids are the contour lines of the residual sums of squares, and the blue areas are the penalizations. From a geometrical perspective, the optimal solution of the penalized models is the point in which the red ellipsoids touch the blue area. Intuitively, one can see that the usage of the $L_1$ norm produces vertices where some of the coefficients have value zero, effectively performing variable selection.

There are many situations in which the variables in the model have a natural grouped structure. For example, one can consider the case of a categorical variable encoded as a set of dummy variables. In these situations, it may be desirable to perform variable selection at the group level rather than at the individual level, including (or excluding) from the model all the variables associated to the same factor. There are other situations where the data may have a grouped structure, for example genetic pathways. Yuan and Lin [2006] introduced the group lasso as a solution to this problem, defined as,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{K} \sqrt{p_j} \left\| \boldsymbol{\beta}^j \right\|_2 \right\}, \tag{1.14}$$

where $K$ is the number of groups, $\boldsymbol{\beta}^j \in \mathbb{R}^{p_j}$ are vectors of components of $\boldsymbol{\beta}$ from the j-$th$ group, and $p_j$ is the size of the j-$th$ group. Group lasso goes one step ahead of lasso, enhancing sparsity at the group level by performing a lasso type penalty between groups, while doing a ridge type penalty within groups. Observe that group lasso includes lasso as a particular case.

If a group of variables is selected using group lasso, then all the coefficients in that group will be different than zero. However, there are situations where it would be interesting to perform variable selection not only between groups, but also within groups. For example, when dealing with genetical pathways it is interesting to select the most important important genes from certain groups. Simon et al. [2013] proposed the sparse group lasso (SGL) as a solution to face this within group sparsity requirement as,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \alpha\lambda \left\| \boldsymbol{\beta} \right\|_1 + (1-\alpha)\lambda \sum_{j=1}^{K} \sqrt{p_j} \left\| \boldsymbol{\beta}^j \right\|_2 \right\}, \tag{1.15}$$

where the parameter $\alpha$ is a hyper parameter that controls the linear combination between lasso and group lasso. Observe that both penalizations are particular cases of the sparse group lasso.

All the penalizations described in this section are widely used and perform great from an experimental point of view. However, they are all based on the concept of the bias-variance tradeoff. By penalizing the model, we increase the bias of the estimator and reduce the variance. When dealing with variable selection, this means that it is possible that the variables selected by the estimators are not the truly significant variables. This problem was faced by [Zou, 2006], who considered the usage of adaptive weights in the lasso penalization that could correct this bias and defined the adaptive lasso as,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} w_j |\beta_j| \right\}, \tag{1.16}$$

where $w_j$ are the adaptive weights. This thesis makes important contributions to the sparse group lasso in quantile regression by proposing a new definition based

on the adaptive idea that helps correcting the bias, improving this way prediction accuracy and variable selection.

## 1.3 Dimension reduction

We can see variable selection as dimension reduction techniques based on using a penalization that selects the most important variables from the original dataset. However, when dealing with colinearity or high dimensional data, there is another alternative that is widely used and which is usually referred to as dimension reduction: dimension reduction based on projections. The key idea is to project the data into a new subspace (usually of lower dimension) that explains most of the information from the original subspace.

### 1.3.1 Principal component analysis

Principal component analysis (PCA) is probably the most famous dimension reduction technique. Consider a data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times p}$ and assume without loss of generality that it is centered so that each column variable has mean zero. PCA seeks to obtain a new group of variables computed as linear combinations of the original variables in $\boldsymbol{X}$ such that the first one explains the largest possible amount of variability from $\boldsymbol{X}$, the second one explains the largest variability subject to the restriction of being orthogonal to the first one, and so on. The final result is a set of uncorrelated variables sorted based on the percentage of variability they explain. Mathematically, we assume a latent relation of the form,

$$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{P}, \tag{1.17}$$

where $\boldsymbol{T} \in \mathbb{R}^{N\times d}$ is the scores matrix storing the new set of variables, $d \leq \text{rank}(X)$ is the number of new variables, and $\boldsymbol{P} \in \mathbb{R}^{p\times d}$ is an orthogonal change of basis matrix. One can pose PCA as a variance maximization problem,

$$\hat{p}_i = \underset{p_i^t p_j=0,\ \|p_i\|=1}{\arg\max} \{\text{var}(t_i)\}, \tag{1.18}$$

where $p_i$ and $t_i$ refer to the $i$th column of matrices $\boldsymbol{P}$ and $\boldsymbol{T}$ respectively, and the condition $\|p_i\| = 1$ is included in the formulation to ensure the uniqueness of solution (otherwise, any random rescaling of $\boldsymbol{P}$ would affect the optimal solution of the problem). Using the spectral theorem, one can see that,

$$\text{var}(\boldsymbol{T}) = \boldsymbol{P}^t\boldsymbol{\Sigma}\boldsymbol{P} = \boldsymbol{P}^t\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^t\boldsymbol{P}, \tag{1.19}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{X}, \boldsymbol{U}$ is a matrix of eigenvectors of $\boldsymbol{\Sigma}$ and $\boldsymbol{D}$ is a diagonal matrix of the ordered eigenvalues associated to the eigenvectors. This shows that the optimal solution to equation (1.18) is $u_i$, the $i$th eigen vector of $\boldsymbol{\Sigma}$.

### 1.3.2 Partial least squares

Partial least squares (PLS) is a problem closely related to PCA where, instead of maximizing the variability from one matrix, the objective is to maximize the covariance between two data matrices $\boldsymbol{X} \in \mathbb{R}^{N\times p}$ and $\boldsymbol{Y} \in \mathbb{R}^{N\times l}$. This algorithm was initially proposed in the field of chemometrics [Wold, 1973] as an alternative to traditional least squares regression, as it is common in this field to deal with highly correlated, high dimensional data like NIR spectra or other lab measurements where least squares was not a feasible option. Without loss of generality, assume that both matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are centered and consider the existence of a latent structure,

$$X = TP^t + E; \quad Y = TQ^t + F, \tag{1.20}$$

where $T \in \mathbb{R}^{n\times d}$ is the scores matrix formed by $d$ (usually being $d \ll p$) linear combinations of the original variables, $P \in \mathbb{R}^{p\times d}$ and $Q \in \mathbb{R}^{l\times d}$ are loadings matrices and $E \in \mathbb{R}^{n\times p}$ and $F \in \mathbb{R}^{n\times l}$ are random error matrices. The objective of PLS regression is to regress the response matrix $Y$ onto the $d$ latent variables, stored in the scores matrix $T$. There are multiple definitions of PLS in the literature, being NIPALS Wold [1973] and SIMPLS [de Jong, 1993] the most widespread ones. We describe here the original version of the multivariate NIPALS algorithm.

To compute the first component, define $\boldsymbol{X}_0 = \boldsymbol{X}$, $\boldsymbol{Y}_0 = \boldsymbol{Y}$ and randomly select a column $\boldsymbol{u}_1$ from matrix $Y_0$.

1. Obtain the weights of $\boldsymbol{X}$ by regressing the columns of $\boldsymbol{X}_0$ onto $\boldsymbol{u}_1$.

2. Normalize the result. Steps 1 and step 2 can be combined into one step as,

$$\boldsymbol{w}_1 = \frac{\boldsymbol{X}_0^t \boldsymbol{u}_1}{\|\boldsymbol{X}_0^t \boldsymbol{u}_1\|}$$

3. Obtain the scores of $\boldsymbol{X}$ by regressing the rows of $\boldsymbol{X}_0$ onto $\boldsymbol{w}_1$.

$$\boldsymbol{t}_1 = \boldsymbol{X}_0^t \boldsymbol{w}_1$$

4. Obtain the weights of $\boldsymbol{Y}$ by regressing the columns of $\boldsymbol{Y}_0$ onto $\boldsymbol{t}_1$.

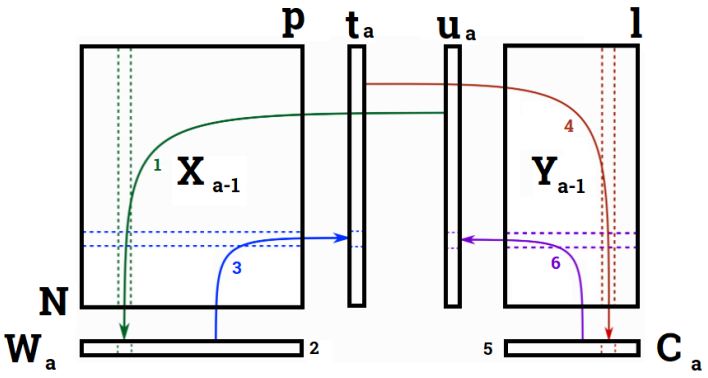
5. Normalize the result. Steps 4 and 5 can be combined into one step as,

$$\boldsymbol{c}_1 = \frac{\boldsymbol{Y}_0^t \boldsymbol{t}_1}{\|\boldsymbol{Y}_0^t \boldsymbol{t}_1\|}$$

6. Obtain the scores of $Y$ by regressing the rows of $Y_1$ onto $c_1$.

$$\boldsymbol{u}_1 = \boldsymbol{Y}_0 \boldsymbol{c}_1$$

Figure 1.4: Overview of multivariate NIPALS algorithm.




Iterate through steps 1-6 until changes in $\boldsymbol{t}_1$ are small. On convergence, go to step 7.

7. Obtain the loadings of $\boldsymbol{X}$ by regressing the columns of $\boldsymbol{X}_0$ onto $\boldsymbol{t}_1$.

$$\boldsymbol{p}_1 = \frac{\boldsymbol{X}_0^t \boldsymbol{t}_1}{\boldsymbol{t}_1^t \boldsymbol{t}_1}$$

8. Obtain the loadings of $\boldsymbol{Y}$ by regressing the columns of $\boldsymbol{Y}_0$ onto $\boldsymbol{t}_1$.

$$\boldsymbol{q}_1 = \frac{\boldsymbol{Y}_0^t \boldsymbol{t}_1}{\boldsymbol{t}_1^t \boldsymbol{t}_1}$$

9. Deflat the matrix $\boldsymbol{X}_0$ from the information already explained by scores $\boldsymbol{t}_1$ and obtain $\boldsymbol{X}_1 = \boldsymbol{X}_0 - \boldsymbol{t}_1 \boldsymbol{p}_1^t$.

10. Deflat the matrix $\boldsymbol{Y}_0$ from the information already explained by scores $\boldsymbol{t}_1$ and obtain $\boldsymbol{Y}_1 = \boldsymbol{Y}_0 - \boldsymbol{t}_1 \boldsymbol{q}_1^t$.

Once the first component is computed, iterate through steps 1-10 to compute the next ones until all the required components are obtained. Figure 1.4 shows an overview of the iterative process of steps 1-6 of NIPALS algorithm. Observe that, with a bit of matrix algebra one can rewrite the weight vector $\boldsymbol{w}_a$,

$$\boldsymbol{w}_a = \frac{\boldsymbol{X}_{a-1}^t \boldsymbol{u}_a}{\|\boldsymbol{X}_{a-1}^t \boldsymbol{u}_a\|} = \frac{\boldsymbol{X}_{a-1}^t \boldsymbol{Y}_{a-1} \boldsymbol{Y}_{a-1}^t \boldsymbol{X}_{a-1} \boldsymbol{w}_{a-1}}{\|\boldsymbol{X}_{a-1}^t \boldsymbol{Y}_{a-1} \boldsymbol{Y}_{a-1}^t \boldsymbol{X}_{a-1} \boldsymbol{w}_{a-1}\|}, \tag{1.21}$$

that can be rewritten as,

$$(\boldsymbol{Y}_{a-1}^t \boldsymbol{X}_{a-1})^t (\boldsymbol{Y}_{a-1}^t \boldsymbol{X}_{a-1}) \boldsymbol{w}_a = \lambda \boldsymbol{w}_a. \tag{1.22}$$

Recurring again to the spectral theorem, the computation of $\boldsymbol{w}_a$ can then be posed as a covariance maximization problem,

$$\boldsymbol{w}_a = \underset{\|\boldsymbol{w}_a\|=1}{\arg\max} \left\{ \mathrm{cov}(\boldsymbol{Y}_{a-1}, \boldsymbol{X}_{a-1} \boldsymbol{w})^t \mathrm{cov}(\boldsymbol{Y}_{a-1}, \boldsymbol{X}_{a-1} \boldsymbol{w}) \right\}. \tag{1.23}$$

Partial least squares is a widely used alternative to least squares regression, as it can deal with high dimensional or colinear data. However, it is essentially an iterative process based on least squares, and as such, it faces the same problems as least squares regarding outliers, skewness, and heteroscedasticity. This thesis makes contributions to this field by proposing a quantile based alternative to partial least squares.

## 1.4 Functional data analysis

A functional variable is characterized by the fact that its observations are functions that in most of the times represent the change of a scalar variable along time. Examples of this are observed in environmental variables like temperature or contamination, economic variables like stock market prices, clinical variables like physical activity, etc. These variables can be analyzed from a traditional multivariate perspective by considering each time point as an independent observation, however, this implies giving up the clear temporal factor in the data. Functional data analysis is a statistical field centered in the study of functional variables preserving the time dependence. Let us consider $X = \{X(t) : t \in \mathcal{T}\}$, a second order stochastic process whose sample functions belong to the Hilbert space $L^2(\mathcal{T})$ of square integrable functions with the following scalar product,

$$\langle f, g \rangle = \int_T f(t)\, g(t)\, dt, \text{ for all } f, g \in L^2(\mathcal{T}), \tag{1.24}$$

and define $\{x_i(t) : t \in \mathcal{T}, i = 1, \ldots, n\}$ a sample of functions taken from the functional variable $X$. In practice, sample functions are usually measured on a finite set of time points that can vary between observations $\{t_{i0}, t_{i1}, \ldots, t_{im_i} \in \mathcal{T}\}$, and represented as finite dimensional vectors $\boldsymbol{x}_i = (x_{i0}, \ldots, x_{im_i})'$, where $x_{ik}$ represents the value of the $i$th observation at time point $k$. The first step when dealing with functional data is to reconstruct the functional form of the curves using the discrete observations available. Let us assume that the sample paths $x_i(t)$ belong to a finite dimensional space generated by a set of basis functions $\left\{\phi_1(t), \ldots, \phi_p(t)\right\}$ and express them in terms of these basis as,

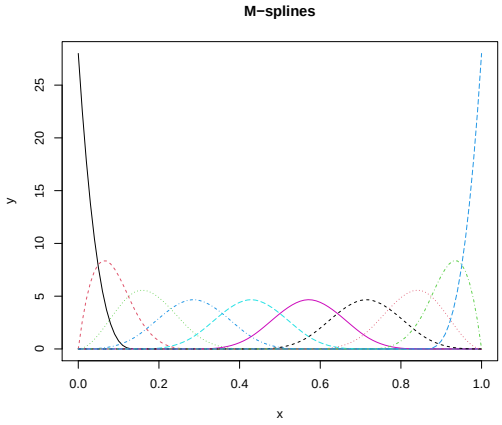
$$x_i(t) = \sum_{j=1}^{p} b_{ij}\phi_j(t), i = 1, \ldots, n, \tag{1.25}$$

where $b_{ij}$ is the basis coefficient for curve $i$ and basis $j$.

### 1.4.1 Spline basis

There are various types of basis that can be considered depending on the characteristics of the data, although in recent years the usage of splines has gained considerable

Figure 1.5: Representation of an M spline basis of degree 3.




importance due, among other factors, to the benefits of its computation and the flexibility they provide. This work considers the usage of M-splines.

A basis of M splines of degree $q$ generates a space of piece wise polynomial curves (called splines) of the same degree that join smoothly at a set of definition knots and that has $q-2$ continuous derivatives at interior knots. Probably the most used degree in practice is 3, called cubic splines, because it provides first and second continuous derivatives. Figure 1.5 shows an example of an M-spline basis of degree 3.

Regarding the estimation of the basis coefficients $bij$, there are also different alternatives depending on the nature of the data. In many applications the data are smooth functions observed with error,

$$x_{ik} = x_i(t_{ik}) + \varepsilon_{ik} \quad k = 0, \ldots, m_i, \quad i = 1, \ldots, n. \tag{1.26}$$

In this situation, one can consider the usage of a smooth approximation method like least squares regression,

$$\hat{\boldsymbol{b}}_i = \left\{ \arg\min \|\boldsymbol{x}_i - \Phi_i \boldsymbol{b}_i\|_2^2 \right\}, \tag{1.27}$$

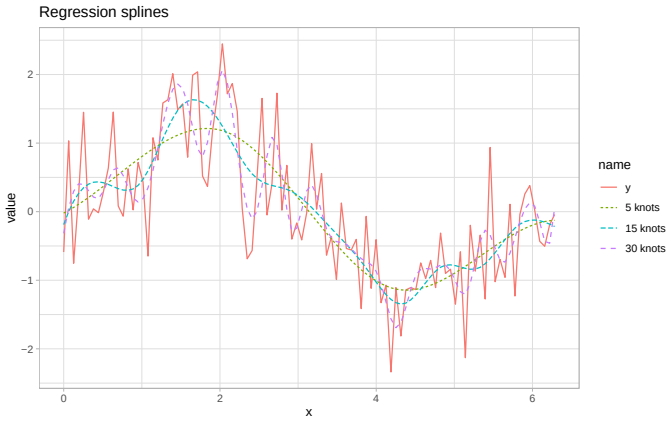
where $\Phi_i = \left(\phi_j(t_{ik})\right)_{m_i \times p}$. Although other regression formulations like quantile regression can also be used. Then, the closed form solution to this problem using least squares is,

$$\hat{\boldsymbol{b}}_i = \left(\Phi_i' \Phi_i\right)^{-1} \Phi_i' \boldsymbol{x}_i. \tag{1.28}$$

The smoothness obtained using regression splines is controlled by the size of the M-spline basis, which depends on the number of knots and the degree of the spline. Selecting a large number of knots can produce an overfitted estimation that is not filtering the noise. But selecting too small a number can produce an underfitted estimation.

**Example:** *This example shows the effect of the number of knots in regression splines. Consider the time period* $\mathcal{T} = [0, 2\pi]$ *and the functional model,* $x_k = x(t_k) + \varepsilon_k$ *with*

Figure 1.6: Regression splines using 5, 15 and 30 knots.




$x(t_k) = sin(t_k)$ *and* $\varepsilon_k \sim N(0, 0.7)$. *Figure 1.6 shows a comparison of regression splines using* 5, 15 *and* 30 *knots. One can see that the estimation using the largest number of knots produces an overfitted model that does not filter the noise.*

The smoothing splines defines a solution to the problem of the selection of the knots in regression splines by including a penalization term on the model formulation that controls the roughness of the function. This way, one can simply take a large number of knots and then control the smoothness via the penalization. A natural way of quantifying the roughness of a curve $x_i(t)$ is centered on the usage of it's derivatives of some order $d$, $D^d(x_i(t)) = x_i^d(t)$, $d \geq 1$. [O'Sullivan, 1986] proposed using the squared of the second derivative, which is called curvature, and defined the penalization,

$$\text{PEN}_2(x_i) = \int \left[D^2 x_i(t)\right]^2 dt = \boldsymbol{b}_i' R \boldsymbol{b}_i, \tag{1.29}$$

where $\boldsymbol{R}$ is the matrix of the cross inner products of the second order derivatives of basis functions $\phi$,

$$R = \int D^2\phi(s) D^2\phi(s)' ds. \tag{1.30}$$

Functions with large variability are expected to have large values of $\text{PEN}_2(x)$, and by controlling it, one can obtain smooth estimates and prevent overfitting the data. The smoothing splines estimator can be posed then as,

$$\hat{\boldsymbol{b}}_i = \left\{\arg\min \|\boldsymbol{x}_i - \Phi_i \boldsymbol{b}_i\|_2^2 + \boldsymbol{b}_i' R \boldsymbol{b}_i\right\}, \tag{1.31}$$

and has the following solution,

$$\hat{\boldsymbol{b}}_i = \left(\Phi_i'\Phi_i + \lambda R\right)^{-1} \Phi_i' \boldsymbol{x}_i, \tag{1.32}$$

where the hyper parameter $\lambda$ controls the effect of the penalization and can be optimized using some sort of cross validation process.

### 1.4.2 Functional principal component analysis

Principal component analysis was one of the first techniques to be extended to the functional framework with the objective to reduce the infinite dimension of a functional predictor and to explain its variation in terms of a reduced set of uncorrelated functions. Without loss of generality, let us assume that the observed curves are centered so that the sample mean $\frac{1}{n}\sum_{i=1}^{n} x_i(t)$ is zero. The functional principal components are linear combinations that maximize the variance from the data subject to the restriction of being orthogonal to other components. The $j$th component scores are expressed as,

$$\xi_{ij} = \int_T x_i(t) f_j(t) dt, i = 1, \ldots, n, \tag{1.33}$$

where the loading function $f_j$ is obtained as a solution to the following optimization problem,

$$\begin{cases} \hat{f(t)} = \arg\max \left\{ \text{var} \left[ \int_T x_i(t) f(t) dt \right] \right\} \\ \text{s.t. } \|f\|^2 = 1 \text{ and } \int f_\ell(t) f(t) dt = 0, \quad \ell = 1, \ldots, j-1 \end{cases} \tag{1.34}$$

Equivalently to traditional principal component analysis introduced in Section 1.3.1, it is possible to see that the loading functions are precisely the eigenfunctions of the covariance operator $C$ defined as,

$$C\left(f_j\right)(t) = \int C(t,s) f_j(s) ds = \lambda_j f_j(t), \tag{1.35}$$

where $C(t,s) = E[(X(t) - \mu(t))(X(s) - \mu(s))]$ is the covariance function.

Functional principal component analysis is widely used as a first step in functional data analysis. However, despite its many advantages, it is unable to capture the variability in the data coming from shifts in the scale that may affect the quantiles. This thesis makes contributions to this field by proposing a quantile based alternative.

## 1.5 Main contributions

The contributions of this thesis are built around two core ideas: the development of quantile based methodologies, suitable for high dimensional frameworks. The thesis is divided into four independent chapters that nonetheless have a shared main objective. Chapter 2 studies the usage of variable selection methods for quantile regression. It starts reviewing the available lasso based penalizations in the literature of quantile regression, and extends the sparse group lasso (SGL) to the quantile regression framework, where it was still not defined. A problem with lasso based penalizations is that they produce biased estimations, and this bias can affect the

quality of the variable selection and the prediction accuracy. As a solution to this problem [Zou et al., 2006] defined the adaptive lasso based on the usage of adaptive weights that help correcting this bias. In Chapter 2 this adaptive idea is used to define a new estimator, the adaptive sparse group lasso for quantile regression. Different alternatives for calculating the adaptive weights based on PCA and PLS are also proposed and studied through synthetic datasets and a genetic real dataset.

Chapter 3 studies an alternative solution to the high dimensional problem based on the usage of dimension reduction techniques. A widely used dimension reduction technique for regression problems is partial least squares (PLS). This methodology is capable of projecting the data into a subspace of orthogonal variables that maximize the covariance with the response data, and can be defined in terms of a covariance maximization problem. However, it provides mean based estimates and is greatly affected by outliers and skewness. Taking this as a starting point, this chapter studies different definitions for a quantile covariance metric and based on these metrics define the fast partial quantile regression (fPQR), an algorithm that enjoys the nice properties of PLS: it is a dimension reduction technique that obtains uncorrelated scores maximizing the quantile covariance between predictors and responses. But additionally, it is also a robust, quantile based methodology suitable for dealing with outliers, heteroscedastic or heavy tailed datasets. When the center of the response variable is of interest, the median estimation of fPQR is a robust alternative to PLS, but fPQR can also provide estimations of different quantile levels, giving a complete picture of the distribution of the data.

On a parallel study conducted as part of a research stay at the Columbia University Mailman School of Public Health, Chapter 3 defines the functional quantile factor model. This model is related to the field of functional data analysis (FDA), in which observations are not treated as multivariate vectors of scalars, but functions that change along time, and it is motivated by a real dataset study measuring levels of physical activity in children. Understanding the differences in patterns of physical activity between children can help developing better strategies to promote activity. This objective is usually achieved using functional principal component analysis (FPCA), a methodology based on multivariate PCA that provides an estimation of a set of orthogonal basis functions that best explain the variability in the functional data. However, FPCA is based on the mean, and as such, it is greatly affected by the same problems as other mean based estimations: outliers, skewness and heteroscedasticity. Additionally, understanding not just the mean trend, but the quantile trends of physical activity can provide very useful insights. As a solution to this problem, this chapter proposes the functional quantile factor model, a methodology that extends the concept of functional principal component analysis to the quantile regression framework, being able to obtain an estimate of the quantiles of the functional data conditional on a set of common functions.

Chapter 5 introduces **asgl**, a Python package that solves penalized least squares

and quantile regression models in low and high dimensional frameworks. It is possible to find implementations of lasso, group lasso and sparse group lasso penalizations for least squares and an implementation for lasso in quantile regression in R. However, none of the adaptive penalizations mentioned in Chapter 2 are available in this programming language. In Python the situation is worst, as only lasso penalized least squares is available, and there is no implementation of any penalized quantile regression model. The **asgl** package fills this gap, providing implementations of lasso, group lasso, sparse group lasso, and its adaptive counterparts for least squares and quantile regression, as well as different alternatives for the computation of the adaptive weights.

Finally, Chapter 6 summarizes the main results achieved in this thesis and discusses further lines of research.

# Bibliography


Sijmen de Jong. SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3):251–263, 3 1993. ISSN 0169-7439. doi: 10.1016/0169-7439(93)85002-X.

Roger Koenker. *Quantile Regression*. Cambridge university Press, 2005. ISBN 0521338255.

Finbarr O'Sullivan. A Statistical Perspective on Ill-posed Inverse Problems. *Statistical Science*, 1(4):502–527, 1986. ISSN 2168-8745. doi: 10.1214/ss/1177013525.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 4 2013. ISSN 10618600. doi: 10.1080/10618600.2012.681250.

Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. doi: 10.2307/2346178.

H Wold. Nonlinear Iterative Partial Least Squares (NIPALS) Modelling: Some Current Developments. In Paruchuri R Krishnaiah, editor, *Multivariate Analysis?III*, pages 383–407. Academic Press, 1973. ISBN 978-0-12-426653-7.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 68 (1):49–67, 2006.

Hui Zou. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 12 2006. ISSN 0162-1459. doi: 10.1198/016214506000000735.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. doi: 10.1198/106186006X113430.

# CHAPTER 2

## Adaptive Sparse Group LASSO in Quantile Regression

In *Advances in Data Analysis and Classification*, 15(3), 2021:547-573.

Álvaro Méndez Civieta[1,2], M. Carmen Aguilera-Morillo[2,3] and Rosa E. Lillo[1,2].

1. Department of Statistics, Universidad Carlos III de Madrid.
2. uc3m-Santander Big Data Institute.
3. Department of Applied Statistics and Operational Research, and Quality, Universitat Politècnica de València.

## Abstract

This paper studies the introduction of sparse group LASSO (SGL) to the quantile regression framework. Additionally, a more flexible version, an adaptive SGL is proposed based on the adaptive idea, this is, the usage of adaptive weights in the penalization. Adaptive estimators are usually focused on the study of the oracle property under asymptotic and double asymptotic frameworks. A key step on the demonstration of this property is to consider adaptive weights based on a initial $\sqrt{n}$-consistent estimator. In practice this implies the usage of a non penalized estimator that limits the adaptive solutions to low dimensional scenarios. In this work, several solutions, based on dimension reduction techniques PCA and PLS, are studied for the calculation of these weights in high dimensional frameworks. The benefits of this proposal are studied both in synthetic and real datasets.

**keywords:** High-dimension; Penalization; Regularization; Prediction; Weight calculation.

## 2.1 Introduction

Along years, regression has become a key method in statistics. Least squares (LS) regression estimates the conditional mean response of a variable as a function of the covariates. Usually, these models assume the errors to be centered, homoscedastic and independent. Making this assumptions, it is guaranteed that the LS estimator is the best linear unbiased estimator, or a BLUE estimator. Additionally, if the errors are assumed to be Gaussian one can perform finite sample studies. However, these hypothesis are not always verified in practical applications, and the LS estimator is known to be extremely sensitive to the presence of outliers or heavy tailed distributions, making it perform poorly when the errors are non Gaussian. Ever since the seminal work of Koenker and Bassett [1978], quantile regression (QR) models have gained importance when dealing with this kind of situations. QR models allow for a relaxation of the classical first two moment conditions over the model error. In addition, the errors in QR are not required to be Gausian. This means that QR offers robust estimators capable of dealing with heteroscedasticity and outliers. QR models can also estimate different quantile levels of a response variable, giving a precise insight of the relation between response and covariates at upper and lower tails. This can provide a much richer point of view than OLS regression. For a full review on quantile regression, we recommend [Koenker, 2005].

In recent years, high dimensional data in which the number of covariates $p$ is larger than the number of observations $n$ ($p \gg n$), has become increasingly common. This problem can be found in many different areas like computer vision and pattern recognition [Wright et al., 2010], climate data over different land regions [Chatterjee et al., 2011], and prediction of cancer recurrence based on patients genetic information [Simon et al., 2013], [Yahya Algamal and Hisyam Lee, 2019]. In these scenarios, variable selection gains in special importance offering sparse modeling alternatives that help identifying significant covariates and enhancing prediction accuracy. One of the first and most popular sparse regularization alternatives is LASSO, which was proposed by Tibshirani [1996] and adapted to the QR framework by Li and Zhu [2008], who developed the piece-wise linear solution of this technique. LASSO is a technique that penalizes each variable individually, enhancing thus individual sparsity. However, in many real applications variables are structured into groups, and group sparsity rather than individual sparsity is desired. One can think for example of a genetic dataset grouped into gene pathways. This problem was faced by the group LASSO penalization of Yuan and Lin [2006], and opened the doors to more complex penalizations like the sparse group LASSO [Friedman et al., 2010], which is a linear combination of LASSO and group LASSO providing solutions that are both between and within group sparse. With the same objective in mind, Zhou and Zhu [2010] proposed a hierarchical LASSO. Other studies have worked on properties for robust estimators in regression when the number of covariates increase with sample

size (see for example Huber and Ronchetti [2009]). In the same line, it is also worth mentioning the work from Loh [2017], that extends the usage of robust estimators, like those obtained using Hubert or Tuckey loss functions (among others) to high dimensional settings, introducing a set of generalized M-estimators capable of dealing with outliers in both the errors and the covariates terms. To the best of our knowledge, the SGL technique has not been studied in the framework of QR models, so this gap is addressed first, extending the SGL penalization to quantile regression.

Zou [2006] was the first to propose the usage of adaptive weights for each variable on the LASSO penalization as a way to increase the model flexibility and correct the estimator bias. This idea, generally known as the adaptive idea, was then extended to other penalizations. The weights of the adaptive idea are defined in the literature based on an initial $\sqrt{n}$-consistent estimator. Typically, this is the result of a nonpenalized model. This definition is a key step for the demonstration of the oracle property of the estimators (in the sense of Fan and Li [2001]), but it is also restrictive, as it limits the usage of adaptive penalizations just to the situations in which solving a nonpenalized model is a feasible first step. This approach, focused on the oracle property under asymptotic, or even double asymptotic frameworks is observed in Nardi and Rinaldo [2008] for the adaptive group LASSO, Ghosh [2011] for an adaptive elastic net, Ciuperca [2019] for the adaptive group LASSO in QR, Ciuperca [2017] for the adaptive fused LASSO in QR, Wu and Liu [2009] for the adaptive LASSO and SCAD penalizations in QR, and Zhao et al. [2014] for an adaptive hierarchical LASSO in QR among others. It is especially interesting to remark the work developed by Poignard [2018], in which an adaptive sparse group LASSO estimator suitable for low dimensional scenarios (with $n > p$) is proposed, studying its theoretical properties for a set of general convex loss functions.

The main contribution of this work lies here. An adaptive sparse group LASSO (ASGL) for quantile regression estimator is defined, working especially on enabling the usage of the ASGL estimator in high dimensional scenarios (with $p \gg n$). In order to achieve this objective, four alternatives for the weight calculation step are proposed. It is worth noting that these weight calculation alternatives can be used not only in the case of the ASGL estimator, but also in the rest of the adaptive-based estimators available in the literature. The performance of these alternatives is also studied in the case of low dimensional scenarios, making the proposed work a good alternative for both high dimensional and low dimensional problems.

The rest of the paper is organized as follows. In Section 2.2 some basic theoretical concepts are introduced, along with the formal definition of the sparse group LASSO in quantile regression. This definition is extended to the adaptive idea in Section 2.3, proposing the ASGL estimator. Section 2.4 discusses the main results regarding asymptotic behavior of adaptive estimators, and Section 2.5 introduces the weights calculation alternatives for high dimensional scenarios, as well as some remarks regarding the asymptotic behavior of the proposed alternatives. Simula-

tion results are divided into two blocks: Section 2.6 shows the advantages of this proposal in synthetic datasets in high and low dimensional scenarios considering a symmetric error distribution while the supplementary material shows a sensitivity analysis of the proposed methods under skewed distribution errors as well as the effect of different hyperparameter values. In Section 2.7 the proposed model is used in a real dataset, a genomic dataset including gene expression data of rat eye disease first shown in Scheetz et al. [2006]. The computational aspects of the problem are briefly commented in Section 2.8, and the conclusions are provided in Section 2.9.

## 2.2 Penalized quantile regression

Consider a sample of $n$ observations structured as $\mathbb{D} = (y_i, \boldsymbol{x}_i),\ i = 1, \ldots, n$ from some unknown population and define the following linear model,

$$y_i = \boldsymbol{x}_i^t \boldsymbol{\beta} + \varepsilon_i,\ i = 1, \ldots, n \tag{2.1}$$

where $y_i$ is the i-$th$ observation of the response variable, $\boldsymbol{x}_i \equiv (x_{i1}, \ldots, x_{ip})$ is the vector of $p$ covariates for observation $i$ and $\varepsilon_i$ is the error term.

Let us introduce now the quantile regression framework by defining the loss check function,

$$\rho_\tau(u) = u(\tau - I(u < 0)) \tag{2.2}$$

where $I(\cdot)$ is the indicator function. In their seminal work Koenker and Bassett [1978] proved that the $\tau$-$th$ quantile of the response variable can be estimated by solving the following optimization problem,

$$\tilde{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) \right\}. \tag{2.3}$$

where $R(\boldsymbol{\beta})$ defines the risk function of quantile regression,

$$R(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^t \boldsymbol{\beta}) \tag{2.4}$$

Quantile regression models allow for a relaxation of the classical first two moment conditions over the model errors $\varepsilon_i$ defined in equation 2.1. These errors are no longer required to be centered, homoscedastic or normally distributed, as stated in Koenker [2005], offering robust estimators capable of dealing with heteroscedasticity and outliers.

We call high dimensional scenarios to the datasets in which $p$ is much larger than $n$ ($p \gg n$). This problem is becoming more and more common nowadays, and can be observed in many different fields of research such as computer vision and pattern recognition [Wright et al., 2010], climate data over different land regions [Chatterjee et al., 2011] or prediction of cancer recurrence based on patients genetic information

[Simon et al., 2013]. An alternative that has been intensively studied in recent years for dealing with these scenarios is the penalization approach. By penalizing a regression model it is possible to perform variable selection and improve the accuracy and interpretability of the models.

One of the best known variable selection penalization methods is the least absolute selection and shrinkage operator, generally known as LASSO, proposed initially by Tibshirani [1996] which, in the case of the QR framework solves,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \lambda \left\|\boldsymbol{\beta}\right\|_1 \right\}, \tag{2.5}$$

where $R(\boldsymbol{\beta})$ is the QR risk function defined in equation (2.4). The LASSO penalization sends many $\boldsymbol{\beta}$ components to zero, offering sparse solutions and performing automatic variable selection. In the last years, many LASSO-based algorithms have been proposed. Yuan and Lin [2006] introduced the group LASSO penalization as an answer for the need to select variables not individually but at the group level. This penalization solves the following problem,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \lambda \sum_{l=1}^{K} \sqrt{p_l} \left\|\boldsymbol{\beta}^l\right\|_2 \right\}, \tag{2.6}$$

where $K$ is the number of groups, $\boldsymbol{\beta}^l \in \mathbb{R}^{p_l}$ are vectors of components of $\boldsymbol{\beta}$ from the l-$th$ group, and $p_l$ is the size of the l-$th$ group. The group LASSO penalization works in a similar way to LASSO, but while LASSO enhances sparsity at individual level, group LASSO enhances sparsity at group level, selecting, or sending to zero whole groups of variables.

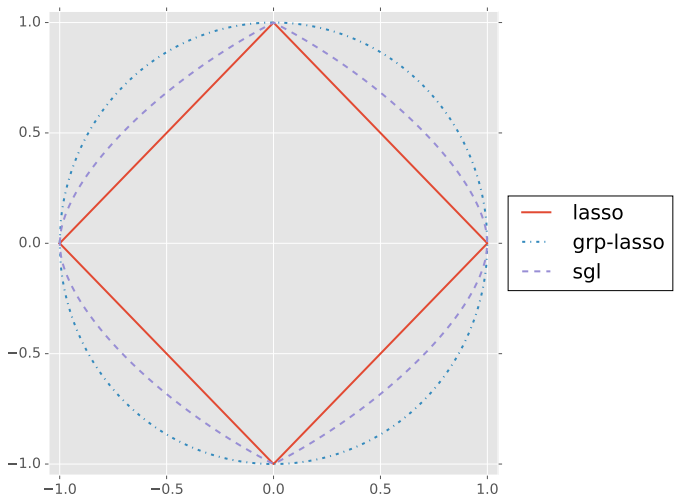
Initially proposed by Friedman et al. [2010], the sparse group LASSO (SGL) is a linear combination of LASSO and group LASSO penalizations. Well known in linear regression and other GLM models, to the best of our knowledge SGL has not been adapted to QR, and as a first step in the paper, this penalization is introduced.

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \alpha\lambda \left\|\boldsymbol{\beta}\right\|_1 + (1-\alpha)\lambda \sum_{l=1}^{K} \sqrt{p_l} \left\|\boldsymbol{\beta}^l\right\|_2 \right\}. \tag{2.7}$$

As in LASSO and group LASSO, SGL solutions are, in general, sparse, sending many of the predictor coefficients to zero. However, while LASSO solutions are sparse at individual level, and group LASSO solutions are sparse at group level, SGL offers both between and within group sparsity, outperforming both alternatives.

From an optimization perspective, equation (2.7) defines a sum of convex functions. This convexity ensures that the solution of the minimization problem is a global minimum. Figure 2.1 shows the constrains defined by LASSO, group LASSO and SGL in the case of a single 2-dimensional group of predictors.

Figure 2.1: Contour lines for LASSO, group-LASSO and sparse-group-LASSO penalties in the case of a single 2-dimensional group




## 2.3 Adaptive sparse group LASSO

From an empirical perspective, sparse group LASSO shows great performance. However, due to its mathematical formulation, it applies a constant penalization rate that provides biased estimates for large coefficients. The adaptive idea, initially introduced by Zou [2006] is considered here as a way to correct this limitation. In this work, a variant of the SGL penalization, the adaptive sparse group LASSO (ASGL) for quantile regression is defined. The ASGL estimator for QR is the result of the following minimization process,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \alpha\lambda \sum_{j=1}^{p} \tilde{w}_j |\beta_j| + (1-\alpha)\lambda \sum_{l=1}^{K} \sqrt{p_l}\tilde{v}_l \left\|\boldsymbol{\beta}^l\right\|_2 \right\}, \tag{2.8}$$

where $\tilde{\boldsymbol{w}} \in \mathbb{R}^p$ and $\tilde{\boldsymbol{v}} \in \mathbb{R}^K$ are known weights vectors and $R(\boldsymbol{\beta})$ is the risk function for quantile regression defined in equation 2.4. The intuition behind these weights is that if a variable (or group of variables) is important, it should have a small weight, and this way would be lightly penalized. On the other hand, if it is not important, by setting a large weight it is heavily penalized. This enhances the model flexibility and improves variable selection and prediction accuracy. It is worth saying that this formulation defines a convex function and thus, the global minimum can be found.

## 2.4 The oracle property

An estimator is oracle if it can correctly select the nonzero coefficients in a model with probability converging to one, and if the nonzero coefficients are asymptotically normally distributed. These properties were initially defined in Fan and Li [2001],

where they proved that the SCAD was an oracle estimator under an asymptotic framework of fixed dimension $p$. The oracle property of the SCAD estimator was then extended in Fan and Peng [2004] to a double asymptotic framework of $p$ depending on $n$. This is, $p \to \infty$ as $n \to \infty$, but $p$ growing at a lower rate and always $n > p$. Zou [2006] proved that the LASSO was not an oracle estimator due to the bias generated by the constant penalization rate. They proposed the usage of adaptive weights as a means to correct the bias, showing that the adaptive LASSO was an oracle estimator under the asymptotic framework of fixed $p$, as long as the weights required by the adaptive idea were computed based on a initial $\sqrt{n}$-consistent estimator. Actually, they proposed using the result from a non penalized model for the computation of the weights $\tilde{\boldsymbol{w}}$,

$$\tilde{w}_i = \frac{1}{|\tilde{\beta}_i|^\gamma}, \tag{2.9}$$

where $w_i$ and $\tilde{\beta}_i$ correspond to the i-$th$ element of vectors $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{\beta}}$ respectively, $|\cdot|$ denotes the absolute value function, $\gamma$ is a non negative constant and $\tilde{\boldsymbol{\beta}}$ is the solution vector obtained from the unpenalized model (described, in the case of the QR framework, in equation (2.3)).

Ever since then, the adaptive idea has been extended to many LASSO-based formulations in OLS, GLM and QR models among others. One can see for instance [Ghosh, 2011] where an adaptive elastic net is defined, [Wu and Liu, 2009] that introduces the adaptive LASSO in QR, [Ciuperca, 2017] where an adaptive fused LASSO in QR is defined, [Zhao et al., 2014] who proposes an adaptive hierarchical LASSO in QR or [Poignard, 2018], where an adaptive sparse group LASSO estimator is defined in a general set of convex functions, among others. All these works are centered on the demonstration of the oracle property under the asymptotic or double asymptotic framework, being the usage of an initial $\sqrt{n}$-consistent estimator on the calculations of the weights a key step in the demonstration. A major drawback of this approach in our opinion is precisely that the asymptotic or double asymptotic frameworks are limited to low dimensional scenarios where $n > p$ but do not consider high dimensional scenarios where $p \gg n$. This is remarked by the fact that usually, the initial $\sqrt{n}$-consistent estimators used in the weight calculations are taken from non penalized models, only feasible in low dimensional scenarios.

Dealing with the problem of an increasing number of covariates is, however, challenging. When an OLS model is considered, the third order term of the taylor expansion on the loss function vanishes, but out of this framework, for example in GLM or QR models, this term does not vanish, and additional boundaries on the convergence rates of $p$ (the number of variables) and $n$ (the number of observations) are required in order to demonstrate the consistency and the oracle property of the estimators. This is pointed out in detail, for a general framework of convex functions, in Poignard [2018].

When considering a high dimensional scenario it is possible to find very interest-

ing results from recent years. One can see for example [Huang et al., 2008a], who considers the oracle property of a bridge penalized least squares model under the $p \gg n$ framework as long as the bridge parameter is strictly between 0 and 1 (leaving out of the formulation the LASSO estimator). In order to achieve these results, they require additional conditions on the design matrix $\boldsymbol{X}$, namely, they require partial orthogonality between the set of significant variables and the set of non significant variables. Similar results can be observed for the adaptive LASSO in least squares [Huang et al., 2008b] where partial orthogonality conditions are required to demonstrate the oracle property in high dimensions, for the SCAD penalization in linear models in Kim et al. [2008] and for the SCAD and MCP penalizations in quantile regression in Wang et al. [2012]. However, the conditions required on the design matrix (and therefore on the covariates) to fit the oracle property are difficult to verify in practice. Thus, the results have an important mathematical relevance that should be landed in more realistic hypotheses.

## 2.5 Adaptive weights calculation

The objective of this section is to introduce different alternatives for the calculation of weights in the adaptive framework. The intuitive idea is to find a way to substitute $\tilde{\beta}$, the solution from the unpenalized model, unfeasible in high dimensional scenarios, in the calculation of the adaptive weights. This problem will be faced making use of two dimensionality reduction techniques, principal component analysis (PCA) and partial least squares (PLS). The proposed weight calculation alternatives can be used both in high dimensional and low dimensional scenarios. It is worth highlighting that these alternatives can be applied not only to the ASGL algorithm, but also to other adaptive based algorithms.

### 2.5.1 Principal components analysis

Given the covariates matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ defined in equation (2.1), with maximum rank $r = \min\{n, p\}$, consider the matrix of principal components $\boldsymbol{Q} \in \mathbb{R}^{p \times r}$ defined in a way such that the first principal component has the largest possible variance, and each succeeding component has the largest possible variance under the constraint that it is orthogonal to the preceding components. From an algebra perspective, the principal components in $\boldsymbol{Q}$ define an orthogonal change of basis matrix that maximize the variance explained from $\boldsymbol{X}$. Consider $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{Q} \in \mathbb{R}^{n \times r}$ the projection of $\boldsymbol{X}$ into the principal components subspace. Two weight calculation alternatives based on principal components are proposed.

**Based on a subset of components**

Consider the submatrix $\boldsymbol{Q}_d = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_d]^t$ where $\boldsymbol{q}_i \in \mathbb{R}^p$ is the i-*th* column of the matrix $\boldsymbol{Q}$, and $d \in \{1, \ldots, r\}$ is the number of components chosen. Let $\alpha_{pca,d} \in [0, 100]$ be the percentage of variability from $\boldsymbol{X}$ that the principal components in $\boldsymbol{Q}_d$ are able to explain. If $d = r$ then the principal components in $\boldsymbol{Q}_d$ are able to explain all the original variability from $\boldsymbol{X}$, and $\alpha_{pca,d} = 100$. If $d < r$ then $\alpha_{pca,d} < 100$. The number of components chosen in order to explain up to a certain percentage of variability is fixed by the researcher. Obtain $\boldsymbol{Z}_d = \boldsymbol{X}\boldsymbol{Q}_d \in \mathbb{R}^{n \times d}$ the projection of $\boldsymbol{X}$ into the subspace generated by $\boldsymbol{Q}_d$ and solve the unpenalized model,

$$\tilde{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^d}{\arg\min} \left\{ \frac{1}{n} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{z}_i^t \boldsymbol{\beta}) \right\}. \tag{2.10}$$

This model defines a low dimensional scenario where $\tilde{\boldsymbol{\beta}} \in \mathbb{R}^d$. Using this solution, it is possible to obtain an estimation of the high dimensional scenario solution, $\hat{\boldsymbol{\beta}} = \boldsymbol{Q}_d \tilde{\boldsymbol{\beta}} \in \mathbb{R}^p$. Finally, the weights are estimated as,

$$\tilde{w}_j = \frac{1}{|\hat{\beta}_j|^{\gamma_1}} \quad \text{and} \quad \tilde{v}_l = \frac{1}{\left\| \hat{\boldsymbol{\beta}}^l \right\|_2^{\gamma_2}}, \tag{2.11}$$

where $\hat{\beta}_j$ is the j-*th* component from $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\beta}}^l$ is the vector of components of $\boldsymbol{\beta}$ from the l-*th* group, and $\gamma_1$ and $\gamma_2$ are non negative constants usually taken in $[0, 2]$.

**Based on the first component**

A more straightforward approach based on the first principal component is also proposed. The principal components are no more than linear combinations of the original variables. Therefore, the first principal component $\boldsymbol{q}_1 \in \mathbb{R}^p$, which is the first column of the matrix $\boldsymbol{Q}$, includes one weight for each of the $p$ original variables. This proposal consists of calculating the weights as,

$$\tilde{w}_j = \frac{1}{|q_{1j}|^{\gamma_1}} \quad \text{and} \quad \tilde{v}_l = \frac{1}{\left\| \boldsymbol{q}_1^l \right\|_2^{\gamma_2}}, \tag{2.12}$$

where $q_{1j}$ is the j-*th* component from $\boldsymbol{q}_1$ and defines the weight associated to the j-*th* original variable, $\boldsymbol{q}_1^l$ is the vector of components of $\boldsymbol{q}_1$ from the l-*th* group and $\gamma_1$ and $\gamma_2$ are non negative constants usually taken in $[0, 2]$.

### 2.5.2 Partial least squares

The principal components are defined in a way such that they capture the maximum possible variance from $\boldsymbol{X}$ under the constraint that they are orthogonal to the rest of the principal components. However, being relevant for describing the variance of

$\boldsymbol{X}$ does not necessarily mean that a principal component is relevant for predicting the value of $\boldsymbol{y}$. Partial least squares (PLS) is a dimensionality reduction technique centered on maximizing the covariance between $\boldsymbol{X}$ and $\boldsymbol{y}$.

Given the covariates matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ defined in equation (2.1), with maximum rank $r = \min\{n, p\}$, consider the matrix of PLS components $\boldsymbol{T} \in \mathbb{R}^{p \times s}$ and the projection of $\boldsymbol{X}$ into the subspace generated by $\boldsymbol{T}$: $\boldsymbol{U} = \boldsymbol{X}\boldsymbol{T} \in \mathbb{R}^{n \times s}$. The matrix of PLS components $\boldsymbol{T}$ defines a nonorthogonal change of basis matrix whose projection $\boldsymbol{U}$ is computed in a way such that the first projection vector, $\boldsymbol{u_1} \in \mathbb{R}^n$ has the largest possible covariance with $\boldsymbol{y}$, and each succeeding projection vector has the largest possible covariance with $\boldsymbol{y}$ under the constraint that it is uncorrelated to the rest of the projection vectors.

Given the submatrix $\boldsymbol{T}_d = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_d]^t$ where $\boldsymbol{t}_i \in \mathbb{R}^p$ is the i-*th* column of the matrix $\boldsymbol{T}$, and $d \in \{1, \ldots, s\}$ is the number of components chosen, let $\alpha_{pls,d} \in [0, 100]$ be the percentage of variability from $\boldsymbol{X}$ that the PLS components in $\boldsymbol{T}_d$ are able to explain. The nonorthogonality of $\boldsymbol{T}$ implies that the total number of PLS components available to be computed is smaller than the rank of $\boldsymbol{X}$, $s \leq r$, and that the maximum possible percentage of variability explained by the PLS components $\alpha_{pls,s}$ is then lower than 100%.

In the case of principal components analysis, the matrix of principal components $\boldsymbol{Q}$ defines an orthogonal change of basis matrix that results into an orthogonal projection matrix $\boldsymbol{Z}$ maximizing the variance of $\boldsymbol{X}$. On the other hand, PLS defines a nonnecesarily orthogonal change of basis matrix $\boldsymbol{T}$ that results into an uncorrelated projection matrix $\boldsymbol{U}$ maximizing the covariance between $\boldsymbol{U}$ and $\boldsymbol{y}$. In the same way as for the PCA alternatives proposed, two alternatives of weight calculation using PLS are considered: based on a subset of PLS components, and based just on the first PLS component.

### 2.5.3 Influence of PCA and PLS on the oracle property

As commented in Section 2.4, a key condition in the demonstration of the oracle property in adaptive estimators is to assume that the initial estimator used in the weights calculation is $\sqrt{n}$-consistent.

The usage of $pca_d$ or $pls_d$ weight calculation proposes to consider a subset of $d$ components in the estimation of the weights. A question that may arise here is whether these PCA (or PLS) estimator is $\sqrt{n}$-consistent or not. We propose the following simple low dimensional example in the OLS framework that can help answering this question.

*Example:*
Given the random variables $X_1 \sim N(0, 0.99)$ and $X_2 \sim N(0, 0.01)$, consider the

random vector : $X = (X_1, X_2)$, for which

$$cov(X) = \begin{pmatrix} 0.99 & 0 \\ 0 & 0.01 \end{pmatrix}.$$

And thus, the eigenvalues from cov(X) are $\lambda_1 = 0.99$ and $\lambda_2 = 0.01$, and the matrix of eigenvectors is

$$P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

If PCA is applied on this random vector $X$, the rotation matrix obtained will be $P$, yielding to a first principal component that explains 99% of the original variability and a second principal component that explains the remaining 1%.

Consider now the following linear model,

$$y = X\beta + \varepsilon,$$

where $\beta = (0, 100)^t$ and $\varepsilon \sim N(0, 0)$. Following the steps described in Section 2.5.1, consider a subset of components that explain up to a certain percentage of variability, for example, 99% of the variability. This implies that $X$ will be projected onto the subspace spanned just by the first principal component $P_1$, $Z = XP_1 = X_1$. Solve now the linear model $\tilde{y} = Z\tilde{\beta}$, where

$$\tilde{\beta} = \frac{cov(Z, y)}{var(Z)} = \frac{cov(X_1, y)}{var(X_1)} = 0.$$

Then, the projection of the estimator $\tilde{\beta}$ into the original subspace is given by $\hat{\beta} = P_1\tilde{\beta} = (0, 0)^t$. Now, in order to be $\sqrt{n}$-consistent, an estimator should verify:

$$(\hat{\beta} - \beta) \text{ is } O_p(n^{-1/2}) \text{ if for all } \varepsilon > 0 \ \exists K > 0 \text{ s.t.}$$

$$P_{n\to\infty}(\sqrt{n}|\hat{\beta} - \beta| > K) < \varepsilon$$

Taking into account that $\beta = (0, 100)^t$, it is clear that the $\sqrt{n}$-consistency property is not verified by $\hat{\beta}$. The problem arises because the variability in variable $Y$ is explained by $X_2$, which is not selected because it explains only 1% of the total variability of $X$.

We would like to point out that this example is meant to be a counterexample of a situation in which the $pca_d$ is not $\sqrt{n}$-consistent. However, in our opinion, it clarifies the conditions required by the estimator in order to be consistent, as stated in the following remarks.

*Remark 1.* Consider an ASGL estimator, where the weights are computed based on a subset of principal components $pca_d$ in the asymptotic or double asymptotic frameworks. If all the components are selected (this is, if the components explain 100% of the original variability), then the initial estimator used in the weights

calculation is $\sqrt{n}$-consistent, and therefore, the ASGL estimator is an oracle estimator. Observe that by selecting all the components, $\hat{\beta} = Q\tilde{\beta}$ is equal to the unpenalized estimator defined in equation (2.3).

*Remark 2.* As shown in Section 2.4, the proof of the oracle property of an estimator in high dimensional scenarios is much more complex than in low dimensional scenarios. We conjecture that in the high dimensional context, the $pca_d$ estimator will behave in a similar way as in low dimensional scenarios, requiring to achieve a 100% of explained variability, but requiring also additional hypothesis similar to the ones observed in, for example, Wang et al. [2012]. In this paper, a set of 5 previous conditions is required for the demonstration of the oracle property in a high dimensional framework in quantile regression while considering non convex penalizations (such as SCAD). Among other things, the proposed conditions include restrictions on the design matrix, for example, that given the design matrix $\boldsymbol{X}$, $\boldsymbol{S} = \frac{1}{n}\boldsymbol{X}^t\boldsymbol{X}$ should be bounded, and the eigenvalues of $\boldsymbol{S}$ should be bounded as well. We consider that due to the complexity of the required results, studying the theoretical aspect of the estimator in high dimensional scenarios is a topic for further work. However, we study the behavior of this estimator in high dimensional scenarios both in synthetic and real datasets in Sections 2.6 and 2.7, and in the supplementary material, obtaining very good results.

*Remark 3.* The study of the oracle property of the $pls_d$ estimator is much more complex than this of $pca_d$. As commented in section 2.5.2, the maximum percentage of variability explained by the PLS components can be smaller than 100%, and thus, we would be facing the same issues described in the example above. This situation will also be a topic for further work.

## 2.6 Simulation study: symmetric errors

This section shows the performance of the proposed ASGL estimator under different synthetic dataset examples focused on symmetric errors as it is usual in OLS models. The proposed ASGL estimator is studied here under the framework of the following model,

$$y = X\beta + \varepsilon, \;\; \varepsilon \sim t(3),$$

where the data matrix $X$ is generated from a standard Gaussian distribution. Variables are organized in groups, considering a within group correlation of 0.5 and a between group correlation of 0. A quantile level $\tau = 0.5$ is considered. The scheme used here is an adaptation of other simulation schemes used in Wu and Liu [2009] and Zhao et al. [2014].

Given that the ASGL formulation in equation (2.8) includes a weight penalization on the group LASSO part based on the group size (the term $\sqrt{p_l}$), two model formulations are considered:

- Adaptive LASSO in sparse group LASSO (AL-SGL), where $\tilde{\boldsymbol{w}} \neq \mathbf{1}$ but $\tilde{\boldsymbol{v}} = \mathbf{1}$, in which the adaptive idea is only applied to the LASSO part.
- Adaptive sparse group LASSO (ASGL), where $\tilde{\boldsymbol{w}} \neq \mathbf{1}$ and $\tilde{\boldsymbol{v}} \neq \mathbf{1}$.

Furthermore, the four weight calculation alternatives proposed are studied:

- PCA weights based on regression on a subset of principal components, we denote this as $pca_d$;
- PCA weights based on the first principal component, we denote this as $pca_1$;
- PLS weights based on regression on a subset of PLS components, we denote this as $pls_d$;
- PLS weights based on the first PLS component, we denote this as $pls_1$.

The total number of components $d$ used in the weight estimation in $pls_d$ and $pca_d$ is chosen such that in both cases the percentage of variability explained from the original matrix $\boldsymbol{X}$ is $\alpha_{pca,d} = 80\%$, $\alpha_{pls,d} = 80\%$. As commented along Section 2.5, due to the non orthogonality of the PLS components it can happen that the maximum possible variability explained by the PLS components $\alpha_{pls,s}$ is smaller than 80%. In these cases we consider $d$ such that $\alpha_{pls,d} = \alpha_{pls,s}$.

The results obtained by the models proposed in this work are compared with the results from LASSO and SGL formulations. For each dataset $\mathbb{D}$, a partition into three disjoint subsets, $\mathbb{D}_{train}$, $\mathbb{D}_{val}$ and $\mathbb{D}_{test}$ is considered. $\mathbb{D}_{train}$ is used for training the models, this is, solving the model equations. $\mathbb{D}_{val}$ is used for validation, this is, optimizing the model parameters. This optimization is performed based on grid-search. Finally, $\mathbb{D}_{test}$ is used for testing the models prediction accuracy. The model parameters are optimized based on the minimization of the quantile error, defined as,

$$E_v = \frac{1}{\#\mathbb{D}_{val}} \sum_{(y_i, \boldsymbol{x_i}) \in \mathbb{D}_{val}} \rho_\tau(y_i - \boldsymbol{x}_i^t \hat{\boldsymbol{\beta}}), \tag{2.13}$$

where $\rho_\tau(\cdot)$ denotes the quantile function defined at (2.2), and # denotes the cardinal of a set. The final model error is calculated over $\mathbb{D}_{test}$ as,

$$E_t = \frac{1}{\#\mathbb{D}_{test}} \sum_{(y_i, \boldsymbol{x_i}) \in \mathbb{D}_{test}} \rho_\tau(y_i - \boldsymbol{x}_i^t \hat{\boldsymbol{\beta}}). \tag{2.14}$$

Additionally, the following metrics evaluating the performance of the methods are considered:

- $\left\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right\|_2$ the euclidean distance between the estimated vector and the true vector;
- true positive rate (TPR)= $\mathrm{P}(\hat{\beta}_i \neq 0 | \beta_i \neq 0)$;
- true negative rate (TNR)= $\mathrm{P}(\hat{\beta}_i = 0 | \beta_i = 0)$;
- correct selection rate (CSR)= $\mathrm{P}(\hat{\boldsymbol{\beta}} = \boldsymbol{\beta})$.

We are interested in studying the performance of the proposed models under different situations. An aspect to be analysed is the effect of an increase on the number of variables, and regarding this aspect, three cases will be considered:

- high-dimensional case with 625 variables;
- high-dimensional case with 225 variables;
- low dimensional case with 100 variables.

Additionally, another important factor is the spread of the significant variables among different groups. In order to study this aspect, two cases will be considered:

- sparse distribution of significant variables: significant variables are spread among many groups, but there is no group fully formed by significant variables;
- dense distribution of significant variables: significant variables are concentrated into a few number of groups, fully formed by significant variables.

Varying the number and the spread of the variables, six cases will be studied:

*Case 1: sparse distribution of* 625 *variables*
There are 25 groups of size 25 each, a total number of 625 variables. Among these groups, 7 groups with 8 significant variables each are defined, a total number of 56 significant variables. For $l \in \{1 \ldots, 25\}$, coefficients inside each group are defined as,

$$\begin{cases} \beta^l = (1, 2, \ldots, 8, \underbrace{0, \ldots, 0}_{17}), \ l = 1, \ldots, 7 \\ \beta^l = (\underbrace{0, \ldots, 0}_{25}), \ l = 8, \ldots, 25. \end{cases}$$

*Case 2: dense distribution of* 625 *variables*
There are 25 groups of size 25 each, a total number of 625 variables. Among these

groups, 3 groups with 25 significant variables each are defined, a total number of 75 significant variables. For $l \in \{1 \ldots, 25\}$, coefficients inside each group are defined as,

$$\begin{cases} \beta^l &= (1, 2, \ldots, 25), \ l = 1, \ldots, 3 \\ \beta^l &= (\underbrace{0, \ldots, 0}_{25}), \ l = 4, \ldots, 25. \end{cases}$$

*Case 3: sparse distribution of* 225 *variables*
There are 15 groups of size 15 each, a total number of 225 variables. Among these groups, 7 groups with 8 significant variables each are defined, a total number of 56 significant variables. For $l \in \{1 \ldots, 15\}$, coefficients inside each group are defined as,

$$\begin{cases} \beta^l &= (1, 2, \ldots, 8, \underbrace{0, \ldots, 0}_{7}), \ l = 1, \ldots, 7 \\ \beta^l &= (\underbrace{0, \ldots, 0}_{15}), \ l = 8, \ldots, 15. \end{cases}$$

*Case 4: dense distribution of* 225 *variables*
There are 15 groups of size 15 each, a total number of 225 variables. Among these groups, 3 groups with 15 significant variables each are defined, a total number of 45 significant variables. For $l \in \{1 \ldots, 15\}$, coefficients inside each group are defined as,

$$\begin{cases} \beta^l &= (1, 2, \ldots, 15), \ l = 1, \ldots, 3 \\ \beta^l &= (\underbrace{0, \ldots, 0}_{15}), \ l = 4, \ldots, 15. \end{cases}$$

*Case 5: sparse distribution of* 100 *variables*
There are 10 groups of size 10 each, a total number of 100 variables. Among these groups, 5 groups with 6 significant variables each are defined, a total number of 30 significant variables. For $l \in \{1 \ldots, 10\}$, coefficients inside each group are defined as,

$$\begin{cases} \beta^l &= (1, 2, \ldots, 6, \underbrace{0, \ldots, 0}_{4}), \ l = 1, \ldots, 5 \\ \beta^l &= (\underbrace{0, \ldots, 0}_{10}), \ l = 6, \ldots, 10. \end{cases}$$

*Case 6: dense distribution of* 100 *variables*
There are 10 groups of size 10 each, a total number of 100 variables. Among these groups, 3 groups with 10 significant variables each are defined, a total number of 30

significant variables. For $l \in \{1 \ldots, 10\}$, coefficients inside each group are defined as,

$$\begin{cases} \beta^l &= (1, 2, \ldots, 10), \ l = 1, \ldots, 3 \\ \beta^l &= (\underbrace{0, \ldots, 0}_{10}), \ l = 4, \ldots, 10. \end{cases}$$

We consider that *Case 1* is the most representative example in further applications, and therefore it will be intensively studied here, and also in the simulations regarding the sensitivity analysis shown in the supplementary material. Each simulation example has been executed 50 times considering 100/100/5000 observations in the train / validate / test samples, except in the low dimensional simulations (*Case* 5 and 6) where 500/500/5000 observations were considered. The large test sets formed by 5000 observations help increase the stability of the results, however, models are built using train and validate sets, making the 625 variables and 225 variables simulations high dimensional ($p > n$). The results have been summarized in terms of the mean and standard deviation values (shown in parenthesis), and the best result from each metric is highlighted.

As it was commented in Section 2.4, the general tendency found in the literature regarding the weights in adaptive models is to define them based on the results of the unpenalized model,

$$\tilde{w}_i = \frac{1}{|\tilde{\beta}_i|^\gamma}, \tag{2.15}$$

where $w_i$ and $\tilde{\beta}_i$ correspond to the i-*th* element of vectors $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{\beta}}$ respectively, $|\cdot|$ denotes the absolute value function, $\gamma$ is a non negative constant and $\tilde{\boldsymbol{\beta}}$ is the solution vector obtained from the unpenalized model (described, in the case of the QR framework, in equation (2.3)). This approach is limited just to low dimensional scenarios, where the unpenalized model can actually be solved. For this reason, in the low dimensional cases, the results of the proposed models are compared with the results from the weights based on the unpenalized model.

### 2.6.1 Simulation 1: sparse distribution of significant variables.

This simulation shows the results obtained under simulation *Case* 1, considering 625 variables, *Case* 3, considering 225 variables and *Case* 5, considering 100 variables. In all of them, the variables are sparsely distributed among groups, and a symmetric error from a t(3) is considered.

Results from this simulation scheme are displayed in Table 2.1, which is divided into three parts related to the three *Cases* under study. The first part of the table analyses *Case* 1, which considers 625 variables. In this part, the results from LASSO and SGL are compared against the eight proposed weight calculation alternatives commented before. Additionally, the performance of sparse variations of PCA and PLS is studied. These alternatives appear denoted as $spca_d$ (from sparse PCA)

Table 2.1: Simulation 1. Sparse distribution of variables. Considering a t(3) error.

| | $\|\hat{\beta} - \beta\|$ | $E_t$ | CSR | TPR | TNR |
|---|---|---|---|---|---|
| | | | $p$ = 625 variables | | |
| LASSO | 23.37 (4.61) | 7.85 (1.70) | **0.89** (0.01) | 0.76 (0.07) | **0.90** (0.01) |
| SGL | 19.62 (3.28) | 6.29 (1.08) | 0.76 (0.10) | 0.90 (0.04) | 0.75 (0.12) |
| AL-SGL-$pca_d$ | 17.97 (3.56) | 5.68 (1.13) | 0.83 (0.07) | 0.88 (0.05) | 0.83 (0.08) |
| AL-SGL-$pca_1$ | 21.41 (2.78) | 6.88 (0.93) | 0.70 (0.10) | 0.90 (0.04) | 0.68 (0.12) |
| AL-SGL-$pls_d$ | 17.60 (3.28) | 5.78 (1.14) | 0.83 (0.06) | 0.89 (0.04) | 0.83 (0.07) |
| AL-SGL-$pls_1$ | 19.40 (2.99) | 6.23 (0.99) | 0.78 (0.09) | 0.90 (0.04) | 0.77 (0.10) |
| ASGL-$pca_d$ | 15.19 (3.43) | 4.65 (1.04) | 0.84 (0.04) | **0.92** (0.03) | 0.83 (0.04) |
| ASGL-$pca_1$ | 21.38 (2.58) | 6.80 (0.87) | 0.73 (0.10) | 0.91 (0.04) | 0.71 (0.11) |
| ASGL-$pls_d$ | **13.23** (3.35) | **4.07** (0.99) | 0.85 (0.03) | 0.91 (0.04) | 0.84 (0.04) |
| ASGL-$pls_1$ | 17.56 (3.98) | 5.61 (1.33) | 0.81 (0.01) | 0.91 (0.04) | 0.80 (0.07) |
| ASGL-$spls_d$ | 14.31 (3.30) | 4.36 (0.99) | 0.85 (0.03) | 0.92(0.04) | 0.84 (0.04) |
| ASGL-$spca_d$ | 18.05 (3.19) | 5.75 (1.06) | 0.78 (0.07) | 0.91(0.03) | 0.77 (0.08) |
| | | | $p$ = 225 variables | | |
| LASSO | 8.09 (2.48) | 2.66 (0.81) | **0.80** (0.02) | 0.96 (0.03) | **0.75** (0.02) |
| SGL | 6.43 (2.02) | 2.12 (0.60) | 0.76 (0.06) | 0.98 (0.02) | 0.69 (0.07) |
| AL-SGL-$pca_d$ | 6.66 (2.33) | 2.20 (0.76) | 0.78 (0.06) | 0.97 (0.03) | 0.71 (0.08) |
| AL-SGL-$pca_1$ | 7.06 (1.98) | 2.30 (0.61) | 0.73 (0.06) | 0.98 (0.02) | 0.65 (0.09) |
| AL-SGL-$pls_d$ | 6.95 (1.79) | 2.28 (0.56) | 0.77 (0.06) | 0.97 (0.02) | 0.70 (0.08) |
| AL-SGL-$pls_1$ | 7.27 (2.46) | 2.39 (0.78) | 0.74 (0.06) | 0.98 (0.02) | 0.66 (0.08) |
| ASGL-$pca_d$ | 5.09 (1.32) | 1.70 (0.38) | 0.73 (0.09) | **0.99** (0.01) | 0.65 (0.12) |
| ASGL-$pca_1$ | 7.07 (1.98) | 2.31 (0.62) | 0.75 (0.06) | 0.98 (0.02) | 0.67 (0.07) |
| ASGL-$pls_d$ | **5.05** (1.30) | **1.68** (0.37) | 0.74 (0.09) | **0.99** (0.02) | 0.66 (0.12) |
| ASGL-$pls_1$ | 6.21 (1.78) | 2.04 (0.52) | 0.74 (0.05) | 0.98 (0.02) | 0.66 (0.06) |
| | | | $p$ = 100 variables | | |
| LASSO | 0.59 (0.08) | 0.59 (0.01) | 0.79 (0.09) | **1.00** (0.00) | 0.69 (0.14) |
| SGL | 0.60 (0.08) | 0.59 (0.01) | 0.75 (0.11) | **1.00** (0.00) | 0.64 (0.16) |
| ASGL-$pca_d$ | 0.55 (0.08) | 0.58 (0.01) | 0.81 (0.10) | **1.00** (0.00) | 0.73 (0.14) |
| ASGL-$pls_d$ | **0.45** (0.07) | **0.58** (0.06) | 0.95 (0.07) | **1.00** (0.00) | 0.93 (0.09) |
| ASGL-unpenalized | **0.45** (0.07) | **0.58** (0.05) | **0.96** (0.07) | **1.00** (0.00) | **0.95** (0.07) |

Figure 2.2: Simulation 1. Sparse distribution of 625 variables. Considering a t(3) error. Box-plots showing the test error of the different models.

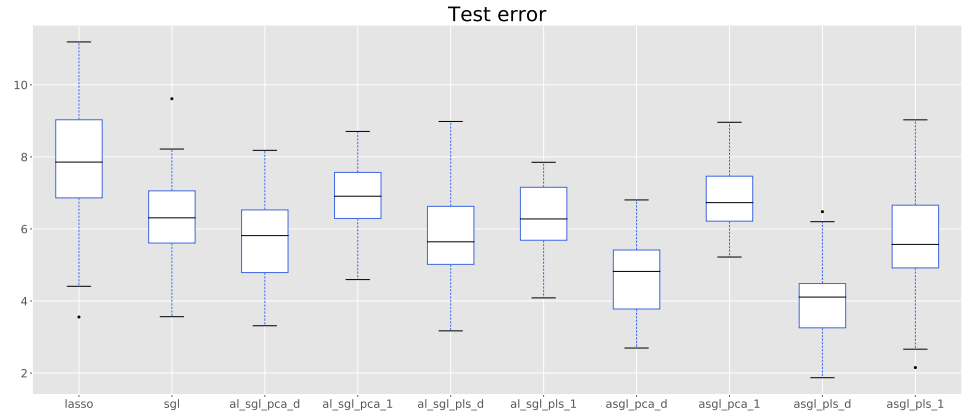
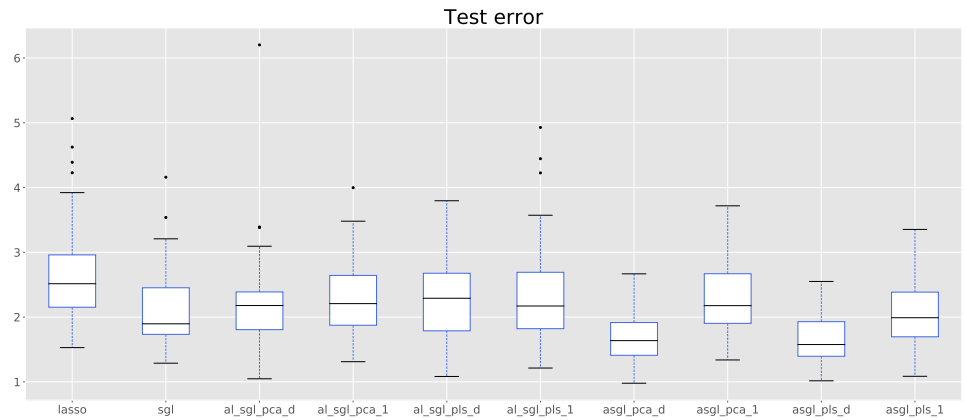



Figure 2.3: Simulation 1. Sparse distribution of 225 variables. Considering a t(3) error. Box-plots showing the test error of the different models.



and $spls_d$ (from sparse PLS). Sparse PCA was initially proposed by [Zou et al., 2006] as a method that computes principal components adding a LASSO based penalization to standard PCA. This yields to principal components that are sparse linear combinations of the original variables, though are no longer orthogonal. In the same sense, Chun and Keleş [2010] proposed an sparse alternative to PLS. Both alternatives are studied in this simulation.The best results here are obtained by the ASGL model using $pls_d$ weights, closely followed by $spls_d$ and $pca_d$ weights. This model outperforms LASSO and SGL both in terms of the distance between predicted and true $\boldsymbol{\beta}$, and in terms of the test error $E_t$. Given that LASSO enhances individual sparsity, LASSO solutions are more sparse than the solutions obtained by the proposed models , and this is shown in the TNR values. However, LASSO offers poor results in terms of the TPR (this is, in terms of the selection of the truly significant variables). SGL shows the opposite behavior, producing solutions with large TPR values but low TNR values. Compared to these techniques, the proposed ASGL formulations achieve good variable selection results both in terms of TNR and TPR. It is worth highlighting the results achieved using the sparse PCA ($spca_d$) and sparse PLS ($spls_d$) weights alternatives. As can be seen, the performance of $spca_d$ and $spls_d$ is worse than that of $pls_d$. Our guess is that establishing a double-sparsity framework, namely, sparse components used to estimate prior weights for an adaptive sparse group LASSO, is not that beneficial, and that simple PLS may be sufficient for the weight calculation, leaving the achievement of sparse solutions to the effect of the ASGL estimator. Additionally, using sparse PCA or sparse PLS in the weight calculation requires to optimize a series of parameters related to these techniques, and then another series of parameters related to the ASGL estimator. Finding the optimal solution in such a grid of parameters can be numerically cumbersome and time-consuming.

A similar behavior is observed in *Case 3*, that considers 225 variables. As before, the best results in terms of prediction accuracy are provided by ASGL $pls_d$ and $pca_d$ alternatives. Finally, the study performed in the low dimensional *Case* 5 is centered on the models achieving the best results among the proposals considered, namely $pls_d$ and $pca_d$ weights, that are compared against LASSO and SGL penalizations, and against the ASGL unpenalized, which is feasible only in this low dimensional framework and that consists in estimating the weights based on a unpenalized model (as it is usually done in the literature). It is worth to remark here that the $pls_d$ alternative performs just as well as the *unpenalized* one, which is a nice finding of this approach.

Figures 2.2 and 2.3 display box-plots of the test error $E_t$ for different models in the high dimensional frameworks, showing that the spread of $E_t$ is much smaller in the ASGL $pls_d$ and $pca_d$ than in the LASSO and SGL, indicating that these models provide more stable solutions in terms of prediction accuracy.

Table 2.2: Simulation 2. Dense distribution of variables. Considering a t(3) error.

| | $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|$ | $E_t$ | CSR | TPR | TNR |
|---|---|---|---|---|---|
| | | $p$ = 625 variables | | | |
| LASSO | 21.00 (13.00) | 7.13 (4.67) | **0.95** (0.01) | 0.96 (0.03) | **0.95** (0.01) |
| SGL | 6.02 (1.77) | 1.99 (0.56) | 0.82 (0.09) | **1.00** (0.01) | 0.80 (0.10) |
| AL-SGL-$pca_d$ | 4.32 (0.99) | 1.45 (0.28) | 0.94 (0.04) | **1.00** (0.01) | 0.93 (0.05) |
| AL-SGL-$pca_1$ | 7.17 (2.47) | 2.30 (0.75) | 0.72 (0.09) | **1.00** (0.01) | 0.68 (0.11) |
| AL-SGL-$pls_d$ | 4.81 (1.47) | 1.60 (0.44) | 0.92 (0.06) | **1.00** (0.01) | 0.90 (0.07) |
| AL-SGL-$pls_1$ | 5.38 (1.20) | 1.77 (0.57) | 0.87 (0.08) | **1.00** (0.01) | 0.85 (0.09) |
| ASGL-$pca_d$ | **3.61** (0.78) | **1.23** (0.20) | 0.92 (0.10) | **1.00** (0.01) | 0.90 (0.12) |
| ASGL-$pca_1$ | 7.60 (3.20) | 2.46 (1.01) | 0.74 (0.09) | **1.00** (0.01) | 0.71 (0.11) |
| ASGL-$pls_d$ | 3.85 (0.83) | 1.29 (0.21) | 0.85 (0.03) | **1.00** (0.01) | 0.89 (0.13) |
| ASGL-$pls_1$ | 4.17 (1.17) | 1.40 (0.32) | 0.90 (0.11) | **1.00** (0.01) | 0.87 (0.09) |
| | | $p$ = 225 variables | | | |
| LASSO | 4.43 (1.10) | 1.57 (0.35) | 0.87 (0.03) | 0.99 (0.01) | 0.83 (0.05) |
| SGL | 3.29 (0.75) | 1.21 (0.21) | 0.73 (0.13) | 0.99 (0.01) | 0.64 (0.17) |
| AL-SGL-$pca_d$ | 2.88 (0.50) | 1.07 (0.14) | 0.78 (0.06) | **1.00** (0.01) | 0.84 (0.11) |
| AL-SGL-$pca_1$ | 3.63 (0.73) | 1.30 (0.22) | 0.61 (0.15) | 0.99 (0.01) | 0.47 (0.21) |
| AL-SGL-$pls_d$ | 2.92 (0.57) | 1.09 (0.16) | 0.84 (0.12) | **1.00** (0.01) | 0.78 (0.16) |
| AL-SGL-$pls_1$ | 3.14 (0.65) | 1.16 (0.18) | 0.76 (0.14) | **1.00** (0.01) | 0.67 (0.20) |
| ASGL-$pca_d$ | **2.56** (0.49) | **0.98** (0.13) | **0.89** (0.12) | **1.00** (0.01) | **0.85** (0.16) |
| ASGL-$pca_1$ | 3.49 (0.79) | 1.25 (0.22) | 0.62 (0.15) | **1.00** (0.01) | 0.49 (0.21) |
| ASGL-$pls_d$ | 2.59 (0.43) | 0.99 (0.10) | 0.88 (0.16) | **1.00** (0.01) | 0.83 (0.21) |
| ASGL-$pls_1$ | 2.80 (0.53) | 1.05 (0.14) | 0.81 (0.12) | **1.00** (0.01) | 0.74 (0.17) |
| | | $p$ = 100 variables | | | |
| LASSO | 0.52 (0.08) | 0.58 (0.01) | 0.82 (0.10) | **1.00** (0.00) | 0.75 (0.13) |
| SGL | 0.50 (0.08) | 0.58 (0.01) | 0.74 (0.17) | **1.00** (0.00) | 0.63 (0.24) |
| ASGL-$pca_d$ | 0.45 (0.07) | **0.57** (0.01) | 0.92 (0.11) | **1.00** (0.00) | 0.88 (0.15) |
| ASGL-$pls_d$ | **0.44** (0.07) | **0.57** (0.01) | **0.95** (0.07) | **1.00** (0.00) | **0.93** (0.10) |
| ASGL-unpenalized | 0.45 (0.07) | **0.57** (0.01) | 0.92 (0.12) | **1.00** (0.00) | 0.89 (0.17) |

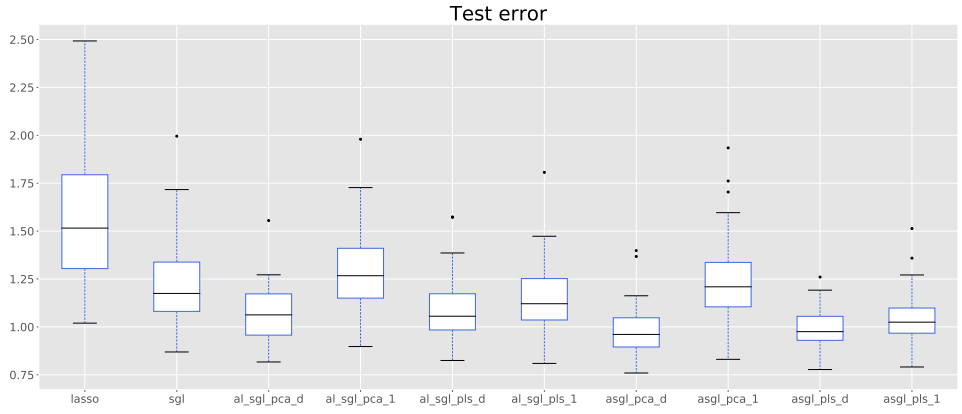
### 2.6.2 Simulation 2: dense distribution of significant variables.

This simulation shows the results obtained under simulation *Case* 2, considering 625 variables, *Case* 4, considering 225 variables and *Case* 6, considering 100 variables. In all of them, the variables are densely distributed among groups, and a symmetric error from a t(3) is considered.

The results from this simulation scheme are displayed in Table 2.2. Similar to the situation shown in the sparse distribution simulation, the ASGL model using $pls_d$ or $pca_d$ weights shows the best results in terms of the distance between predicted and true $\boldsymbol{\beta}$, and the value of $E_t$ in the high dimensional cases. These proposals offer also the best compromise between TPR and TNR. It is worth saying that under a more "compact" distribution of the significant variables in a small number of groups,

Figure 2.4: Simulation 2. Dense distribution of 625 variables. Considering a t(3) error. Box-plots showing the test error of the different models.


Figure 2.5: Simulation 2. Dense distribution of 225 variables. Considering a t(3) error. Box-plots showing the test error of the different models.



the proposed methods show a great improvement in terms of prediction accuracy compared to LASSO and SGL. As before, the low dimensional *case* is studied centered on the models achieving the best results among the proposals considered, $pls_d$ and $pca_d$ weights, that are compared against LASSO, SGL and ASGL *unpenalized* penalizations. It can be seen here that $pls_d$ is the one achieving the best results in this framework, closely followed by $pca_d$ and *unpenalized* results.

Figures 2.4 and 2.5 display box-plots of test error value $E_t$ in high dimensional scenarios, showing, as in the previous simulation scheme, that ASGL models with $pls_d$ or $pca_d$ weights also provide more stable results in terms of spread. Based on previous simulations, we conclude that the best performance both in the high dimensional and low dimensional frameworks, considering sparse or dense distribution of significant variables is achieved by ASGL models with $pls_d$ or $pca_d$ weights.

Additionally to the simulations shown here, a comprehensive sensitivity analysis that studies the behavior of the proposed methodology under different non symmetric error distributions, when varying the powers $\gamma_1$ and $\gamma_2$ entering the weights and when varying the number of PCA and PLS components chosen in the weight calculation can be found in the supplementary material.

## 2.7 Real application

The performance of the ASGL estimator is shown here using a genomic dataset first reported in Scheetz et al. [2006]. The dataset consists of 120 twelve-week-old male offspring animals chosen for tissue harvesting from the eyes and for microarray analysis. The dataset contains expression values from 31042 different probe-sets (Affymetric GeneChip Rat Genome 230 2.0 Array) on a logarithmic scale. As described in Huang et al. [2008b] and Wang et al. [2012], a two-steps preprocessing is performed, selecting, among the 31042 probe-sets, the ones that are sufficiently expressed, and sufficiently variable. A probe is considered to be sufficiently expressed if the maximum expression value observed for that probe among the 120 animals is greater than the 25-*th* percentile of the entire set of RMA expression values. A probe is considered to be sufficiently variable if it shows at least 2-fold variation in the expression value among the 120 rats. There are 18986 probes that meet these criteria.

We study how expression level of gene TRIM32, corresponding to probe 1389163_at, is related to expression levels at other probes. Chiang et al. [2006] pointed out that gene TRIM32 was found to cause Bardet-Biedl syndrome, a disease of multiple organ systems including the retina.[Scheetz et al., 2006, :1] stated: "Any genetic element that can be shown to alter the expression of a specific gene or gene family known to be involved in a specific disease is itself an excellent candidate for involvement in the disease, either primarily or as a genetic modifier." Here the

sample size is 120 (the number of animals selected for micro-array analysis), and the number of covariates (probes that pass the preprocessing steps) is 18985. The correlation coefficients of the 18985 probes and the probe corresponding to gene TRIM32 is calculated, and the genes in which the absolute value of the correlation exceeds 0.5 are selected. There are 3734 probes meeting this criteria. Finally, this dataset is standardized. Only a few genes are expected to be related to gene TRIM32, making this a high dimensional sparse problem.

From a biological perspective it is clear that genes do not work individually. The problem of grouping genes based on a medical criteria is nowadays under intense study, and it is possible to find some group structures for human genetic information based, for example, in cytogenetic positions [Subramanian et al., 2005]. It is interesting to remark that groups built based on biological criteria are usually formed just by a few dozens of genes. For example, in the case of groups based on cytogenetic positions, groups averaged 30 genes, as stated in Simon et al. [2013]. However, these group structures are not available for all the genetic information, and to the best of our knowledge there is no genetic grouping alternative for the dataset under study here.

We address the grouping problem from an statistical perspective, using principal components analysis to create groups of genes that are similar. It is worth to remark that in Section 2.5.1 PCA was used for estimating the ASGL weights, while here it will be used for variable clustering.

*Variable clustering using PCA*

1. Given a matrix of covariates $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ as in Section 2.5.1, obtain the matrix of principal components $\boldsymbol{Q} \in \mathbb{R}^{p \times r}$ $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ defined in Section 2.5.1.

2. Consider $r$ possible groups, as many as principal components.

3. Each principal component $\boldsymbol{q}_i \in \boldsymbol{Q}$, $i \in 1, \ldots, r$, is a linear combination of the original variables from $\boldsymbol{X}$. Assign each original variable to the group associated to the principal component in which that variable had its maximum weight (in absolute value).

The intuition behind this process is that variables with a large weight in the same principal component are likely to be related and should be included in the same group.

In the case of the dataset used in this section, there are 120 observations from 3734 different genes. The maximum rank of $\boldsymbol{X}$ here is 120, for this reason 120 possible groups are initially considered. Each gene is assigned to the group associated to the principal component in which that gene had its maximum weight. No gene was assigned to one of the groups, and therefore 119 groups averaging 32 genes per

Figure 2.6: Gene expression data of rat eye disease. Box-plot showing the sizes of the groups built using PCA.

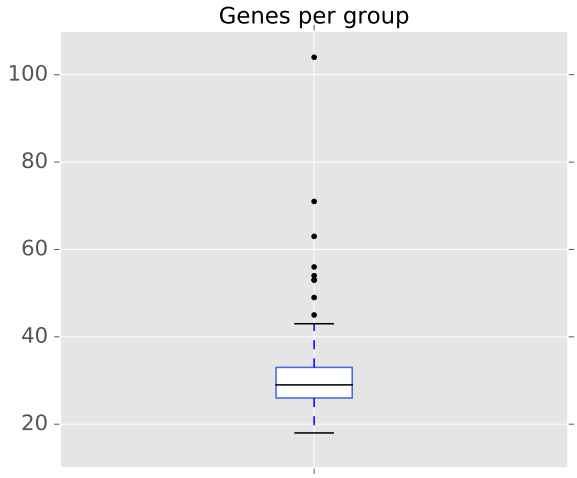



Table 2.3: Gene expression data of rat eye disease. 20 random dataset divisions were considered. Results displayed as mean value, with standard errors in parenthesis.

| | $E_t$ | # Variables selected |
|---|---|---|
| LASSO | 0.34 (0.08) | 18.9 (15.4) |
| SGL | 0.31 (0.07) | 189.5 (156.6) |
| ASGL-$pca_d$ | **0.28** (0.06) | 56.35 (70.86) |
| ASGL-$pls_d$ | 0.29 (0.06) | 101.7 (85.56) |

group are created this way. It is worth highlighting that the average group size obtained based on this proposal is close to the expected group size in terms of the cytogenetic position. Figure 2.6 shows a box-plot of the group sizes.

The dataset is randomly divided into 80/20/20 train / validate / test observations and LASSO, SGL, ASGL $pls_d$ and ASGL $pca_d$ models are solved. For each model, the test error $E_t$ and the significant variables selected are obtained. This process is repeated 20 times as a way to gain stability.

The results obtained are shown in Table 2.3. The best results in terms of the test error are obtained by the proposed ASGL models. LASSO offers a test error approximately 20% greater while SGL test error is 11% greater. Figure 2.7 displays box-plots of the test error $E_t$, showing that the spread of $E_t$ is also smaller in the proposed ASGL models providing more stable results. Figure 2.8 displays box-plots of the number of genes each model selected as significant. The LASSO is the one offering more sparse solutions, using only 19 variables (in mean) per model. SGL is the one using the largest number of variables, approximately 190, and also the one with the largest variability in this metric. Both ASGL $pca_d$ and ASGL $pls_d$ selected a smaller number of variables than SGL but still larger than LASSO, and they achieve the best prediction results of the four models.

Figure 2.7: Gene expression data of rat eye disease. 20 random dataset divisions were considered. Box-plot showing the test error.

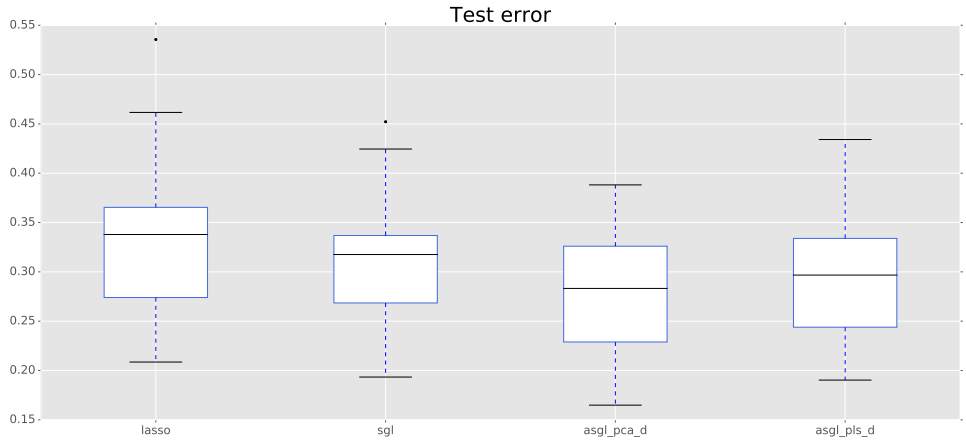



Figure 2.8: Gene expression data of rat eye disease. 20 random dataset divisions were considered. Box-plot showing the number of significant genes.

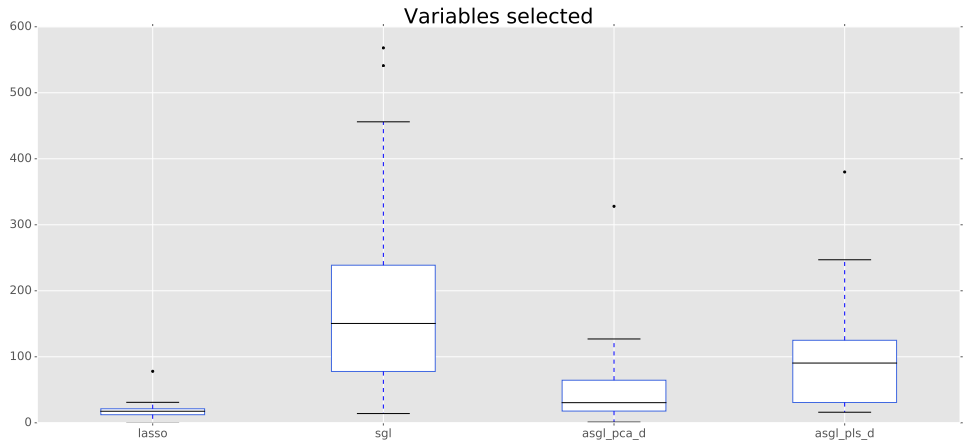
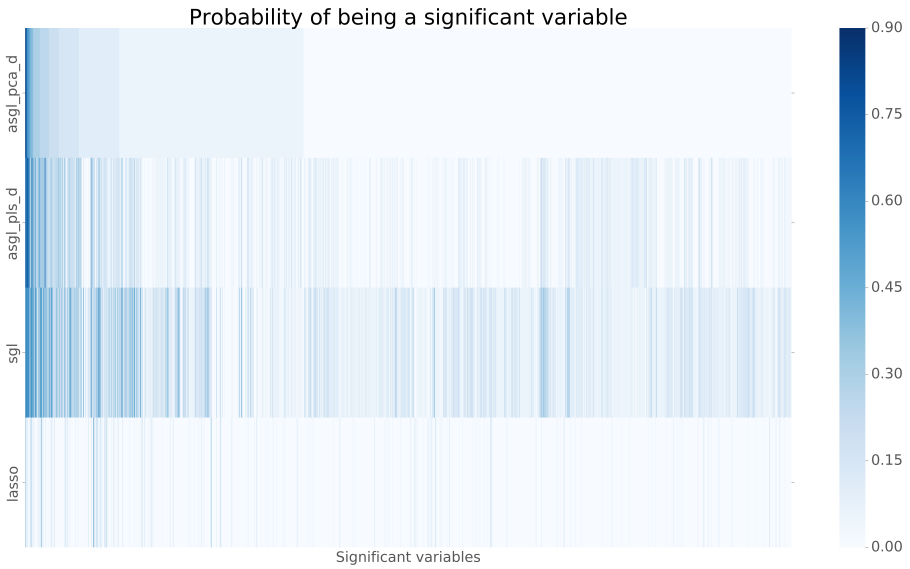


Figure 2.9: Gene expression data of rat eye disease. 20 random dataset divisions were considered. Heatmap showing the probability of being a significant gene. Each row represents a model and each column represents a gene.




Given that we have the results obtained from 20 repetitions, it is possible to count the number of times each gene has been selected as significant by one of the models in any of the repetitions. Dividing this number by the total number of repetitions, a sort of "probability of being a significant gene" associated to each gene for each model considered is obtained. Out of the 3734 genes in the dataset, 1612 genes were selected at least one time by any of the models in any of the repetitions (the majority being selected by SGL models). Figure 2.9 shows the probability of being a significant gene for these 1612 variables and for each model. Rows represent the different models considered and columns represent each gene. Genes are sorted based on the probabilities obtained in the ASGL model with $pca_d$ weights.

Considering a probability threshold of 0.5, only 1 gene in the LASSO models reach a probability of significance above the threshold, showing no stability on the gene selection along the 20 repetitions, and anticipating problems with possible further biological interpretation of the statistical results. In the case of the SGL model, 35 genes are above the probability threshold, being 0.6 the maximum probability achieved. On the other hand, the ASGL model with $pls_d$ weights includes 17 genes with probabilities above the threshold with a maximum probability value of 0.75, and the ASGL model with $pca_d$ weights has 9 genes above the probability threshold with a maximum probability value of 0.9, showing more stability on the selection along the 20 repetitions and possibly better biological interpretation of the results than the other models.

Results displayed in Table 2.3 and Figure 2.9 have been obtained using estimators of the median of the response variable, however, it can be interesting to compare the genes selected at different quantiles. For this reason, the process described above

Table 2.4: Gene expression data of rat eye disease. 20 random dataset divisions were considered. Number of genes above the probability threshold for different quantile levels.

| Number of genes above the probability threshold | | | | |
|---|---|---|---|---|
| | $\tau = 0.3$ | $\tau = 0.5$ | $\tau = 0.7$ | Three quantiles |
| LASSO | 0 | 1 | 1 | 0 |
| SGL | 19 | 35 | 17 | 0 |
| ASGL-$pca_d$ | 23 | 9 | 17 | 7 |
| ASGL-$pls_d$ | 41 | 17 | 37 | 9 |

is repeated and LASSO, SGL, ASGL $pls_d$ and ASGL $pca_d$ models are solved for quantile levels $\tau = 0.3$ and $\tau = 0.7$, obtaining probabilities of being a significant gene for each quantile level and each model. Considering a probability threshold of 0.5, Table 2.4 show the number of genes above the probability threshold for each quantile, and also the number of genes in the same model that have been selected along the different quantile levels.

The LASSO model shows no stability on the variable selection, having only one gene above the threshold for $\tau = 0.5$ and $\tau = 0.7$, and no gene with probability of being significant above 0.5 on the three quantiles simultaneously. The SGL shows some stability across the 20 repetitions considering each quantile independently, but when considering all the quantiles simultaneously it has no gene above the probability threshold. On the other hand, in the case of the ASGL $pls_d$ model, 9 genes had a probability of being significant greater than 0.5 in the 3 quantiles, and in the case of the ASGL $pca_d$ models, 7 genes fulfilled this, showing more robust results than the other estimators.

We conclude that the best results in this real dataset study are provided by the ASGL model with $pca_d$ weights, given that this model is the one with the smallest prediction error and showing great stability on the gene selection.

## 2.8 Computational aspects

All the simulations and data analysis commented in Sections 2.6, and 2.7 and in the supplementary material were run in a cluster node with two Intel (R) Xeon(R) CPU E5-2630 v3 (2.4GHz, 20MB Smart Cache) processors, with 32Gb of RAM memory running CentOS 6.5 Final (Rocks 6.1.1 Sand Boa). The computation itself has been developed in Python 2.7.15 (Anaconda Inc.). All the optimization problems have been solved using the CVXPY optimization framework for Python [Diamond and Boyd, 2016] and the open source solver ECOS [Domahidi et al., 2013].

## 2.9 Conclusion

In this paper the definition of the SGL estimator has been extended to the QR framework. A new estimator for quantile regression based on the usage of adaptive weights, the adaptive sparse group LASSO in quantile regression has also been proposed. As shown in Section 2.4, adaptive penalizations are typically centered on the study of the oracle property in both asymptotic and double asymptotic frameworks. A key step on the demonstration of this property is the usage of an initial $\sqrt{n}$-consistent estimator that is usually the result of a nonpenalized model. However, this definition limits the usage of adaptive estimators to low dimensional scenarios. As a solution to this problem, four weight calculation alternatives that can be used in high dimensional scenarios when working with adaptive estimators have been proposed. Section 2.5.3 conjectures about the relation between these alternatives and the oracle property. Additionally, the performance of the proposed alternatives have been analyzed in a set of synthetic data scenarios that includes high dimensional and low dimensional examples and symmetric error distributions (Section 2.6). Moreover, a thorough sensitivity analysis studying the behavior of the estimator under different error distributions, and under changes in parameter values has been performed in the supplementary material. The performance of the proposed work is also studied in a real high dimensional dataset including gene expression values of rat eye disease. Previous synthetic data analysis showed that the ASGL estimator is a competitive option in both high and low dimensional scenarios, especially when the adaptive weights are calculated based on subsets of PCA or PLS components. However, when dealing with the real dataset, the ASGL $pca_d$ estimator achieved better results in terms of prediction error and stability of the variables selected. For this reason we conclude that the ASGL $pca_d$ provides the best results among the options proposed in this work.

This work has risen some questions that will require further investigation. One interesting problem is the optimization of the hyper-parameters. In this work we make use of grid-search, but it is worth commenting that new hyper-parameter tuning alternatives have appeared in recent years [Laria et al., 2019], and it can be interesting to investigate the usage of this or other options in the optimization of the parameters of the models introduced in this work.

Section 2.5.3 has shown some concluding remarks related to the oracle property of the $pca_d$ weight calculation alternative. The $pls_d$ alternative based on PLS, however, is more complex and will require further research. In any case, it is worth mentioning the interesting work performed by Chun and Keleş [2010], that studies the consistency of the PLS estimator in the asymptotic and double asymptotic frameworks, reaching the conclusion (in *Theorem 1*) that given some previous as-

sumptions, if $\frac{p}{n} \to 0$, then

$$\left\|\beta^{PLS} - \beta\right\|_2 \to 0 \text{ in probability.}$$

This result would prove the consistency of the estimator, but It would not be enough for proving the $\sqrt{n}$-consistency, for this reason, we consider that the asymptotic property of the $pls_d$ alternative is a topic for future work.

Finally, simulations from Section 2.6 have studied different model formulations, including (suggested by a referee) the usage of sparse PCA and sparse PLS in the weight calculation process. The simulations showed that this alternative did not yield to better results than the non sparse PCA or PLS alternatives, but it can be interesting to study other sparse techniques.

## 2.10 Supplementary material

### 2.10.1 Simulation study: sensitivity analysis

This supplementary material shows a sensitivity analysis studying the effect of variations on the error distribution of the model as well as different parameters of the ASGL estimator proposed in the article.

#### Variation on the model errors

In order to perform well, OLS estimators need to set certain hypothesis on the model errors, namely, being centered, homoscedastic and normally distributed, that are no longer required in quantile regression models. Along this section, the behavior of the proposed ASGL QR estimator is studied under the framework of different error distributions that do not fulfill the OLS hypothesis, showing this way the benefits of the QR formulation.

**Simulation 3: Cauchy(0,3) error** In this section the proposed ASGL estimator is studied under the framework of the following model,

$$y = X\beta + \varepsilon,\ \varepsilon \sim \text{Cauchy}(0, 3),$$

The main characteristic of the Cauchy distribution is that the central moments in this distribution do not exist, making it an interesting variation on the model error. This distribution is a good example of heavy tail distributions which often appear in practical situations. This simulation show the results obtained under simulation *Case* 1, considering 625 variables sparsely distributed and *Case* 2, considering 625 variables densely distributed.

Table 2.5: Simulation 3. Considering 625 variables and a Cauchy(0, 3) error.

| | $\|\hat{\beta} - \beta\|$ | $E_t$ | CSR | TPR | TNR |
|---|---|---|---|---|---|
| 625 variables. Sparse distribution of variables | | | | | |
| LASSO | 33.69 (4.62) | 21.33 (10.53) | **0.87** (0.02) | 0.57 (0.08) | **0.91** (0.02) |
| SGL | 25.81 (1.92) | 18.43 (10.38) | 0.67 (0.12) | **0.89** (0.07) | 0.66 (0.13) |
| ASGL-$pca_d$ | **25.24** (2.08) | **17.89** (10.34) | 0.80 (0.05) | 0.87 (0.07) | 0.79 (0.06) |
| ASGL-$pca_1$ | 25.81 (2.07) | 18.40 (10.36) | 0.68 (0.14) | **0.89** (0.07) | 0.69 (0.15) |
| ASGL-$pls_d$ | 25.47 (2.14) | 18.15 (10.33) | 0.74 (0.08) | **0.89** (0.06) | 0.72 (0.09) |
| ASGL-$pls_1$ | 25.57 (2.16) | 18.19 (10.31) | 0.75 (0.09) | 0.87 (0.006 | 0.73 (0.10) |
| 625 variables. Dense distribution of variables | | | | | |
| LASSO | 57.52 (16.14) | 27.85 (10.71) | **0.95** (0.02) | 0.86 (0.06) | **0.96** (0.01) |
| SGL | 26.13 (5.30) | 17.65 (8.76) | 0.73 (0.13) | **0.99** (0.01) | 0.70 (0.15) |
| ASGL-$pca_d$ | **22.05** (4.90) | **16.25** (8.76) | 0.91 (0.11) | **0.99** (0.01) | 0.90 (0.13) |
| ASGL-$pca_1$ | 22.65 (5.36) | 17.50 (8.78) | 0.75 (0.11) | **0.99** (0.01) | 0.71 (0.13) |
| ASGL-$pls_d$ | 22.13 (5.07) | 16.28 (8.84) | 0.89 (0.11) | **0.99** (0.01) | 0.88 (0.13) |
| ASGL-$pls_1$ | 22.17 (4.84) | 16.28 (8.74) | 0.90 (0.09) | **0.99** (0.01) | 0.89 (0.10) |

Figure 2.10: Simulation 3. Sparse distribution of 625 variables. Considering a Cauchy(0, 3) error. Box-plots showing the test error of the different models.

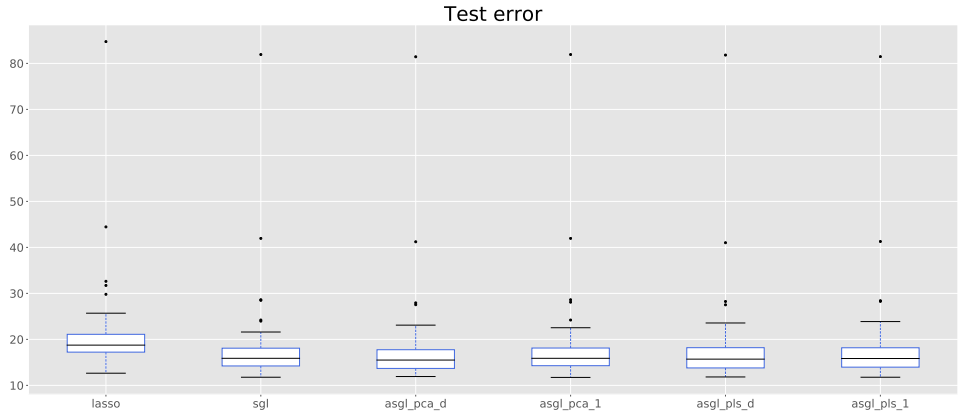



Figure 2.11: Simulation 3. Dense distribution of 625 variables. Considering a Cauchy(0, 3) error. Box-plots showing the test error of the different models.

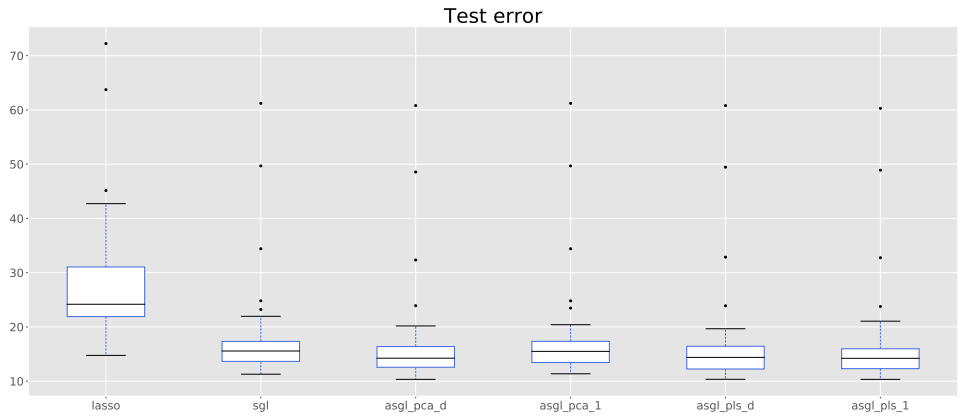


Table 2.6: Simulation 4. Considering 625 variables and a $\chi^2(3)$ error.

| | $\|\hat{\beta} - \beta\|$ | $E_t$ | CSR | TPR | TNR |
|---|---|---|---|---|---|
| 625 variables. Sparse distribution of variables | | | | | |
| LASSO | 23.36 (4.00) | 7.88 (1.54) | **0.89** (0.01) | 0.75 (0.06) | **0.90** (0.01) |
| SGL | 18.97 (2.99) | 6.10 (1.00) | 0.78 (0.09) | 0.88 (0.04) | 0.77 (0.10) |
| ASGL-$pca_d$ | **14.77** (3.19) | **4.62** (0.97) | 0.84 (0.04) | 0.90 (0.03) | 0.83 (0.04) |
| ASGL-$pca_1$ | 18.84 (2.97) | 6.07 (1.00) | 0.78 (0.07) | 0.88 (0.03) | 0.77 (0.08) |
| ASGL-$pls_d$ | 15.09 (3.07) | 4.71 (0.90) | 0.83 (0.04) | **0.91** (0.03) | 0.82 (0.04) |
| ASGL-$pls_1$ | 15.09 (3.16) | 4.75 (0.99) | 0.82 (0.04) | 0.90 (0.03) | 0.82 (0.04) |
| 625 variables. Dense distribution of variables | | | | | |
| LASSO | 20.06 (11.52) | 6.71 (3.88) | **0.95** (0.01) | 0.96 (0.03) | **0.95** (0.01) |
| SGL | 8.89 (2.23) | 2.80 (0.69) | 0.78 (0.10) | **0.99** (0.01) | 0.75 (0.12) |
| ASGL-$pca_d$ | 5.79 (1.00) | 1.96 (0.28) | 0.90 (0.13) | **0.99** (0.01) | 0.88 (0.14) |
| ASGL-$pca_1$ | 8.17 (2.30) | 2.73 (0.71) | 0.80 (0.09) | **0.99** (0.01) | 0.77 (0.11) |
| ASGL-$pls_d$ | **5.75** (1.04) | **1.95** (0.29) | 0.89 (0.12) | **0.99** (0.01) | 0.87 (0.13) |
| ASGL-$pls_1$ | 5.92 (1.09) | 1.99 (0.29) | 0.89 (0.08) | **0.99** (0.01) | 0.88 (0.09) |

The results from this simulation scheme are displayed in Table 2.5. Both in the case of the sparse or the dense distribution of the significant variables, the best results in terms of the distance between predicted and true $\boldsymbol{\beta}$, and the value of $E_t$ are achieved by the proposed ASGL estimator using $pca_d$ weights. The difference in terms of prediction error among models (excepting LASSO, which shows by far the largest error) is smaller in this simulation than in symmetric error ones shown in the article, probably due to the large tails of Cauchy distributions and the associated outliers. However, even under this framework, it is interesting to see that the proposed models offer a good variable selection performance both in terms of TPR and TNR as opposed to lasso (with large TNR but very low TPR) or SGL (with large TPR but low TNR). Figures 2.10 and 2.11 display box-plots of the test error value $E_t$, showing clearly the presence of outliers.

**Simulation 4: $\chi^2(3)$ error** In this section the proposed ASGL estimator is studied under the framework of the following model,

$$y = X\beta + \varepsilon,\ \varepsilon \sim \chi^2(3),$$

The $\chi^2$ distribution is non symmetric as opposed to previous error distributions $t$ and Cauchy that were symmetric. This simulation show the results obtained under simulation *Case* 1, considering 625 variables sparsely distributed and *Case* 2, considering 625 variables densely distributed.

The results from this simulation scheme are displayed in Table 2.6. The best results in terms of the distance between predicted and true $\boldsymbol{\beta}$, and in terms of the test error $E_t$ are obtained by the ASGL model using $pca_d$ weights in the sparse

Figure 2.12: Simulation 4. Sparse distribution of 625 variables. Considering a $\chi(3)$ error. Box-plots showing the test error of the different models.

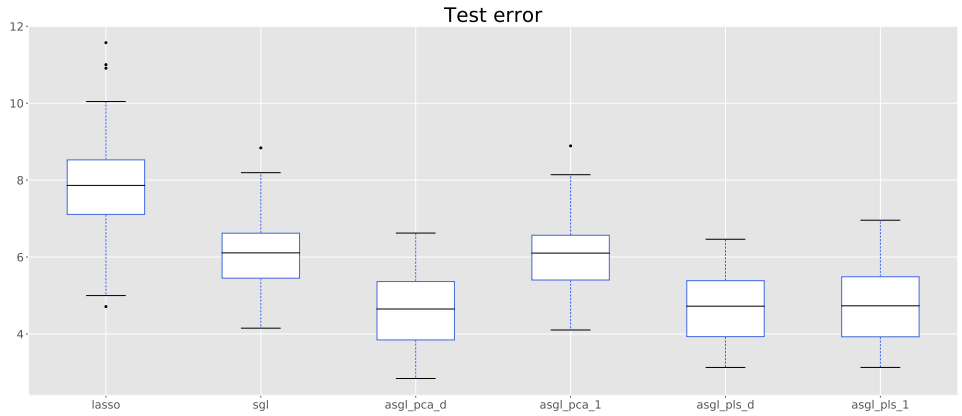
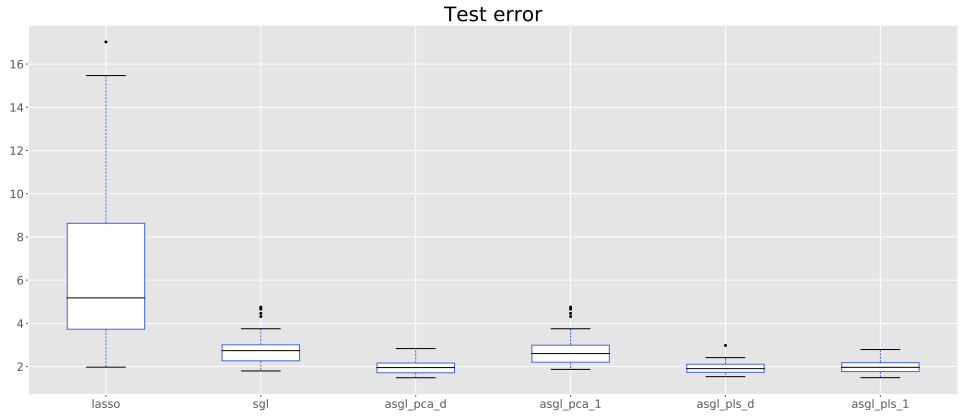



Figure 2.13: Simulation 4. Dense distribution of 625 variables. Considering a $\chi(3)$ error. Box-plots showing the test error of the different models.



*Case* 1 and $pls_d$ weights in the dense *Case* 2, though both methods provide quite similar solutions. As in previous simulations, LASSO show a larger TNR value, being the most sparse solution, but also the worst TPR performance, meaning that the selection of significant variables is not very accurate. Opposed to this behavior, SGL show good TPR value but worse TNR, selecting too many non significant variables. The proposed ASGL estimator provides good results both in terms of TPR and TNR. Figures 2.12 and 2.13 display box-plots of the test error $E_t$ for the different models, showing that the spread of $E_t$ is much smaller in the ASGL $pls_d$ and $pca_d$ than in the LASSO and SGL (especially in the dense case), indicating that these models provide more stable solutions in terms of prediction accuracy.

**Simulation 5: influence of $\gamma_1$ and $\gamma_2$**

Given equations

$$\tilde{w}_j = \frac{1}{|\hat{\beta}_j|^{\gamma_1}} \quad \text{and} \quad \tilde{v}_l = \frac{1}{\left\|\hat{\boldsymbol{\beta}}^l\right\|_2^{\gamma_2}}, \tag{2.16}$$

and

$$\tilde{w}_j = \frac{1}{|q_{1j}|^{\gamma_1}} \quad \text{and} \quad \tilde{v}_l = \frac{1}{\left\|\boldsymbol{q}_1^l\right\|_2^{\gamma_2}}, \tag{2.17}$$

for the calculation of the weights, one can see that the formulation includes two nonnegative parameters, $\gamma_1$ in the lasso weights part and $\gamma_2$ in the group lasso weights part that are the powers entering the weights. Along this section a simulation studying the influence of the value of these parameters is performed. The simulation scheme is that of *Case 1*: 625 variables sparsely distributed. Additionally, a t(3) distribution error is considered, and the weights are calculated based on a subset of PCA components $pca_d$. Two situations are studied: the behavior of the ASGL estimator while varying the value of $\gamma_2$ and leaving $\gamma_1 = 1$ and the behavior of the ASGL estimator while varying the value of $\gamma_1$ and leaving $\gamma_2 = 1$. The results are compared against the LASSO, SGL and the ASGL estimator optimizing both $\gamma_1$ and $\gamma_2$.

The results obtained in this simulation are displayed in Table 2.7 and Figure 2.14. The best results in terms of the distance between predicted and true $\boldsymbol{\beta}$, and the value of $E_t$ are provided by the ASGL estimator while optimizing both $\gamma_1$ and $\gamma_2$, highlighting the importance of the selection of this parameters. It is also interesting to observe how, while fixing $\gamma_1$, errors decrease as $\gamma_2$ increase, but while fixing $\gamma_2$, the opposite behavior appears, and errors increase as $\gamma_1$ increase.

**Influence of $\alpha_{pca,d}$ and $\alpha_{pls,d}$**

The weight calculation alternatives $pca_d$ and $pls_d$ are based on selecting a subset of $d$ either PCA or PLS components that explain up to a certain percentage of

Table 2.7: Simulation 5. Sparse distribution of 625 variables. Considering a t(3) error. Analysis of $\gamma_1$ and $\gamma_2$ influence.

| | $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|$ | $E_t$ | CSR | TPR | TNR |
|---|---|---|---|---|---|
| LASSO | 23.88 (4.35) | 8.02 (1.60) | **0.88** (0.01) | 0.75 (0.06) | **0.90** (0.01) |
| SGL | 19.40 (2.74) | 6.19 (0.88) | 0.77 (0.07) | 0.89 (0.04) | 0.76 (0.08) |
| ASGL | **15.14** (2.97) | **4.66** (0.87) | 0.83 (0.03) | **0.92** (0.03) | 0.82 (0.04) |
| $\gamma_1 = 1$ fixed. Varying $\gamma_2$ | | | | | |
| ASGL-$\gamma_2 = 0.0$ | 19.74 (2.94) | 6.23 (0.94) | 0.81 (0.07) | 0.89 (0.05) | 0.81 (0.08) |
| ASGL-$\gamma_2 = 0.2$ | 19.42 (2.97) | 6.08 (0.92) | 0.72 (0.07) | 0.89 (0.05) | 0.81 (0.08) |
| ASGL-$\gamma_2 = 0.4$ | 19.08 (2.83) | 5.95 (0.87) | 0.83 (0.05) | 0.89 (0.05) | 0.82 (0.06) |
| ASGL-$\gamma_2 = 0.6$ | 18.74 (2.79) | 5.80 (0.85) | 0.83 (0.05) | 0.89 (0.04) | 0.83 (0.05) |
| ASGL-$\gamma_2 = 0.8$ | 18.65 (2.97) | 5.75 (0.88) | 0.84 (0.04) | 0.90 (0.04) | 0.84 (0.05) |
| ASGL-$\gamma_2 = 1.0$ | 18.38 (3.07) | 5.66 (0.90) | 0.85 (0.04) | 0.90 (0.04) | 0.85 (0.04) |
| ASGL-$\gamma_2 = 1.2$ | 18.24 (3.19) | 5.61 (0.94) | 0.86 (0.03) | 0.90 (0.04) | 0.85 (0.04) |
| ASGL-$\gamma_2 = 1.4$ | 18.08 (3.32) | 5.56 (0.97) | 0.87 (0.02) | 0.90 (0.04) | 0.86 (0.03) |
| $\gamma_2 = 1$ fixed. Varying $\gamma_1$ | | | | | |
| ASGL-$\gamma_1 = 0.0$ | 16.23 (2.79) | 5.03 (0.80) | 0.80 (0.04) | 0.91 (0.03) | 0.79 (0.05) |
| ASGL-$\gamma_1 = 0.2$ | 16.23 (2.91) | 5.03 (0.85) | 0.82 (0.04) | 0.91 (0.03) | 0.81 (0.08) |
| ASGL-$\gamma_1 = 0.4$ | 16.54 (2.93) | 5.12 (0.87) | 0.84 (0.03) | 0.91 (0.03) | 0.83 (0.04) |
| ASGL-$\gamma_1 = 0.6$ | 17.07 (2.94) | 5.28 (0.88) | 0.84 (0.04) | 0.90 (0.04) | 0.84 (0.04) |
| ASGL-$\gamma_1 = 0.8$ | 17.69 (2.96) | 5.46 (0.89) | 0.85 (0.03) | 0.90 (0.04) | 0.84 (0.04) |
| ASGL-$\gamma_1 = 1.0$ | 18.38 (3.07) | 5.66 (0.90) | 0.85 (0.03) | 0.90 (0.04) | 0.85 (0.04) |
| ASGL-$\gamma_1 = 1.2$ | 18.90 (3.07) | 5.81 (0.89) | 0.85 (0.04) | 0.90 (0.05) | 0.84 (0.05) |
| ASGL-$\gamma_1 = 1.4$ | 19.47 (3.02) | 5.96 (0.86) | 0.85 (0.04) | 0.90 (0.05) | 0.85 (0.04) |

Figure 2.14: Simulation 5. Sparse distribution of 625 variables. Considering a t(3) error. Analysis of $\gamma_1$ and $\gamma_2$ influence. Box-plots showing the test error of the different models.

Table 2.8: Simulation 6. Sparse distribution of 625 variables. Considering a t(3) error. Analysis of $\alpha_{pca,d}$ influence.

| | $\|\hat{\beta}-\beta\|$ | $E_t$ | CSR | TPR | TNR |
|---|---|---|---|---|---|
| 625 variables. Sparse distribution of variables. | | | | | |
| LASSO | 21.85 (4.77) | 7.40 (1.77) | 0.89 (0.07) | 0.77 (0.07) | 0.90 (0.08) |
| SGL | 18.14 (3.28) | 5.80 (1.07) | 0.80 (0.06) | 0.89 (0.05) | 0.79 (0.10) |
| ASGL-$pca$ – 10% | 17.96 (3.32) | 5.76 (1.09) | 0.80 (0.08) | 0.89 (0.05) | 0.79 (0.09) |
| ASGL-$pca$ – 20% | 17.54 (3.47) | 5.60 (1.13) | 0.81 (0.07) | 0.89 (0.05) | 0.80 (0.07) |
| ASGL-$pca$ – 30% | 17.54 (3.45) | 5.60 (1.12) | 0.82 (0.06) | 0.90 (0.05) | 0.79 (0.09) |
| ASGL-$pca$ – 40% | 16.73 (3.78) | 5.33 (1.22) | 0.84 (0.04) | 0.90 (0.04) | 0.80 (0.08) |
| ASGL-$pca$ – 50% | 15.47 (3.78) | 4.92 (1.25) | 0.84 (0.04) | 0.90 (0.04) | 0.82 (0.07) |
| ASGL-$pca$ – 60% | 13.35 (3.47) | 4.15 (1.16) | 0.84 (0.04) | 0.92 (0.04) | 0.83 (0.05) |
| ASGL-$pca$ – 70% | 12.76 (3.37) | 3.92 (1.04) | 0.84 (0.04) | 0.93 (0.04) | 0.83 (0.05) |
| ASGL-$pca$ – 80% | 12.98 (3.36) | 4.01 (1.02) | 0.84 (0.04) | 0.93 (0.04) | 0.83 (0.04) |
| ASGL-$pca$ – 90% | 13.04 (3.41) | 4.04 (1.03) | 0.84 (0.04) | 0.92 (0.04) | 0.84 (0.04) |
| ASGL-$pca$ – 100% | 14.08 (3.76) | 4.34 (0.16) | 0.84 (0.03) | 0.92 (0.04) | 0.84 (0.03) |
| 100 variables. Sparse distribution of variables. | | | | | |
| LASSO | 0.58 (0.08) | 0.59 (0.01) | 0.73 (0.01) | 1.00 (0.00) | 0.66 (0.14) |
| SGL | 0.60 (0.08) | 0.59 (0.01) | 0.72 (0.12) | 1.00 (0.00) | 0.57 (0.17) |
| ASGL-$pca$ – 10% | 0.59 (0.07) | 0.59 (0.01) | 0.83 (0.10) | 1.00 (0.00) | 0.60 (0.14) |
| ASGL-$pca$ – 20% | 0.60 (0.07) | 0.59 (0.01) | 0.75 (0.10) | 1.00 (0.00) | 0.60 (0.17) |
| ASGL-$pca$ – 30% | 0.59 (0.07) | 0.59 (0.01) | 0.78 (0.10) | 1.00 (0.00) | 0.61 (0.14) |
| ASGL-$pca$ – 40% | 0.58 (0.07) | 0.59 (0.01) | 0.79 (0.10) | 1.00 (0.00) | 0.64 (0.14) |
| ASGL-$pca$ – 50% | 0.56 (0.07) | 0.58 (0.01) | 0.78 (0.10) | 1.00 (0.00) | 0.68 (0.13) |
| ASGL-$pca$ – 60% | 0.55 (0.08) | 0.58 (0.01) | 0.79 (0.10) | 1.00 (0.00) | 0.70 (0.14) |
| ASGL-$pca$ – 70% | 0.55 (0.07) | 0.58 (0.01) | 0.78 (0.11) | 1.00 (0.00) | 0.69 (0.17) |
| ASGL-$pca$ – 80% | 0.54 (0.07) | 0.58 (0.01) | 0.79 (0.10) | 1.00 (0.00) | 0.70 (0.16) |
| ASGL-$pca$ – 90% | 0.52 (0.07) | 0.58 (0.01) | 0.82 (0.11) | 1.00 (0.00) | 0.74 (0.17) |
| ASGL-$pca$ – 100% | 0.44 (0.05) | 0.57 (0.01) | 0.94 (0.07) | 1.00 (0.00) | 0.92 (0.10) |

variability, $\alpha_{pca,d}$ or $\alpha_{pls,d}$ respectively, priorly fixed by the researcher. Along this section, the effect of changes in the percentage of explained variability is studied. The simulation schemes are these of *Case 1* (625 variables sparsely distributed) and *Case 5* (100 variables sparsely distributed). Additionally, a t(3) distribution error is considered. Finally, two situations will be studied: variations on the percentage of variability affecting $pca_d$ technique and variations on the percentage of variability affecting $pls_d$ technique.

**Simulation 6: Influence of $\alpha_{pca,d}$** This simulation is centered on the effect of variations in the percentage of explained variability using PCA. Since PCA technique defines an orthogonal change of basis matrix, it is possible to recover all the variability from the original variables, and thus, different ASGL $pca_d$ models are solved ranging the percentage of explained variability from 10% to 100%.

Figure 2.15: Simulation 6. Sparse distribution of **625** variables. Considering a t(3) error. Analysis of $\alpha_{pca,d}$ influence. Box-plots showing the test error of the different models.

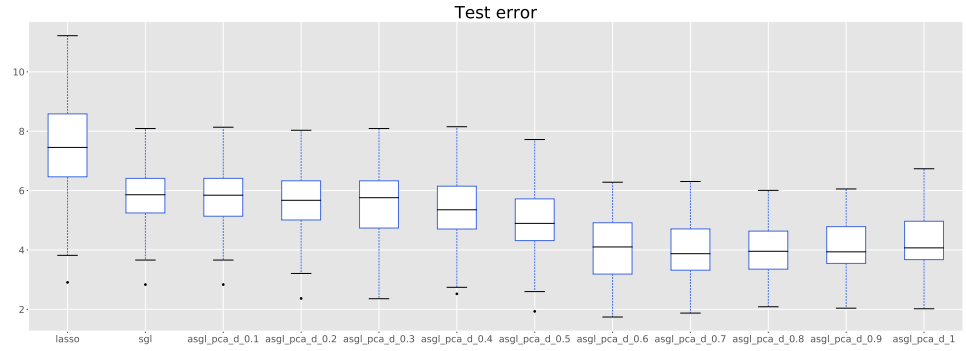



Figure 2.16: Simulation 6. Sparse distribution of **625** variables. Considering a t(3) error. Analysis of $\alpha_{pca,d}$ influence. Box-plots showing the correct selection rate of the different models.

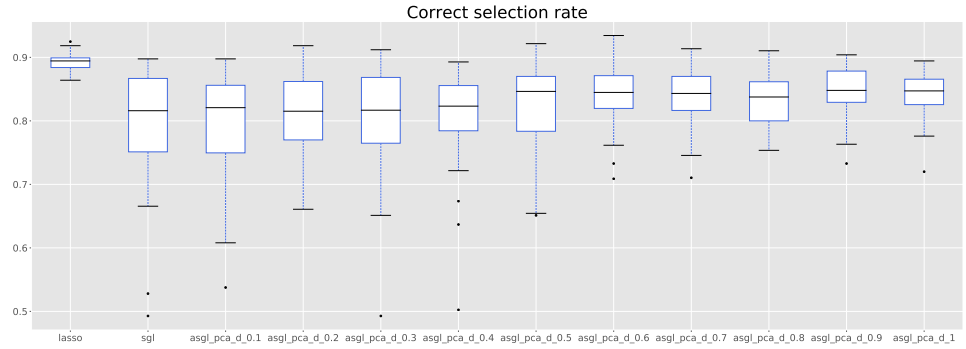



Figure 2.17: Simulation 6. Sparse distribution of **100** variables. Considering a t(3) error. Analysis of $\alpha_{pca,d}$ influence. Box-plots showing the test error of the different models.

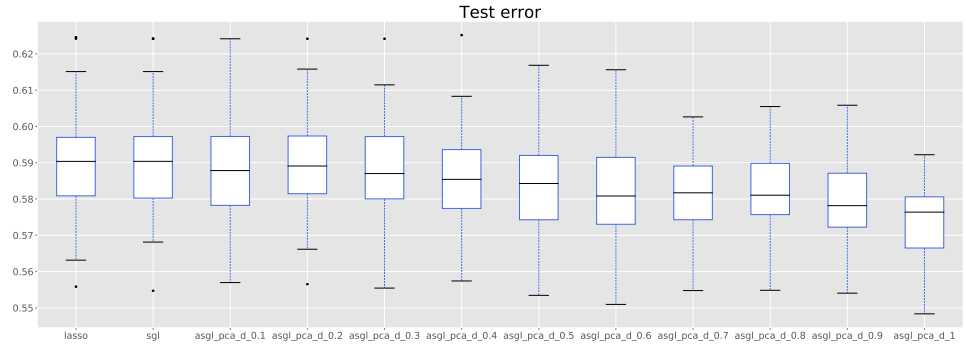
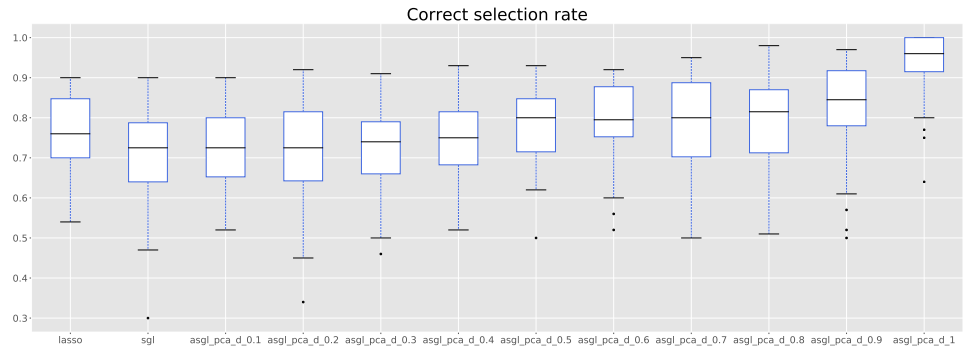


Figure 2.18: Simulation 6. Sparse distribution of 100 variables. Considering a t(3) error. Analysis of $\alpha_{pca,d}$ influence. Box-plots showing the correct selection rate of the different models.




The results obtained are shown in Table 2.8. In the low dimensional framework considering 100 variables it is possible to see how as the percentage of variability increases, all the metrics are improved achieving smaller prediction errors and better variable selection. A similar behavior is observed in the high dimensional framework for the explained variability ranging between 10% up to, approximately, 80%. However, when further increasing the percentage of explained variability up to 100%, the results get worse. Our guess is that in high dimensional frameworks, attaining a 100% of explained variability in PCA requires obtaining as many principal components as rows in the data matrix, producing overfitted solutions and adding noise to the predictions. Figures 2.15 and 2.16 show boxplots of the prediction error $E_t$ and the correct selection rate in the high dimensional framework, while Figures 2.17 and 2.18 show the same boxplots in the low dimensional framework. In these boxplots the behavior described above can be easily seen.

**Simulation 7: Influence of $\alpha_{pls,d}$** This simulation is focused on the effect of variations in the percentage of explained variability using PLS. PLS defines a nonnecesarily orthogonal change of basis matrix, and therefore, it is not possible to recover all the variability from the original variables. Actually, in the scheme considering 100 variables, PLS technique could recover at most 70% of the original variabiity, while in the simulation scheme considering 625 variables, PLS could recover at most 60%. For this reason, in the low dimensional framework different ASGL $pls_d$ models are solved ranging the percentage of explained variability from 10% to 70%, while in the high dimensional framework the variability ranges from 10% to 60%.

The results obtained in this simulation are shown in Table 2.9. In the low dimensional framework considering 100 variables it is possible to see how as the percentage of variability increases from 10% up to 30% results improve slightly in terms of prediction accuracy. Further increases up to 70% produce small improvements in the TNR, but overall, changes in the percentage of explained variability in PLS do not

Table 2.9: Simulation 7. Sparse distribution of 625 variables. Considering a t(3) error. Analysis of $\alpha_{pls,d}$ influence.

| | $\|\hat{\beta} - \beta\|$ | $E_t$ | CSR | TPR | TNR |
|---|---|---|---|---|---|
| 625 variables. Sparse distribution of variables. | | | | | |
| LASSO | 23.66 (4.97) | 7.99 (1.82) | 0.85 (0.04) | 0.76 (0.07) | 0.90 (0.01) |
| SGL | 18.63 (3.95) | 6.06 (1.35) | 0.84 (0.04) | 0.90 (0.04) | 0.79 (0.08) |
| ASGL-$pls$ – 10% | 13.88 (4.23) | 4.42 (1.30) | 0.84 (0.04) | 0.92 (0.03) | 0.84 (0.04) |
| ASGL-$pls$ – 20% | 14.19 (4.20) | 4.42 (1.30) | 0.84 (0.04) | 0.92 (0.03) | 0.83 (0.04) |
| ASGL-$pls$ – 30% | 14.19 (4.20) | 4.42 (1.30) | 0.84 (0.04) | 0.92 (0.03) | 0.83 (0.04) |
| ASGL-$pls$ – 40% | 14.19 (4.20) | 4.42 (1.30) | 0.84 (0.04) | 0.92 (0.03) | 0.83 (0.04) |
| ASGL-$pls$ – 50% | 14.19 (4.20) | 4.42 (1.30) | 0.84 (0.04) | 0.92 (0.03) | 0.83 (0.04) |
| ASGL-$pls$ – 60% | 14.19 (4.20) | 4.42 (1.30) | 0.84 (0.04) | 0.92 (0.03) | 0.83 (0.04) |
| 100 variables. Sparse distribution of variables. | | | | | |
| LASSO | 0.60 (0.07) | 0.60 (0.01) | 0.77 (0.09) | 1.00 (0.00) | 0.67 (0.13) |
| SGL | 0.60 (0.07) | 0.60 (0.01) | 0.73 (0.12) | 1.00 (0.00) | 0.90 (0.12) |
| ASGL-$pls$ – 10% | 0.50 (0.07) | 0.58 (0.01) | 0.87 (0.09) | 1.00 (0.00) | 0.92 (0.13) |
| ASGL-$pls$ – 20% | 0.46 (0.06) | 0.58 (0.01) | 0.93 (0.08) | 1.00 (0.00) | 0.93 (0.12) |
| ASGL-$pls$ – 30% | 0.45 (0.06) | 0.57 (0.01) | 0.94 (0.08) | 1.00 (0.00) | 0.93 (0.11) |
| ASGL-$pls$ – 40% | 0.45 (0.06) | 0.57 (0.01) | 0.95 (0.07) | 1.00 (0.00) | 0.93 (0.11) |
| ASGL-$pls$ – 50% | 0.45 (0.06) | 0.57 (0.01) | 0.95 (0.07) | 1.00 (0.00) | 0.93 (0.09) |
| ASGL-$pls$ – 60% | 0.45 (0.06) | 0.57 (0.01) | 0.96 (0.05) | 1.00 (0.00) | 0.95 (0.07) |
| ASGL-$pls$ – 70% | 0.45 (0.06) | 0.57 (0.01) | 0.96 (0.05) | 1.00 (0.00) | 0.95 (0.07) |

Figure 2.19: Simulation 7. Sparse distribution of 625 variables. Considering a t(3) error. Analysis of $\alpha_{pls,d}$ influence. Box-plots showing the test error of the different models.

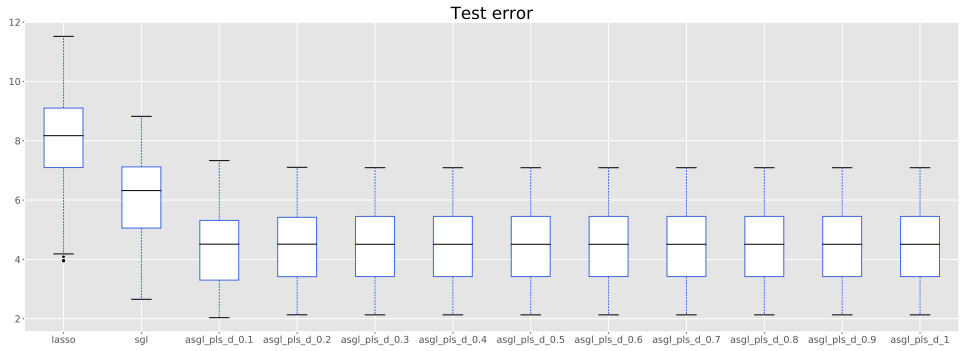


Figure 2.20: Simulation 7. Sparse distribution of **625** variables. Considering a t(3) error. Analysis of $\alpha_{pls,d}$ influence. Box-plots showing the correct selection rate of the different models.

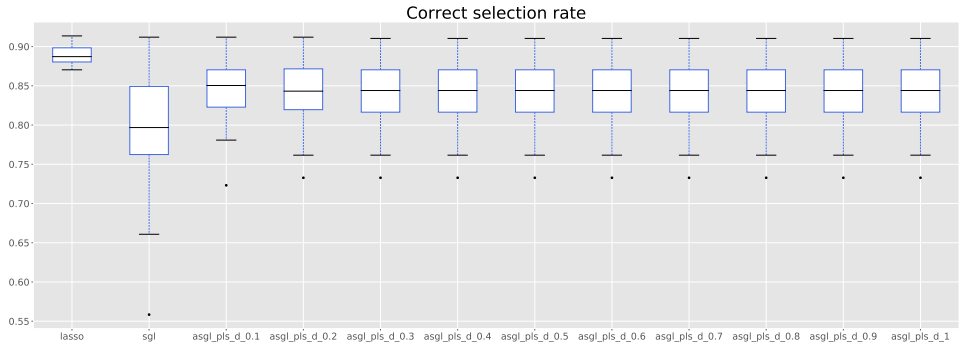



Figure 2.21: Simulation 7. Sparse distribution of **100** variables. Considering a t(3) error. Analysis of $\alpha_{pls,d}$ influence. Box-plots showing the test error of the different models.

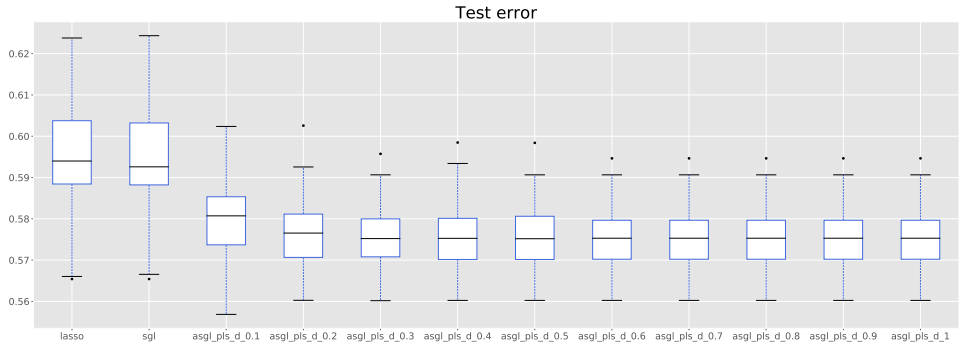
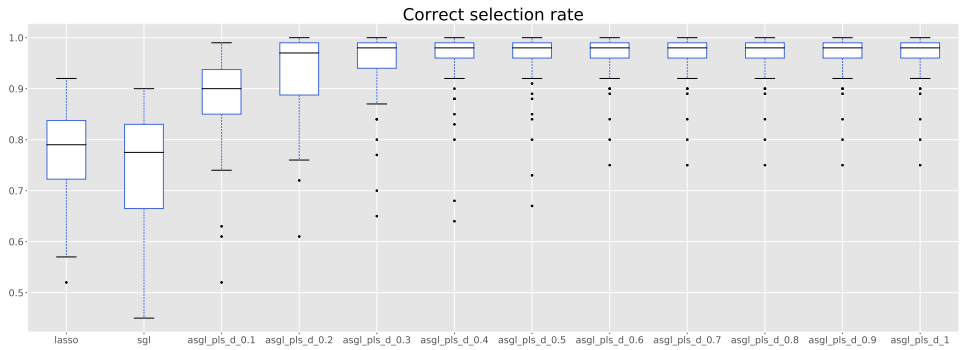



Figure 2.22: Simulation 7. Sparse distribution of **100** variables. Considering a t(3) error. Analysis of $\alpha_{pls,d}$ influence. Box-plots showing the correct selection rate of the different models.



affect heavily the performance of the estimator. This is probabily due to the way the PLS components are obtained, based on the maximization of the covariance between the response variable and the covariates. This means that the first PLS components already hold the information most related to the response variable, providing very good results. A similar behaviour is observed in the high dimensional fraework, where the prediction accuracy stabilizes while considering a 20% of explained variability. Figures 2.19 and 2.20 show boxplots of the prediction error $E_t$ and the correct selection rate in the high dimensional framework, while Figures 2.21 and 2.22 show the same boxplots in the low dimensional framework. In these boxplots the behavior described above can be easily seen.

## Acknowledgments

We appreciate the work of the referees that has contributed to substantially improve the scientific contributions of this work. In this research we have made use of Uranus, a supercomputer cluster located at University Carlos III of Madrid and funded jointly by EU-FEDER funds and by the Spanish Government via the National Projects No. UNC313-4E-2361, No. ENE2009-12213- C03-03, No. ENE2012-33219 and No. ENE2015-68265-P. This research was partially supported by research grants and Project ECO2015-66593-P from Ministerio de Economía, Industria y Competitividad, Project MTM2017-88708-P from Ministerio de Economía y Competitividad, FEDER funds and Project IJCI-2017-34038 from Agencia Estatal de Investigación, Ministerio de Ciencia, Innovación y Universidades.

# Bibliography

Soumyadeep Chatterjee, Snigdhanshu Banerjee, Arindam, and Auroop R. Ganguly. Sparse Group Lasso for Regression on Land Climate Variables. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 1–8. IEEE, 12 2011. ISBN 978-1-4673-0005-6. doi: 10.1109/ICDMW.2011.155.

A. P. Chiang, J. S. Beck, H.-J. Yen, M. K. Tayeh, T. E. Scheetz, R. E. Swiderski, D. Y. Nishimura, T. A. Braun, K.-Y. A. Kim, J. Huang, K. Elbedour, R. Carmi, D. C. Slusarski, T. L. Casavant, E. M. Stone, and V. C. Sheffield. Homozygosity mapping with SNP arrays identifies TRIM32, an E3 ubiquitin ligase, as a Bardet-Biedl syndrome gene (BBS11). *Proceedings of the National Academy of Sciences*, 103(16):6287–6292, 4 2006. doi: 10.1073/pnas.0600158103.

Hyonho Chun and Sündüz Keleş. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 72(1):3–25, 1 2010. ISSN 13697412. doi: 10.1111/j.1467-9868.2009.00723.x.

Gabriela Ciuperca. Adaptive fused LASSO in grouped quantile regression. *Journal of Statistical Theory and Practice*, 11(1):107–125, 1 2017. ISSN 15598616. doi: 10.1080/15598608.2016.1258601.

Gabriela Ciuperca. Adaptive group LASSO selection in quantile models. *Statistical Papers*, 60(1):173–197, 2 2019. ISSN 09325026. doi: 10.1007/s00362-016-0832-1.

Steven Diamond and Stephen Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *arXiv:1603.00943*, 3 2016.

Alexander Domahidi, Eric Chu, and Stephen Boyd. ECOS: An SOCP Solver for Embedded Systems. In *European Control Conference (ECC)*, 2013. ISBN 9783952417348. doi: 10.0/Linux-x86{_}64.

Jianqing Fan and Runze Li. Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties. *Journal of the American Statistical Association*, 96 (456):1348–1360, 2001. ISSN 0162-1459. doi: 10.2307/3085904.

Jianqing Fan and Heng Peng. Nonconcave penalized likelihood with a diverging number of parameters. *Annals of Statistics*, 32(3):928–961, 2004. ISSN 00905364. doi: 10.1214/009053604000000256.

J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *ArXiv:1001.0736*, pages 1–8, 2010. ISSN 15410420. doi: 10.1111/biom.12292. URL http://arxiv.org/abs/1001.0736.

Samiran Ghosh. On the grouped selection and model complexity of the adaptive elastic net. *Statistics and computing*, 21:451–462, 2011. doi: 10.1007/s11222-010-9181-4. URL https://link.springer.com/content/pdf/10.1007%2Fs11222-010-9181-4.pdf.

Jian Huang, Joel L. Horowitz, and Shuangge Ma. Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 36(2):587–613, 4 2008a. ISSN 00905364. doi: 10.1214/009053607000000875.

Jian Huang, Shuangge Ma, and Cun-Hui Zhang. Adaptive Lasso for Sparse High-dimensional Regression. *Statistica Sinica*, 1(374):1–28, 2008b.

Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics: Second Edition*. Wiley Series in Probability and Statistics. wiley, Hoboken, NJ, USA, 2 2009. ISBN 9780470434697. doi: 10.1002/9780470434697. URL http://doi.wiley.com/10.1002/9780470434697.

Yongdai Kim, Hosik Choi, and Hee Seok Oh. Smoothly clipped absolute deviation on high dimensions. *Journal of the American Statistical Association*, 103(484):1665–1673, 2008. ISSN 01621459. doi: 10.1198/016214508000001066.

Roger Koenker. *Quantile Regression*. Cambridge university Press, 2005. ISBN 0521338255.

Roger Koenker and Gilbert Bassett. Regression Quantiles. *Econometrica*, 46(1):33–50, 1 1978. ISSN 00129682. doi: 10.2307/1913643.

Juan C. Laria, M. Carmen Aguilera-Morillo, and Rosa E. Lillo. An iterative sparse-group lasso. *Journal of Computational and Graphical Statistics*, pages 1–21, 2 2019. doi: 10.1080/10618600.2019.1573687.

Youjuan Li and Ji Zhu. L1- -Norm Quantile Regression. *Journal of Computational and Graphical Statistics*, 17(1):1–23, 3 2008. doi: 10.1198/106186008X289155.

Po Ling Loh. Statistical consistency and asymptotic normality for high-dimensional robust m-estimators. *Annals of Statistics*, 45(2):866–896, 2017. ISSN 00905364. doi: 10.1214/16-AOS1471.

Yuval Nardi and Alessandro Rinaldo. On the asymptotic properties of the group lasso estimator for linear models. *Electronic Journal of Statistics*, 2(0):605–633, 2008. ISSN 19357524. doi: 10.1214/08-EJS200.

Benjamin Poignard. Asymptotic theory of the adaptive Sparse Group Lasso. *Annals of the Institute of Statistical Mathematics*, 72:297–328, 2018. ISSN 15729052. doi: 10.1007/s10463-018-0692-7.

T. E. Scheetz, K.-Y. A. Kim, R. E. Swiderski, A. R. Philp, T. A. Braun, K. L. Knudtson, A. M. Dorrance, G. F. DiBona, J. Huang, T. L. Casavant, V. C. Sheffield, and E. M. Stone. Regulation of gene expression in the mammalian eye and its relevance to eye disease. *Proceedings of the National Academy of Sciences*, 103(39):14429–14434, 9 2006. doi: 10.1073/pnas.0602562103.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 4 2013. ISSN 10618600. doi: 10.1080/10618600.2012.681250.

A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 10 2005. doi: 10.1073/pnas.0506580102.

Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. doi: 10.2307/2346178.

Lan Wang, Yichao Wu, and Runze Li. Quantile regression for analyzing heterogeneity in ultra-high dimension. *Journal of the American Statistical Association*, 107 (497):214–222, 2012. ISSN 01621459. doi: 10.1080/01621459.2012.656014.

John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S. Huang, and Shuicheng Yan. Sparse Representation for Computer Vision and Pattern Recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 6 2010. ISSN 0018-9219. doi: 10.1109/JPROC.2010.2044470.

Yichao Wu and Yufeng Liu. Variable selection in quantile regression. *Statistica Sinica*, 19(2):801–817, 2009.

Zakariya Yahya Algamal and Muhammad Hisyam Lee. A two-stage sparse logistic regression for optimal gene selection in high-dimensional microarray

data classification. *Advances in Data Analysis and Classification*, 13:753–771, 2019. doi: 10.1007/s11634-018-0334-1. URL https://doi.org/10.1007/s11634-018-0334-1.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 68 (1):49–67, 2006.

Weihua Zhao, Riquan Zhang, and Jicai Liu. Sparse group variable selection based on quantile hierarchical Lasso. *Journal of Applied Statistics*, 41(8):1658–1677, 8 2014. ISSN 0266-4763. doi: 10.1080/02664763.2014.888541.

Nengfeng Zhou and Ji Zhu. Group Variable Selection via a Hierarchical Lasso and Its Oracle Property. *Statistics and Its Interface*, 3:557–574, 2010. URL http://arxiv.org/abs/1006.2871.

Hui Zou. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 12 2006. ISSN 0162-1459. doi: 10.1198/016214506000000735.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. doi: 10.1198/106186006X113430.

# CHAPTER 3

## A quantile based dimension reduction technique

In *Working paper. Statistics and Econometrics 21-06.* Universidad Carlos III de Madrid. Departamento de Estadística.

Álvaro Méndez Civieta[1,2], M. Carmen Aguilera-Morillo[2,3] and Rosa E. Lillo[1,2].

1. Department of Statistics, Universidad Carlos III de Madrid.
2. uc3m-Santander Big Data Institute.
3. Department of Applied Statistics and Operational Research, and Quality, Universitat Politècnica de València

## Abstract

Partial least squares (PLS) is a dimensionality reduction technique used as an alternative to ordinary least squares (OLS) in situations where the data is colinear or high dimensional. Both PLS and OLS provide mean based estimates, which are extremely sensitive to the presence of outliers or heavy tailed distributions. In contrast, quantile regression is an alternative to OLS that computes robust quantile based estimates. In this work, the multivariate PLS is extended to the quantile regression framework, obtaining a theoretical formulation of the problem and a robust dimensionality reduction technique that we call fast partial quantile regression (fPQR), that provides quantile based estimates. An efficient implementation of fPQR is also derived, and its performance is studied through simulation experiments and the chemometrics well known biscuit dough dataset, a real high dimensional example.

**keywords:** Partial-least-squares; Quantile-regression; Dimension-reduction; Outliers; Robust.

## 3.1 Introduction

Partial least squares (PLS) [Wold, 1973], [Wold et al., 2001] is a dimensionality reduction technique commonly applied to two data blocks (predictors and responses) that works by projecting the available data into a latent structure. The key idea behind PLS is that it can summarize the predictors into a small set of uncorrelated latent variables that have maximal covariance with the responses. PLS has proven to be a versatile alternative to ordinary least squares (OLS), obtaining parsimonious models even when dealing with ill-posed multicollinear problems, commonly found in different areas of scientific research such as chemometrics, social science or medicine. See for example [Nguyen and Rocke, 2002], where it is used in a tumor classification problem. In recent years PLS has also received attention when dealing with the increasingly common problem of high dimensional data, in which the number of observations is small and the number of variables is very large. In this regard, Boulesteix and Strimmer [2006] successfully applied PLS to a genomic dataset. Partial least squares is based on the cross-covariance matrix between predictors and response, and on least squares models. Least squares models are known to behave nicely when the errors are normally distributed, but there is no guarantee that the normality will be satisfied in many experimental data problems, where heavy tailed distributions, and even outliers are expected to be found. This makes PLS extremely sensitive to the presence of outliers or non normal data. The solution to this problem has traditionally been centered in robustifying the least squares estimator in which PLS is based, see for example [Serneels et al., 2005] where they make use of a robust M-regression estimator, or [Acitas et al., 2020], where a partial robust adaptive modified maximum likelihood estimator is proposed, among others.

Quantile regression [Koenker and Bassett, 1978] is an important statistical methodology that allows to describe the conditional quantiles of a response given a set of covariates. Fitting the data at a set of quantiles provides a more comprehensive picture of the response distribution than does the mean, and as opposed to least squares, quantile regression is resistant to outliers, and can deal with heavy tailed distributions and heteroscedasticity, the situation when variances depend on some covariates. Specifically, when the center of the distribution is of interest, the least absolute deviation (LAD), also called median regression, a particular case of quantile regression, provides more robust estimators than least squares regression. In recent years many papers have been published extending quantile regression to the high dimensional framework by performing variable selection, see for example Wu and Liu [2009] where an adaptive lasso for quantile regression is introduced, or [Mendez-Civieta et al., 2021], where an adaptive sparse group lasso for quantile regression is proposed. However, to the best of our knowledge there is very little work on quantile based dimension reduction techniques. A well known PLS implementation is given by the NIPALS algorithm [Wold, 1973]. Dodge and Whittaker [2009]

extended the NIPALS algorithm for univariate response problems to the quantile regression framework. They proposed a quantile covariance metric based on the quantile regression slope and used this metric to modify the univariate NIPALS, a modification that they called partial quantile regression (PQR). The work from Dodge and Whittaker [2009] lays the foundation for an extension of PLS to the quantile regression framework, however we find some shortcomings in the development of the methodology and the algorithmic implementation that should be addressed. First, it has no background on what is the optimization problem that their PQR algorithm is solving. Second, it is centered in univariate response problems, providing no solution for multivariate response problems commonly found in fields such as chemometrics. Third, the computation time of their quantile coviariance, key in the algorithmic implementation, grows linearly with the number of variables, making solving high dimensional problems computationally expensive. The main contribution of our work is centered in addressing these problems. We define the optimization problem that the fPQR algorithm solves and study different quantile covariance alternatives [Li et al., 2015], [Choi and Shin, 2018]. We provide an efficient implementation of fPQR, significantly reducing the computation time when compared with that of [Dodge and Whittaker, 2009] while achieving more accurate predictions. We also provide an implementation suitable for multivariate response settings. The result is a methodology that parallels the nice properties of PLS: it is a dimension reduction technique that obtains uncorrelated scores maximizing the quantile covariance between predictors and responses. But additionally, it is also a robust, quantile linked methodology suitable for dealing with outliers, heteroscedastic or heavy tailed datasets. The median estimator of the fPQR algorithm is a robust alternative to PLS, while other quantile levels can provide additional information on the tails of the responses.

The rest of the paper is organized as follows. In Section 3.2 a brief introduction of the PLS algorithm for multivariate response is provided. Section 3.3 introduces the fPQR algorithm and studies different options for a quantile covariance metric. Section 3.4 tests the performance of the proposed fPQR algorithm in three synthetic dataset frameworks studying the quality of the estimated $\beta$ coefficients and the prediction error. In Section 3.5, the proposed algorithm is used in a real high dimensional data example. Some computational aspects are briefly commented in Section 3.6, and the conclusions are provided in Section 3.7.

## 3.2 The PLS model for multivariate response

Let $X \in \mathbb{R}^{n \times m}$ and $Y \in \mathbb{R}^{n \times l}$ be two data matrices, samples drawn from some unknown population following the linear model,

$$\boldsymbol{y}_i = \boldsymbol{x}_i B + \boldsymbol{\varepsilon}_i, \quad i = 1, \ldots, n, \tag{3.1}$$

where $\boldsymbol{y}_i \equiv (y_{i1}, \ldots, y_{il})$ is the vector containing the response variables for the $i$-th observation, $\boldsymbol{x}_i \equiv (x_{i1}, \ldots, x_{im})$ contains the predictive variables, $B \in \mathbb{R}^{m \times l}$ is the matrix containing the coefficients from the linear relations, and $\boldsymbol{\varepsilon_i} \equiv (\varepsilon_{i1}, \ldots, \varepsilon_{il})$ is the error term. Without loss of generality, consider that both $X$ and $Y$ are mean centered. The PLS regression methodology works by assuming the existence of a latent structure,

$$X = TP^t + E; \quad Y = TQ^t + F, \tag{3.2}$$

where $T \in \mathbb{R}^{n \times h}$ is the scores matrix formed by $h$ (usually being $h \ll m$) linear combinations of the original variables, $P \in \mathbb{R}^{m \times h}$ and $Q \in \mathbb{R}^{l \times h}$ are loadings matrices and $E \in \mathbb{R}^{n \times m}$ and $F \in \mathbb{R}^{n \times l}$ are random error matrices. The aim of PLS regression is precisely to regress the response matrix $Y$ onto the $h$ latent variables, stored in the scores matrix $T$, defining this way a low-dimensional regression model,

$$\boldsymbol{y}_i = \boldsymbol{t_i}\Gamma + \boldsymbol{\varepsilon}_i^*, \quad i = 1, \ldots, n, \tag{3.3}$$

where $\Gamma$ is the matrix of regression coefficients. PLS is an iterative algorithm in which the scores in $T$ are obtained sequantially. There are multiple definitions of the PLS algorithm available in the literature, being NIPALS [Wold, 1973] and SIMPLS [de Jong, 1993] the most frequently used ones. Here a version of NIPALS that will be useful in the implementation of the fPQR algorithm is considered:

1. Define $X_0 = X$ and $Y_0 = Y$.
2. Compute $S_1 = X_0^t Y_0$ the sample covariance matrix.
3. Obtain the eigen decomposition of $S_1 S_1^t$ and take $\boldsymbol{w}_1$ as the eigenvector associated to the largest eigenvalue.
4. Calculate the $X$ score vector as $\boldsymbol{t}_1 = X_0 \boldsymbol{w}_1$.
5. Calculate the $X$ loading vector as $\boldsymbol{p}_1 = \dfrac{X_0^t \boldsymbol{t}_1}{\boldsymbol{t}_1^t \boldsymbol{t}_1}$.
6. Calculate the $Y$ loading vector as $\boldsymbol{q}_1 = \dfrac{Y_0^t \boldsymbol{t}_1}{\boldsymbol{t}_1^t \boldsymbol{t}_1}$.
7. Deflat the matrix $X_0$ from the information already explained by scores $\boldsymbol{t}_1$ and obtain $X_1 = X_0 - \boldsymbol{t}_1 \boldsymbol{p}_1^t$.
8. Deflat the matrix $Y_1$ from the information already explained by scores $\boldsymbol{t}_1$ and obtain $Y_1 = Y_0 - \boldsymbol{t}_1 \boldsymbol{q}_1^t$.

Iterate through steps 2-8 until all $h$ components are computed. Observe that the deflation process stated in step 7 ensures that the score matrix $T$ will be orthogonal.

Once all the required components have been computed, the parameter estimates $\hat{\Gamma}$ from equation (3.3) are obtained solving the low dimensional least squares model,

$$\hat{\Gamma} = \arg\min_{\Gamma} \left\{ \|Y - T\Gamma\|^2 \right\}. \tag{3.4}$$

Finally, one can project the estimate $\hat{\Gamma}$ back into the original sub-space spawned by $X$ and obtain,

$$\hat{B} = W(P^t W)^{-1}\hat{\Gamma}. \tag{3.5}$$

PLS is essentially a covariance maximization problem where, at each iteration $a+1$, the objective function being solved is defined as,

$$\boldsymbol{w}_{a+1} = \arg\max_{\boldsymbol{w}, \|\boldsymbol{w}\|=1} \left\{ \operatorname{cov}(X_a \boldsymbol{w}, Y_a) \operatorname{cov}(X_a \boldsymbol{w}, Y_a)^t \right\}, \tag{3.6}$$

where $X_0 = X$ and $Y_0 = Y$, and the solution is the eigenvector associated to the largest eigenvalue $\lambda_1$,

$$S_a S_a^t \boldsymbol{w}_a = \lambda_1 \boldsymbol{w}_a. \tag{3.7}$$

Posing PLS as a covariance optimization problem opens the door to the possibility of using alternative covariance definitions. Traditionally, robust versions have been considered in order to obtain robustified PLS algorithms, see for example [Hubert and Branden, 2003]. In this work we are interested in defining not only a robust PLS estimator, but an estimator linked to the quantiles of the response matrix, giving the possibility to study the tails of the response matrix and not just the central behavior. As a solution to this question, a robust quantile based dimension reduction technique that we call fast partial quantile regression (fPQR) is introduced in the next section.

## 3.3 Fast partial quantile regression

There are two key steps in the definition of the fPQR methodology. First, the usage of a quantile covariance metric linked to the quantiles, instead of the traditional covariance, that is linked to the mean. As it will be discussed in Section 3.4, the metric that we consider to be the best alternative was proposed by Li et al. [2015], although other alternatives [Dodge and Whittaker, 2009], [Choi and Shin, 2018] will also be studied along Sections 3.3.3 and 3.4. Second, the estimation of the $\Gamma$ coefficients defined in equation (3.3). In the PLS algorithm, these coefficients are estimated using ordinary least squares, but in the fPQR algorithm a quantile regression model is used instead, ensuring that the $\hat{\Gamma}$ estimates remain linked to the quantiles of the response matrix $Y$.

### 3.3.1 A quantile covariance

In a very interesting work, Li et al. [2015] extended the usage of autoregressive models to the quantile framework by defining a novel measure suitable for examining the linear relationships between any two random variables for a given quantile $\tau \in (0, 1)$, a measure that they called quantile correlation. Given two random variables $Z_1$ and $Z_2$, take $Q_{\tau,Z_2}$ as the $\tau$-th quantile of $Z_2$ and $Q_{\tau,Z_2}(Z_1)$ as the $\tau$-th quantile of $Z_2$ conditional to $Z_1$. Then it is possible to demonstrate that $Q_{\tau,Z_2}(Z_1)$ is independent of $Z_1$ if and only if the random variables $I(Z_2 - Q_{\tau,Z_2} > 0)$ and $Z_1$ are independent, where $I(\cdot)$ is the indicator function. This fact motivated the definition of the quantile covariance proposed in their work as,

$$\begin{aligned} \text{qcov}_\tau\{Z_1, Z_2\} &= \text{cov}\left\{I\left(Z_2 - Q_{\tau,Z_2} > 0\right), Z_1\right\} \\ &= E\left\{\psi_\tau\left(Z_2 - Q_{\tau,Z_2}\right)(Z_1 - EZ_1)\right\}, \end{aligned} \tag{3.8}$$

where $\psi_\tau(w) = \tau - I(w < 0)$. Being based on a traditional covariance makes this quantile covariance easy and fast to compute. Additionally, although this definition is proposed for random variables, it can be extended to random vectors, making it possible to adapt to the data matrices found in multidimensional problems. Observe however that, opposed to the traditional covariance, this quantile covariance does not enjoy the symmetry property, that is, $\text{qcov}_\tau(Z_1, Z_2) \neq \text{qcov}_\tau(Z_2, Z_1)$. A complete definition of this metric can be found in [Li et al., 2015] where they study a nice relation between this metric and the slope from a quantile regression model, and also the asymptotic properties of the estimator.

### 3.3.2 The fPQR algorithm

The objective function that the fPQR algorithm solves is obtained by adapting the objective function from a PLS model as it was defined in equation (3.6) using the quantile covariance introduced in Section 3.3.1,

$$\begin{aligned} \boldsymbol{w}_{a+1} &= \underset{\boldsymbol{w}, \|\boldsymbol{w}\|=1}{\arg\max} \left\{\text{qcov}_\tau(X_a\boldsymbol{w}, Y_a)^t \, \text{qcov}_\tau(X_a\boldsymbol{w}, Y_a)\right\} \\ &= \underset{\boldsymbol{w}, \|\boldsymbol{w}\|=1}{\arg\max} \left\{\boldsymbol{w}^t X_a^t \psi_\tau(Y_a - Q_{\tau,Y_a})\psi_\tau(Y_a - Q_{\tau,Y_a})^t X_a\boldsymbol{w}\right\}, \end{aligned} \tag{3.9}$$

where $\psi_\tau(w) = \tau - I(w < 0)$. The solution to this equation is the eigenvector associated to the largest eigenvalue $\lambda_1$,

$$X_a^t \psi_\tau(Y_a - Q_{\tau,Y_a})\psi_\tau(Y_a - Q_{\tau,Y_a})^t X_a \boldsymbol{w_a} = \lambda_1 \boldsymbol{w_a}. \tag{3.10}$$

Based on this idea, the main steps of the fPQR algorithm are defined below,

1. Take $\tau \in (0, 1)$ the quantile level of interest.

2. Define $X_0 = X$ and $Y_0 = Y$.

3. Compute $S_{1,\tau} = \text{qcov}_\tau(X_0, Y_0)$ the sample quantile covariance matrix.

4. Obtain the eigen decomposition of $S_{1,\tau}S_{1,\tau}^t$ and take $\boldsymbol{w}_1$ as the eigenvector associated to the largest eigenvalue.

5. Calculate the $X$ score vector as $\boldsymbol{t}_1 = X_0\boldsymbol{w}_1$.

6. Calculate the $X$ loading vector as $\boldsymbol{p}_1 = \dfrac{X_0^t\boldsymbol{t}_1}{\boldsymbol{t}_1^t\boldsymbol{t}_1}$.

7. Calculate the $Y$ loading vector as $\boldsymbol{q}_1 = \dfrac{Y_0^t\boldsymbol{t}_1}{\boldsymbol{t}_1^t\boldsymbol{t}_1}$.

8. Deflat the matrix $X_1$ from the information already explained by scores $\boldsymbol{t}_1$ and obtain $X_1 = X_0 - \boldsymbol{t}_1\boldsymbol{p}_1^t$.

9. Deflat the matrix $Y_1$ from the information already explained by scores $\boldsymbol{t}_1$ and obtain $Y_1 = Y_0 - \boldsymbol{t}_1\boldsymbol{q}_1^t$.

Iterate through steps 2-8 until all $h$ components are computed. In order to obtain the parameter estimates $\hat{\Gamma}$ in the PLS algorithm, a least squares model was solved following equation (3.4), but in the fPQR algorithm this is substituted by a quantile regression model solving,

$$\tilde{\Gamma} = \arg\min_{\beta}\left\{\frac{1}{n}\sum_{i=1}^{n}\rho_\tau(\mathbf{y}_i - \boldsymbol{t}_i^t\Gamma)\right\}, \tag{3.11}$$

where $\rho_\tau(u) = u(\tau - I(u < 0))$ is the quantile regression loss check function. Using a quantile regression model here ensures that the $\hat{\Gamma}$ estimates remain linked to the quantile of the response matrix $Y$. Finally, one can project $\hat{\Gamma}$ back into the original sub-space spawned by $X$ as it was done in the PLS models in equation (3.5). The fPQR is an algorithm that shares many of the benefits of PLS:

- It is a dimension reduction technique suitable for multicollinear or high dimensional data;
- The new scores obtained by the algorithm are orthogonal;
- It maximizes the quantile covariance between predictor and response.

But it also has some additional properties:

- It is a robust methodology, suitable for dealing with outliers or heteroscedastic data;
- It can provide an estimation of the central behavior of the response conditional to the predictors, but additionally can provide an estimation of any other quantile of the response, conditional to the predictors, obtaining a complete view of the distribution of the response.

### 3.3.3 Other quantile covariance metrics

In Section 3.3.2, the fPQR algorithm was defined as an optimization problem where a quantile covariance metric is maximized. Although the metric proposed by Li et al. [2015] was used in the definition of the algorithm, it is possible to consider alternative versions of fPQR based on other quantile covariance metrics. Along this section, two other candidates, defined by [Dodge and Whittaker, 2009] and [Choi and Shin, 2018] are considered, showing their definition and some properties related to the fPQR performance.

#### A quantile covariance from Dodge and Whittaker [2009]

Take two random variables $Z_1$ and $Z_2$ following the linear model,

$$Z_2 = Z_1\beta + \varepsilon. \tag{3.12}$$

The analytical solution of the ordinary least squares estimator for model (3.12) is,

$$\hat{\beta} = \text{var}(Z_1)^{-1}\,\text{cov}(Z_1, Z_2). \tag{3.13}$$

Dodge and Whittaker [2009] take advantage of this fact and define a quantile covariance in terms of the quantile regression estimator, mimicking the relation between the OLS estimator and the traditional covariance displayed in equation (3.13). Consider the quantile regression estimator,

$$\tilde{\beta} = \arg\min_{\beta} \left\{ \text{E}\,\rho_\tau(Z_2 - \beta Z_1) \right\}, \tag{3.14}$$

where $\rho_\tau(u) = u(\tau - I(u < 0))$ is the quantile regression loss check function. Then the quantile covariance proposed by Dodge and Whittaker [2009] is obtained as,

$$\text{qcov}^*_\tau(Z_1, Z_2) = \text{var}(Z_1)\tilde{\beta}, \tag{3.15}$$

where $\tilde{\beta}$ is the quantile regression estimator defined in equation (3.14). Here the superscript "$*$" differentiates this quantile covariance from the one defined in Section 3.3.1. There are some remarks worth mentioning:

- The extension of this quantile covariance to a multidimensional setting is not as straightforward as in the traditional covariance or in the quantile covariance proposed by Li et al. [2015]. This means that given a random vector $U \equiv (U_1, \ldots, U_m)$,

$$\text{qcov}^*_\tau(U, Z_2) \neq (\text{qcov}^*_\tau(U_1, Z_2), \ldots, \text{qcov}^*_\tau(U_m, Z_2)). \tag{3.16}$$

  This implies that, in order to ensure that the quantile covariance (in the sense of Dodge and Whittaker [2009]) between two random variables remains the

same regardless of the computation affecting a random vector or not, it must be computed univariatedly. This way, the computation of the quantile covariance between $U$ and $Z_2$ requires to solve $m$ univariate quantile regression models, where $m$ is the dimension of $U$, greatly affecting the computation time as the number of variables increase;

- As happened with the quantile covariance defined by Li et al. [2015], this quantile covariance is not symmetric. This means that $\text{qcov}^*_\tau(Z_1, Z_2) \neq \text{qcov}^*_\tau(Z_2, Z_1)$

Additionally to the quantile covariance described above, the key contribution of Dodge and Whittaker [2009] was the adaptation of the univariate NIPALS algorithm to the quantile regression framework. The main differences between their proposal (PQR) and the work developed here (fPQR) are listed below:

- In the work developed here, the optimization problem that the fPQR algorithm solves is clearly defined, and based on this definition, the algorithm is proposed. Opposed to this, Dodge and Whittaker [2009] simply defined the algorithm as a modification of the univariate PLS NIPALS, without studying the optimization problem;

- The fPQR algorithm allows $Y$ to be a multivariate response matrix while the PQR algorithm is limited to univariate responses;

- As it will be seen in Section 3.4, the covariance considered in the fPQR algorithm allows the algorithm to run significantly faster than the PQR algorithm.

**A quantile covariance from Choi and Shin [2018]**

Given two random variables $Z_1$ and $Z_2$, the Pearson correlation between the two variables can be seen as the geometric mean of two OLS slopes, $\beta_{2.1}$ of $Z_1$ on $Z_2$ and $\beta_{1.2}$ of $Z_2$ on $Z_1$,

$$\text{cor}(Z_1, Z_2) = \text{sign}(\beta_{2.1}) \sqrt{\beta_{2.1}\beta_{1.2}}. \tag{3.17}$$

Based on this idea, Choi and Shin [2018] proposed a quantile correlation coefficient defined as the geometric mean of two quantile regression slopes,

$$\text{qcor}^{**}_\tau(Z_1, Z_2) = \text{sign}(\beta_{2.1}(\tau)) \sqrt{\beta_{2.1}(\tau)\beta_{1.2}(\tau)}, \tag{3.18}$$

where the superscript "$**$" is used to differentiate this metric from the ones from [Li et al., 2015] and [Dodge and Whittaker, 2009]. A full review of the properties of this metric can be found in the original paper [Choi and Shin, 2018] but there are some remarks that are worth mentioning:

- As it happened with the quantile covariance defined by Dodge and Whittaker [2009], given a random vector $U \equiv (U_1, \ldots, U_m)$,

$$\text{qcor}_\tau^{**}(U, Z_2) \neq (\text{qcor}_\tau^{**}(U_1, Z_2), \ldots, \text{qcor}_\tau^{**}(U_m, Z_2)). \tag{3.19}$$

  This implies that in order to ensure consistency of the results when dealing with random vectors, this metric must also be computed univariatedly. The computation of $\text{qcor}_\tau^{**}(U, Z_2)$ requires thus to solve $2m$ univariate quantile regression models, where $m$ is the dimension of $U$, significantly affecting the computation time;

- Opposed to the other quantile metrics under study, this is the only metric that is symmetric, meaning that $\text{qcor}_\tau^{**}(Z_1, Z_2) = \text{qcor}_\tau^{**}(Z_2, Z_1)$.

Observe that the fPQR algorithm requires a quantile covariance, and not a quantile correlation. Although not defined in the original paper, it is possible to obtain an estimation of a quantile covariance based on equation (3.18). Observe that,

$$\begin{aligned} \text{qcor}_\tau^{**}(Z_1, Z_2) &= \text{sign}\left(\beta_{2,1}(\tau)\right) \sqrt{\beta_{2,1}(\tau)\beta_{1,1}(\tau)} \\ &= \text{sign}\left(\beta_{2,1}(\tau)\right) \sqrt{\frac{\text{qcov}_\tau^{*}(Z_1, Z_2)\, \text{qcov}_\tau^{*}(Z_2, Z_1)}{\text{var}(Z_1)\, \text{var}(Z_2)}}, \end{aligned} \tag{3.20}$$

where $\text{qcov}^{*}(\cdot, \cdot)$ refers to the quantile covariance introduced in Section 3.3.3. This way, a symmetric quantile covariance can be defined as,

$$\text{qcov}_\tau^{**}(Z_1, Z_2) = \text{sign}(\beta_{2,1}(\tau)) \sqrt{\text{qcov}_\tau^{*}(Z_1, Z_2)\, \text{qcov}_\tau^{*}(Z_2, Z_1)}. \tag{3.21}$$

## 3.4 Numerical simulation

This section shows the performance of the proposed fPQR methodology under different synthetic datasets. The three quantile covariances under study, proposed by [Li et al., 2015], [Dodge and Whittaker, 2009] and [Choi and Shin, 2018] are compared here. Additionally, the algorithm is compared against PLS, taken as a benchmark model, and the partial robust adaptive modified maximum likelihood estimator (PRAMML), proposed by Acitas et al. [2020], which is a robust PLS alternative for univariate response models. In order to compare the quantile estimation provided by fPQR with the mean estimations from PLS and PRAMML, the quantile level of the fPQR is fixed at $\tau = 0.5$ (the median estimation). For each dataset $\mathbb{D}$, a partition into two disjoint subsets, $\mathbb{D}_{train}$ and $\mathbb{D}_{test}$ is considered. $\mathbb{D}_{train}$ is used for training the models, this is, solving the model equations. $\mathbb{D}_{test}$ is used for testing the models prediction accuracy. The following metrics are computed, where "#" denotes the cardinal of a set:

- $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2$: the euclidean distance between the estimated coefficients and the true coefficients;
- $\frac{1}{\#\mathbb{D}_{test}} \sum(\hat{y}_i - y_i)^2$: the mean squared error between the estimated response and the true response;
- The execution time of each algorithm measured in seconds.

*Remark.* These simulations compare the results of the fPQR algorithm with the results from PLS and PRAMML. For this reason, the quantile level is fixed at $\tau = 0.5$ and the metric considered is the mean squared error. However, when dealing with other quantile levels, the mean squared error is not a suitable metric, as it does not take into account the quantile being computed. In such scenarios the following quantile error metric can be used instead,

$$E_\tau = \frac{1}{\#\mathbb{D}_{test}} \sum_{(y_i, \boldsymbol{x_i}) \in \mathbb{D}_{test}} \rho_\tau(y_i - \boldsymbol{x}_i^t \hat{\boldsymbol{\beta}}). \tag{3.22}$$

### 3.4.1 Simulation 1

The following simulation scheme is an adaptation taken from Mendez-Civieta et al. [2021]. The idea behind this scheme is to simulate the behavior found in the increasingly common problem of sparse high dimensional data, where the number of variables is very large, and not all the variables affect the response, being some of them just noise. This problem can be found in many different areas of scientific research such as genetics [Boulesteix and Strimmer, 2006] or climate data [Chatterjee et al., 2011], and an interesting solution is the usage of dimension reduction techniques like PLS or the proposed fPQR algorithm. Take the model,

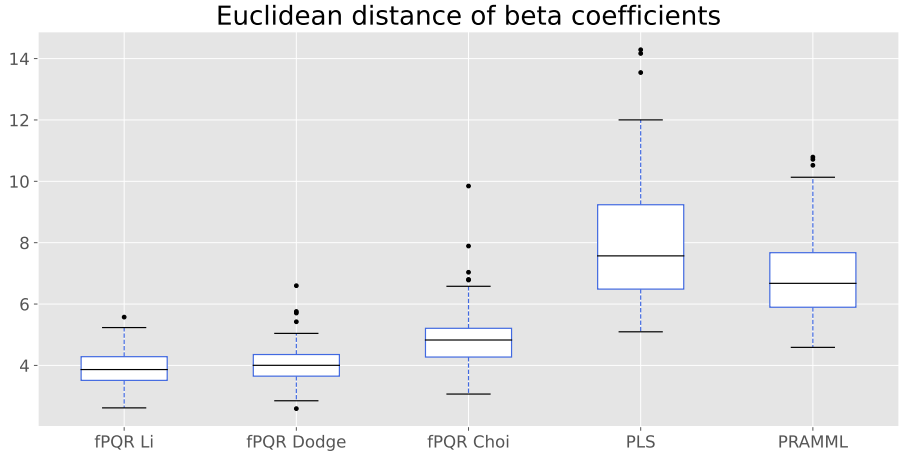
$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{3.23}$$

where the predictors matrix $X$ is generated from a standard normal distribution and the error term is generated following a chi squared distribution with 3 degrees of freedom, a distribution known to be heavy tailed and non symmetric. This will favor the usage of robust estimators. Since we are interested in the high dimensional framework, a sample size of $n = 100$ training observations and $m = 100$ predictive variables is considered. Out of the 100 predictive variables, 30 are generated from a standard uniform distribution and the remaining 70 have value 0, meaning that these 70 variables do not affect the response variable and are simply noise in the model. Although in real datasets the number of components in the model should be found based on some sort of cross-validation process, in this simulation it is fixed, taken equal to the number of significant variables, $h = 30$. Additionally, a sample of 500 observations is generated as test set. Observe that this fact does not affect

Table 3.1: Simulation 1. Sparse high dimensional framework considering a $\chi^2(3)$ error.

| | $\|\hat{\beta} - \beta\|$ | $\frac{1}{\#\mathbb{D}_{test}}\|\hat{y} - y\|_2^2$ | Execution time |
|---|---|---|---|
| fPQR Li | 3.88 (0.58) | 21.59 (5.13) | 0.038 (0.01) |
| fPQR Dodge | 4.05 (0.62) | 23.02 (5.98) | 38.65 (1.649) |
| fPQR Choi | 4.95 (0.94) | 31.48 (11.40) | 76.78 (2.716) |
| PLS | 8.03 (2.03) | 75.42 (37.21) | 0.004 (0.001) |
| PRAMML | 6.64 (1.37) | 52.11 (20.66) | 0.358 (0.047) |

Figure 3.1: Simulation 1. Mean squared error of $\beta$ coefficients.




the consideration of the simulation being high dimensional, as the algorithms are trained with a number of observations equal to the number of variables. This data generation process is repeated 100 times, and the results are summarized in terms of the mean value and standard deviation value (shown in parenthesis) of each metric computed.

Results from this simulation scheme are displayed in Table 3.1 and Figures 3.1, 3.2 and 3.3. In terms of the euclidean distance of the $\beta$ coefficients, the best results are obtained by the fPQR Li estimator, followed by the other quantile based alternatives, while PLS obtains the worst results, as expected since the normality assumptions are not met. Observe also that the standard deviation of this metric is smallest in the fPQR Li, indicating more stable results. In terms of prediction accuracy, the best results are obtained also by the fPQR Li algorithm, closely followed by the fPQR Dodge and achieving the smallest standard deviation values again. Finally, regarding the execution time the fastest algorithm was PLS and the second fastest was fPQR Li, while PRAMML took on average 10 times longer than fPQR Li. One can also see the large execution times using fPQR Dodge or fPQR Choi alternatives. This is due to the way these covariances are computed, requiring to solve, at each iteration of the algorithm, $m = 100$ univariate quantile regression models in the case of Dodge metric, and $2m = 200$ models in the case of Choi metric,

Figure 3.2: Simulation 1. Mean squared error of the response variable $y$.

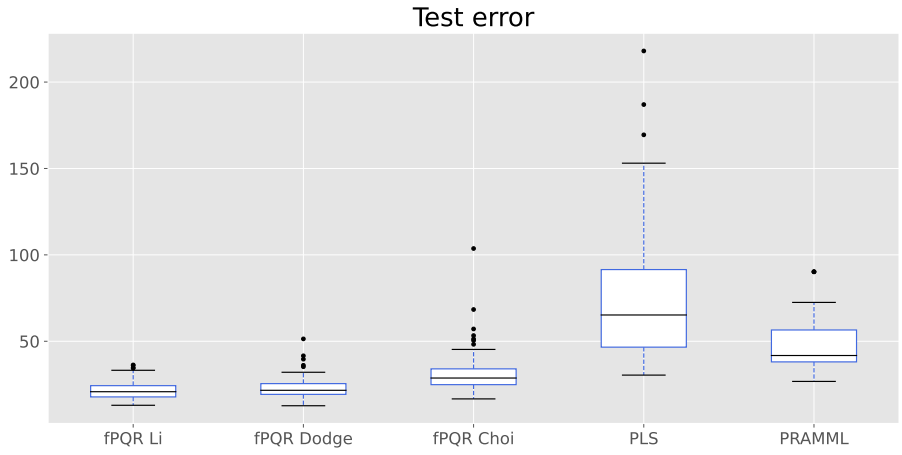



Figure 3.3: Simulation 1. Execution time measured in seconds.

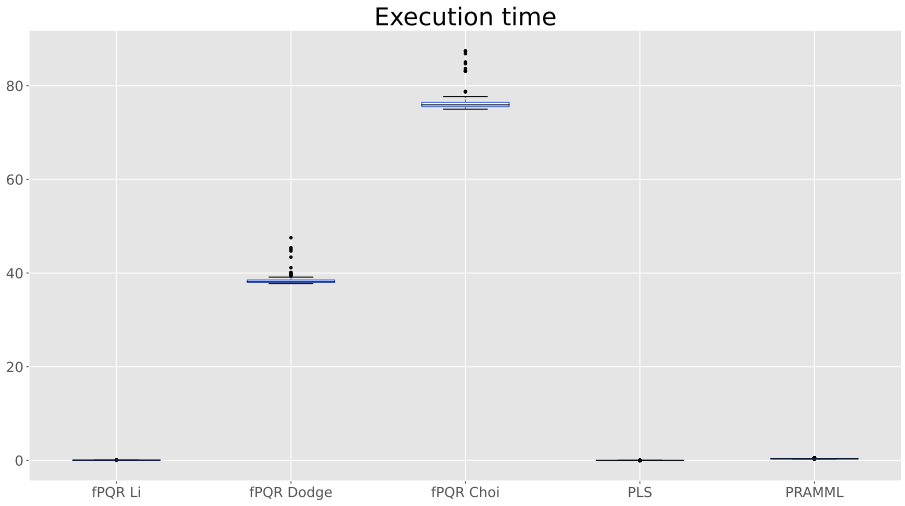


Table 3.2: Simulation 2. Sparse high dimensional framework with multidimensional response, considering a $\chi^2(3)$ error.

| | $\left\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right\|$ | $\frac{1}{\#\mathbb{D}_{test}}\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$ | Execution time |
|---|---|---|---|
| fPQR Li | 5.16 (0.42) | 15.14 (1.85) | 0.10 (0.013) |
| fPQR Dodge | 6.11 (0.48) | 16.37 (2.18) | 116.645 (2.139) |
| fPQR Choi | 6.70 (0.51) | 21.55 (3.89) | 232.64 (7.088) |
| PLS | 8.61 (1.01) | 32.01 (6.55) | 0.023 (0.004) |
| PRAMML | 12.06 (1.38) | 56.03 (11.74) | 1.02 (0.063) |

as it was discussed in Section 3.3.3.

### 3.4.2 Simulation 2

A second simulation is considered where we study the problem of having a multivariate response variable, very common in the field of chemometrics. Take,

$$Y = XB + \varepsilon, \tag{3.24}$$

where the predictors matrix $X$ of size $n = 100$ and $m = 100$ is generated from a standard normal distribution, and the matrix of coefficients $B$ has size $m = 100$ and $l = 3$. This defines a problem where the response matrix $Y$ has $l = 3$ dimensions. Out of the 100 predictive variables, 30 are generated from a standard uniform distribution and the remaining 70 have value 0, and finally the error term is generated following a chi squared distribution with 3 degrees of freedom. In this simulation, the number of components obtained by the algorithms is taken equal to the number of significant variables, $h = 30$. Additionally, a sample of 500 observations is generated as test set and the simulation is repeated 100 times to ensure the stability of the results. Algorithms PLS and fPQR can deal directly with multivariate response matrices, but PRAMML solves only univariate models, for this reason in this simulation the predictions from PRAMML are obtained by solving $l = 3$ independent univariate models.

Results from this simulation scheme are displayed in Table 3.2. The best results, both in terms of the euclidean distance and prediction error, are achieved by the fPQR Li algorithm, closely followed by fPQR Dodge. The fPQR Li algorithm also displays the smallest standard deviations, meaning that the results are stable. The PRAMML estimator is outperformed here by all the other algorithms including PLS, probably due to the inability to directly solve multivariate problems, requiring to solve those in a univariate manner. In terms of execution time, the fastest algorithm is PLS, while fPQR Li is the second fastest running 10 times faster than PRAMML. The fPQR Dodge and Choi algorithms are again the slowest.

Table 3.3: Simulation 3. Euclidean distance of $\boldsymbol{\beta}$ coefficient estimations under different error distributions.

| | $N(0,1)$ | $t_1$ | Slash |
|---|---|---|---|
| $(n,m,h) = (100,10,2)$ | | | |
| fPQR Li | 0.19 (0.13) | 0.25 (0.15 | 0.37 (0.23) |
| fPQR Dodge | 0.19 (0.13) | 0.26 (0.16) | 0.38 (0.24) |
| fPQR Choi | 0.49 (1.37) | 3.46 (55.95) | 1.69 (5.70) |
| PLS | 0.19 (0.10) | 6.23 (22.57) | 12.00 (58.51) |
| PRAMML | 0.16 (0.10) | 0.23 (0.14) | 0.31 (0.19) |
| $(n,m,h) = (15,60,4)$ | | | |
| fPQR Li | 0.79 (0.33) | 1.61 (1.25) | 2.21 (1.45) |
| fPQR Dodge | 0.90 (0.40) | 1.84 (1.56) | 2.49 (1.70) |
| fPQR Choi | 6.74 (42.19) | 18.78 (176.08) | 28.25 (231.58) |
| PLS | 1.14 (0.42) | 14.94 (68.17) | 29.55 (183.84) |
| PRAMML | 0.61 (0.31) | 1.02 (0.62) | 1.42 (0.98) |

### 3.4.3 Simulation 3

The last simulation considered takes the scheme from [Serneels et al., 2005] and [Acitas et al., 2020]. Consider the model,

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon} = TP^t\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{3.25}$$

where $X = TP^t \in \mathbb{R}^{n\times m}$ is the predictor matrix, $T \in \mathbb{R}^{n\times h}$ is a scores matrix and $P \in \mathbb{R}^{m\times h}$ is a loadings matrix. $T$ and $P$ are generated based on a $N(0,1)$ distribution, and $\boldsymbol{\beta} \in \mathbb{R}^m$ is the vector of true coefficients, generated based on a normal distribution with mean 0 and standard deviation 0.001. Three possible error distributions are considered for $\boldsymbol{\varepsilon} \in \mathbb{R}^n$: a standard normal distribution, a $t_1$ distribution, which is symmetric as the normal distribution but with heavier tails, and a slash distribution (defined as a standard normal distribution divided by a standard uniform distribution), which is heavy tailed and non symmetric. The number of components in the model is fixed, equal to the dimension of the latent loadings $h$. This process is repeated 500 times. Two cases are defined based on changes in the number of training observations $n$, variables $m$ and components $h$,

- A low dimensional example: $(n,m,h) = (100,10,2)$;
- A high dimensional example: $(n,m,h) = (15,60,4)$.

Results from this simulation are shown in Tables 3.3 and 3.4. In terms of the euclidean distance of the $\beta$ coefficients, one can see that PRAMML estimator obtains the best results closely followed by fPQR Li and Dodge algorithms, being both competitive alternatives. It is worth remarking the fact that fPQR Li and Dodge

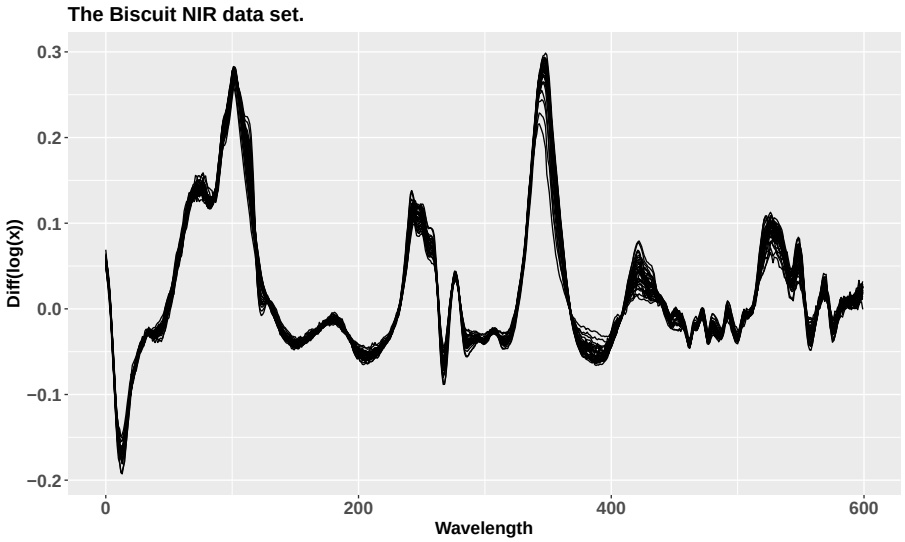
Table 3.4: Simulation 3. Execution time

| fPQR Li | fPQR Dodge | fPQR Choi | PLS | PRAMML |
|---|---|---|---|---|
| $(n, m, h) = (100, 10, 2)$ | | | | |
| 0.015 | 0.27 | 0.54 | 0.0006 | 0.017 |
| $(n, m, h) = (15, 60, 4)$ | | | | |
| 0.017 | 2.99 | 5.94 | 0.0007 | 0.021 |

outperformed PLS even when considering a normal distribution for the error term, where PLS is expected to excel. Finally, fPQR Choi consistently provides the worst results. The execution time is affected by the number of observations $n$, variables $m$ and $l$, and components $h$, but not by the error distribution, for this reason Table 3.4 shows the execution time regardless of the error distribution. PLS is the fastest algorithm, while fPQR Li is the second fastest, closely followed by PRAMML. Results regarding prediction accuracy are not included in this simulation scheme because the error distributions considered generated outliers with very large values, providing predictions where the mean squared error values were very large and very similar regardless of the algorithm.

The three simulations displayed in this section remark the fact that, among the three quantile covariances under study, the best alternative for the fPQR algorithm is the quantile covariance proposed by Li et al. [2015], as it consistently provides the smallest prediction errors and the smallest euclidean distance of the $\beta$ coefficients. Additionally, it is by far the fastest of the three algorithms, having a computation based on a traditional covariance rather than in solving univariate quantile regression models, as is the case with the other quantile covariances considered. Comparing the fPQR Li algorithm for the median with PLS shows that it outperformed PLS in all the scenarios considered in terms of prediction accuracy and euclidean distance of the $\beta$ coefficients. When comparing it with robust PLS alternatives like PRAMML, it is worth remarking the fact that fPQR Li can be used to solve multidimensional response problems while PRAMML requires to face this situation by solving univariate models, as discussed in Section 3.4.2. Additionally, one can see that fPQR Li is a competitive alternative in terms of prediction accuracy and euclidean distance of the $\beta$ coefficients, providing better estimations in two of the three simulations, and being competitive in the last one. In terms of execution time, fPQR Li also outperformed PRAMML in all the simulations. But the fPQR algorithm has an additional advantage when compared with any PLS based methodology: PLS based methodologies can only obtain estimations for the mean of the response matrix, while fPQR can obtain estimations for different quantile levels. This allows to study not only the central behavior of the response variable, but also the behavior at any other quantile of interest, like the tails of the distribution.

Figure 3.4: Biscuit dataset: NIR spectra of the biscuit dataset.




## 3.5 Real data analysis: Biscuit data

The biscuit data was first introduced in Osborne et al. [1984]. This dataset contains four response variables, concentration of fat, flour, sucrose and water, of 72 biscuit dough samples, where 40 observations usually define a training set and 32 a prediction set. In this analysis, and following the steps from [Hubert and Branden, 2003], the variable fat was removed because it showed small correlation coefficients with the other constituents and a larger variance. The rest of the response variables show larger correlations and similar variances, and for this, a multivariate analysis is considered. The objective is to predict the values of the three response variables based on NIR spectra measurements taken every 2 nm from 1200 up to 2400. The same preprocessing steps as in [Hubert et al., 2002] and [Hubert and Branden, 2003] were performed, obtaining a NIR spectra prediction matrix of $m = 600$ dimensions, shown in Figure 3.4, and a response matrix of $l = 3$ dimensions. Though observation 23 is known to be an outlier, it is kept in the dataset.

Using this dataset, a comparison of fPQR Li, PLS and PRAMML estimators is performed. The quantile level is taken as $\tau = 0.5$ so that quantile based results can be compared with the mean based results from PRAMML and PLS, and since the PRAMML estimator solves only univariate models, the predictions from this estimator are obtained by solving 3 independent univariate models. The first step is to select the number of components to be computed. This is done by performing 5-fold cross validation on the training set, and the objective is to minimize the mean squared error of the predictions. Figure 3.5 shows the CV results, concluding that three is the best number of components for any of the models considered.

The final models are built using the 40 observations from the training set and 3 components, and the mean squared error of the prediction of each model is computed

Figure 3.5: Biscuit dataset: CV mean squared error on the number of components.

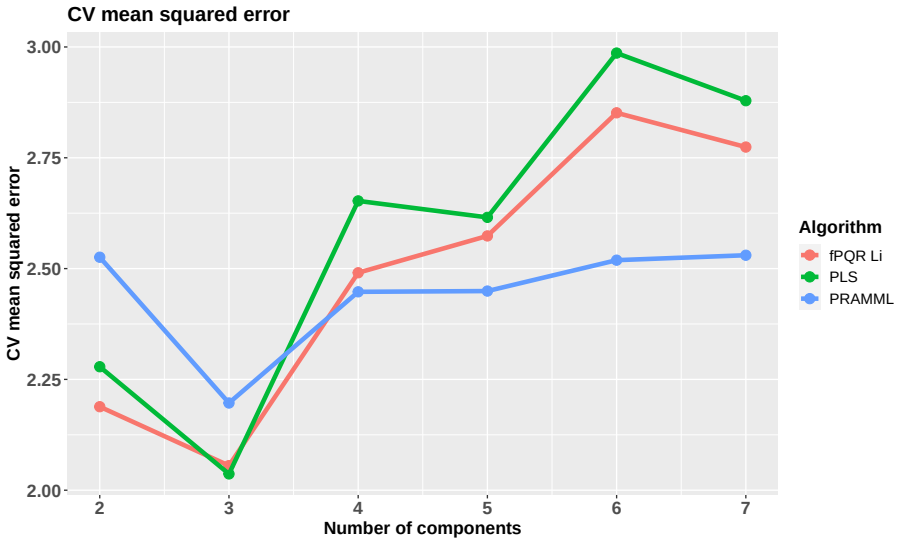



Table 3.5: Biscuit data: Test mean squared error.

| fPQR Li | PLS | PRAMML |
|---|---|---|
| 0.491 | 0.614 | 0.527 |

on the test set. Table 3.5 shows the results. One can see that best result is obtained by fPQR Li, followed by the PRAMML estimator, and PLS obtains the worst result, presumably due to the presence of outliers in the dataset. An additional advantage of fPQR Li is that it can provide estimations for different quantile levels. Take for example observation 41, which is the first one in the test set. This observation has values flour= 16.44, sucrose= 47.65 and water= 12.57, and the median prediction obtained using fPQR Li for $\tau = 0.5$ is flour= 15.68, sucrose= 48.39 and water= 12.82. But one can also calculate an estimation of any other quantile of interest, obtaining this way prediction intervals. For example, the prediction for the 10% percentile of the response is flour= 15.24, sucrose= 47.67 and water= 12.39 for a small biscuit dough given the associated NIR spectra values, while the 90% percentile for a large biscuit dough has values flour= 17.22, sucrose= 48.41 and water= 13.10. The fPQR Li algorithm can thus provide a complete picture of the distribution of the response matrix.

## 3.6 Computational aspect

All the simulations and analysis commented in Sections 3.4 and 3.5 were run in a computer with an Intel Core i7-10750H CPU (2.6GHz) processor with 32GB RAM memory running the O.S. Windows 10. The computation of the fPQR has been developed in Python 3.8.5 (Anaconda Inc.). The quantile covariance metrics in-

troduced in Section 3.3.3 required solving quantile regression models. Those were solved using the Python package ASGL, built on top of the CVXPY optimization framework for Python [Diamond and Boyd, 2016] and Mosek solver [ApS, 2021]. The PRAMML estimator was computed using the R package 'rpls' [Filzmoser et al., 2020], as there was no Python implementation for this methodology.

## 3.7 Conclusion

In this paper the fast partial quantile regression (fPQR) algorithm has been introduced. This algorithm extends the PLS models to the quantile regression framework. The result is a dimensionality reduction technique that parallels the nice properties of PLS models but that is linked to the quantiles of the response matrix, being robust to the presence of outliers or heteroscedastic data. As discussed in Section 3.3, the key idea behind fPQR is the definition of the objective function that it maximizes in terms of a quantile covariance metric, and in this work different metrics are considered [Li et al., 2015], [Dodge and Whittaker, 2009], [Choi and Shin, 2018]. Section 3.4 studies the performance of the fPQR algorithm using the different quantile metrics in a set of synthetic datasets, concluding that the best results in terms of prediction accuracy, euclidean distance of the $\beta$ coefficients and execution time are obtained using the quantile covariance defined by Li et al. [2015]. Additionally, the performance of the fPQR algorithm is compared with PLS and PRAMML [Acitas et al., 2020] estimators, showing that, if the median estimation is computed, fPQR is a competitive alternative to other robust PLS algorithms, but additionally, fPQR can obtain estimates for different quantile levels of the response matrix, providing a complete picture of its distribution. The performance of the proposed work is also studied in a real high dimensional dataset containing NIR spectra measurements, where fPQR Li obtains the best prediction accuracy.

## Acknowledgments

This research was partially supported by research grants and projects PID2020-113961GB-I00 and PID2019-104901RB-I00 from Agencia Estatal de Investigación.

# Bibliography


Sukru Acitas, Peter Filzmoser, and Birdal Senoglu. A new partial robust adaptive modified maximum likelihood estimator. *Chemometrics and Intelligent Laboratory Systems*, 204:104068, 2020. ISSN 18733239. doi: 10.1016/j.chemolab.2020.104068. URL https://doi.org/10.1016/j.chemolab.2020.104068.

Mosek ApS. MOSEK Optimizer API for Python 9.3.6, 2021. URL https://docs.mosek.com/9.3/pythonapi/index.html.

Anne-laure Boulesteix and Korbinian Strimmer. Partial least squares : a versatile tool for the analysis of high-dimensional genomic data. *Briefings in Bioinformatics*, 8(1):32–44, 2006. doi: 10.1093/bib/bbl016.

Soumyadeep Chatterjee, Snigdhanshu Banerjee, Arindam, and Auroop R. Ganguly. Sparse Group Lasso for Regression on Land Climate Variables. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 1–8. IEEE, 12 2011. ISBN 978-1-4673-0005-6. doi: 10.1109/ICDMW.2011.155.

Ji-Eun Choi and Dong Wan Shin. *Quantile correlation coefficient: a new tail dependence measure*. 2018. ISBN 8223277360. URL http://arxiv.org/abs/1803.06200.

Sijmen de Jong. SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3):251–263, 3 1993. ISSN 0169-7439. doi: 10.1016/0169-7439(93)85002-X.

Steven Diamond and Stephen Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *arXiv:1603.00943*, 3 2016.

Yadolah Dodge and Joe Whittaker. Partial quantile regression. *Metrika*, 70:35–57, 2009. ISSN 00261335. doi: 10.1007/s00184-008-0177-4.

Peter Filzmoser, Sukru Acitas, and Birdal Senoglu. rpls: Robust Partial Least Squares, 2020. URL https://cran.r-project.org/package=rpls.

M. Hubert and K. Vanden Branden. Robust methods for partial least squares regression. *Journal of Chemometrics*, 17(10):537–549, 10 2003. ISSN 0886-9383. doi: 10.1002/cem.822. URL http://doi.wiley.com/10.1002/cem.822.

Mia Hubert, Peter J. Rousseeuw, and Sabine Verboven. A fast method for robust principal components with applications to chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 60(1-2):101–111, 2002. ISSN 01697439. doi: 10.1016/S0169-7439(01)00188-5.

Roger Koenker and Gilbert Bassett. Regression Quantiles. *Econometrica*, 46(1): 33–50, 1 1978. ISSN 00129682. doi: 10.2307/1913643.

Guodong Li, Yang Li, and Chih Ling Tsai. Quantile Correlations and Quantile Autoregressive Modeling. *Journal of the American Statistical Association*, 110 (509):246–261, 2015. ISSN 1537274X. doi: 10.1080/01621459.2014.892007.

Alvaro Mendez-Civieta, M. Carmen Aguilera-Morillo, and Rosa E. Lillo. Adaptive sparse group LASSO in quantile regression. *Advances in Data Analysis and Classification*, 15(3):547–573, 2021. ISSN 18625355. doi: 10.1007/s11634-020-00413-8.

Danh V. Nguyen and David M. Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18(1):39–50, 2002. ISSN 13674803. doi: 10.1093/bioinformatics/18.1.39.

Brian G. Osborne, Thomas Fearn, Andrew R. Miller, and Stuart Douglas. Application of near infrared reflectance spectroscopy to the compositional analysis of biscuits and biscuit doughs. *Journal of the Science of Food and Agriculture*, 35 (1):99–105, 1984. ISSN 10970010. doi: 10.1002/jsfa.2740350116.

Sven Serneels, Christophe Croux, Peter Filzmoser, and Pierre J. Van Espen. Partial robust M-regression. *Chemometrics and Intelligent Laboratory Systems*, 79(1-2): 55–64, 2005. ISSN 01697439. doi: 10.1016/j.chemolab.2005.04.007.

H Wold. Nonlinear Iterative Partial Least Squares (NIPALS) Modelling: Some Current Developments. In Paruchuri R Krishnaiah, editor, *Multivariate Analysis?III*, pages 383–407. Academic Press, 1973. ISBN 978-0-12-426653-7.

Svante Wold, Michael Sjöström, and Lennart Eriksson. PLS-regression: A basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58(2): 109–130, 2001. ISSN 01697439. doi: 10.1016/S0169-7439(01)00155-1.

Yichao Wu and Yufeng Liu. Variable selection in quantile regression. *Statistica Sinica*, 19(2):801–817, 2009.

# CHAPTER 4

## Functional Quantile Factor Models

Álvaro Méndez Civieta[1,2], Ying Wei[3] and Jeff Goldsmith[3].

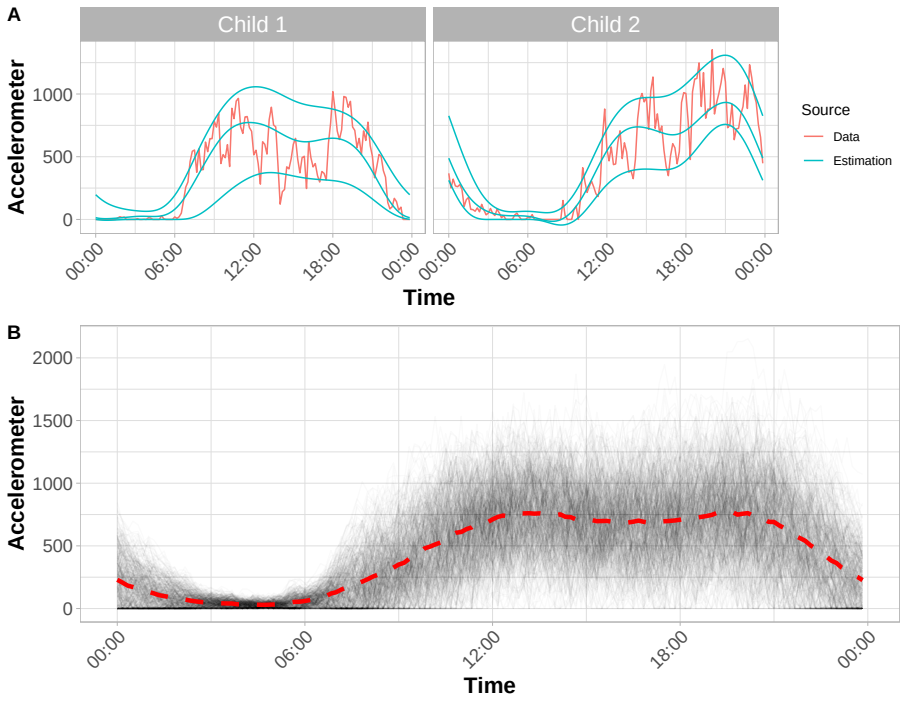
1. Department of Statistics, Universidad Carlos III de Madrid.
2. uc3m-Santander Big Data Institute.
3. Department of Biostatistics, Columbia University, New York, U.S.A.

## Abstract

This paper introduces the Functional Quantile Factor Model (FQFM), a dimensionality reduction technique that extends the concept of functional principal components to the quantile regression framework, obtaining a model that can explain the quantiles of the data conditional on a set of common functions. The need for such methodology is exemplified by our motivating example: a study in which the level of physical activity in 420 children was measured during one week. To the best of our knowledge, there is currently no work understanding the quantile trends of physical activity from the functional perspective, but it can certainly provide useful information. FQFM is able to capture shifts on the scale of the data affecting the quantiles, and is also a robust methodology suitable for dealing with outliers and heteroscedastic data. The model is estimated using penalized M splines, and can deal with sparse and irregular time measurements. The proposed methodology is evaluated in synthetic data and real data analyses, and is implemented in R programming language.

**keywords:** Accelerometer data; Functional data; Quantile regression; Penalized splines; Dimension reduction.

Figure 4.1: Accelerometer measurements from 420 children. (A) includes two observations from the dataset, showing a clear difference in the pattern of physical activity. (B) shows the full dataset, including an estimate of the mean behavior as a dashed line.




## 4.1 Introduction

Wearables are a new generation of electronic devices that can be easily worn as accessories like wrist watches, and that allow to quantify different aspects of a user's daily activity, for example heart rate, brainwave, activity counts, etc. This information is obtained in an objective and unbiased manner, in minute by minute or even finer granularity providing almost a continuous stream, and can then be analyzed to improve the understanding of the relation between human behavior and health. Some examples of the usage of wearables in the field of biomedicine include the effect of age on physical activity [Varma et al., 2017], the study of circadian rhythms [Xiao et al., 2015], or a study of activity levels on children [Morris et al., 2006] among others. Traditionally, analysis of these kinds of data has been centered on simple summaries such as the average daily activity, however this approach ignores the temporal information provided by the continuous stream. This temporal factor can be better accounted for using functional data analysis (FDA) [Ramsay and Silvermann, 1998], where each observation represents a curve of activity usually recorded over a 24 hours period. FDA is a statistics field that has shown an intense methodological development in recent years and that can be used in many different areas of scientific research such as chemometrics, economics, engineering, or any other field where there is a space or temporal factor.

This work is motivated by the study of a wearable dataset. Four hundred and twenty children participating in a Head Start program from centers in northern Manhattan, the Bronx, and Brooklyn were recruited. Then, field staff attached an accelerometer to the child's non-dominant wrist with a hospital band, measuring their physical activity during a period of 6 days using 1-minute epochs. Each diurnal activity profile is regarded as a functional data point, and denoted as $X_i(t)$ for child $i$ at time point $t$. The profile is obtained by averaging for each separate $t$ across the 6 days, and additionally data is aggregated into 10-minute epochs. For a previous analysis of this dataset using functional regression, see [Goldsmith et al., 2016].

A very interesting problem that can be studied here is the difference in physical activity patterns between children. For example, some environmental factors such as neighborhoods can affect the development of physical activity during the day but not at night. Describing and understanding the different profiles of physical activity can be key in the development of effective programs to increase the overall activity. Figure 4.1 (A) is formed by two specific observations from the accelerometer dataset showing different physical activity patterns. It includes the estimations provided by the proposed FQFM methodology for quantiles 10%, 50% and 90%. (B) shows an estimation of the mean function as a dashed line that describes a clear trend in the data with the presence of two peaks around 12PM and 8PM, as well as a decrease of the activity during the night. But the mean alone does not explain all the variability in the data. Usually, this problem is faced by making use of dimension reduction techniques such as functional principal component analysis (FPCA). FPCA provides a set of orthonormal basis functions that best describe variation across curves. See [Ramsay and Silvermann, 1998] for an introduction on the topic, or [James et al., 2000] for a usage example of FPCA on growth dataset.

However, FPCA is known to compute mean based estimates, and does not capture hidden aspects that may affect the scale, shifting the quantiles. It can be certainly interesting to study not just the variation in the mean but in the quantiles, providing insight on the behavior at the tails of the distribution. There is very little work on quantile trends for physical activity. The problem of modeling quantiles was first studied in the context of multivariate regression by Koenker and Bassett [1978], and has gained special importance in recent years as an alternative to least squares regression. Quantile regression can provide robust estimates of the quantiles of a response variable, suitable for dealing with non normal distributions, heteroscedasticity and outliers. It can also help to understand the behavior of the data at different quantile levels, providing a complete picture of its distribution. Quantile regression was probably first extended to the functional framework by Cardot et al. [2005], where they considered a scalar on function model based on smoothing splines. Then [Kato, 2012] studied the same model under a PCA basis framework, and [Chen and Müller, 2012] considered the estimation of the conditional quantile function by inverting the corresponding conditional distribution function.

The problem of a function on scalar model was studied by Yang et al. [2019], where they modeled the quantile function of each subject as a function of subject-specific covariates. All these works are centered in a functional regression setting, but do not consider the study of a dimension reduction technique like FPCA but based on quantiles. The objective of our work is to develop new methods that allow dimension reduction and modeling of within and between subject variation in quantiles over time. To the best of our knowledge, there is no existing approach for modeling quantiles in FDA.

There have been some advances in the field of quantile factor analysis for panel data, especially for economic data. Chen et al. [2021] introduced a quantile factor model that effectively extended factor analysis to quantile regression. They proposed an iterative algorithm based on the power method [Hotelling, 1933] (used in PCA estimation) and the quantile regression loss check function. Their methodology is able to provide quantile dependent loadings and common factors, as opposed to PCA where neither loadings nor factors are allowed to vary across distributional characteristics. We take this work as a starting point and introduce our main contribution. We propose a functional quantile factor model (FQFM) that extends the work by [Chen et al., 2021] to the FDA framework. Instead of treating the data as single points and estimating the factor curves independently across time, we consider each observation as a smooth curve measured on a grid of time points and propose an algorithm that makes use of M-splines with a roughness penalty for the estimation of the common factor curves. The main advantages of this approach are: (i) this approach can deal with irregular and sparse sets of time points which can differ across individuals, while the algorithm by [Chen et al., 2021] is only suitable for working on traditional panel data where fine grids are taken at the same time points for all observations, and (ii) the usage of penalized splines ensure that the common factors are smooth curves, eliminating sharp changes in direction. Additionally, we introduce the concept of an intercept quantile curve in the algorithm, and study the deviations of the common factor curves from this quantile trend. Observe that the usage of penalized splines reduce the impact of the choice of basis, as well as the impact of the number and position of the knots. It remains to select the value for the smoothing parameter, and here we consider a cross validation procedure that minimizes a quantile error measurement.

The rest of the paper is organized as follows. In Section 4.2 the mathematical formulation of the quantile dependent latent structure that the FQFM solves is posed. Section 4.2.1 introduces the conceptual definition of the model and a first version of the iterative algorithm required to solve this problem. In Section 4.2.2 the model is formulated in terms of a basis expansion, and a roughness penalty is introduced. Section 4.2.3 proposes the matrix formulation of the iterative algorithm encompassing the basis expansion and the roughness penalty. Section 4.3 tests the performance of the proposed FQFM algorithm in two synthetic dataset frameworks,

studying the integrated mean squared errors between the true quantile distributions and the quantiles estimated by the algorithm, as well as a quantile based error metric. In Section 4.4 the proposed algorithm is used in the motivating accelerometer data example. Some computational aspects are briefly commented in Section 4.5, and the conclusions are provided in Section 4.6.

## 4.2 Functional Quantile Factor Analysis

Let $X_i(t),\ i = 1, \dots N$ be a set of trajectories assumed to be independent realizations of a real valued second order stochastic process $\{X(t) : t \in \mathcal{T}\}$ defined on a bounded close interval $\mathcal{T}$. Without loss of generality, consider $\mathcal{T} = [0, 1]$. Let us assume that the sample paths of $X$ are continuous and consider the existence of a latent structure modelling the quantiles of observation $i$,

$$Q_{X_i(t)}(\tau|f(t,\tau)) = f_0(t) + \sum_{j=1}^{r} \lambda_{ij}(\tau) f_j(t,\tau) = f_0(t) + \lambda_i(\tau)' f(t,\tau), \tag{4.1}$$

where $\tau$ is the quantile level, $r$ is the number of latent functional factors ($r \ll N$), $\lambda_i(\tau) = (\lambda_{i1}(\tau), \dots, \lambda_{ir}(\tau))'$ is an $r \times 1$ quantile dependent vector of factor scores, $f_0(t)$ is a quantile dependent intercept curve, $f(t,\tau) = (f_1(t,\tau), \dots, f_r(t,\tau))'$ is the vector of quantile dependent common functions evaluated at time $t$ and $Q_{X_i(t)}(\tau|f(t,\tau))$ denotes the $\tau$th quantile of $X_i(t)$ conditional on the common functions. Observe that the first curve $f_0(t,\tau)$ estimates the general quantile trend of function $X(t)$, and the rest of the functions $f_j(t,\tau),\ j = 1, \dots, r$ are factor curves measuring deviations from this trend. This behavior parallels that of FPCA, where the functional principal components show deviations from the mean trend of the data. Taking $\lambda_i(\tau) = (\mathbf{1}, \lambda_{i1}(\tau), \dots, \lambda_{ir}(\tau))'$ and $f(t,\tau) = (f_0(t), f_1(t,\tau), \dots, f_r(t,\tau))'$, equation (4.1) can also be posed as,

$$X_i(t) = \lambda_i(\tau)' f(t,\tau) + u_i(t,\tau), \tag{4.2}$$

where the quantile dependent idiosyncratic error $u_i(t,\tau)$ satisfies the condition,

$$P\left(u_i(t,\tau) \leq 0 | f(t,\tau)\right) = \tau, \forall t \tag{4.3}$$

Just as it happens with other dimensionality reduction techniques such as PCA or factor analysis, in order to ensure that the solution to the model posed in equation (4.1) is unique, it is necessary to include some restrictions on both the scores and the common functions. Without loss of generality, the following conditions are proposed here,

$$\begin{aligned} &\int_0^1 f_j(t,\tau)^2 dt = 1 \text{ and } \int_0^1 f_j(t,\tau) f_m(t,\tau) dt = 0 \text{ for } m < j, \\ &\frac{1}{N}\sum_{i=1}^{N} \lambda_i(\tau)\lambda_i(\tau)' \text{ is diagonal with non increasing diagonal elements.} \end{aligned} \tag{4.4}$$

### 4.2.1 Conceptual formulation

This section is centered on the conceptual formulation of the strategy to solve the model proposed in equation 4.1 from a functional perspective, while the actual model resolution over a finite grid of time points will be studied afterwards in Section 4.2.3. In order to keep the notation clear, the dependence of the scores $\lambda(\tau)$ and the common functions $f(t,\tau)$ on the quantile level $\tau$ will be assumed, but removed from further equations keeping these elements simply as $\lambda$ and $f(t)$ respectively. Consider $\Lambda = (\lambda_1, \ldots, \lambda_N)'$ and define,

$$\mathbb{M}(\Lambda, f(t)) = \frac{1}{N}\sum_{i=1}^{N}\int_0^1 \rho_\tau\left(X_i(t) - \lambda_i' f(t)\right) dt, \tag{4.5}$$

where $\rho_\tau(u) = u(\tau - I(u<0))$ is the quantile regression loss check function as proposed by [Koenker and Bassett, 1978] and $I(\cdot)$ is the indicator function. Then, assuming a parametric approach on both $\Lambda$ and $f(t)$, the FQFA estimator is obtained as the solution to the following optimization problem,

$$(\hat{\Lambda}, \hat{f(t)}) = \arg\min\ \mathbb{M}(\Lambda, f(t)), \tag{4.6}$$

subject to the restrictions posed in equation (4.4). The minimization of this function defines a non convex optimization problem with no direct analytical solution. However, following the steps from Chen et al. [2021], an iterative process capable of finding the stationary points of the objective function can be defined. The key idea is centered in dividing the objective function into two sub functions,

$$\text{Given } f(t), \text{ define } \mathbb{M}(\lambda_i) = \int_0^1 \rho_\tau\left(X_i(t) - \lambda_i' f(t)\right) dt, \tag{4.7}$$

$$\text{Given } \Lambda, \text{ define } \mathbb{M}(f(t)) = \frac{1}{N}\sum_{i=1}^{N}\rho_\tau\left(X_i(t) - \lambda_i' f(t)\right). \tag{4.8}$$

As it happens, $\mathbb{M}(\lambda_i)$ is convex in $\lambda$ for each observation $i$, and $\mathbb{M}(f(t))$ is convex in $f$ at any time point $t$. The following iterative algorithm is then proposed,

1. Take a random initialization for the common curves $f^{(0)}(t) = (f_0^{(0)}(t), \ldots, f_r^{(0)}(t))$.
2. Given $f^{(l-1)}(t)$ solve,

   $$\lambda_i^{(l-1)} = \arg\min\ \mathbb{M}(\lambda_i) = \arg\min\ \int_0^1 \rho_\tau\left(X_i(t) - \lambda_i' f(t)^{(l-1)}\right) dt.$$

   Repeat for $i = 1, \ldots, N$ until matrix $\Lambda^{(l-1)}$ is obtained.
3. Given $\Lambda^{(l-1)}$, solve,

   $$f(t)^{(l)} = \arg\min\ \mathbb{M}(f(t)) = \arg\min\ \frac{1}{N}\sum_{i=1}^{N}\rho_\tau\left(X_i(t) - \lambda_i^{(l-1)\prime} f(t)\right).$$

Iterate through steps 2-3 until the objective function value $\mathbb{M}(\Lambda, f(t))$ has converged. Finally, perform a normalization process to ensure restrictions from equation (4.4) are satisfied.

### 4.2.2 Basis expansion

In practice, the functions $X_i(t)$ are usually measured on a finite set of time points and represented as finite dimensional vectors. From a multivariate setting perspective, Chen et al. [2021] studied the case of decomposing panel data into a set of quantile dependent scores and common factors when one has measurements taken on a fine grid at the same time points for all individuals. However, functions are often measured at irregular and sparse sets of time points that may differ across individuals. In this work we are interested in handling this more difficult situation. As it is common in FDA, the first step in the actual estimation of the common curves $f(t)$ is the reconstruction of the functional form of the data from discrete observations, handled using an expansion into a splines basis representation,

$$f_j(t) = \sum_{k=1}^{p} \beta_{jk}\phi_k, \;\; j = 1, \ldots, r, \tag{4.9}$$

where $\beta_{jk}$ is a spline coefficient for curve factor $j$ and basis $k$, and $\phi(t) = (\phi_1, \ldots, \phi_p)'$ is a $p$ dimensional basis. One drawback of [Chen et al., 2021] is the fact that being a multivariate approach, the estimation of the common curves is performed independently across time and smoothness is not taken into account, often obtaining rough curve estimations for the factors, with sharp spikes that difficult interpretability. A main objective in this work is to obtain smooth estimates of the common curves, ensuring that the quantile curves estimates of $X_i(t)$ will also be smooth. There are different ways in which this can be achieved. Probably the simplest is by controlling the number of spline basis functions, however this is a discontinuous process. In this work, a continuous alternative based on a roughness penalty is considered. A natural way of measuring the roughness of a function $f_j(t)$ is centered on the usage of it's derivatives of some order $d$, $D^d(f_j(t)) = f_j^d(t)$, $d \geq 1$. [O'Sullivan, 1986] proposed using the squared of the second derivative, which is traditionally called curvature, and defined the penalization,

$$\text{PEN}_2(f_j) = \int \left[D^2 f_j(t)\right]^2 dt. \tag{4.10}$$

Functions with large variability are expected to have large values of $\text{PEN}_2(f)$, and by controlling it, one can obtain smooth estimates and prevent overfitting the data. Taking into account the basis expansion defined in equation (4.9), the roughness penalty is then given by,

$$\text{PEN}_2(f_j) = \int \left[D^2 \beta'_{j\cdot}\phi(t)\right]^2 dt = \beta'_{j\cdot} R \beta_{j\cdot}, \tag{4.11}$$

where $\beta_{j\cdot} \in \mathbb{R}^p$ is the vector of spline coefficients associated to curve $f_j(t)$ and $\boldsymbol{R}$ is the matrix of the cross inner products of the second order derivatives of basis functions $\phi$.

### 4.2.3 Matrix computation

Let us go now from the conceptual formulation of the model to the actual resolution considering a over a finite grid of time points. First, we include the basis expansion and the roughness penalty into the iterative algorithm implementation proposed in Section 4.2.1. Given the function $X_i(t)$ measured at time points $\{t_{i1}, \ldots, t_{im_i}\}$ let us take $X_{i\cdot} \in \mathbb{R}^{T_i}$ as the vector storing the values of the function at the time points where it was measured and define $\text{vec}(X) = (X_{1\cdot}, \ldots, X_{N\cdot})'$ a vector of dimension $T_{\max} = \sum_{i=1}^{N} T_i$ result of row stacking the observations across time points and individuals. Observe that different individuals may be measured at different time points. This means that $\{t_i\} \neq \left\{t_j\right\}$ for $i \neq j$, $i, j \in \{1, \ldots, N\}$. Take $\Phi_i \in \mathbb{R}^{p \times T_i}$ the basis matrix associated to the spline basis functions $\phi(t)$ measured at the same time points as observation $i$. Take $B \in \mathbb{R}^{(r+1)\times p}$ the matrix of spline coefficients and $\text{vec}(B) = (\beta_{0\cdot}, \ldots, \beta_{r\cdot})'$ a vector of dimension $rp \times 1$ result of row stacking matrix $B$. Using the Kronecker product, "$\otimes$", define $\left[\lambda_i' \otimes \Phi_i'\right]$ the super matrix of dimension $(T_i \times rp)$ consisting of sub matrices $\lambda_{ij}\Phi_i'$, and $[\Lambda \otimes \Phi']$ the matrix of dimension $T_{\max} \times rp$ result of concatenating $\left[\lambda_i' \otimes \Phi_i'\right]$ across different observations. Finally, define $R_{r+1}$ a block diagonal matrix made of $r+1$ replications of the penalization matrix $R$ defined in equation (4.11). Then, the objective function of the FQFM estimator is rewritten as,

$$\mathbb{M}(\Lambda, B) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \rho_\tau \left(X_{it} - \lambda_i' B \Phi_i\right), \tag{4.12}$$

and the iterative algorithm is,

1. Take a random initialization of the matrix of spline coefficients $B^{(0)}$

2. Given $B^{(l-1)}$ solve,

$$\lambda_i^{(l-1)} = \arg\min \ \frac{1}{T_i} \sum_{t=1}^{T_i} \rho_\tau \left(x_{i\cdot} - \lambda_i' B^{(l-1)} \Phi_i\right). \tag{4.13}$$

   Repeat for $i = 1, \ldots, N$ until matrix $\Lambda^{(l-1)}$ is obtained.

3. Given $\Lambda^{(l-1)}$, solve,

$$\begin{aligned} \text{vec}(B) = \arg\min \ & \frac{1}{T_{\max}} \sum_{a=1}^{T_{\max}} \rho_\tau \left(\text{vec}(X)_a - \left[\Lambda^{(l-1)} \otimes \Phi'\right]_a' \text{vec}(B)\right) + \\ & + \alpha \text{vec}(B)' R_{r+1} \text{vec}(B). \end{aligned} \tag{4.14}$$

Iterate through steps 2-3 until the objective function value $\mathbb{M}(\Lambda, B)$ has converged. Finally, perform a normalization process to ensure restrictions from equation (4.4) are satisfied. Observe that the roughness penalty has been included in the spline coefficients estimation step (equation (4.14)) controlled by a hyper parameter $\alpha$. Larger values of $\alpha$ yield to more smooth, smaller curvature estimations of the

common factor functions $f(t)$, while smaller values of $\alpha$ produce more variable solutions. The value of this parameter is selected at the beginning of the algorithm execution and can be optimized using a cross validation based process. Additionally, observe that, opposed to the algorithm proposed by Chen et al. [2021] where the common factors are estimated independently across time, the formulation in equation (4.14) estimates the splines coefficients using all the available time points simultaneously.

## 4.3 Numerical simulation

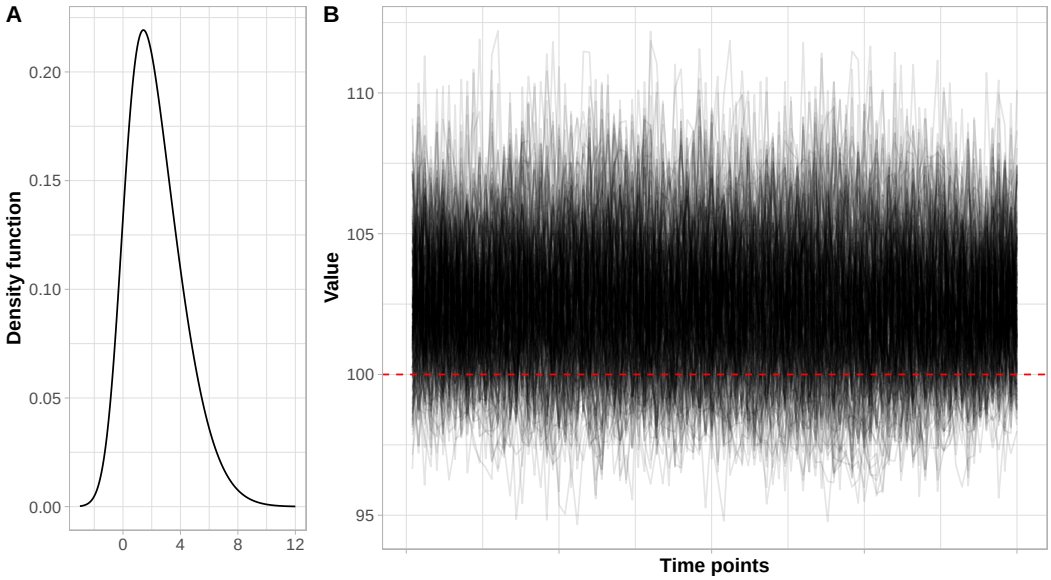
This section shows the performance of the proposed FQFM methodology under different synthetic datasets. The results are studied at three different quantile levels: 10%, 50% and 90%, and the algorithm is compared against the QFM methodology proposed by [Chen et al., 2021]. Additionally, the median estimations provided by the FQFM and QFM estimators are compared against the FPCA estimation of the curves mean. The algorithm proposed by Chen et al. [2021] is unable to solve problems where the data is measured at irregular time grids that vary across individuals, for this reason the simulations considered here have equally spaced measurements taken at the same time points for all individuals. For each dataset $\mathbb{D}$, a partition into two disjoint subsets, $\mathbb{D}_{train}$ and $\mathbb{D}_{test}$ is considered. $\mathbb{D}_{train}$ is used for training the models, this is, solving the model equations. $\mathbb{D}_{test}$ is used for testing the models prediction accuracy. Given that the FQFM estimator can handle the situation where data is measured at irregular time points, a last estimator denoted as $\text{FQFM}_{50\%}$ is considered. This estimator is built using only a random selection of 50% of the data points from the training set, effectively making this estimator work with irregular time grids across individuals. The following metrics are computed for all the estimators,

- IMSE= $\frac{1}{T_{\max}}\|\hat{Q}_{X_{it}}(\tau) - Q_{X_{it}}(\tau)\|_2$. The integrated mean squared error (IMSE) between the estimation of the quantiles for function $X_i(t)$ and the true value of the quantiles.

- QE= $\frac{1}{T_{\max}} \sum_{a=1}^{T_{\max}} \rho_\tau \left(\text{vec}(\hat{Q}_{X_{it}}(\tau))_a - \text{vec}(X_{it})_a\right)$. The quantile error (QE) measured as the value of the quantile regression loss check function.

Observe that the value of the IMSE is only available in synthetic datasets where the true value of the quantiles of $X_i(t)$ is known, but the QE value can always be computed and is thus considered as the reference metric when dealing with real data in which the true quantiles are unknown.

Figure 4.2: Simulation 1. (A) shows the density function of the asymmetric error term $\varepsilon_i(t)$. (B) shows a subset of the generated dataset.




### 4.3.1 Simulation 1

Consider the following data generation process,

$$X_i(t) = 100 + \lambda_{i1} + \lambda_{i2}\sin(t) + \lambda_{i3}\sin(\frac{1}{2}t) + \varepsilon_i(t) \tag{4.15}$$

where

- $\lambda_{i1} \sim N(0, 0.5)$ is the score associated to an additive factor that shifts the location of the data,
- $\lambda_{i2} \sim N(0, 1)$ is the score associated to a multiplicative factor that shifts the scale of the data,
- $\lambda_{i3} \sim N(0, 1)$ is the score associated to another multiplicative factor that shifts the scale of the data,
- $\varepsilon_i(t)$ is the error term, that follows a skewed normal distribution with location= 0, scale= 3 and $\alpha = 3$.

A grid of 100 equally spaced time points taken in the interval $[0, 2\pi]$ is considered. A training dataset formed by 500 observations and a test set formed by additional 500 observations are generated following this process. Figure 4.2 (A) shows the asymmetric density function of $\varepsilon_i(t)$, while (B) shows a subset of the dataset. In this simulation the number of factors is taken equal to the number of true factors, which is 3, a location shift and two scale shifts. Using this data generation process, a first dataset is created and used for the tuning of the $\alpha$ parameter controlling the

Table 4.1: Simulation 1. IMSE and QE of estimators FQFM, $\text{FQFM}_{50\%}$, QFM and PCA for three different quantile levels.

| | FQFM | $\text{FQFM}_{50\%}$ | QFM | FPCA |
|---|---|---|---|---|
| | | $\tau = 0.1$ | | |
| IMSE | 0.119 | 0.120 | 0.183 | – |
| QE | 0.259 | 0.258 | 0.261 | – |
| | | $\tau = 0.5$ | | |
| IMSE | 0.166 | 0.169 | 0.226 | 0.184 |
| QE | 0.729 | 0.729 | 0.733 | 0.744 |
| | | $\tau = 0.9$ | | |
| IMSE | 0.547 | 0.561 | 0.738 | – |
| QE | 0.370 | 0.368 | 0.373 | – |

roughness penalty of the FQFM estimator. After selecting an optimal value for $\alpha$, the data generation process is repeated 100 times, and the results are summarized in terms of the mean value of each metric.

Results from this simulation are shown in Table 4.1. One can see that the smallest error in both metrics, regardless of the quantile level, is achieved by the FQFM estimator. Additionally, the $\text{FQFM}_{50\%}$ estimator, built using a random 50% of the data points in the training set including irregular time frames varying across observations, outperforms both QFM and FPCA. Outperforming FPCA in this scenario is expected, as FPCA is known to work well under symmetric error distributions, but the error considered here is non symmetric. However, the QFM estimator fails to outperform it in terms of the IMSE, probably due to the lack of smoothness. Figure 4.3 shows a comparison of the 3 common curves estimated by FQFM and QFM algorithms at the three different quantile levels considered. Observe that the estimations of FQFM are smooth and easily interpretable: factor 1 recovers the additive shift, factor 2 recovers the $\sin(t)$ true factor and factor 3 recovers the $\sin(0.5t)$ true factor . However, this smoothness and interpretability is lost when considering QFM estimations. Additionally, Figure 4.4 shows the estimation of the intercept curve provided by FQFM at the three quantile levels. Observe that it is correctly capturing the skewness in $\varepsilon_i(t)$, as the distance between the 90% quantile and the median is larger than the distance between the 10% quantile and the median. In Figure 4.5, an observation from the test set is randomly selected and represented against its reconstruction provided by QFM and FQFM for the three different quantiles. Observe how the curve reconstruction provided by FQFM is a smooth estimation, closer to the true quantiles than the QFM estimation. Finally, Figure 4.6 (A) shows the basis functions provided by the FPCA algorithm. Observe that FPCA,

Figure 4.3: Simulation 1. Comparison of the common curves $f(t)$ estimated using FQFM and QFM at three quantile levels.

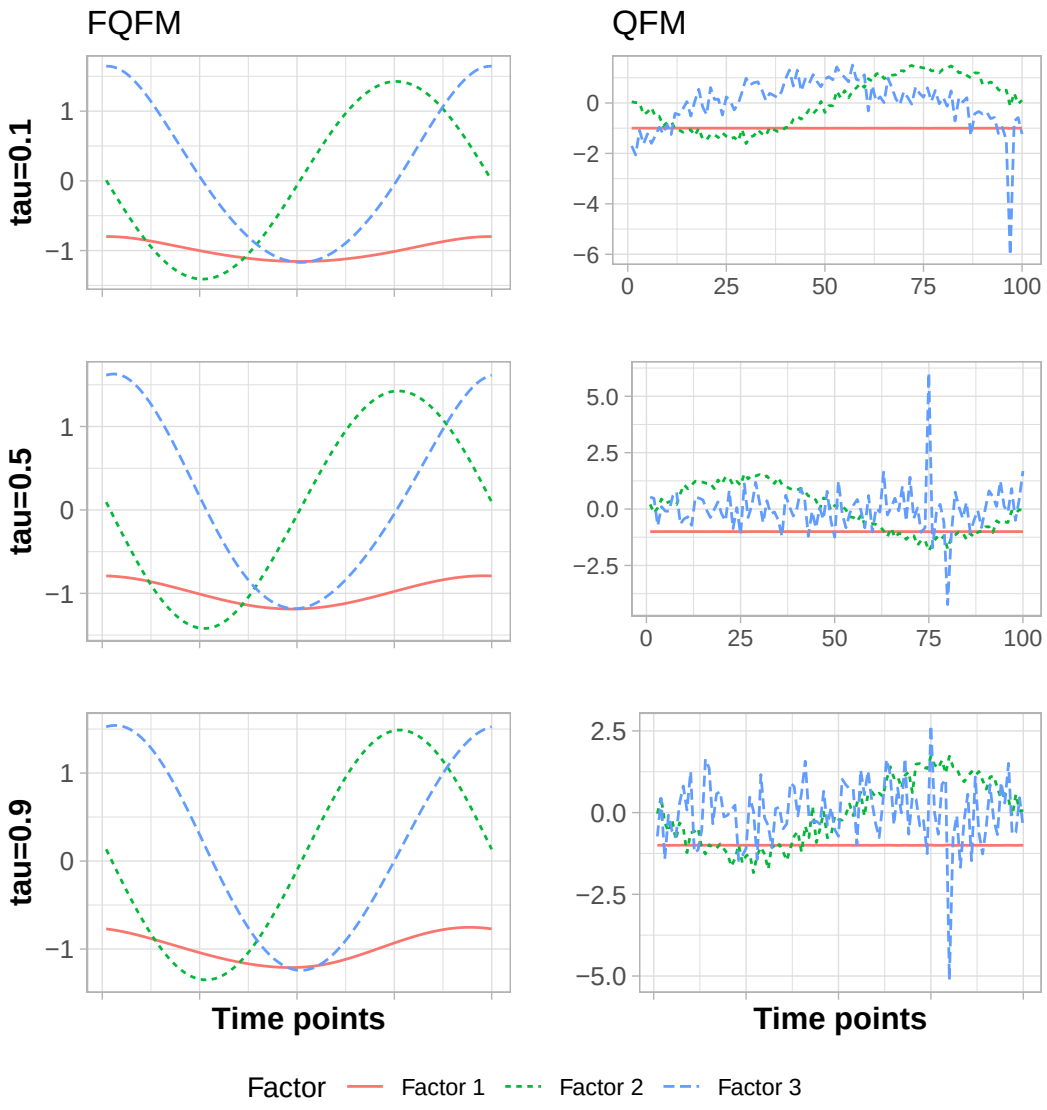



Figure 4.4: Simulation 1. Intercept curve estimation provided by FQFM algorithm at three quantile levels.

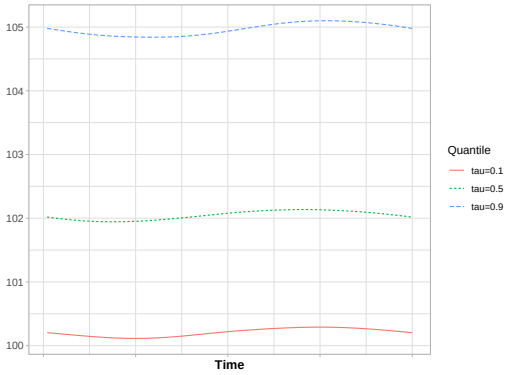
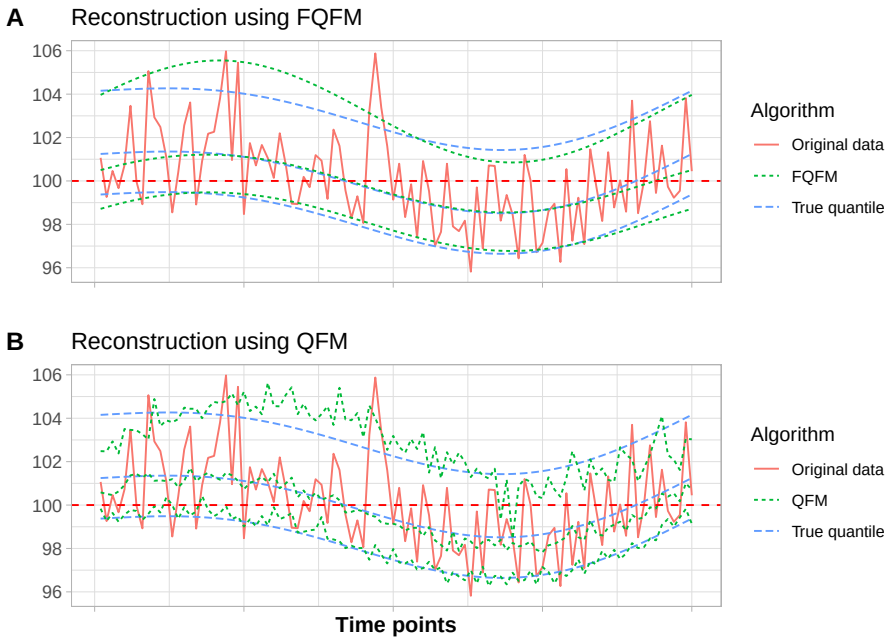


Figure 4.5: Simulation 1. Randomly selected observation from the test set compared against the reconstruction provided by FQFM (A) and QFM (B) at different quantiles.




as FQFM, correctly recovers the true underlying common curves. In plot (B) the mean estimation of a randomly selected curve is shown.

### 4.3.2 Simulation 2

Consider the following data generation process,

$$X_i(t) = 100 + \lambda_{i1} + \lambda_{i2}\sin(t) + \varepsilon_i(t) \tag{4.16}$$

where

- $\lambda_{i1} \sim N(0, 0.5)$ is the score associated to an additive factor that shifts the location of the data,
- $\lambda_{i2} \sim N(0, 1)$ is the score associated to a multiplicative factor that shifts the scale of the data,
- $\varepsilon_i(t) \sim N(0, \sigma_t)$ is a symmetric error term whose variance is a piecewise linear, monotonically increasing function of time.

A grid of 100 equally spaced time points taken in the interval $[0, 2\pi]$ is considered. A training dataset formed by 500 observations and a test set formed by additional 500 observations are generated following this process. Figure 4.7 (A) shows how the variance of $\varepsilon_i(t)$ increases along time, while (B) shows a subset of the dataset.

Figure 4.6: Simulation 1. Results from the FPCA algorithm. (A) shows the basis functions estimation. (B) shows the mean estimation of a randomly selected observation from the test set.

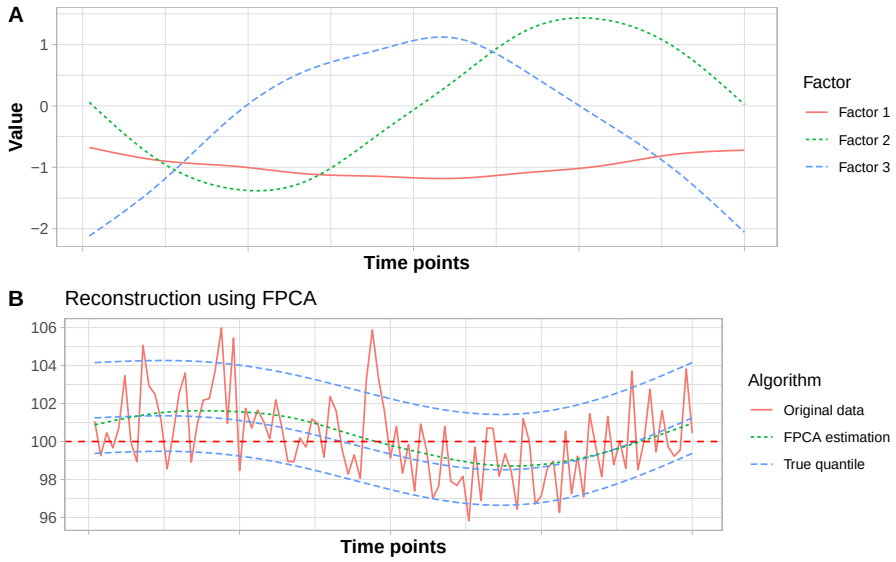



Figure 4.7: Simulation 1. plot (A) shows the increasing variance of the $\varepsilon_i(t)$ error term. plot (B) shows a subset of the generated dataset

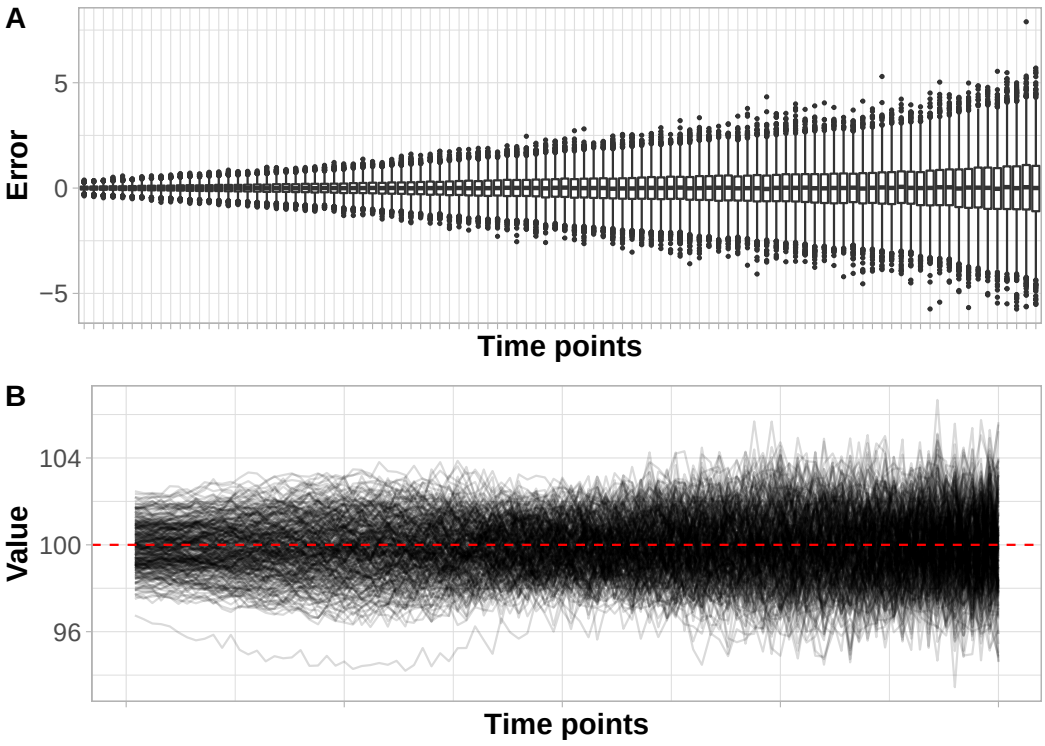


Table 4.2: Simulation 1. IMSE and QE of estimators FQFM, $\text{FQFM}_{50\%}$, QFM and PCA for three different quantile levels.

| | FQFM | $\text{FQFM}_{50\%}$ | QFM | FPCA |
|---|---|---|---|---|
| | | $\tau = 0.1$ | | |
| IMSE | 0.026 | 0.030 | 0.032 | – |
| QE | 0.115 | 0.116 | 0.116 | – |
| | | $\tau = 0.5$ | | |
| IMSE | 0.013 | 0.014 | 0.016 | 0.012 |
| QE | 0.265 | 0.265 | 0.266 | 0.267 |
| | | $\tau = 0.9$ | | |
| IMSE | 0.026 | 0.030 | 0.032 | – |
| QE | 0.115 | 0.116 | 0.116 | – |

Observe that larger time points effectively show larger variability. As in simulation 4.3.1, the number of factors estimated here will be taken equal to the number of true factors, which is 2, a location shift and a scale shift. As in the previous simulation, the roughness penalty is optimized on a first repetition of this dataset, and once optimized, the data generation process is repeated 100 additional times.

Results from this simulation are shown in Table 4.2. One can see that for quantiles 10% and 90%, the smaller error in both metrics is achieved by the FQFM estimator, while the $\text{FQFM}_{50\%}$ estimator outperforms the QFM algorithm. The best estimation for the median is provided by the FPCA estimator. This is somehow expected, as the distribution error considered in this simulation is symmetric, but the FQFM provides a very competitive alternative even in this framework. Figure 4.8 shows a comparison of the 3 common curves estimated by FQFM and QFM algorithms at the three different quantile levels considered. Observe that, as in the previous simulation, the estimations of FQFM are smooth and easily interpretable: factor 1 recovers the additive shift and factor 2 recovers the $\sin(t)$ true factor, while estimations from QFM offer some interpretability but lack smoothness. Additionally, Figure 4.9 shows the estimation of the intercept curve provided by FQFM at the three quantile levels. Observe how it is correctly estimating the increasing variability from the error term $\varepsilon_i(t)$. In Figure 4.10, an observation from the test set is randomly selected and represented against it's reconstruction provided by QFM and FQFM for the three different quantiles. As in the previous simulation, FQFM offers a smoother reconstruction, closer to the true quantiles, than QFM. Finally, Figure 4.11 (A) shows the basis functions provided by the FPCA algorithm and (B) the mean estimation of a randomly selected curve.

Figure 4.8: Simulation 2. Comparison of the common curves $f(t)$ estimated using FQFM and QFM at three quantile levels.

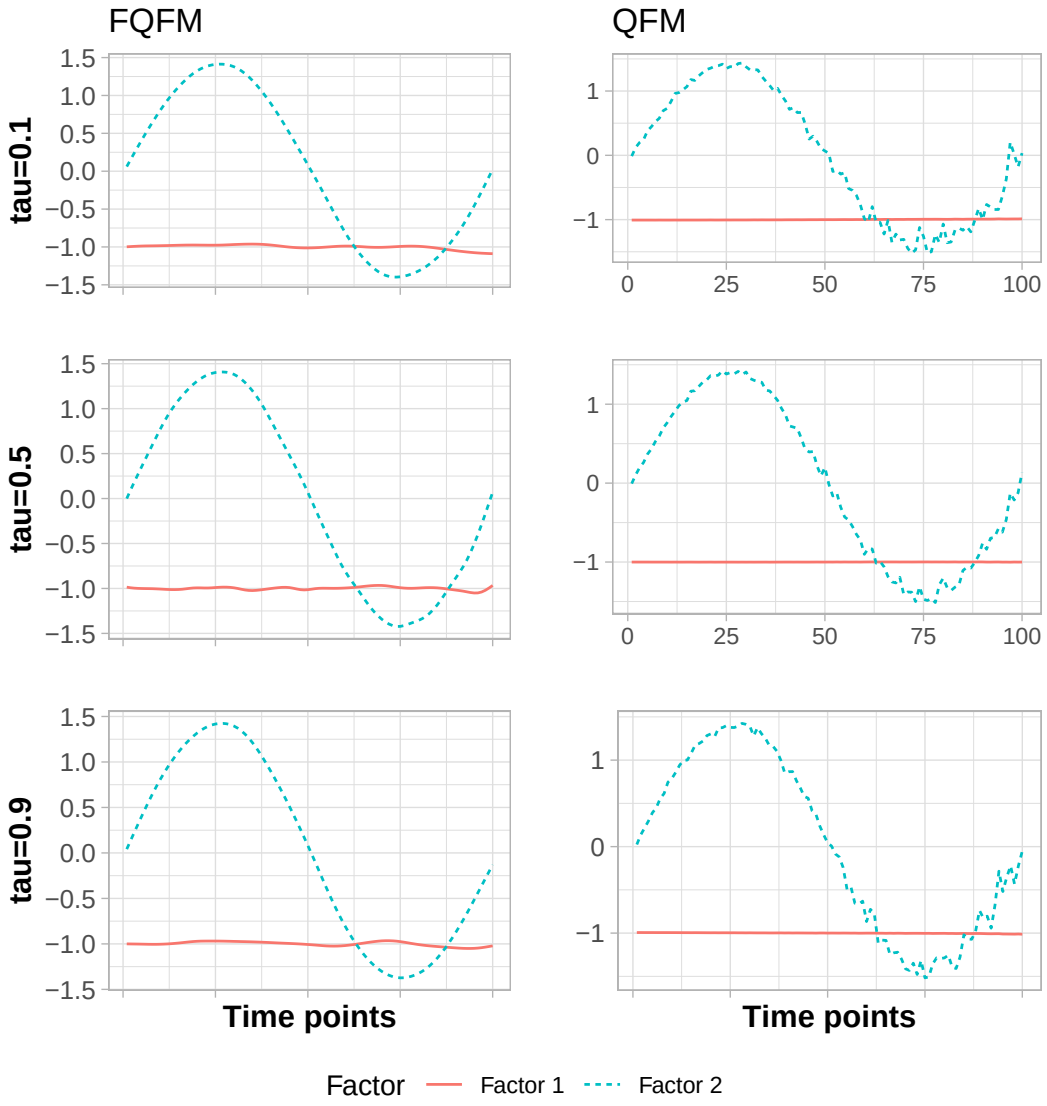



Figure 4.9: Simulation 2. Intercept curve estimation provided by FQFM algorithm at three quantile levels.

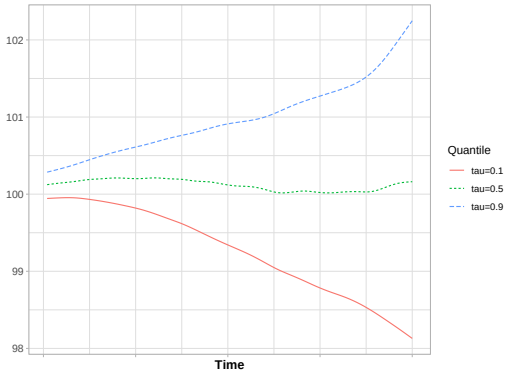


Figure 4.10: Simulation 2. Randomly selected observation from the test set compared against the reconstruction provided by FQFM (A) and QFM (B) at different quantiles.

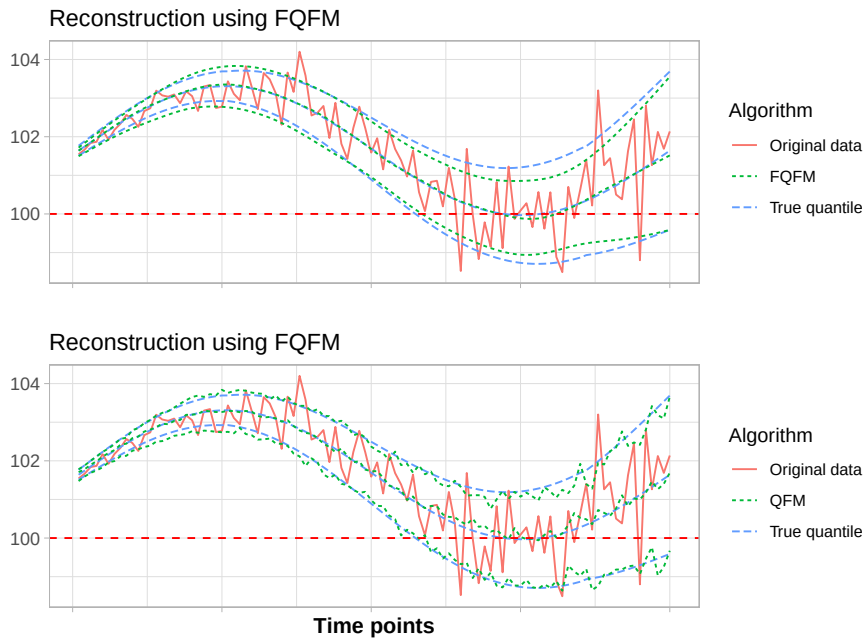



Figure 4.11: Simulation 2. Results from the FPCA algorithm. (A) shows the basis functions estimation. (B) shows the mean estimation of a randomly selected observation from the test set.

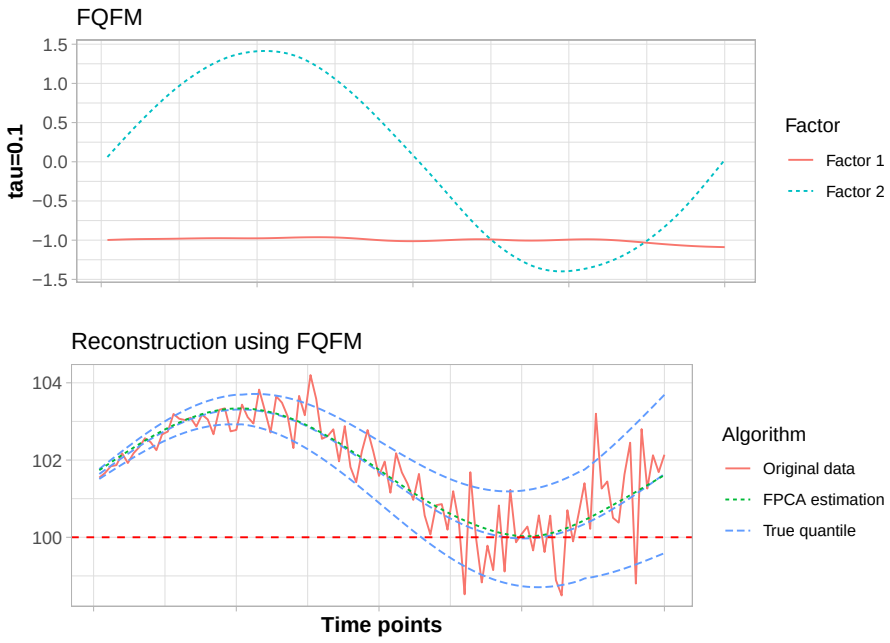
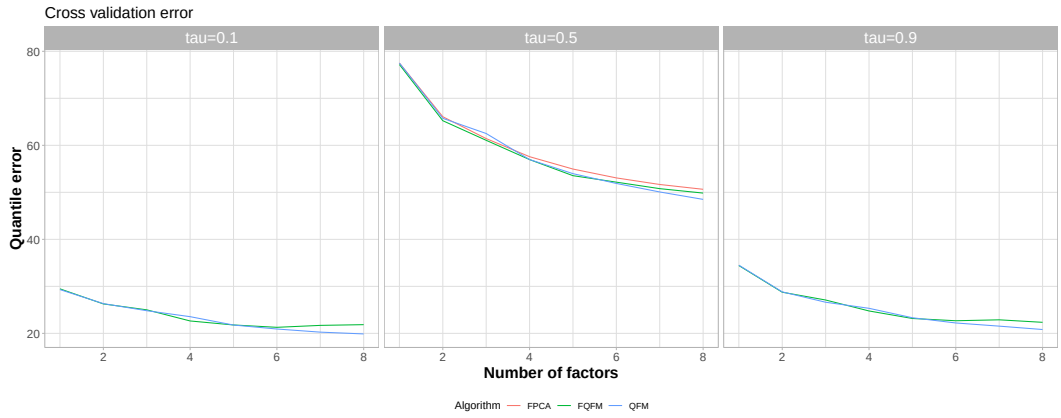


Figure 4.12: Real data analysis. Cross validation results on the number of factors measuring the quantile error.




## 4.4 Real data analysis

In this section we undertake the analysis of the child's accelerometer dataset introduced in Section 4.1. This dataset includes measurements on a grid of 144 time points (a measurement every ten minutes for a period of 24 hours) taken in 420 children. Using this dataset, a comparison of FQFM, $\text{FQFM}_{50\%}$, QFM and FPCA is performed. The selection of the number of factors is performed using 3-fold cross validation. Like in PCA, including more factors tend to reduce the overall error, but after a certain point, the reduction of the error produced by adding a factor to the model is not worth the increase in the model complexity. Figure 4.12 shows the results for FQFM, QFM and FPCA in terms of the quantile error as the number of factors increase. One can see a very large decrease when the models go from one factor to two factors, but after that the decrease stabilizes. For this reason we select 2 as the number of factors for the models.

Once the number of factor is selected, we divide the dataset into a train set formed by 300 observations and a test set formed by 120 observations, and run the proposed methodologies for quantiles 10%, 50% and 90%. The quantile errors obtained are displayed in Table 4.3. The FQFM provides the smaller quantile error for all the quantiles, outperforming even FPCA when comparing it against the median estimator. It is also worth remarking the very good performance achieved by the $\text{FQFM}_{50\%}$ even though it was built with only half the data points from the dataset. This shows great robustness in the algorithm when dealing with missing data.

Figure 4.13 shows a comparison of the 2 common curves estimated by FQFM and QFM algorithms at the three different quantile levels considered. Observe how FQFM achieves smoother curve estimates again. The first factor curve seems to capture the afternoon activity trend, while the second factor seems to capture the contrast between morning and evening activity.

Table 4.3: Real data. QE of estimators FQFM, $\text{FQFM}_{50\%}$, QFM and PCA for three different quantile levels.

| | FQFM | $\text{FQFM}_{50\%}$ | QFM | FPCA |
|---|---|---|---|---|
| | | $\tau = 0.1$ | | |
| QE | 26.13 | 26.51 | 26.17 | – |
| | | $\tau = 0.5$ | | |
| QE | 65.33 | 65.54 | 66.09 | 66.23 |
| | | $\tau = 0.9$ | | |
| QE | 28.89 | 28.92 | 29.26 | – |

Figure 4.13: Real data. Comparison of the common curves $f(t)$ estimated using FQFM and QFM at three quantile levels.

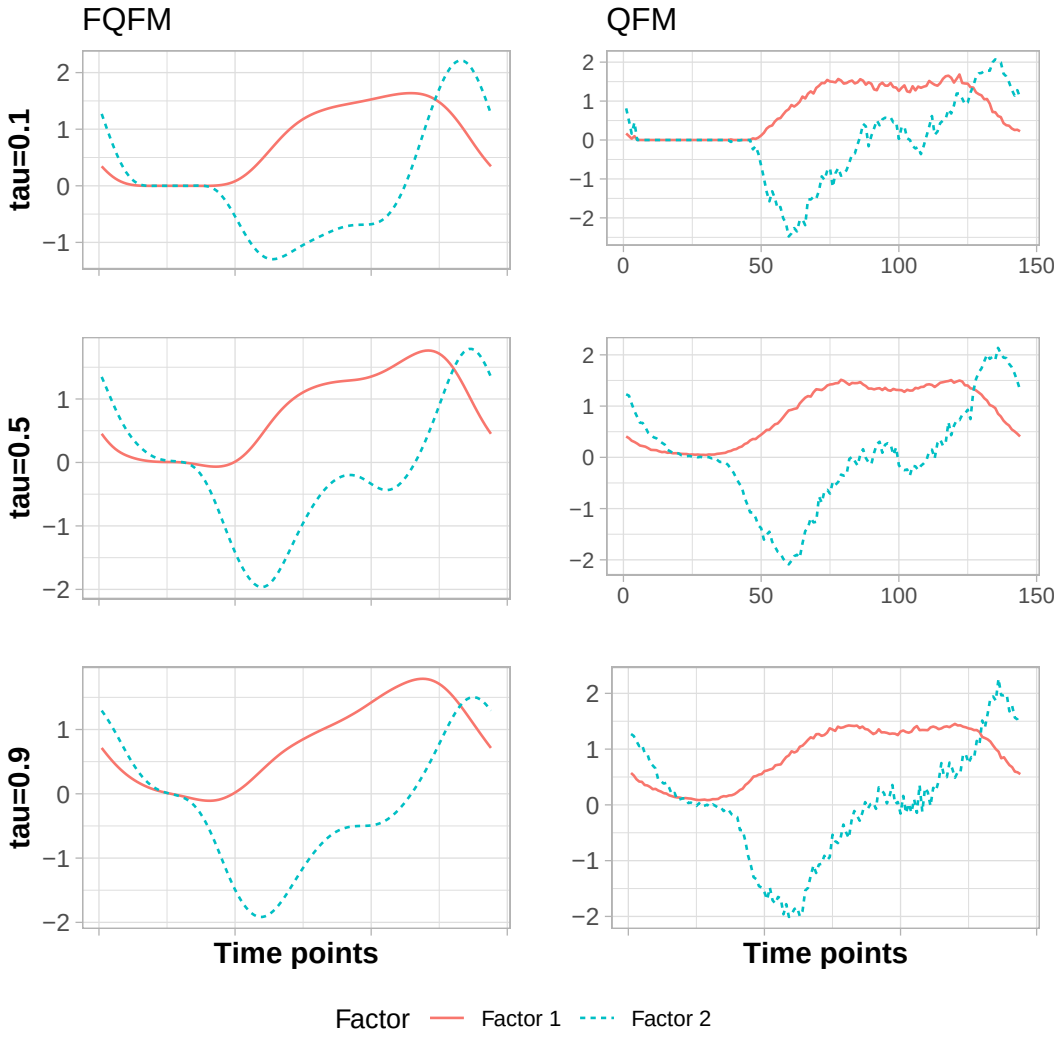
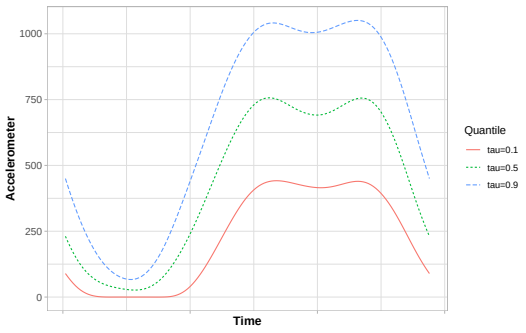


Figure 4.14: Real data. Intercept curve estimation provided by FQFM algorithm at three quantile levels.




Additionally, Figure 4.14 shows the estimation of the intercept curve provided by FQFM at the three quantile levels. In Figure 4.15, an observation from the test set is randomly selected and represented against its reconstruction provided by QFM and FQFM for the three different quantiles. As in the previous simulation, FQFM offers a smoother reconstruction than QFM. Finally, Figure 4.16 (A) shows the basis functions provided by the FPCA algorithm and (B) the mean estimation of a randomly selected curve.

## 4.5 Computational aspects

All the simulations and analysis commented in Sections 4.3 and 4.4 were run in a computer with an Intel Core i7-10750H CPU (2.6GHz) processor with 32GB RAM memory running the O.S. Windows 10. The programming of the FQFM algorithm has been developed in R [R Core Team, 2021]. The splines penalization step described in Section 4.2.2 required solving a penalized quantile regression model. This was programmed using the R package CVXR [Fu et al., 2020], a framework for convex optimization in R, and Mosek solver [ApS, 2021]. Mosek is a very fast solver with a freely available academic license, but the FQFM algorithm cal also be executed using the (slightly slower) open source solver SCS [Brendan et al., 2016], or any other solver that can be integrated into CVXR.

## 4.6 Conclusion

This work introduces the functional quantile factor model (FQFM). This algorithm extends the concept of functional principal components to the quantile regression framework. The result is a dimensionality reduction technique capable of estimating the different quantiles of the data conditional on a set of common functions. Being based on the quantiles, this estimator is robust to the presence of outliers and can deal with heteroscedastic data. Also, understanding the quantile trends found in

Figure 4.15: Real data. Randomly selected observation from the test set compared against the reconstruction provided by FQFM (A) and QFM (B) at different quantiles.

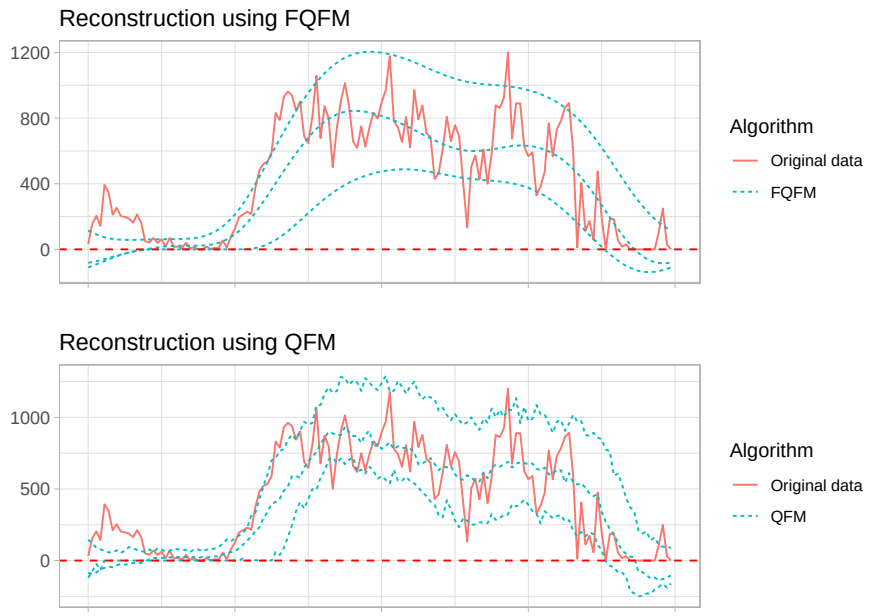
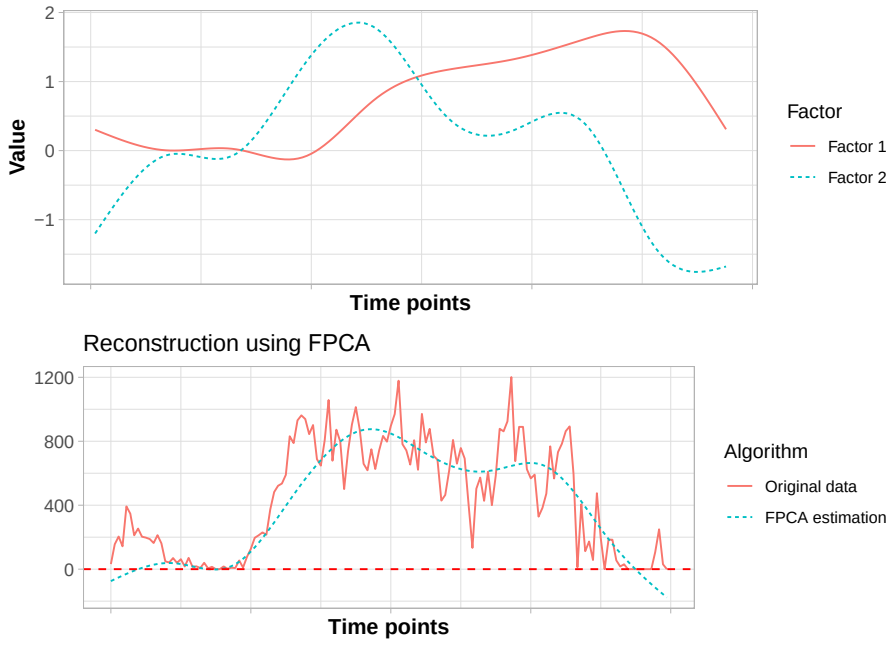



Figure 4.16: Real data Results from the FPCA algorithm. (A) shows the basis functions estimation. (B) shows the mean estimation of a randomly selected observation from the test set.



the data can provide useful insight. Section 4.3 studies the performance of the FQFM algorithm in a skewed dataset and a scenario where the error variability is a function of time, while in Section 4.4 its performance is studied in a real dataset measuring levels of physical activity in childrens during the day. When comparing it against the QFM method proposed by [Chen et al., 2021] one can see the advantages of our proposal, as it provides smaller error measurements, and smoother, more interpretable results. Additionally, the algorithm can be executed even under the situation when observations are measured at irregular time points that vary across individuals, and shows great robustness in the results achieved even under this more difficult scenario. If the median estimation is computed, one can see the FQFM estimation as a robust alternative of FPCA, showing an advantage in asymmetric scenarios and obtaining competitive results in symmetric scenarios.

## Acknowledgments

This research was partially supported by research grant PID2019-104901RB-I00 from Agencia Estatal de Investigación, as well as the National Institute of Neurological Disorders and Stroke Award R01NS097423-01 from the National Institutes of Health.

# Bibliography


Mosek ApS. MOSEK Optimizer API for Python 9.3.6, 2021. URL https://docs.mosek.com/9.3/pythonapi/index.html.

O'Donoghue Brendan, Chu Eric, Parikh Neal, and Boyd Stephen. Operator Splitting for Conic Optimization via Homogeneous Self-Dual Embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 6 2016. doi: 10.1007/s10957-016-0892-3. URL https://doi.org/10.1007/s10957-016-0892-3.

Hervé Cardot, Christophe Crambes, and Pascal Sarda. Quantile regression when the covariates are functions. *Journal of Nonparametric Statistics*, 17(7):841–856, 2005. ISSN 10485252. doi: 10.1080/10485250500303015.

Kehui Chen and Hans Georg Müller. Conditional quantile analysis when covariates are functions, with application to growth data. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 74(1):67–89, 2012. ISSN 13697412. doi: 10.1111/j.1467-9868.2011.01008.x.

Liang Chen, Juan J. Dolado, and Jesús Gonzalo. Quantile Factor Models. *Econometrica*, 89(2):875–910, 2021. ISSN 0012-9682. doi: 10.3982/ecta15746.

Anqi Fu, Balasubramanian Narasimhan, and Stephen Boyd. CVXR: An R Package for Disciplined Convex Optimization. *Journal of Statistical Software*, 94(14):1–34, 2020. doi: 10.18637/jss.v094.i14.

Jeff Goldsmith, Xinyue Liu, Judith Jacobson, and Andrew Rundle. New Insights into Activity Patterns in Children, Found Using Functional Data Analyses. *Med Sci Sports Exerc*, 48(9):1723–1729, 2016. doi: 10.1249/MSS.0000000000000968.

Harold Hotelling. Analysis of a complex of statistical variables into Principal Components. *The Journal of Educational Psychology*, 24:417–441, 1933.

Gareth M. James, Trevor J. Hastie, and Catherine A. Sugar. Principal component models for sparse functional data. *Biometrika*, 87(3):587–602, 2000. ISSN 00063444. doi: 10.1093/biomet/87.3.587.

Kengo Kato. Estimation in functional linear quantile regression. *Annals of Statistics*, 40(6):3108–3136, 2012. ISSN 00905364. doi: 10.1214/12-AOS1066.

Roger Koenker and Gilbert Bassett. Regression Quantiles. *Econometrica*, 46(1): 33–50, 1 1978. ISSN 00129682. doi: 10.2307/1913643.

Jeffrey S. Morris, Cassandra Arroyo, Brent A. Coull, Louise M. Ryan, Richard Herrick, and Steven L. Gortmaker. Using Wavelet-Based Functional Mixed Models to Characterize Population Heterogeneity in Accelerometer Profiles: A Case Study. *Journal of the American Statistical Association*, 101(576):1352–1364, 2006. doi: 10.1198/016214506000000465.

Finbarr O'Sullivan. A Statistical Perspective on Ill-posed Inverse Problems. *Statistical Science*, 1(4):502–527, 1986. ISSN 2168-8745. doi: 10.1214/ss/1177013525.

R Core Team. R: A Language and Environment for Statistical Computing, 2021. URL https://www.r-project.org/.

J.O. Ramsay and B.W. Silvermann. *Functional Data Analysis. Springer Series in Statistics*. Springer, 1998. ISBN 9780387400808. doi: 10.1007/b98888.

Vijay R. Varma, Debangan Dey, Andrew Leroux, Junrui Di, Jacek Urbanek, Luo Xiao, and Vadim Zipunnikov. Re-evaluating the effect of age on physical activity over the lifespan. *Physiology & behavior*, 101:102–108, 2017. doi: 10.1016/j.ypmed.2017.05.030.

Luo Xiao, Lei Huang, Jennifer A. Schrack, Luigi Ferrucci, Vadim Zipunnikov, and Ciprian M. Crainiceanu. Quantifying the lifetime circadian rhythm of physical activity: A covariate-dependent functional approach. *Biostatistics*, 16(2):352–367, 2015. ISSN 14684357. doi: 10.1093/biostatistics/kxu045.

Hojin Yang, Veerabhadran Baladandayuthapani, Arvind U.K. Rao, and Jeffrey S. Morris. Quantile Function on Scalar Regression Analysis for Distributional Data. *Journal of the American Statistical Association*, 0(0):1–39, 2019. ISSN 1537274X. doi: 10.1080/01621459.2019.1609969. URL https://doi.org/10.1080/01621459.2019.1609969.

# CHAPTER 5

## **asgl**: A Python Package for Penalized Linear and Quantile Regression

In *arXiv preprint arXiv:2111.00472*, (2021).

Álvaro Méndez Civieta[1,2], M. Carmen Aguilera-Morillo[2,3] and Rosa E. Lillo[1,2].

1. Department of Statistics, Universidad Carlos III de Madrid.
2. uc3m-Santander Big Data Institute.
3. Department of Applied Statistics and Operational Research, and Quality, Universitat Politècnica de València

## Abstract

**asgl** is a Python package that solves penalized linear regression and quantile regression models for simultaneous variable selection and prediction, for both high and low dimensional frameworks. It makes very easy to set up and solve different types of lasso based penalizations among which the asgl (adaptive sparse group lasso, that gives name to the package) is remarked. This package is built on top of **cvxpy**, a Python-embedded modeling language for convex optimization problems and makes extensive use of **multiprocessing**, a Python module for parallel computing that significantly reduces computation times of **asgl**.

**keywords:** Regression, Variable-selection, High-dimension, Python.

## 5.1 Introduction

In this paper, **asgl**, an open source Python [van Rossum and Drake, 2009] package for solving penalized linear regression and quantile regression models is introduced. While linear regression is very well known and need no introduction, quantile regression is less known. Quantile regression provides an estimation of the conditional quantile of a response variable as a function of the covariates, it is robust against outliers and can deal with heteroscedastic datasets (as opposed to linear regression). Ever since the seminal work from Koenker and Bassett [1978] it has become more and more used in practical applications thanks to these properties.

Penalized regression is a field under intense research nowadays due to the increasing number of problems where high dimensional data (in which the number of variables $p$ is larger than the number of observations $n$) can be seen. It is not uncommon to find this scenario in problems from fields such as genetics [Simon et al., 2013], finances [Rapach et al., 2013] or pattern recognition [Wright et al., 2010] among many others. One of the best known automatic prediction and variable selection alternatives for high dimensional data is lasso (least absolute shrinkage and selection operator) [Tibshirani, 1996], which makes use of an $\ell_1$ norm to provide sparse estimations of the coefficients of the model. After Yuan and Lin [2006] proposed the group lasso penalization, a considerable literature was generated about the selection of variables both at group and within-group levels, among which the sparse group lasso, proposed by Friedman et al. [2010] is worth to remark. This penalization generalizes lasso and group lasso providing solutions that are both between and within group sparse.

The penalizations mentioned above achieve good prediction results, but face the same underlying theoretical problem. They all provide biased solutions due to the usage of constant penalization rates, fact that can affect the quality of variable selection and prediction accuracy. This problem was addressed by Zou [2006] who proposed using what is known as the adaptive idea, which consists of adding some weights previously defined by the researcher to the penalization term. Traditionally, these weights are computed based on non penalized models, a fact that limited the usage of adaptive formulations to low dimensional frameworks where non penalized models could be solved. However, Mendez-Civieta et al. [2021] defined an asgl (adaptive sparse group lasso) estimator and proposed a series of weight calculation alternatives based on dimensionality reduction techniques pca (principal component analysis) and pls (partial least squares), that provide very good results in both high dimensional and low dimensional frameworks.

It is possible to find implementations of many of the penalization alternatives discussed above in different programming languages. One can find lasso, group lasso and sparse group lasso penalized linear regression models in R (for example, using the

Table 5.1: Overview of availability of penalizations mentioned along Section 5.1 in R, Python and Matlab.

| | R | Python | Matlab |
|---|---|---|---|
| Linear regression | Lasso | Lasso | Lasso |
| | Group lasso | Group lasso | |
| | Sparse group lasso | | |
| Quantile regression | Lasso | | |

**sgl** package), lasso and group lasso linear models in Python (in **sklearn** and **group-lasso** respectively) and lasso models in Matlab. Regarding quantile regression, it is possible to find lasso penalized models in the **quantreg** package for R, but no quantile regression penalization alternatives are available in Python or Matlab. Additionally, to the best of our knowledge, there is no statistical software currently implementing the usage of adaptive penalizations, which are known to provide much better results than the non-adaptive counterparts. An overview of this situation is shown in Table 5.1.

The **asgl** package solves this problem. It provides an easy to use framework where lasso, group lasso, sparse group lasso, and adaptive versions of all these penalizations can be solved for both linear models and quantile regression models. It also implements the main weight calculation alternatives proposed in Mendez-Civieta et al. [2021] for adaptive formulations as well as a grid search based system for optimizing the parameter values using cross validation.

The rest of the paper is organized as follows. In Section 5.2, the linear regression and quantile regression models are specified, along with lasso, group lasso, sparse group lasso, and adaptive penalizations. In Section 5.3 the main **asgl** classes are described in detail. Section 5.4 then illustrates the usage of this package to a variety of examples. Section 5.6 concludes.

## 5.2 Theoretical background

Given a sample of $n$ observations structured as $\mathbb{D} = (y_i, \boldsymbol{x}_i),\ i = 1, \ldots, n$ from some unknown population, define a linear model,

$$y_i = \boldsymbol{x}_i^t \boldsymbol{\beta} + \varepsilon_i,\ \ i = 1, \ldots, n, \tag{5.1}$$

where $y_i$ is the i-$th$ observation of the response variable, $\boldsymbol{x}_i \equiv (x_{i1}, \ldots, x_{ip})$ is the vector of $p$ covariates for observation $i$ and $\boldsymbol{\varepsilon}_i$ is the error term.

### 5.2.1 Least squares and quantile regression

#### Least squares regression

Least squares regression, usually known simply as linear regression, provides an estimation of the conditional mean of the response variable as a function of the covariates. The solution of this type of models is obtained by minimizing the risk function,

$$R(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^t\boldsymbol{\beta})^2. \tag{5.2}$$

#### Quantile regression

Quantile regression, initially proposed by Koenker and Bassett [1978] provides an estimation of the conditional quantile of the response variable as a function of the covariates. Opposed to least squares regression, quantile regression offers robust estimators when dealing with heteroscedasticity and outliers, fact that has made it increasingly popular on recent years. For a full review on quantile regression we recommend Koenker [2005]. The solution of quantile regression models is obtained from the minimization of the risk function,

$$R(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - \boldsymbol{x}_i^t\boldsymbol{\beta}), \tag{5.3}$$

where $\rho_\tau(u)$ is the loss check function defined as $\rho_\tau(u) = u(\tau - I(u < 0))$.

### 5.2.2 Penalized regression

We call high dimensional to a framework in which the number of covariates $p$ is larger than the number of observations $n$ ($n < p$). This situation is becoming more and more common, and can be found in problems from different fields such as genetics (Yahya Algamal and Hisyam Lee [2019]), finance (Rapach et al. [2013]) or climate data (Chatterjee et al. [2011]) among others.

#### Lasso

Regression models including a penalization constraint have become very popular in high dimensional problems, and one of the best known penalization alternatives is lasso. Lasso, initially proposed by Tibshirani [1996] solves the problem,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\left\{R(\boldsymbol{\beta}) + \lambda\left\|\boldsymbol{\beta}\right\|_1\right\}, \tag{5.4}$$

where $R(\beta)$ can be either the linear regression or the quantile regression risk function defined in equations 5.2 and 5.3 respectively, and $\lambda$ is a parameter controlling the weight given to the penalization. Lasso sends part of the beta coefficients to zero, yielding solutions that are sparse and performing automatic prediction and variable selection.

### Group lasso

There are situations in which data has a natural grouped structure, and group sparsity rather than individual sparsity is desired. One can think, for example, in groups of genetical pathways in the field of genetics or groups of technical indicators in finances. The problem of selecting the most important groups, rather than the most important variables was faced by Yuan and Lin [2006] who proposed the usage of a group lasso estimator,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \lambda \sum_{l=1}^{K} \sqrt{p_l} \left\| \boldsymbol{\beta}^l \right\|_2 \right\}, \tag{5.5}$$

where $K$ is the number of groups, $\boldsymbol{\beta}^l \in \mathbb{R}^{p_l}$ are vectors of components of $\boldsymbol{\beta}$ from the l-$th$ group, and $p_l$ is the size of the l-$th$ group. Group lasso works in a similar way to lasso, but while lasso provides individual sparse solutions, group lasso provides group sparse solutions, selecting or sending to zero whole groups of variables.

### Sparse group lasso

Sparse group lasso, initially proposed by Friedman et al. [2010], is defined as a linear combination between lasso and group lasso that generalizes both penalizations providing solutions that are between and within group sparse. This penalization solves the problem,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \alpha\lambda \left\| \boldsymbol{\beta} \right\|_1 + (1-\alpha)\lambda \sum_{l=1}^{K} \sqrt{p_l} \left\| \boldsymbol{\beta}^l \right\|_2 \right\}, \tag{5.6}$$

where $\alpha$ controls the balance between lasso and group lasso.

### Adaptive penalizations

All the penalizations defined above face the same problem: they apply a constant penalization rate that provides biased estimates, fact that can affect the quality of the variable selection increasing prediction error. As a solution to this problem, Zou [2006] proposed using what is now known as the adaptive idea, and defined an adaptive lasso. This idea consists of adding some weights previously defined by the

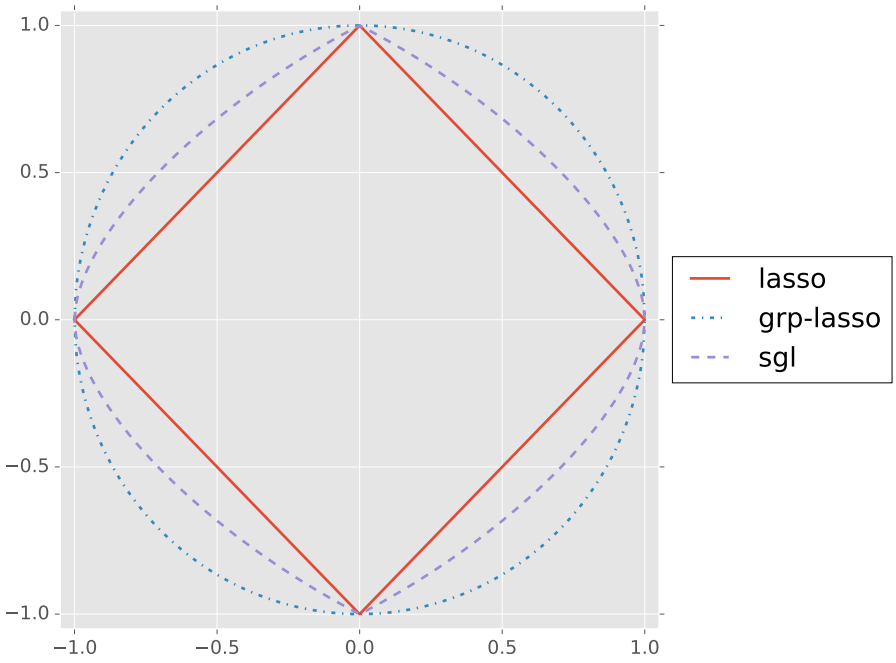



Figure 5.1: Contour lines for lasso, group lasso and sparse group lasso penalties in the case of a single 2-dimensional group

researcher to the penalization. Using the adaptive idea, Mendez-Civieta et al. [2021] defined the adaptive sparse group lasso as,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \left\{ R(\boldsymbol{\beta}) + \alpha\lambda \sum_{j=1}^{p} \tilde{w}_j |\beta_j| + (1-\alpha)\lambda \sum_{l=1}^{K} \sqrt{p_l}\tilde{v}_l \left\|\boldsymbol{\beta}^l\right\|_2 \right\}, \tag{5.7}$$

where $\tilde{\boldsymbol{w}} \in \mathbb{R}^p$ and $\tilde{\boldsymbol{v}} \in \mathbb{R}^K$ are known weights vectors. Intuitively, if a variable (or group of variables) is important, it should have a small weight, and this way would be lightly penalized. On the other hand, if it is not important, by setting a large weight it is heavily penalized. The adaptive idea enhances model flexibility, improving variable selection and prediction. Observe that adaptive lasso and adaptive group lasso can be defined just in the same way by adding weights to lasso and group lasso respectively.

### 5.2.3 Weight calculation

The usage of adaptive penalizations opens the door to a question on how to estimate the adaptive weights $\tilde{\boldsymbol{w}} \in \mathbb{R}^p$ and $\tilde{\boldsymbol{v}} \in \mathbb{R}^K$. These weights need to be specified by the researcher prior of solving the optimization problem, and in low dimensional frameworks (where $n > p$), they can be computed as,

$$\tilde{w}_i = \frac{1}{|\tilde{\beta}_i|^\gamma}, \tag{5.8}$$

where $\tilde{w}_i$ and $\tilde{\beta}_i$ correspond to the i-$th$ element of vectors $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{\beta}}$ respectively and $\tilde{\boldsymbol{\beta}}$ is the solution vector obtained from the unpenalized (linear or quantile) regression

model, $|\cdot|$ denotes the absolute value function and $\gamma$ is a non negative constant. A small $\beta$ coefficient results into a large weight, which is heavily penalized and more likely left outside the final model. On the opposite hand, large $\beta$ coefficients result into small weights that are likely to remain in the final model. However, when dealing with a high dimensional framework solving an unpenalized model is not feasible, rendering impossible the usage of adaptive formulations. As a solution to this problem, Mendez-Civieta et al. [2021] proposed a series of weight calculation alternatives based on dimensionality reduction techniques pca (principal component analysis) and pls (partial least squares).

**pca based on a subset of components**

Given the covariates matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ with maximum rank $r = \min\{n, p\}$, the matrix of pca loadings $\boldsymbol{Q} \in \mathbb{R}^{p\times r}$ and the matrix of pca scores $\boldsymbol{Z} = \boldsymbol{XQ} \in \mathbb{R}^{n\times r}$, this proposal takes advantage of the fact that the pca scores define a low dimensional framework. Consider the submatrix $\boldsymbol{Q}_d = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_d]^t$ where $\boldsymbol{q}_i \in \mathbb{R}^p$ is the i-*th* column of the matrix $\boldsymbol{Q}$, and $d \in \{1, \ldots, r\}$ is the number of components chosen in order to explain up to a certain percentage of variability, which is fixed by the researcher. Obtain $\boldsymbol{Z}_d = \boldsymbol{XQ}_d \in \mathbb{R}^{n\times d}$ the projection of $\boldsymbol{X}$ into the subspace generated by $\boldsymbol{Q}_d$ and solve the unpenalized model,

$$\tilde{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^d}{\arg\min} \left\{ \frac{1}{n} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{z}_i^t \boldsymbol{\beta}) \right\}. \tag{5.9}$$

Using this solution, it is possible to obtain an estimation of the high dimensional scenario solution, $\hat{\boldsymbol{\beta}} = \boldsymbol{Q}_d \tilde{\boldsymbol{\beta}} \in \mathbb{R}^p$ and compute the weights as,

$$\tilde{w}_j = \frac{1}{|\hat{\beta}_j|^{\gamma_1}} \quad \text{and} \quad \tilde{v}_l = \frac{1}{\left\|\hat{\boldsymbol{\beta}}^l\right\|_2^{\gamma_2}}, \tag{5.10}$$

where $\hat{\beta}_j$ is the j-*th* component from $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\beta}}^l$ is the vector of components of $\boldsymbol{\beta}$ from the l-*th* group, and $\gamma_1$ and $\gamma_2$ are non negative constants usually taken in $[0, 2]$. An in-depth explanation of this process can be read in the original paper.

**pca based on the first component**

Each pca score is built as a linear combination of the original variables. This means that another alternative for estimating the weights can be defined as simply using the weights from the first pca loading as weights for the adaptive sparse group lasso model,

$$\tilde{w}_j = \frac{1}{|q_{1j}|^{\gamma_1}} \quad \text{and} \quad \tilde{v}_l = \frac{1}{\left\|\boldsymbol{q}_1^l\right\|_2^{\gamma_2}}, \tag{5.11}$$

where $q_{1j}$ is the j-*th* component from $\boldsymbol{q}_1$ and defines the weight associated to the j-*th* original variable, $\boldsymbol{q}_1^l$ is the vector of components of $\boldsymbol{q}_1$ from the l-*th* group and $\gamma_1$ and $\gamma_2$ are non negative constants usually taken in $[0, 2]$.

#### pls based alternatives

In the same way as for the pca proposal, two alternatives of weight calculation using pls are considered: based on a subset of pls components, and based just on the first pls component. It is worth to remark that due to the way in which the pls components are obtained, this method cannot recover all the original variability from matrix $\boldsymbol{X}$ as opposed to what happens in pca, where this is ensured by the orthogonal loadings.

#### sparse pca

Sparse pca was initially proposed by [Zou et al., 2006] as a method that computes principal components adding a lasso based penalization to standard pca. This yields to principal components that are sparse linear combinations of the original variables, though are no longer orthogonal. Using the `SparsePCA` from **sklearn** package, the **asgl** package incorporates a weight calculation alternative based on a subset of sparse pca components.

#### lasso

In addition to the weight calculation alternatives proposed above, the **asgl** package implements the usage of a lasso penalized model, that can be directly solved in high dimensional frameworks, as a first step for the calculation of adaptive based formulations.

## 5.3 Python implementation

The **asgl** package is built around four main class objects: `ASGL`, `WEIGHTS`, `CV` and `TVT`, that will be described in detail along this section.

### 5.3.1 `ASGL` class

The `ASGL` class is the central part of the package. It includes all the necessary code in order to define and solve penalized regression models. The default parameters from the `ASGL` class are,

```
model = asgl.ASGL(model, penalization, intercept=True, tol=1e-5,
    lambda1=1, alpha=0.5, tau=0.5, lasso_weights=None,
    gl_weights=None,  parallel=False, num_cores=None, solver=None,
    max_iters=500)
```

where the meaning of the arguments is,

- `model`: A string specifying the model to be solved. Valid alternatives are:
  - `"lm"` for solving linear regression models ,
  - `"qr"` for solving quantile regression models,
- `penalization`: A string specifying the type of penalization to be applied. Valid alternatives are:
  - `"lasso"` for using a lasso penalization,
  - `"gl"` for using a group lasso penalization,
  - `"sgl"` for using an sparse group lasso penalization,
  - `"alasso"` for using an adaptive sparse group lasso penalization,
  - `"agl"` for using an adaptive group lasso penalization,
  - `"asgl"` for using an adaptive sparse group lasso penalization,
  - `"asgl_lasso"` for using an sparse group lasso with adaptive weights in the lasso part,
  - `"asgl_gl"` for using an sparse group lasso with adaptive weights in the group lasso part,
- `intercept`: Boolean indicating `True` if the intercept should be fitted or `False` otherwise,
- `tol`: A floating point number, tolerance for a coefficient in the model to be considered as 0,
- `lambda1`: A number, Python `list` of numbers or 1D `numpy.array` of numbers. Parameter $\lambda$ from the penalization definitions in Section 5.2.2. It controls the level of shrinkage applied on penalizations. If more than one value is provided, the program solves one model for each possible parameter combination found,
- `alpha`: A floating point number, `list` of numbers or 1D `numpy.array` of numbers. Parameter $\alpha$ from the penalization definitions in Section 5.2.2 used in `"sgl"`, `"asgl"`, `"asgl_lasso"` and `"asgl_gl"` penalizations. It must be always bounded between 0 and 1. If more than one number is provided, the program solves one model for each possible parameter combination found,

- `tau`: A floating point number between 0 and 1. Parameter $\tau$ indicating the quantile level in quantile regression models. Value 0.5 solves the median regression. Value 0.25 solves the first quartile regression. If the model is set to `"lm"`, then `tau` has no effect,

- `lasso_weights`: A `list` of numbers, 1D `numpy.array` of numbers, or a `list` made of `list` of numbers or 1D `numpy.array` of numbers. Parameter $\boldsymbol{w}$ from Section 5.2.3 used only in penalizations including the adaptive lasso weights (`"alasso"`, `"asgl"` and `"asgl_lasso"`), it includes the values of the adaptive lasso weights, so each inner `list` or `numpy.array` must have length equal to the number of predictors in the dataset. Example: given $\boldsymbol{X} \in \mathbb{R}^{100\times 10}$ a predictor matrix with 10 covariates, `lasso_weights` can be equal to a `list` of length 10, an 1D `numpy.array` of length 10, or it can be a `list` (of any length) made of `lists` of length 10. In the latter case, each `list` will be used in order to fit different adaptive models,

- `gl_weights`: A `list` of numbers, 1D `numpy.array` of numbers, or a `list` made of `list` of numbers or 1D `numpy.array` of numbers. Parameter $\boldsymbol{v}$ from Section 5.2.3 used only in penalizations including the adaptive group lasso weights (`"agl"`, `"asgl"` and `"asgl_gl"`), it includes the values of the adaptive group lasso weights, so each inner `list` or `numpy.array` must have length equal to the number of groups in the dataset. Example: given $\boldsymbol{X} \in \mathbb{R}^{100\times 10}$ a predictor matrix with 10 covariates divided in three groups, `gl_weights` can be equal to a `list` of length 3, an 1D `numpy.array` of length 3, or it can be a `list` (of any length) made of `lists` of length 3. In the latter case, each `list` will be used in order to fit different adaptive models,

- `parallel`: Boolean that takes the value `False` if it is required that the solver run sequentially (using only one core) or `True` if it is required to run in parallel (using more than one core),

- `num_cores`: An integer number indicating the number of cores to be used if `parallel` is set to `True`. By default, the program will detect the number of cores in the machine and execute the code in the number of cores minus 1. If `parallel` is `False`, `num_cores` has no effect,

- `solver`: A string indicating the solver to be used by **cvxpy** package. It is recommended to leave this parameter with the default value, as the program will automatically select the best solver among the available options,

- `max_iters`: An integer number indicating the number of iterations to be performed by the **cvxpy** solver. It is recommended to leave this parameter with the default value,

The main methods included in this class object are `fit`, `predict` and `retrieve_parameters_value`.

### `fit` function

`fit(x, y, group_index)` is the main function in this class. Once an `ASGL` class object is created, a call to the `fit` function solves the model defined there. The parameters in this function are,

- `x`: A 2-dimensional `numpy.ndarray` matrix of predictors where the rows define the observations in the dataset and the columns define the variables.
- `y`: A 1-dimensional `numpy.ndarray` array. The response vector in the problem to solve,
- `group_index`: A 1-dimensional `numpy.ndarray` array of length equal to the number of variables. An index indicating to which group each variable belongs. Example: `group_index=numpy.array([1,2,2,3])` would mean that there are 4 variables, the first belongs to group 1, second and third belong to group 2 and the fourth to group 3. Observe that this parameter is only required when using group based penalizations `"gl"`, `"agl"`, `"sgl"`, `"asgl"`, `"asgl_gl"`.

As stated in the parameter class `ASGL` definition, the parameters `lambda1` and `alpha` can be defined as either a single number or as a `list` or `numpy.ndarray` of values, and `lasso_weights` and `gl_weights` can be defined as `list`s of values or as `list`s containing those `list`s of values. The reason for this is that the `fit` function is programmed in a way that, if lists of values are provided for the parameters, it will solve all the models yielding from all possible combinations of the different values. In other words, it creates a grid of possible parameter value combinations and solves all the models in the proposed grid, computing the $\boldsymbol{\beta}$ coefficients from each case. These results are stored as a `list` in the `ASGL` class object. Usage example:

```
>>> import asgl

>>> import numpy as np

>>> from sklearn.datasets import load_boston

>>> boston = load_boston()

>>> x = boston.data

>>> y = boston.target
```

```
>>> group_index = np.array([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4,
                            4, 5])

>>> model = asgl.ASGL(model="lm", penalization="sgl",
                      lambda1=[0.001, 0.01, 0.1],
                      alpha=[0.2, 0.5, 0.7])

>>> model.fit(x, y, group_index)

>>> coefficients = model.coef_

>>> len(coefficients)

    9
```

Firs, the **asgl** package is imported. Then, the `BostonHousing` dataset, a well known regression dataset, is imported from **sklearn** package. This dataset will be used in this and the following examples. The dataset is not known to have a natural grouped structure but observe that here a fake `group_index` is provided for the sake of the example. Finally, an `ASGL` class object is created, and the `fit` function is called, solving a model for each possible combination of parameter values `lambda1` (3 values) and `alpha` (3 values), yielding to 9 solutions. If `parallel=True` these 9 models would be solved in parallel, otherwise, they would be solved sequentially.

#### `predict` function

`predict(x_new)` function computes predictions based on the coefficients provided after executing the `fit` function. The only parameter in this function is,

- `x_new`: A 2-dimensional `numpy.ndarray` with the number of columns equal to the number of columns in the original matrix `x`. `x_new` is the matrix to be used for computing the predictions.

```
>>> predictions = model.predict(x_new=x)

>>> len(predictions)

    9
```

From `model` (the `ASGL` class object created in the example above), the `predict` function is called. In this example, `x_new` is simply taken as the original matrix `x`,

although in a real analysis, `x_new` would be the test set, or any new set from which a prediction would be required. `predictions` is a list where each element of the list stores the predictions computed using each possible solution from `coefficients`.

`retrieve parameters value` **function**

`retrieve_parameters_value(param_index)` function returns a Python `dictionary` storing the value of the parameters associated to a solution index from the array of solutions `coefficients`. The only parameter in this function is,

- `param_index`: An integer number no larger than the length of the `model.coef_` list.

```
>>> model.retrieve_parameters_value(5)

    {"lambda1": 0.01, "alpha": 0.7, "lasso_weights": None,
     "gl_weights": None}
```

This Python `dictionary` is formed by 4 elements, one for each possible parameter in the model. If the model solved did not use any of the parameters, the function will retrieve the value `None` for such parameter. For example, in this case an sparse group lasso model was solved, penalization that does not use adaptive weights. For this reason `lasso_weights` and `gl_weights` are `None`. This function is used mostly along with the `cross_validation` function that will be introduced in section 5.3.3.

### 5.3.2 `WEIGHTS` class

The `WEIGHTS` class includes all the weight calculation methods discussed in Section 5.2.3. The default parameters from the `WEIGHTS` class are,

```
weights = asgl.WEIGHTS(model="lm", penalization="asgl", tau=0.5,
    weight_technique="pca_pct", weight_tol=1e-4,
    lasso_power_weight=1, gl_power_weight=1, variability_pct=0.9,
    lambda1_weights=1e-1, spca_alpha=1e-5, spca_ridge_alpha=1e-2)
```

where the meaning of the arguments is,

- `model`: A string specifying the model to be used when solving non penalized models required in the weights computation . Valid alternatives are:
  - `"lm"` for solving linear regression models ,

  - `"qr"` for solving quantile regression models,

  The `model` specified here must be the same as in the `ASGL` class,

- `penalization`: A string specifying the type of penalization to be applied. Valid alternatives are adaptive based ones:

  - `"alasso"` for using an adaptive sparse group lasso penalization,
  - `"agl"` for using an adaptive group lasso penalization,
  - `"asgl"` for using an adaptive sparse group lasso penalization,
  - `"asgl_lasso"` for using an sparse group lasso with adaptive weights in the lasso part,
  - `"asgl_gl"` for using an sparse group lasso with adaptive weights in the group lasso part.

  The `penalization` specified here must be the same as in the `ASGL` class,

- `tau`: A floating point number between 0 and 1. Parameter $\tau$ indicating the quantile level in quantile regression models. Value 0.5 solves the median regression. Value 0.25 solves the first quartile regression. If the model is set to `"lm"`, then `tau` has no effect. The `tau` specified here must be the same as in the `ASGL` class,

- `weight_technique`: A string indicating the weight technique to use for fitting the adaptive weights. Valid values are:

  - `"pca_pct"`: It computes the weights based on pca using a subset of components,
  - `"pca_1"`: It computes the weights based on pca using the first component,
  - `"pls_pct"`: It computes the weights based on pls using a subset of components,
  - `"pls_1"`: It computes the weights based on pls using the first component,
  - `"unpenalized"`: It computes the weights using a non penalized model. This alternative is only suitable for low dimensional scenarios where the number of observations is larger than the number of variables,
  - `"lasso"`: It computes the weights using a lasso penalized model.
  - `"spca"`: It computes the weights based on sparse pca using a subset of components

- `weight_tol`: A floating point number, tolerance for a coefficient to be considered as 0,

- `lasso_power_weight`: A positive floating point number or a `list` of floating point numbers. It is the parameter $\gamma_1$ from the penalizations described in Section 5.2.2, the power at which the lasso weights $\boldsymbol{w}$ are risen,

- `gl_power_weight`: A positive floating point number or a `list` of floating point numbers. It is the parameter $\gamma_2$ from the penalizations described in Section 5.2.2, the power at which the group lasso weights $\boldsymbol{v}$ are risen,

- `variability_pct`: A floating point number between 0 and 1. Percentage of variability explained by pca or pls components used in `"pca_pct"`, `"pls_pct"` and `"spca"`. Default value is 0.9 meaning that enough coefficients in order to explain up to 90% of the variability are computed. This parameter only affects the weight techniques mentioned here, but has no effect on the rest of the alternatives,

- `lambda1_weights`: A floating point number. The value of the parameter $\lambda$ to be used when computing the weights based on `"lasso"` `weight_technique`.

- `spca_alpha`: A floating point number. The value of the `alpha` parameter in the `sparsePCA` formulation when using the `"spca"` `weight_technique`. The name of the parameter is inherited from the original source in the **sklearn** package,

- `spca_ridge_alpha`: A floating point number. The value of the `ridge_alpha` parameter in the `sparsePCA` formulation when using the `"spca"` `weight_technique`. The name of the parameter is inherited from the original source in the **sklearn** package.

The main function in the `WEIGHTS` class is `fit`

#### `fit` function

`fit(x, y, group_index)` function computes the weights for adaptive based penalizations using the information in the `WEIGHTS` class object. The parameters from this function are the same as for the `fit` function from `ASGL` class. Since this function computes adaptive weights, the usual usage of this function (if one is interested on using an adaptive penalization) would require to create a `WEIGHTS` class object, run the `WEIGHTS.fit` function, obtain the adaptive weights and pass this adaptive weights as parameters to the `lasso_weights` and `gl_weights` parameters from the `ASGL` class. For example:

```
>>> weights = asgl.WEIGHTS(
    model="qr", penalization="asgl", tau=0.5,
    lasso_power_weight=[1, 1.2],
```

```
    gl_power_weight=[0.7, 0.9, 1.1, 1.3, 1.7])

>>> lasso_weights, gl_weights = weights.fit(x, y, group_index)

>>> model = asgl.ASGL(model="qr", penalization="asgl", tau=0.5,
                      lambda1=[0.001, 0.01, 0.1],
                      alpha=[0.2, 0.5, 0.7, 0.9],
                      lasso_weights=lasso_weights,
                      gl_weights=gl_weights)

>>> model.fit(x, y, group_index)

>>> coefficients = model.coef_

>>> len(coefficients)

    120
```

Here the `weights` object, from the `WEIGHTS` class is created. Since the penalization specified is `"asgl"`, by running the `weights.fit` function, the `lasso_weights` and `gl_weights` are computed. Then, an object from the class `ASGL` is created, and the weights computed above are passed as arguments to this object. By calling the function `model.fit`, the quantile regression models with adaptive sparse group lasso penalizations are solved. The penalization parameters are:

- `lambda1`: 3 possible values
- `alpha`: 4 possible values
- `lasso_power_weight`: 2 possible values
- `gl_power_weight`: 5 possible values

Thus, a grid formed of $3 \times 4 \times 2 \times 5 = 120$ values is formed, and 120 models are solved, the results stored in `model.coef_`.

### 5.3.3 `CV` class

The `CV` class inherits all the methods from `ASGL` and `WEIGHTS` class and provides methods for performing cross validation in order to find the optimal parameter values. The default parameters from the `CV` class are,

```
cv = asgl.CV(model, penalization, intercept=True, tol=1e-5,
    lambda1=1, alpha=0.5, tau=0.5, lasso_weights=None,
    gl_weights=None, parallel=False, num_cores=None, solver=None,
    max_iters=500,weight_technique="pca_pct", weight_tol=1e-4,
    lasso_power_weight=1, gl_power_weight=1, variability_pct=0.9,
    lambda1_weights=1e-1, spca_alpha=1e-5, spca_ridge_alpha=1e-2,
    error_type="MSE", random_state=None, nfolds=5)
```

where all the arguments have been already explained in Sections 5.3.1 and 5.3.2 except for the last three, directly related with the cross validation process,

- `error_type`: A string indicating the error metric to consider. Valid values are:
  - `"MSE"`: mean squared error,
  - `"MAE"`: mean absolute error,
  - `"MDAE"`: median absolute error,
  - `"QRE"`: quantile regression error,
- `random_state`: An integer number. Seed for the pseudo-random number generation. If a value is provided, it ensures reproducibility of results,
- `nfolds`: An integer number. Number of folds to be used in cross validation.

The main function in the `CV` class is `cross_validation`

#### `cross validation` function

`corss_validation(x, y, group_index)` function performs a cross validation process based on the parameters defined in the `CV` class object. The parameters in this function are the same as in functions `fit` from `ASGL` and `WEIGHTS` classes: `x`, `y` and `group_index`. The function returns an error matrix of shape (number of models, nfolds). Usage example:

```
>>> cv_class = asgl.CV(model="qr", penalization="sgl",
                       lambda1=[0.01, 0.1, 1, 10],
                       alpha=[0.1, 0.5, 0.9], parallel=True,
                       tau=0.1, nfolds=10, error_type="QRE",
                       random_state=3)

>>> error = cv_class.cross_validation(x, y, group_index)

>>> error.shape
```

```
(12, 10)
```

First, an object from class `CV` is created. It stores information for an sparse group lasso quantile regression model to be solved, and considers a grid of 4 possible `lambda1` values and 3 possible `alpha` values, a total number of 12 models. A 10 folds cross validation process is executed, and the result of this process is an error matrix of shape $\boldsymbol{E} \in \mathbb{R}^{12\times 10}$, where $\boldsymbol{E}_{i,j}$ stores the error value from model $i$ in fold $j$. Additionally, since the `random_state` has been set to 3, future executions of the same dataset specifying the same random state would return same results, ensuring reproducibility.

### 5.3.4 `TVT` class

The `TVT` class inherits all the methods from `ASGL` and `WEIGHTS` class and provides methods for performing train /validate / test analysis. The default parameters from the `CV` class are,

```
tvt_class = asgl.TVT(model, penalization, intercept=True, tol=1e-5,
    lambda1=1, alpha=0.5, tau=0.5, lasso_weights=None,
    gl_weights=None, parallel=False, num_cores=None, solver=None,
    max_iters=500,weight_technique="pca_pct", weight_tol=1e-4,
    lasso_power_weight=1, gl_power_weight=1, variability_pct=0.9,
    lambda1_weights=1e-1, spca_alpha=1e-5, spca_ridge_alpha=1e-2,
    error_type="MSE", random_state=None, train_pct=0.05,
    validate_pct=0.05, train_size=None, validate_size=None)
```

where all the arguments have been already explained in Sections 5.3.1 and 5.3.2 except for the last six, directly related with the train /validate / test process,

- `error_type`: A string indicating the error metric to consider. Valid values are:
  - `"MSE"`: mean squared error,
  - `"MAE"`: mean absolute error,
  - `"MDAE"`: median absolute error,
  - `"QRE"`: quantile regression error,
- `random_state`: An integer number. Seed for the pseudo-random number generation. If a value is provided, it ensures reproducibility of results,
- `train_pct`: A floating point number between 0 and 1. Percentage of the dataset to be used for training the model,

- `validate_pct`: A floating point number between 0 and 1. Percentage of the dataset to be used for validation the model,
- `train_size`: An integer number indicating the number of rows from the dataset to be used for training the model. This parameter takes preference over the `train_pct` parameter,
- `validate_size`: An integer number indicating the number of rows from the dataset to be used for validating the model. This parameter takes preference over the `validate_pct` parameter.

The main function in this class is `train_validate_test`

#### `train validate test` function

`train_validate_test(x, y, group_index)` function performs a train / validate / test process based on the parameters defined in the `TVT` class object. The parameters from this function are the same as in functions `fit` from `ASGL` and `WEIGHTS` classes. The function returns a Python `dictionary` containing:

- `optimal_betas`: A 1-dimensional `numpy.ndarray`. The value of the $\beta$ coefficients that minimized the test error,
- `optimal_parameters`: A Python `dictionary` containing the value of the parameters with which the optimal solution was achieved,
- `test_error`: A positive floating point number. The value of the test error obtained with the optimal solution.

Usage example:

```
>>> tvt_class = asgl.TVT(model="qr", penalization="asgl", tau=0.5,
                         lambda1=[0.01, 0.1, 1, 10],
                         alpha=[0.1, 0.5, 0.9],
                         lasso_power_weight=[0.8, 1, 1.2],
                         gl_power_weight=[0.8, 1, 1.2],
                         parallel=True, error_type="QRE",
                         random_state=99, train_size=300,
                         validate_size=200)

>>> tvt_class.train_validate_test(x, y, group_index)

{"optimal_betas":
    array([ 2.65930767e+01,  0.00000000e+00,  0.00000000e+00,
```

```
            0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
            0.00000000e+00, -5.32798841e-02,  0.00000000e+00,
            0.00000000e+00, -1.52427363e-02,  0.00000000e+00,
            9.51042157e-03,  0.00000000e+00]),
 "optimal_parameters": {"lambda1": 0.01, "alpha": 0.9,
         "lasso_weights":
             array([ 316.50255799,  247.98156613,  336.68766214,
                     7756.72189464, 4777.64968308, 2853.7114713,
                     140.44162671,  1015.40062999, 235.2278634,
                     20.949954,     1144.91563613, 44.53821725,
                     361.15079837]),
         "gl_weights":
             array([ 189.48216987, 2754.77796665,  127.62155584,
                     19.8070448,   361.15079837])},
 "test_error": 3.387824170724728}
```

First, an object of class `TVT` is created containing information for solving a quantile regression model with an adaptive sparse group lasso penalization. A `train_size=300` is indicated, meaning that 300 rows from the dataset will be used in the training process of the models. A `validate_size=200` is indicated. This means that the validation process will be performed using 200 rows from the dataset, and the remaining rows in the dataset ($n - 300 - 200$) will be used in the test process. The result obtained is a `dictionary` as described above, containing the value of the optimal $\beta$ solution, the optimal parameter values, and the test error obtained with this solution.

### 5.3.5 Extra functions

In addition to the class objects and functions described along this section, the **asgl** package includes 2 extra functions worth mentioning: `error_calculator` and `train_test_split`.

`error calculator function`

`error_calculator(y_true, prediction_list, error_type="MSE", tau=None)` is a function that allows to compute the error obtained from a prediction or list of predictions. The parameters of this function are,

- `y_true`: A 1-dimensional `numpy.ndarray` containing the true values of the response variable,
- `prediction_list`: A `list` of predictions, output from the `ASGL.predict` function described in Section 5.3.1

- `error_type`: A string indicating the error metric to consider. Valid values are:
  - `"MSE"`: mean squared error,
  - `"MAE"`: mean absolute error,
  - `"MDAE"`: median absolute error,
  - `"QRE"`: quantile regression error,
- `tau`: A floating point number between 0 and 1. Parameter $\tau$ indicating the quantile level in quantile regression models. This parameter is only used if a quantile regression model was solved.

Usage example:

```
>>> model = asgl.ASGL(model="lm", penalization="sgl",
                      lambda1=[0.001, 0.01, 0.1],
                      alpha=[0.2, 0.5, 0.7])

>>> model.fit(x, y, group_index)

>>> coefficients = model.coef_

>>> predictions = model.predict(x_new=x)

>>> error = asgl.error_calculator(y_true=y,
                                  prediction_list=predictions)

>>> error

[21.895037097213336, 21.894987951816283, 21.894959521292872,
 21.915054953611918, 21.910333713620282, 21.907558654585653,
 22.6425078760837, 22.66227036349894, 22.685544907299743]
```

The result is a list of error values of the same length as the `prediction_list`.

### `train test split` function

`train_test_split(nrows, train_size=None, train_pct=0.7, random_state=None)` function randomly generates a train index and a test index to be used on a train / test split. The parameters in this function are:

- `nrows`: An integer number. The number of rows in the dataset,

- `train_size`: An integer number indicating the number of rows in the dataset that should define the train set,
- `train_pct`: A floating point number between 0 and 1 indicating the percentage of the dataset that should be used for training. The parameter `train_size` takes preference over this one,
- `random_state`: An integer number. Seed for the pseudo-random number generation. If a value is provided, it ensures reproducibility of results.

Usage example:

```
>>> train_idx, test_idx = asgl.train_test_split(
    nrows=10, train_size=7, random_state=5)

>>> train_idx

array([9, 5, 2, 4, 7, 1, 0])
```

In this example a fake `train_idx` of size 7 and a `test_idx` of size 3 are generated for a dataset containing 10 rows.

## 5.4 Examples

### 5.4.1 Non penalized models

Solving a non penalized linear regression or quantile regression model using the **asgl** package is very simple. In this section, the `BostonHousing` regression dataset, available in the **sklearn** package will be used, so the first step is to import the dataset,

```
>>> import numpy as np

>>> from sklearn.datasets import load_boston

>>> boston = load_boston()

>>> x = boston.data

>>> y = boston.target
```

After that, the **asgl** package is imported and the dataset is divided into a 70% train set and a 30% test set. Then an object of the class `ASGL` is created containing the

information about the model to be solved. In this case, it is only necesary to provide the information for parameters `model` and `penalization`, the latter, in the case of a non penalized model, shoud be left as `None`.

```
>>> import asgl

>>> train_idx, test_idx = asgl.train_test_split(
    nrows=x.shape[0], train_pct=0.7, random_state=1)

>>> unpenalized_model = asgl.ASGL(model="lm", penalization=None)

>>> unpenalized_model.fit(x=x[train_idx,], y=y[train_idx])

>>> unpenalized_model.coef_

[array([ 3.42867793e+01, -1.20183515e-01,  3.60204868e-02,
        -1.44737040e-02,  3.11649925e+00, -1.83800932e+01,
         4.33076482e+00, -1.45478637e-02, -1.58421918e+00,
         2.99294537e-01, -1.32561898e-02, -9.49536207e-01,
         9.87429820e-03, -4.29662800e-01])]
```

Once the coefficients have been obtained, one can compute the predictions and obtain the prediction error easily by calling the `unpenalized_model.predict` and `error_calculator` functions:

```
>>> predictions = unpenalized_model.predict(x_new=x[test_idx,])

>>> error = asgl.error_calculator(
    y_true=y[test_idx], prediction_list=predictions,
    error_type="MSE")
>>> error

[29.450990335365578]
```

### 5.4.2 Sparse group lasso model using cross validation

In this example a high dimensional synthetic dataset generated using the freely available **data-generation** package will be used. This package can easily be installed from the Python Package Index repository.

```
>>> import numpy as np
```

```
>>> import data_generation as dgen

>>> data_class = dgen.EqualGroupSize(
    n_obs=1000, group_size=10, num_groups=10, non_zero_groups=5,
    non_zero_coef=6, random_state=1)

>>> x, y, beta, group_index = data_class.data_generation().values()
```

The dataset generated here contains 1000 observations and 100 variables that are divided into 10 groups of size 10 each. Among these groups, 30 variables are significant (have been used in the calculation of the response variable), and the remaining 70 variables are noise. Using this dataset a quantile regression model with an sparse group lasso penalization will be solved using cross validation.

```
>>> import asgl

>>> train_idx, test_idx = asgl.train_test_split(
    nrows=x.shape[0], train_pct=0.7, random_state=1)

>>> lambda1 = (10.0 ** np.arange(-3, 1.51, 0.2))

>>> alpha = np.arange(0, 1, 0.05)

>>> cv_class = asgl.CV(model="qr", penalization="sgl",
                       lambda1=lambda1, alpha=alpha,
                       nfolds=5, error_type="QRE",
                       parallel=True, random_state=1)

>>> error = cv_class.cross_validation(x=x[train_idx,],
    y=y[train_idx], group_index=group_index)

>>> error.shape

(460, 5)
```

The synthetic dataset is divided into a train set, in which the cross validation process is executed, and a test set, used for computing the final prediction error. A grid of 23 possible `lambda1` values and 20 `alpha` values is defined, yielding a total number of 460 possible parameter combinations to be solved. `parallel` is set to `True`, which means that the cross validation process will be executed in parallel using as many cores as the maximum number available in the machine being used minus 1. A `random_state` value of 1 is set in order to ensure the reproducibility of this example.

```
>>> error = np.mean(error, axis=1)

>>> minimum_error_idx = np.argmin(error)

>>> optimal_parameters = \
    cv_class.retrieve_parameters_value(minimum_error_idx)

>>> optimal_lambda = optimal_parameters.get("lambda1")

>>> optimal_alpha = optimal_parameters.get("alpha")
```

Once the cross validation process is finished, the error obtained in the different models and across different folds is stored in `error`. The mean error across different folds is computed and then the index of the model providing the minimum error value is obtained and stored in `minimum_error_idx`. With this index, the function `retrieve_parameters_value` is used and the value of the parameters minimizing the cross validation error is recovered and stored in `optimal_lambda` and `optimal_alpha`. Now, the final model using just the optimal values obtained above can be computed in order to obtain the test error of this model.

```
>>> sgl_model = asgl.ASGL(
    model="qr", penalization="sgl", lambda1=optimal_lambda,
    alpha=optimal_alpha)

>>> sgl_model.fit(
    x=x[train_idx, :], y=y[train_idx], group_index=group_index)

>>> final_beta_solution = sgl_model.coef_[0]

>>> final_prediction = sgl_model.predict(x_new=x[test_idx, :])

>>> final_error = asgl.error_calculator(
    y_true=y[test_idx], prediction_list=final_prediction,
    error_type="QRE", tau=0.5)

>>> final_error

[0.6156290699941908]
```

Additionally, given that it is a synthetic dataset, it is possible to compute the true positive rate, a measure of the accuracy of the variable selection comparing the true `beta` with the predicted `final_beta_solution`. First, we remove the intercept from the `final_beta_solution`,

```
>>> predicted_beta = final_beta_solution[1:]

>>> true_positive = np.sum(
    np.logical_and(np.abs(beta) > 1e-4,
    np.abs(predicted_beta) > 1e-4))

>>> real_positive = np.sum(np.abs(beta) > 1e-4)

>>> true_positive_rate = true_positive / real_positive

>>> true_positive_rate

1.0
```

A `true_positive_rate` equal to 1 means that 100% of the significant variables were correctly detected as significant.

### 5.4.3 Adaptive lasso using train / validate / test

For this example, a synthetic dataset will be generated using the `make_regression` function from the **sklearn** package.

```
>>> import numpy as np

>>> import asgl

>>> import sklearn.datasets

>>> x, y, beta = sklearn.datasets.make_regression(
                        n_samples=100, n_features=200,
                        n_informative=10, n_targets=1,
                        bias=10.0, noise=1.0, shuffle=True,
                        coef=True, random_state=1)
```

The above code generates a synthetic dataset formed by 100 observations and 200 variables (a truly high dimensional framework) where only 10 of the variables are significant (used in the computation of the response variable) and the remaining 190 are noise. Using a train / validate / test split, an adaptive lasso model is computed. Additionally, a simple lasso model is computed to act as benchmark of the benefits of adaptive models.

```
>>> lambda1 = 10.0 ** np.arange(-3, 1.51, 0.1)
```

```
>>> tvt_lasso = asgl.TVT(
    model="lm", penalization="lasso", lambda1=lambda1,
    parallel=True, error_type="MSE", random_state=1,
    train_size=50, validate_size=25)

>>> lasso_result = tvt_lasso.train_validate_test(x=x, y=y)

>>> lasso_prediction_error = lasso_result["test_error"]

>>> lasso_betas = lasso_result["optimal_betas"][1:]

>>> lasso_prediction_error

11.779487290509907
```

The above code performs a train / validate / test optimization of a lasso model using 50 observations in the training process, 25 observations in the validation process and 25 observations in the test process. The optimal betas obtained are stored in `lasso_betas` and the prediction error is stored in `lasso_prediction_error`. Now an adaptive lasso model is solved using a lasso model in the weight computation.

```
>>> tvt_alasso = asgl.TVT(
    model="lm", penalization="alasso", lambda1=lambda1,
    parallel=True, weight_technique="lasso", error_type="MSE",
    random_state=1, train_size=50, validate_size=25)

>>> alasso_result = tvt_alasso.train_validate_test(x=x, y=y)

>>> alasso_prediction_error = alasso_result["test_error"]

>>> alasso_betas = alasso_result["optimal_betas"][1:]

>>> alasso_prediction_error

1.472196185408593
```

While the solution achieved by the lasso model had a prediction error of 11.77, the solution achieved by the adaptive lasso is 1.47, 8 times smaller. Additionally, it is possible to compute the correct selection rate of both alternatives

```
>>> def correct_selection_rate(predicted_beta, true_beta):
        true_positive = np.sum(np.logical_and(
```

```
        np.abs(true_beta) > 1e-4,
        np.abs(predicted_beta) > 1e-4))
        true_negative = np.sum(np.logical_and(
        np.abs(true_beta) <= 1e-4,
        np.abs(predicted_beta) <= 1e-4))
        correct_selection_rate = \
        (true_positive + true_negative) / len(true_beta)
        return correct_selection_rate

>>> lasso_csr = correct_selection_rate(
    predicted_beta=lasso_betas, true_beta=beta)

>>> alasso_csr = correct_selection_rate(
predicted_beta=alasso_betas, true_beta=beta)

>>> print(f"Lasso correct selection rate: {lasso_csr}"
    f"\nAdaptive lasso correct selection rate: {alasso_csr}")


Lasso correct selection rate: 0.13
Adaptive lasso correct selection rate: 1.0
```

While the lasso only selected correctly 13% of the variables, the adaptive lasso was able to correctly select all the variables. Taking as non significant all the non significant variables and taking as significant all the significant variables.

## 5.5 Computational details

The results in this paper were obtained using Python 3.8.3 version . All the packages used in this paper are available from the Python Package Index (Pypi) at https://pypi.org/.

## 5.6 Conclusions

In this paper, a Python package called **asgl** has been introduced. **asgl** is a powerful package capable of solving linear and quantile regression models using different types of penalizations that, to the best of our knowledge, were not available in any other package or programming language so far. The usage of **asgl** has been shown throughout three examples detailed in Section 5.4.

## 5.7 Supplementary material

### 5.7.1 Installation and requirements

The **asgl** package is freely available at the Python Package Index (Pypi), one of the best known Python repositories, and can be easily installed by running the command,

```
pip install asgl
```

Alternatively, it is possible to directly pull the repository from GitHub and run the setup.py file,

```
git clone https://github.com/alvaromc317/asgl.git
cd asgl
Python setup.py
```

The package requires:

- Python (≥ 3.5) programming language version [van Rossum and Drake, 2009],
- **scikit-learn** (≥ 0.23.1) [Pedregosa et al., 2011], an open source machine learning Python library,
- **numpy** (≥ 1.15) [Van Der Walt et al., 2011] a package for efficient manipulation of multidimensional arrays,
- **cvxpy** (≥ 1.1.0) [Diamond and Boyd, 2016], a Python-embedded modeling language for convex optimization problems.
- Additionally, the package makes use of the **multiprocessing** package, which is not required to be installed, since it is included in the base program.

The package has been mainly tested in Microsoft Windows machines running Windows 10 and Windows 8 versions, as well as in Red Hat Linux machines.

## Acknowledgments

This research was partially supported by research grants and Project PID2019-104901RB-I00 from Ministerio de Ciencia e innovación, Project MTM2017-88708-P from Ministerio de Economía y Competitividad, FEDER funds and Project IJCI-2017-34038 from Agencia Estatal de Investigación, Ministerio de Ciencia, Innovación y Universidades.

# Bibliography


Soumyadeep Chatterjee, Snigdhanshu Banerjee, Arindam, and Auroop R. Ganguly. Sparse Group Lasso for Regression on Land Climate Variables. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 1–8. IEEE, 12 2011. ISBN 978-1-4673-0005-6. doi: 10.1109/ICDMW.2011.155.

Steven Diamond and Stephen Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *arXiv:1603.00943*, 3 2016.

J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *ArXiv:1001.0736*, pages 1–8, 2010. ISSN 15410420. doi: 10.1111/biom.12292. URL http://arxiv.org/abs/1001.0736.

Roger Koenker. *Quantile Regression*. Cambridge university Press, 2005. ISBN 0521338255.

Roger Koenker and Gilbert Bassett. Regression Quantiles. *Econometrica*, 46(1): 33–50, 1 1978. ISSN 00129682. doi: 10.2307/1913643.

Alvaro Mendez-Civieta, M. Carmen Aguilera-Morillo, and Rosa E. Lillo. Adaptive sparse group LASSO in quantile regression. *Advances in Data Analysis and Classification*, 15(3):547–573, 2021. ISSN 18625355. doi: 10.1007/s11634-020-00413-8.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, , B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, , R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, , D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

David E Rapach, Jack K Strauss, and Guofu Zhou. International Stock Return Predictability : What Is the Role of the United States ? *The Journal of Finance*, 68(4):1633–1662, 2013. doi: 10.1111/jofi.12041.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 4 2013. ISSN 10618600. doi: 10.1080/10618600.2012.681250.

Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. doi: 10.2307/2346178.

Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13:22, 2011.

G. van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, 2009. ISBN 1441412697.

John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S. Huang, and Shuicheng Yan. Sparse Representation for Computer Vision and Pattern Recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 6 2010. ISSN 0018-9219. doi: 10.1109/JPROC.2010.2044470.

Zakariya Yahya Algamal and Muhammad Hisyam Lee. A two-stage sparse logistic regression for optimal gene selection in high-dimensional microarray data classification. *Advances in Data Analysis and Classification*, 13:753–771, 2019. doi: 10.1007/s11634-018-0334-1. URL https://doi.org/10.1007/s11634-018-0334-1.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 68 (1):49–67, 2006.

Hui Zou. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 12 2006. ISSN 0162-1459. doi: 10.1198/016214506000000735.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. doi: 10.1198/106186006X113430.

# CHAPTER 6

# Conclusions

This thesis has filled important gaps in the literature, proposing a series of quantile based methodologies that address different perspectives of high dimensional problems. In Chapter 2, the usage of variable selection techniques in quantile regression was studied. Although widely used, lasso based penalizations face the problem of being biased. To address this issue, an adaptive sparse group lasso for quantile regression was proposed in the chapter, and different alternatives for the adaptive weight computation based on PCA and PLS were proposed and studied through a series of synthetic and real genetic dataset studies showing the advantages of the proposed estimator over the non adaptive counterparts.

Chapter 3 treated the high dimensional problem from another perspective, studying the usage of dimension reduction techniques. Partial least squares (PLS) is a widely used methodology in the field of chemometrics, and in general, in any field treating with high dimensional or colinear regression problems. However, PLS is a mean based estimator very sensitive to the presence of outliers, heteroscedasticity or skewness. This opened the door for the second contribution of this thesis: the fast partial quantile regression, an algorithm that makes use of a quantile covariance metric to extend the partial least squares methodology to quantile regression, obtaining a robust alternative that can provide not only an estimation of the central trend of the response matrix, but also estimations of other quantiles, giving a complete picture of the distribution. The effectiveness of the fPQR methodology was compared against PLS and against the partial robust adaptive modified maximum likelihood estimator (PRAMML) proposed by Acitas et al. [2020], which is a robust PLS alternative for univariate response models, in a series of synthetic datasets and a real dataset containing NIR spectra measurements, obtaining very good results.

Chapter 4 considered the field of functional data analysis, in which observations are not treated as multivariate vectors, but functions. This work was motivated by a real dataset containing measurements of physical activity in children. Understanding the differences in the patterns of activity can provide very useful information. This is usually done based on functional principal component analysis. However, this technique shows some disadvantages: being based on the mean, it is unable to capture shifts in the scale of the data that may affect the quantiles. Additionally, it is not robust against skewness or the presence of outliers. As a solution to these problems, this chapter proposed the definition of the functional quantile factor model, a methodology that can provide estimations of the quantile levels of the functional data, effectively extending FPCA to quantile regression. The methodology was compared in a series of synthetic datasets and in the motivating accelerometer dataset against FPCA and against the quantile factor models proposed by [Chen et al., 2021], a multivariate approach to this problem, and obtained very good results.

Finally, Chapter 5 described the implementation of a series of lasso based adaptive penalizations in Python, including different alternatives for the computation of the adaptive weights as well as a framework for optimizing the hyper parameters of the penalizations using cross validation.

Besides its main contributions, this thesis has identified areas for further research, which can be summarized as follows.

1. To extend the adaptive based penalizations proposed in Chapter 2 to the case where the response is a multivariate matrix, rather than a vector. This is a typical situation in fields like economics or chemometrics.

2. To study the $\sqrt{n}$-consistency of the PLS estimator. A key requirement in the demonstration of the oracle property of an adaptive estimator is the fact that the adaptive weights are based on an initial $\sqrt{n}$-consistent estimator. Studying whether PLS verifies this property would sustain from a theoretical perspective the effectiveness showed in the numerical simulations of Chapter 2.

3. To study additional alternatives for the weight calculation process described in Chapter 2. On this regard, a master thesis was recently developed under the supervision of my tutor Rosa E. Lillo and mine where a series of weight calculation alternatives based on sparse group lasso, sparse PCA, sparse PLS, support vector machines, and the machine learning algorithm XGBoost were considered, obtaining promising results.

4. To consider the extension of fPQR to a sparse setting. The new components of fPQR are built as linear combinations of all the original variables. In high dimensional problems this affects the interpretability of the results. Following the steps of [Chun and Keleş, 2010], it can be interesting to consider the usage of a variable selection penalization embedded in the fPQR algorithm, obtaining

solutions where the new components are sparse combinations of the original variables and enhancing the interpretability of the results.

5. To study different alternatives for the hyper parameter tuning of the adaptive sparse group lasso estimator. The cross validation process considered in Chapter 5 is known to provide very good results, but can also be time consuming. Other alternatives like random search or bayesian optimization could be the solution.

6. To extend the implementation of the **asgl** package to different risk functions, including generalized linear models like logistic regression for classification.

This thesis started with the idea to develop new methodologies for high dimensional problems that could benefit from a quantile based perspective, and achieved this primary objective. It has extended the quantile regression framework in several ways, providing robust alternatives to well established mean based methodologies in regression, dimension reduction and functional scenarios, and has opened the door for future research, which is already underway.

# Bibliography

Sukru Acitas, Peter Filzmoser, and Birdal Senoglu. A new partial robust adaptive modified maximum likelihood estimator. *Chemometrics and Intelligent Laboratory Systems*, 204:104068, 2020. ISSN 18733239. doi: 10.1016/j.chemolab.2020.104068. URL https://doi.org/10.1016/j.chemolab.2020.104068.

Liang Chen, Juan J. Dolado, and Jesús Gonzalo. Quantile Factor Models. *Econometrica*, 89(2):875–910, 2021. ISSN 0012-9682. doi: 10.3982/ecta15746.

Hyonho Chun and Sündüz Keleş. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 72(1):3–25, 1 2010. ISSN 13697412. doi: 10.1111/j.1467-9868.2009.00723.x.